Heterogeneous Graph Neural Networks with Neighbor-SIM Attention Mechanism for Substitute Product Recommendation

Liyuan Zheng[1]*, **Zhen Zuo**[2], **Wenbo Wang**[2], **Chaosheng Dong**[2], **Michinari Momma**[2], **Yi Sun**[2]

[1]Department of Electrical & Computer Engineering, University of Washington

[2]Amazon

liyuanz8@uw.edu, {zhenzuo, wenbowan, chaosd, michi, yisun}@amazon.com

Abstract

Substitute product recommendation is important to improve customer experience in E-commerce. Recent developments in Graph Neural Networks (GNNs) show superior advantages in recommendation systems, as they are able to leverage both content information of each item, and connections among items that reflect customer behavior interactions. Despite a substantial amount of effort has been made to homogeneous and heterogeneous GNN, few works shown generalization and efficiency in large-scale heterogeneous graph structure and none of them are optimized for substitute product recommendation task. In this paper, first, we present HetSAGE, a novel general framework for heterogeneous graph structure. In HetSAGE, we build a two-level hierarchical information aggregation: neighbor-level aggregation and edge-level aggregation. Specifically, for each node, the neighbor-level aggregator aims to aggregate information from its neighbor nodes connected with the same type of edges, and the edge-level aggregator further aggregates different types of edges. Second, we propose a novel Neighbor-SIM attention mechanism for edge-level aggregation, which is optimized for substitute product recommendation task. Extensive experiments on both proprietary and public Amazon datasets illustrate the best performance of HetSAGE with Neighbor-SIM attention mechanism compared with state-of-the-art GNN methods on the substitute product recommendation tasks.

1 Introduction

Recommendation systems are playing critical roles in E-commerce. They improve user experience by suggesting relevant items from a huge candidate pools to meet users' tastes. Despite the tremendous amount of research on improving recommender systems (Schafer et al. 2007; Adomavicius and Tuzhilin 2011; Bobadilla et al. 2013; Herlocker et al. 2017), few attempts have been made to investigate different relationships among products (Zheng et al. 2009; McAuley, Pandey, and Leskovec 2015; Zhang and Bockstedt 2016). In this paper, we focus on studying the substitute relationship among products in E-commerce. Specifically, substitute products are typically similar and compatible, and equally attractive to the same group of target customers (Shocker, Bayus, and Kim 2004; McAuley, Pandey, and Leskovec 2015; Zhang et al. 2019c). Such a substitute relationship, once discovered, would be greatly helpful in improving customer satisfaction. For instance, Amazon could recommend its customers substitute products to reduce their frustration when the query product is out of stock.

One important source to understand product sustainability is the item's content information, like product image, title, and description. In the field of content-based representation learning, deep learning methods have shown prominent performance (Van den Oord, Dieleman, and Schrauwen 2013; Covington, Adams, and Sargin 2016). For example, Alibaba built a visual search system to find visually similar products given customer's image queries (Zhang et al. 2018). Pinterest applied image search in their content-based product recommendations (Zhai et al. 2019). However, if only content information is available, expert knowledge is required in identifying the key attributes to define "substitute" for different product families (e.g., substitute apparels should have similar images, substitute furnitures should have similar style and dimension). However, this is neither scalable nor guaranteed to be the best definition for all customers.

Another important source to understand product sustainability is the customer behavior data, including click logs and purchase records, which reflect product relationships from the customers' perspective. In the field of behavior-based representation learning, there're also lots of success works. For example, Alibaba proposed a representation learning method using graph to model user-to-item interactions (Cen et al. 2019). Amazon leveraged collaborative filtering to model item-to-item interactions (Linden, Smith, and York 2003). However, one potential issue of only using behavior information is that it suffers from the cold-start problem for new or unseen products. In contrast, the content data can provide useful information for cold items. Therefore, the information from item content and customer behavior are complementary to each other.

Graph Neural Networks (GNNs) are capable of learning product representations from both content information and behavior interactions (Hamilton, Ying, and Leskovec 2017a; Ying et al. 2018; Wu et al. 2020). However, most of the GNNs are designed to handle the homogeneous network with single-typed edge, while real-world behavior interactions are often multi-typed which motivated by different purpose (e.g., people purchase A also purchase B means

*Work done as an intern at Amazon.

they are more likely to be complementary, people view A also view B means they're more likely to be substitutes). To incorporate data heterogeneity, several heterogeneous GNN methods with attention mechanism have been proposed recently (Wang et al. 2019; Zhang et al. 2019a; Hu et al. 2019; Hong et al. 2020). However, none of the existing heterogeneous GNNs have been built to learn product representation optimized for substitute product recommendations.

In this paper, we propose a highly-scalable, generic heterogeneous GNN model named HetSAGE for substitute product recommendation. In particular, our contributions include but not limit to:

- We propose HetSAGE, a two-level hierarchical GNN architecture (i.e., neighbor-level and edge-level aggregation) as a nature extension of the homogeneous GraphSAGE (Hamilton, Ying, and Leskovec 2017a) . The proposed work can be applied as a general framework to large-scale heterogeneous graphs.
- We propose a novel Neighbor-SIM attention mechanism in the edge-level aggregation in HetSAGE, which is optimized for substitute product recommendation tasks.
- We build a large-scale recommendation engine that retrieves and ranks substitute products in Amazon. Experiments on both proprietary and public datasets illustrate the superiority of HetSAGE with Neighbor-SIM attention mechanism compared with the state-of-the-art GNN methods on the substitute product retrieval tasks.

2 Related Work

Although the concept of product substitution has been proposed for over twenty years (Bucklin, Russell, and Srinivasan 1998; Shocker, Bayus, and Kim 2004; Zheng et al. 2009), there has been relatively few works that rigorously study substitute product relationships when designing recommendation systems. An exception to this is the work of McAuley, Pandey, and Leskovec (2015) who developed the first method that is capable of modeling and predicting substitute products from product reviews and descriptions via topic modeling. Recently, several non-GNN deep learning models are proposed to model the substitutable relationship between products (Wang et al. 2018; Rakesh et al. 2019; Zhang et al. 2019b,c; Chen et al. 2020). Motivated by the recent trend of exploiting GNNs designing recommendation systems, e.g., (Hamilton, Ying, and Leskovec 2017b), we propose a heterogeneous GNN model for product substitution that combines product content information (e.g., title, image) and behavior information with different edge types indicating different behavior relations.

GNNs (Bronstein et al. 2017; Hamilton, Ying, and Leskovec 2017b; Wu et al. 2020) which aim to extend the deep neural network to deal with arbitrary graph-structured data have become increasingly studied. Among them, GCN (Kipf and Welling 2016) leverages convolutional operation to propagate structural information of graphs layer by layer; GraphSAGE (Hamilton, Ying, and Leskovec 2017a) provides an inductive approach to combine structural information with node features. GAT (Veličković et al. 2017) employs self-attention mechanism to measure impacts of different neighbors and combine their impacts to obtain node embeddings. However, all these approaches are designed to handle homogeneous graph structure with one edge type.

Recently, several studies attempted to extend GNNs to heterogeneous graphs. HAN (Wang et al. 2019), and HetSANN (Hong et al. 2020) propose heterogeneous GNN models with self attention mechanism. However, as a drawback, the neighbor-level attention precludes sampling, thus lacks the ability to scale to large graph. R-GCN (Schlichtkrull et al. 2018) propose an edge type aware aggregator for dealing with highly multi-relational data in knowledge graph. HetGNN (Zhang et al. 2019a), the most related work to ours, proposes a heterogeneous GNN architecture with two modules to aggregate feature information of those sampled neighboring nodes of different types. In contrast, our model provides a framework that generalizes HetGNN on heterogeneous graph. On top of that, we propose a novel attention mechanism specifically optimized for substitute product recommendation.

3 Method

3.1 Problem Setup

For each product, its *substitute products* are similar and compatible products of itself (Shocker, Bayus, and Kim 2004; McAuley, Pandey, and Leskovec 2015; Zhang et al. 2019c). This substitute relation is *not unique*, namely one product could have several substitute products. In this paper, we assume that we have access to a set of labeled substitute pairs $\mathcal{S}$. For each substitute pair $(i, j) \in \mathcal{S}$, we call i the *query product* and j the *substitute product*. We aim to learn an embedding for each product such that substitute products are closer in the embedding space.

To incorporate the multi-typed item-to-item relations, we consider a *heterogeneous graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which consists of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$. Nodes in the graph represent items and edges are item-to-item relations (e.g., viewed together by a customer). In the graph there is one node type but K edge types, $\mathcal{E} = \{\mathcal{E}_1, \dots, \mathcal{E}_K\}, K > 1$. Denote node pair (i, j) the directed edge that links from the node $i \in \mathcal{V}$ to $j \in \mathcal{V}$ and (j, i) its reversed edge. A node i is a type k neighbor of node j if and only if $(i, j) \in \mathcal{E}_k$. The set of all the type k neighbors of node j is defined as $\{i \in \mathcal{V} : (i, j) \in \mathcal{E}_k\}$. In addition to the graph structure, each node i is associated with real-valued attributes $\mathbf{x}_i$, which contains content information (e.g., title, image) about a node.

Our goal is to leverage both the content informmation and the graph structure to generate embeddings of each node $\mathbf{z}_i \in \mathbb{R}^d$ that preserves the substitute relation, $0 < d \ll |\mathcal{V}|$. Subsequently, these embeddings are used for substitute candidate retrieval and ranking via a nearest neighbor lookup.

3.2 Model Architecture

Now we present HetSAGE, a general framework that leverages both nodes' attributes and graph structure to efficiently generate embeddings on multiplex heterogeneous graph. Figure 1 presents the overall framework of HetSAGE.

The key idea of most GNNs is to aggregate feature information from a node's neighbors in each layer, such as

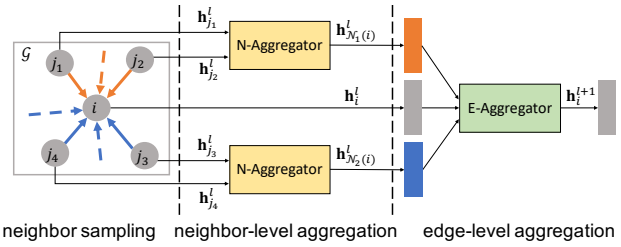

Figure 1: One layer of the proposed HetSAGE architecture (blue and orange arrows are edges and the solid lines of them represent sampled neighbors). It consists of two components: in neighbor-level aggregation, a fixed size of same type neighbors is sampled and aggregated; and in edge-level aggregation, different edge types are aggregated. The output feature of target node is then fed into the next layer.

GCN, GraphSAGE, and GAT. Such an aggregation is also called graph convolution in literature (Duvenaud et al. 2015; Kipf and Welling 2016). However, directly applying these approaches to heterogeneous graphs may raise two issues. First, they are not designed for heterogeneous graph, thus there is no general aggregation framework for multiple types of neighbors, as different edge types could have different meaning of connectivity; Second, aggregating all the neighbors could be computationally inefficient (Hamilton, Ying, and Leskovec 2017a). To that end, we propose the HetSAGE architecture, where a fixed size of neighbors is sampled for each edge type to improve computation efficiency and different types of neighbors are aggregated via a flexible aggregation mechanism that can be tailored to different use cases. Next, we introduce the details of HetSAGE, which consists of two hierarchical components: Neighbor-Level Aggregation and Edge-Level Aggregation.

Neighbor-Level Aggregation Our model firstly aggregates neighbor nodes connected to the query node with the same type of edges. In industry scenario, the graph usually have huge size with large node degree. For computational efficiency, we uniformly sample N_k neighbors of each edge type k, instead of using full neighborhood sets. Denote $\mathcal{N}_k(i)$ the set of node i's sampled type k neighbors, $\mathbf{h}_i^l$ the feature representation of node i at layer l, $\mathbf{h}_{\mathcal{N}_k(i)}^l$ the aggregated neighbor feature, and "N-Aggregator" the same type neighbor-level aggregator. The aggregated neighbor feature can be represented by $\mathbf{h}_{\mathcal{N}_k(i)}^l = \text{N-Aggregator}(\{\mathbf{h}_j^l, \forall j \in \mathcal{N}_k(i)\})$. Similar to GraphSAGE, N-Aggregator here could have several options such as "mean" or "max-pooling". Besides neighbor sampling strategy, parameters (e.g., weights) of HetSAGE with localized neighbor sampling are shared across all nodes, making the parameter complexity of our approach independent of the input graph size.

Edge-Level Aggregation The previous step generates K aggregated neighbor features for the target node i. To further aggregate these edge type depended neighbor features with node i's own feature, we introduce the edge-level aggregator, which can be represented as E-Aggregator: $\mathbf{h}_i^{l+1} = \text{E-Aggregator}(\{\mathbf{h}_i^l\} \cup \{\mathbf{h}_{\mathcal{N}_k(i)}^l, \forall k\})$. Similar to N-Aggregator, this edge-level aggregator is a general form and could have multiple options, e.g., "concatenation", "max-pooling" or "mean". A detailed achitecture of HetSAGE is shown in Algorithm 1 and a more specific example will be discussed in the next section.

Algorithm 1: HetSAGE

Input: the heterogenous graph $\mathcal{G}$, node attributes $\{\mathbf{x}_i\}$, a set of mini-batch nodes $\mathcal{M}$, number of layers L, sample neighbor size for each type N_k, N-Aggregator, E-Aggregator.
Output: Embeddings $\mathbf{z}_i, \forall i \in \mathcal{M}$

```
/* neighbor sampling                    */
```
$\mathcal{S}^L \leftarrow \mathcal{M}$;
for $l = L, \ldots, 1$ **do**
  for $k = 1, \ldots, K$ **do**
    Sample N_k type k neighbors for each node $i \in \mathcal{S}^l$, denoted by $\mathcal{N}_k(i)$;
  end
  $\mathcal{S}^{l-1} \leftarrow \mathcal{S}^l \cup \{\mathcal{N}_k(i), \forall k, \forall i \in \mathcal{S}^l\}$;
end
```
/* generate embeddings                  */
```
$\mathbf{h}_i^0 \leftarrow \mathbf{x}_i, \forall i \in \mathcal{S}^0$;
for $l = 1, \ldots, L$ **do**
  for $k = 1, \ldots, K$ **do**
    $\mathbf{h}_{\mathcal{N}_k(i)}^{l-1} = \text{N-Aggregator}(\{\mathbf{h}_j^{l-1}, \forall j \in \mathcal{N}_k(i)\}), \forall i \in \mathcal{S}^l$;
  end
  $\mathbf{h}_i^l = \text{E-Aggregator}(\{\mathbf{h}_i^{l-1}\} \cup \{\mathbf{h}_{\mathcal{N}_k(i)}^{l-1}, \forall k\}), \forall i \in \mathcal{S}^l$;
end
$\mathbf{z}_i = \mathbf{h}_i^L / \|\mathbf{h}_i^L\|_2, \forall i \in \mathcal{M}$;

3.3 Neighbor-Sim Attention E-Aggregator

Attention mechanism can be adopted by the proposed E-Aggregator to allow models to focus on informative relationships (edges). Recent state-of-the-art heterogeneous GNN model proposed in (Zhang et al. 2019a) falls into this category as it uses attention mechanism in edge-level aggregation, where the attention mechanism was defined as

$$\mathbf{h}_{\mathcal{N}_k(i)}^{l+1} = \mathbf{W}_k^l \cdot \mathbf{h}_{\mathcal{N}_k(i)}^l + \mathbf{b}_k^l, \tag{1}$$

$$\hat{\mathbf{h}}_i^{l+1} = \mathbf{W}^l \cdot \mathbf{h}_i^l + \mathbf{b}^l, \tag{2}$$

$$e_k = \sigma_1(\mathbf{u} \cdot \text{concat}(\hat{\mathbf{h}}_i^{l+1}, \mathbf{h}_{\mathcal{N}_k(i)}^{l+1})), \tag{3}$$

$$e_0 = \sigma_1(\mathbf{u} \cdot \text{concat}(\hat{\mathbf{h}}_i^{l+1}, \hat{\mathbf{h}}_i^{l+1})), \tag{4}$$

$$a_k = e_k / \textstyle\sum_{k=0}^K e_k, \ k = 0, \ldots, K \tag{5}$$

$$\mathbf{h}_i^{l+1} = \sigma_2(a_0 \hat{\mathbf{h}}_i^{l+1} + \textstyle\sum_{k=1}^K a_k \mathbf{h}_{\mathcal{N}_k(i)}^{l+1}), \tag{6}$$

In (1)-(2), the neighbors' and self node's features are fed through a fully connected layer, $\mathbf{W}_k^l$ and $\mathbf{b}_k^l$ are learnable network parameters. In (3)-(4), $\mathbf{u}$ is a learnable attention vector to measure the substitutability of self node and its neighbors. (5) further normalizes the attention weights and

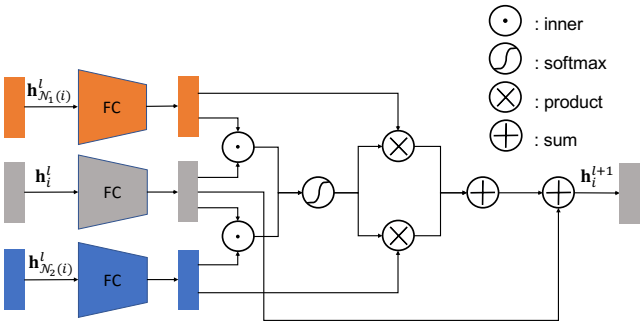

Figure 2: Illustration of Neighbor-SIM attention mechanism (blue and orange blocks are different type neighbors' feature and the gray one is self node's). This architecture serves as the E-Aggregator in the HetSAGE framework (Figure 1), where different type neighbors' and self node's feature are aggregated to generate new self node's feature for next layer.

applies to the $l+1$ layer node representation in (6). However, this attention mechanism is not optimal for product substitutability measurement.

To further optimize for substitute product recommendation task, we present a new novel Neighbor-Sim attention mechanism. The Neighbor-Sim attention E-Aggregator can be defined as:

$$e_k = \sigma_1(\hat{\mathbf{h}}_i^{l+1} \cdot \mathbf{h}_{\mathcal{N}_k(i)}^{l+1}), \tag{7}$$

$$a_k = e_k / \textstyle\sum_{k=1}^{K} e_k, \ k = 1, \ldots, K \tag{8}$$

$$\mathbf{h}_i^{l+1} = \sigma_2(\hat{\mathbf{h}}_i^{l+1} + \textstyle\sum_{k=1}^{K} a_k \mathbf{h}_{\mathcal{N}_k(i)}^{l+1}). \tag{9}$$

We leverage (7)-(9) to aggregate the features of multiple types of neighbors instead of (3)-(6). See Figure 2 for an illustration.

There are two major advantages of the new attention mechanism. First, we directly use the inner product between self node and neighbor $\hat{\mathbf{h}}_i^{l+1} \cdot \mathbf{h}_{\mathcal{N}_k(i)}^{l+1}$ in (7) as the score function to replace the concat$(\hat{\mathbf{h}}_i^{l+1}, \mathbf{h}_{\mathcal{N}_k(i)}^{l+1})$ in (3)-(4) to better measure the substitutability among items. Our design is based on the fact that the substitute products should have similar content features with query products, thus the higher the inner product is, the more likely they are substitutable and the stronger connection is established in our attention mechanism. Second, we construct the attention mechanism only across different types of neighbors in (8) (i.e., $k = 1, \ldots, K$), not including the self node as in (5) (i.e., $k = 0, \ldots, K$). This design is equivalent to setting the weight a_0 of the self node $\hat{\mathbf{h}}_i^{l+1}$ to 1, which preserves and propagates more content information of the self node into the next layer representation $\mathbf{h}_i^{l+1}$.

3.4 Model Training

Finally, we leverage triplet loss for the output embeddings optimization, which aims to maximize the cosine similarity of the of positive substitute examples, i.e., the embedding of the query product $\mathbf{z}_q$ and the substitute product $\mathbf{z}_s$, for $(q, s) \in \mathcal{S}$. Meanwhile, we want to ensure that the inner product of negative examples, $\mathbf{z}_q \cdot \mathbf{z}_n$, is smaller than that of the positive samples with a margin. The loss function for a single pair of node embeddings $(\mathbf{z}_q, \mathbf{z}_s)$ is

$$L(\mathbf{z}_q, \mathbf{z}_s) = \mathbb{E}_{n \sim U(\mathcal{V})} \max\{0, \mathbf{z}_q \cdot \mathbf{z}_n - \mathbf{z}_q \cdot \mathbf{z}_s + \Delta\} \tag{10}$$

where the negative product n is uniformly sampled from all nodes and Δ denotes the margin hyper-parameter.

Dataset	proprietary data	public data
# node	∼1.5M	11,417
# co-purchase	∼24M	9,138
# co-view	∼54M	29,338
# view-to-purchase	∼12M	N/A
# training pairs	∼3.4M	7,854
# testing pairs	∼850K	1,963

Table 1: Statistics of Datasets.

4 Experiments

4.1 Datasets

Amazon Proprietary Data We select three types of product relationships extracted from historical data: *co-purchase*, *co-view*, and *view-to-purchase*, where $\mathcal{E}_{cp}, \mathcal{E}_{cv}, \mathcal{E}_{v2p}$ are denoted as the corresponding edge sets. The product pair edge (i, j) in those set represents one of the following relations:

1. *co-purchase*: customer who purchases i also purchases j,
2. *co-view*: customer who views i also views j,
3. *view-to-purchase*: customer who views i eventually purchases j.

According to Amazon's report (Linden, Smith, and York 2003), the above relationships are collected simply by ranking products according to the cosine similarity of sets of users who purchased/viewed them. In our experiment, we use overlapped pairs from *co-view* and *view-to-purchase* with *co-purchase* removed to be the labeled substitute pairs, $\mathcal{S} = (\mathcal{E}_{cv} \cap \mathcal{E}_{v2p}) \backslash \mathcal{E}_{cp}$. The edges in the graph are considered as directed. Note that *co-purchase* and *co-view* should be bidirectional edges (i.e., $(i, j) \in \mathcal{E}_{cp}, \forall (j, i) \in \mathcal{E}_{cp}$). *view-to-purchase* is not considered as a reversible relation so in the graph we construct two types of edges including *view-to-purchase* and *purchase-from-view*.

Public Dataset We test our proposed methods on public Amazon Review dataset (Ni, Li, and McAuley 2019). This dataset has no *view-to-purchase* edge but contains *co-purchase*, *co-view*, and *similar-item*. We use that *similar-item* as the labeled substitute pairs in the training.

The dimensionalities of those two datasets are summarized in Table 1. For both datasets we randomly select 20% product pairs from $\mathcal{S}$ for testing.

4.2 Experiment Setup

Feature Preparation For both proprietary and public dataset, we extract image and title text feature as the content input of each product. The visual input is extracted from pre-trained Xception (2048 dimension) (Chollet 2017) and the title text input is extracted by pre-trained BERT (768 dimension) (Devlin et al. 2018). They are reduced to 100 dimensions by PCA and concatenated into the input attributes.

Dataset	Amazon Proprietary Dataset				Public Dataset			
Metric	NDCG@5	NDCG@30	MAP@5	MAP@30	NDCG@5	NDCG@30	MAP@5	MAP@30
None Graph NN	0.0778	0.1364	0.0567	0.0747	0.1523	0.2524	0.1221	0.1545
GraphSAGE (co-purchase)	0.0683	0.1308	0.0482	0.0675	0.1442	0.2496	0.1181	0.1513
GraphSAGE (co-view)	0.0853	0.1694	0.0611	0.087	0.1769	0.292	0.1435	0.1805
GraphSAGE (v2p)	0.1144	0.2072	0.0841	0.1139	-	-	-	-
GraphSAGE (all edge types)	0.0744	0.1442	0.0535	0.0746	0.1774	0.2967	0.1443	0.1825
HetGNN	0.1091	0.2113	0.0778	0.1121	0.1788	0.2972	0.1465	0.1845
HetSAGE (concat)	0.1138	0.2184	0.0816	0.1169	0.1958	0.3146	0.1573	0.1958
HetSAGE (max-pooling)	0.1136	0.2092	0.082	0.1137	0.184	0.3055	0.1506	0.1897
HetSAGE (mean)	0.1246	0.2301	**0.0901**	0.1257	**0.2028**	0.3145	0.1624	0.2025
HetSAGE (Neighbor-SIM attention)	**0.1253**	**0.2311**	**0.0904**	**0.1263**	0.2016	**0.3236**	**0.1665**	**0.2067**

Table 2: Substitute retrieval ranking performances on Amazon proprietary dataset and public dateset. See GraphSAGE in (Hamilton, Ying, and Leskovec 2017a), HetGNN in (Zhang et al. 2019a).

Evaluation Metrics To evaluate the trained embedding representations, we select the k' nearest neighbors to each testing query product in the embedding space. The substitute candidates are selected from all the products appear in training and testing pairs. We evaluate the returned top k' candidates by ranking metrics such as Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP) using the labeled pairs in the testing set.

Implementation Details Our models are implemented based on DGL package[1]. We set the output embedding dimension d to be 128 and 32, and hidden size to be 1024 and 256 for proprietary and public dataset, respectively. For both datasets, we use a 2-layer model. We set the training batch size to be 512 and default neighbor sampling size to be 10. While there could be several design choices for N-aggregator in our HetSAGE framework, we focus more on edge-level aggregation and we set the N-aggregator to be "mean" in all experiments. The model are trained and evaluated on one p3.16xlarge AWS SageMaker[2] instance with 8 GPUs.

4.3 Baselines

We evaluate the performance of HetSAGE with Neighbor-SIM attention mechanism against the following state-of-the-art baselines that generate embeddings of products:

None Graph Neural Networks First we compare with the method that use no graph structure (behavior relations). This model is equivalent to a feed-forward neural network trained by the same triplet loss defined in (10).

Homogeneous GNNs Then we compare with GNN methods designed for homogeneous graphs. To have a fair comparison, we compare to the model that also leverages neighbor sampling strategy (GraphSAGE)

Heterogeneous GNNs Finally, we compare with GNN methods designed for heterogeneous graphs. We compare with HetGNN (details illustrated in Section 3.3), as well as HetSAGE with other basic E-Aggregators including "concat", "max-pooling", and "mean".

[1]https://www.dgl.ai/

[2]https://aws.amazon.com/sagemaker/pricing/instance-types/

4.4 Experiment Results

The substitute retrieval ranking performances on both proprietary and public datasets are presented in Table 2. In the table, none graph neural networks, homogeneous GNNs, and Heterogeneous GNNs models are separated in three blocks, where no edge, one type of edge, and all type of edges are used to construct the graph, respectively. Specifically, for GraphSAGE (all edge types), we use the edges of all types in the graph but treat them as one identical type. For our proposed HetSAGE model, we test three basic edge-level aggregation methods: "concat", "max-pooling", and "mean". Finally, we present the results for HetSAGE with Neighbor-SIM attention mechanism.

Performance Analysis First, we compare GraphSAGE, the homogeneous GNN model, with the none GNN model. Results show that in terms of both metrics, using only *co-purchase* edge in the graph leads to a worse performance. This is intuitive as *co-purchase* often represents complement relation instead of substitute. Thus adding only *co-purchase* edge in a homogeneous graph structure could introduce misleading information. On the other hand, using more informative edges with respect to substitute such as *view-to-purchase* (proprietary dataset) and *co-view* (both proprietary and public dataset) increases in the ranking performance.

Second, including all the edge types as one identical type in the graph is not applicable (GraphSAGE (all edge types) in Table 2). In the proprietary dataset, the performance is worse than using only *view-to-purchase* edge or *co-view* edge. This is due to the incapability of homogeneous GNN to learn multiple connection properties on different edge types. Treating *co-purchase* as the same type with *view-to-purchase* results in worse performance. This justifies the demand for heterogeneous graph structure in GNNs.

Third, we observe that HetGNN with multiple edge types could have worse performance than homogeneous GNN model. For example, in the proprietary dataset, HetGNN is worse than using *view-to-purchase* edge only. There are two potential reasons that motivate our design of HetSAGE and Neighbor-SIM attention: The attention scores in HetGNN are computed over neighbor nodes and self node, which might lead to a small coefficient on self node's feature. In our setting the content information of each node is rich and a small coefficient on self node will result in losing such information; The attention is not designed directly based on

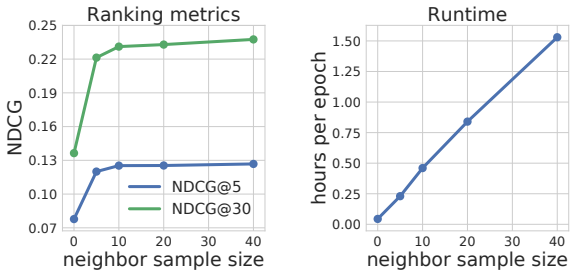

Figure 3: Impact of neighbor sampling size to the performance and runtime in the HetSAGE model.

the similarity over the features of nodes.

Finally, HetSAGE with "concat", "max-pooling", and "mean" overall achieve better performance compared with HetGNN and other homogeneous approaches. These E-Aggregators of HetSAGE compute the features of neighbor nodes and self node separately, thus the disadvantage of losing content information is resolved. Moreover, HetSAGE with Neighbor-SIM attention has the best overall performance, as it computes attention scores directly based on the similarity between features of nodes and establishes stronger connections to types of neighbors with similar features.

Neighbor Size Sensitivity Another important property of HetSAGE is the neighbor sampling strategy leveraged in neighbor-level aggregation. Figure 3 shows the performance and runtime sensitivity with respect to the neighbor sampling size in the HetSAGE model. We observe a performance diminishing when then neighbor sampling size is larger than 10. However, the runtime increases linearly with neighbor sampling size. Similar observations are also presented in (Hamilton, Ying, and Leskovec 2017a) and (Zhang et al. 2019a). This shows the inefficiency of using attention based neighbor-level aggregation methods such as GAT, HAN, and HetSANN in large and dense graph, which aggregate all neighbors without sampling. Our HetSAGE framework, on the other hand, could maintain good performance, while significantly reducing the runtime.

5 Conclusion

In this paper, we present HetSAGE, a general heterogeneous GNN framework that leverages both product content information and multiple types of customer behavior interactions to generate embedding representations for products, and a novel Neighbor-SIM attention mechanism in HetSAGE. We further show HetSAGE achieves the start-of-the-art performance on substitute recommendation task on both proprietary and public datasets. In the future, we propose to extensively compare with more substitute recommendation algorithms and modify HetSAGE to optimize for other recommendation tasks, such as complemantary recommendation.

References

Adomavicius, G.; and Tuzhilin, A. 2011. Context-aware recommender systems. In *Recommender systems handbook*, 217–253. Springer.

Bobadilla, J.; Ortega, F.; Hernando, A.; and Gutiérrez, A. 2013. Recommender systems survey. *Knowledge-based systems* 46: 109–132.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34(4): 18–42.

Bucklin, R. E.; Russell, G. J.; and Srinivasan, V. 1998. A relationship between market share elasticities and brand switching probabilities. *Journal of Marketing Research* 35(1): 99–113.

Cen, Y.; Zou, X.; Zhang, J.; Yang, H.; Zhou, J.; and Tang, J. 2019. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Chen, T.; Yin, H.; Ye, G.; Huang, Z.; Wang, Y.; and Wang, M. 2020. Try This Instead: Personalized and Interpretable Substitute Recommendation. *SIGIR* .

Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017a. Inductive representation learning on large graphs. In *Advances in neural information processing systems*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017b. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* .

Herlocker, J. L.; Konstan, J. A.; Borchers, A.; and Riedl, J. 2017. An algorithmic framework for performing collaborative filtering. In *ACM SIGIR Forum*, volume 51, 227–234. ACM New York, NY, USA.

Hong, H.; Guo, H.; Lin, Y.; Yang, X.; Li, Z.; and Ye, J. 2020. An Attention-Based Graph Neural Network for Heterogeneous Structural Learning. In *AAAI*.

Hu, B.; Zhang, Z.; Shi, C.; Zhou, J.; Li, X.; and Qi, Y. 2019. Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* .

Linden, G.; Smith, B.; and York, J. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7(1): 76–80.

McAuley, J.; Pandey, R.; and Leskovec, J. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*.

Ni, J.; Li, J.; and McAuley, J. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Rakesh, V.; Wang, S.; Shu, K.; and Liu, H. 2019. Linked variational autoencoders for inferring substitutable and supplementary items. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*.

Schafer, J. B.; Frankowski, D.; Herlocker, J.; and Sen, S. 2007. Collaborative filtering recommender systems. In *The adaptive web*, 291–324. Springer.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 593–607. Springer.

Shocker, A. D.; Bayus, B. L.; and Kim, N. 2004. Product complements and substitutes in the real world: The relevance of "other products". *Journal of Marketing* 68(1): 28–40.

Van den Oord, A.; Dieleman, S.; and Schrauwen, B. 2013. Deep content-based music recommendation. In *Advances in neural information processing systems*.

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

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*.

Wang, Z.; Jiang, Z.; Ren, Z.; Tang, J.; and Yin, D. 2018. 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*, 619–627.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* .

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zhai, A.; Wu, H.-Y.; Tzeng, E.; Park, D. H.; and Rosenberg, C. 2019. Learning a Unified Embedding for Visual Search at Pinterest. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019a. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zhang, M.; and Bockstedt, J. 2016. Complements and Substitutes in Product Recommendations: The Differential Effects on Consumers' Willingness-to-pay. In *IntRS@ RecSys*, 36–43.

Zhang, M.; Wei, X.; Guo, X.; Chen, G.; and Wei, Q. 2019b. Identifying Complements and Substitutes of Products: A Neural Network Framework Based on Product Embedding. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13(3): 1–29.

Zhang, S.; Yin, H.; Wang, Q.; Chen, T.; Chen, H.; and Nguyen, Q. V. H. 2019c. Inferring Substitutable Products with Deep Network Embedding. In *IJCAI*.

Zhang, Y.; Pan, P.; Zheng, Y.; Zhao, K.; Zhang, Y.; Ren, X.; and Jin, R. 2018. Visual search at alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM.

Zheng, J.; Wu, X.; Niu, J.; and Bolivar, A. 2009. Substitutes or complements: another step forward in recommendations. In *Proceedings of the 10th ACM conference on Electronic commerce*. ACM.