Recommending Related Products Using Graph Neural Networks in Directed Graphs

Srinivas Virinchi[1]✉, Anoop Saladi[1], and Abhirup Mondal[1]

International Machine Learning, Amazon, Bengaluru, India
{virins,saladias,mabhirup}@amazon.com

Abstract. Related product recommendation (RPR) is pivotal to the success of any e-commerce service. In this paper, we deal with the problem of recommending related products i.e., given a query product, we would like to suggest top-k products that have high likelihood to be bought together with it. Our problem implicitly assumes asymmetry i.e., for a phone, we would like to recommend a suitable phone case, but for a phone case, it may not be apt to recommend a phone because customers typically would purchase a phone case only while owning a phone. We also do not limit ourselves to complementary or substitute product recommendation. For example, for a specific night wear t-shirt, we can suggest similar t-shirts as well as track pants. So, the notion of relatedness is subjective to the query product and dependent on customer preferences. Further, various factors such as product price, availability lead to presence of selection bias in the historical purchase data, that needs to be controlled for while training related product recommendations model. These challenges are orthogonal to each other deeming our problem non-trivial. To address these, we propose DAEMON, a novel Graph Neural Network (GNN) based framework for related product recommendation, wherein the problem is formulated as a node recommendation task on a directed product graph. In order to capture product asymmetry, we employ an asymmetric loss function and learn dual embeddings for each product, by appropriately aggregating features from its neighborhood. DAEMON leverages multi-modal data sources such as catalog metadata, browse behavioral logs to mitigate selection bias and generate recommendations for cold-start products. Extensive offline experiments show that DAEMON outperforms state-of-the-art baselines by 30-160% in terms of HitRate and MRR for the node recommendation task. In the case of link prediction task, DAEMON presents 4-16% AUC gains over state-of-the-art baselines. DAEMON delivers significant improvement in revenue and sales as measured through an A/B experiment.

Keywords: Related product recommendation · Graph Neural Networks · Directed Graphs · Selection bias.

1 Introduction

Related product[1] recommendation (RPR) plays a vital role in helping customers easily find right products on e-commerce websites and hence, critical to their success. It not only helps customers discover new related products, but also simplifies their shopping effort, thereby delivering a great shopping experience. In this paper, we are interested in related product recommendation problem: *given a query product, the goal is to recommend top-k products that have a high likelihood to be bought together with it.* For notation purpose, let a, b and c be any three products. Let R_{cp}, R_{cv} and R_{like} be binary relationships where, $aR_{cp}b$ represents that a is co-purchased with b, $aR_{cv}b$ represents that a is co-viewed with b, and $aR_{like}b$ represents that a is similar to b based on its product features. Consider a sample product graph in Figure 1, where a node corresponds

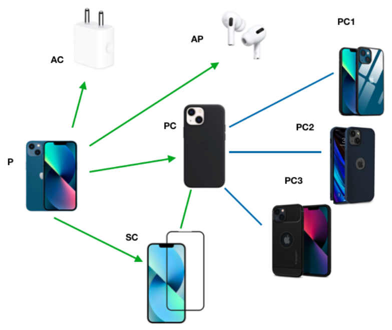

Fig. 1. Sample product graph. Green and blue edges represent co-purchase and co-view edges respectively. P refers to an iPhone 13, AC is an adapater, AP refers to AirPods and SC refers to a screen guard. Phone case PC is co-viewed with other phone cases $PC1$, $PC2$ and $PC3$; $PC1$, $PC2$ and $PC3$ are similar to PC. Undirected edges are bidirectional.

to a product and an edge corresponds to a co-purchase or co-view relationship between the products (refer to Section 4.1 for graph construction). RPR problem entails few challenges which we illustrate using Figure 1 as follows: 1) *product relevance:* Given a query product P, we would like to suggest related products such as an adapter AC, phone case PC, AirPods AP, screen guard SC. 2) *product asymmetry:* For a phone P, we would like to recommend a suitable phone case PC, but for a phone case PC, it may not be apt to recommend a phone P because customers typically would purchase a phone case only while owning a phone. Formally, product asymmetry can be represented as $aR_{cp}b \nRightarrow bR_{cp}a$. 3) *selection bias* is inherent to historical purchase data due to several factors like product availability, price etc. For example, while shopping for a

[1] We use product, item and node interchangeably in this paper.

phone case for phone P, customer c_1 might browse phone cases PC and $PC1$ ($PC2$ and $PC3$ are not shown to him owing to some of the factors mentioned above). Another customer c_2 might browse phone cases PC and $PC2$ ($PC1$ and $PC3$ are not presented to him). A third customer c_3 might browse phone cases PC and $PC3$ ($PC1$ and $PC2$ are not presented to him). Assume that all three customers eventually purchase PC; this could also be due to a bias in the list of products shown to each customer. In our example, across different customers, PC, $PC1$, $PC2$ and $PC3$ are co-viewed (similar). In order to correct for selection bias, we would like to recommend not only phone case PC, but also $PC1$, $PC2$ and $PC3$ as related products given a query product P. Formally, we want to uncover relationships of the form: a) $aR_{cp}b \wedge bR_{cv}c \implies aR_{cp}c$, b) $aR_{cv}b \wedge bR_{cp}c \implies aR_{cp}c$ in order to mitigate selection bias. 4) *cold-start product recommendations:* We not only need to suggest recommendations for existing products, but also suggest recommendations for newly launched i.e. cold-start products. Formally, given two existing products a and b, and a cold-start product c, we want to uncover relationships of the form: a) $cR_{like}a \wedge aR_{cp}b \implies cR_{cp}b$, b) $aR_{cp}b \wedge bR_{like}c \implies aR_{cp}c$. These two rules correspond to the case when c is the query product and the recommended product respectively. Observe that mitigating selection bias and tackling cold-start products deal with modelling transitive relationships, while preserving edge asymmetry. 5) Dealing with millions of products in our catalog demands a *scalable* solution for the recommendation task. These challenges are uncorrelated with each other making our problem non-trivial.

Given multi-modal data sources such as catalog metadata, product co-purchase data, anonymized browse behavioral logs, etc. product graphs serve as an excellent abstraction to seamlessly capture relationships between products. RPR boils down to the node recommendation problem [21,9] in directed product graphs. Recent work on node representation learning in directed graphs such as HOPE [9], APP [21], NERD [5], DGGAN [22] and Gravity Graph VAE [11] learn two embeddings for each node and utilize them for downstream tasks. Further, GNN models like DGCN [16] and DiGraphIB [15] generate real valued embeddings for each node, while MagNet [20] generates complex valued embeddings to preserve edge strength and direction. Selection bias in recommendation systems has been well studied [1] for different applications. Prior work has limitations across different dimensions as follows: a) there is no straightforward approach to extend popular GNN models like GCN [7], GraphSage [3], GAT [18], RGCN [12] to model directed graphs as they do not model product asymmetry. b) GNN models for directed graphs like DGCN [16], DGGAN [22], DiGraphIB [15] and MagNet [20] do not address the problem of recommendation for cold-start products. c) prior work does not address selection bias inherent to historical purchase data, and make specific assumption regarding the availability of unbiased data [19].

In this paper, we formulate RPR as a node recommendation task on a directed product graph. We propose DAEMON, **D**irection **A**war**E** Graph Neural Network **MO**del for **N**ode recommendation. In order to capture product asymmetry, our model generates dual embeddings for each product, by appropriately aggregating features from its neighborhood. We leverage customer co-view pairs from anonymized browse behavioral logs to recommend relevant products that customers clicked but didn't purchase to tackle selection bias. Specifically, during model training, we employ a novel asymmetric loss function which explicitly considers product co-purchases to capture product asymmetry, and product co-views to estimate co-purchase likelihood for products that were previously clicked but not purchased. We exploit product catalog metadata to generate product embeddings for cold-start products, and consequently generate recommendations for them. Further, as a byproduct, the proposed model is also able to suggest substitute product recommendations.

We perform exhaustive experiments offline on real-world datasets to evaluate the performance of our model. Results show that DAEMON outperforms state-of-the-art baselines by 30-160% in terms of HitRate and MRR for the node recommendation task which is the primary task of interest. For link prediction tasks, DAEMON outperforms state-of-the-art baselines by 4-16% and 3-6.5% in terms of AUC for the existence [20] and direction [20] link prediction task respectively. DAEMON delivers significant improvement in revenue and sales as measured through an A/B experiment. To summarize, we make the following contributions:

1. We formulate RPR as a node recommendation task in a directed product graph and propose a Graph Neural Network (GNN) based framework for related product recommendation. To this end, we present DAEMON, a novel GNN model, that leverages dual embeddings to capture node asymmetry in directed graphs.
2. In order to train the model, we employ a novel asymmetric loss function that explicitly deals with modelling co-purchase likelihood, product asymmetry and selection bias. We exploit product catalog metadata to deal with the issue of cold-start products. This is the first work that jointly addresses product asymmetry, generating cold start recommendations and mitigating selection bias in historical purchase data.
3. Offline evaluation shows that DAEMON outperforms state-of-the-art models for node recommendation and link prediction tasks. DAEMON derives significant improvement in product sales and revenue as measured through an A/B experiment.

2 Related Work

Our problem conceptually relates to the node recommendation problem [21,9] in directed graphs. Prior work uses random walk based techniques to model node relationships in directed graphs. VERSE [17], HOPE [9] propose learning two embeddings for each node to preserve higher order proximity and consequently, node asymmetry in directed graphs. APP [21] captures asymmetry by preserving Rooted PageRank between nodes by relying on random walk with restart strategy. ATP [13] addresses the problem of question answering by embeddings nodes of directed graph by preserving asymmetry. However, their approach is restricted to only directed acyclic graphs (DAGs), while an e-commerce product graph can be cyclic. NERD [5] learns a pair of role specific embeddings for each node using a alternating random walk strategy to model edge strength and direction in directed graphs.

Recent work has seen GNN's being designed for directed graphs. DGCN [16] extends the spectral-based GCN model to directed graphs by using first and second-order proximity to expand the receptive field of the convolution operation. APPNP [8] uses a GCN model to approximate personalized PageRank. DGCN [16] uses one first-order matrix and two second order proximity matrices to model asymmetry. DiGraphIB [15] builds upon the ideas of [16] and constructs a directed Laplacian of a PageRank matrix. It uses an inception module to share information between receptive fields. Gravity GAE [11] is inspired from Graph Auto Encoders [6] by applying the idea of gravity to address link prediction in directed graphs. DGGAN [22] is based on Generative Adversarial Network by using a discriminator and two generators to jointly learn each node's source and target embedding. MagNet [20] proposes a GNN for directed graphs based on a complex Hermitian matrix. The magnitude of entries in the complex matrix encodes the graph structure while the directional aspect is captured in the phase parameter. Prior work in this space does not deal with the the issue of cold-start products. Selection bias in recommendation systems has been well studied [1] for different applications. However, prior work does not address the bias inherent to historical purchase data. Further, we make no assumption regarding the availability of unbiased data in order to combat selection bias unlike [19]. Ours is the first work that jointly: a) learns node representations in directed graphs to capture edge strength and direction, b) generates recommendations for cold-start products by leveraging catalog metadata, and c) mitigates selection bias in product co-purchase directed graphs.

3 Related Product Recommendation Problem

We present relevant notation in Table 1. Let G be a directed product graph with products P as the nodes and directed edges corresponding to a co-purchase or co-view relationship between the products (refer section 4.1). Further, every product i ($i \in P, \forall i$) has an input feature X_i from the product catalog metadata. Given a query node[2] q, the goal is to recommend R_k^q, top-k related products that have a high likelihood to be bought together with q.

This corresponds to the node recommendation problem [21,9] in directed graphs. Note that while generating related product recommendations, we must jointly capture product co-purchase likelihood and preserve product asymmetry.

4 Proposed Framework

We show the proposed framework in Figure 2. We leverage the co-purchase data CP and co-view data CV to create the product graph G. We represent

[2] We use product and node interchangeably in this paper based on the context.

Table 1. Notation

Notation	Description
P	set of products in catalog
$i \in P$	product i in catalog
X_i	input feature of product i
θ_i^s	source embedding of product i
θ_i^t	target embedding of product i
G	directed product graph
$q \in P$	query product
R_k^q	top-k related products for query product q
CP	product co-purchase pairs
CV	product co-view pairs
E_{cp}	product co-purchase edges
E_{cv}	product co-view edges

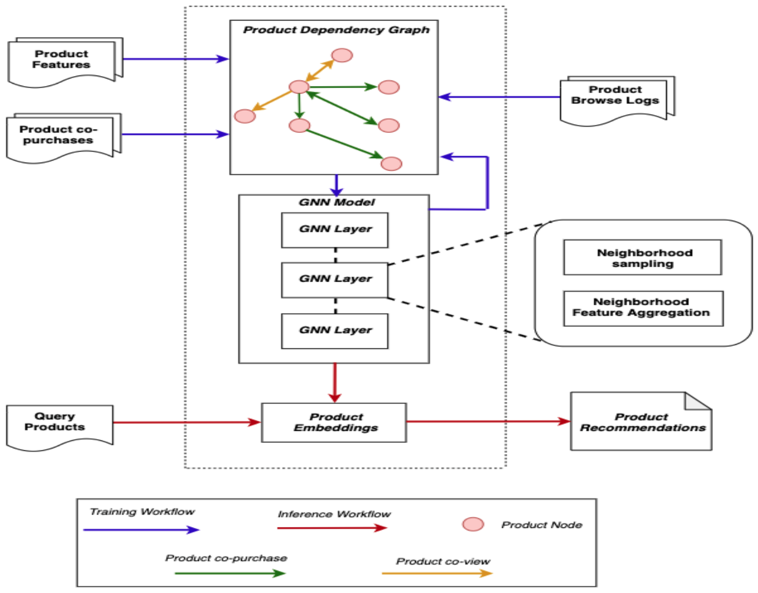

Fig. 2. Proposed Framework for Related Product Recommendation. The computational graphs corresponding to the source (s) and target (t) embedding of node A in the product graph is shown.

each product by an input feature based on its metadata from catalog (product name, product description, product type etc.). In order to jointly model product co-purchase likelihood and product asymmetry, we represent each product using dual i.e. source and target embeddings. We train DAEMON using an asymmetric loss function. The trained model generates product embeddings by appropriately aggregating features from its neighborhood in G. We show the computational graphs corresponding to the source and target embedding of node A in Figure 2. For every node u, we generate a pair of embeddings (denoted by u-s and u-t in Figure 2). In order to address node asymmetry, we use the following rationale: a) while generating the source embedding of a node, we aggregate information from the target embedding of its out-neighbors, b) while generating the target embedding of a node, we aggregate information from the source embedding of its in-neighbors. Compared to undirected graphs, the computational graph corresponding to the source and target embedding of each node is different in directed graphs. Consequently, this trick enables us to model edge strength and edge direction efficiently in directed graphs. Given a query product q, we generate its source embedding and perform a nearest neighbor lookup in the target embedding space to recommend top-k related products for q. We explain each component in detail next.

4.1 Product graph construction

Given a set of product co-purchase pairs CP, we create the co-purchase edges $E_{cp} = \{(u, v) | \forall\, u, v \in P \wedge uR_{cp}v\}$. We use E_{cp} to model the product co-purchase likelihood. However, E_{cp} is prone to selection bias.

In order to combat this issue, we exploit anonymized browse behavioral logs, to create product co-view pairs CV, which we use to create the set of co-view edges $E_{cv} = \{(u, v) | \forall\, u, v \in P \wedge uR_{cv}v\}$. This lets us construct a directed product graph $G = (P, \{E_{cp} \cup E_{cv}\})$, which contains both co-purchase and co-view relationships between products.

For any three products $a, b, c \in P$, G contains product relationships of the form: 1) $aR_{cp}b \not\Rightarrow bR_{cp}a$ 2) $aR_{cp}b \wedge bR_{cv}c \implies aR_{cp}c$ 3) $aR_{cv}b \wedge bR_{cp}c \implies aR_{cp}c$. The goal is to design a GNN model that learns these kind of relationships from G, while preserving co-purchase likelihood; rule 1 corresponds to product asymmetry, and rules 2-3 are specific to the case of selection bias. *Observe that E_{cp} and E_{cv} are not correlated with each other, and we need to provide differential treatment to E_{cp} and E_{cv} during neighborhood feature aggregation.*

4.2 DAEMON: Proposed GNN model

We first describe the product embedding generation procedure i.e. forward pass of the model assuming that the model is already trained (Section 4.2.1). We then describe how the model parameters can be learned using stochastic gradient descent and backpropagation techniques (Section 4.2.2).

4.2.1 Product Embedding Generation: Forward Pass Algorithm 1 describes the forward pass to generate the source and target embedding of each product. It expects the product graph G and product features X as input. We set the initial source and target embedding of each product to its input feature. Let $(h_u^s)^l$ and $(h_u^t)^l$ denote a product's source and target representation in the l^{th} step of Algorithm 1. First, for each product u, extract its co-purchase

Algorithm 1 DAEMON product embedding generation (i.e. forward pass)

Input: product graph $G = (P, \{E_{cp} \cup E_{cv}\})$; input product features $\{X_u, \forall \in P\}$; Number of GNN layers L; weight matrices $W^l, \forall l \in \{1, 2, .., L\}$
Output: source embedding θ^s_u and target embedding θ^t_u, $\forall u \in P$

```
 1: $(h^s_u)^0 \leftarrow X_u$ ; $(h^t_u)^0 \leftarrow X_u \forall u \in P$
 2: for $l$=1,...$L$ do
 3:     for $u \in P$ do
 4:         $(h^s_u)^l \leftarrow \sigma\left(\sum_{(u,v)\in E_{cp}} (h^t_v)^{l-1} W^l\right) + \sigma\left(\sum_{(u,v)\in E_{cv}} (h^s_v)^{l-1} W^l\right)$
 5:         $(h^t_u)^l \leftarrow \sigma\left(\sum_{(v,u)\in E_{cp}} (h^s_v)^{l-1} W^l\right) + \sigma\left(\sum_{(v,u)\in E_{cv}} (h^t_v)^{l-1} W^l\right)$
 6:     end for
 7:     $(h^s_u)^l \leftarrow (h^s_u)^l / ||(h^s_u)^l||_2, \forall u \in P$
 8:     $(h^t_u)^l \leftarrow (h^t_u)^l / ||(h^t_u)^l||_2, \forall u \in P$
 9: end for
10: $\theta^s_u \leftarrow (h^s_u)^L, \forall u \in P$
11: $\theta^t_u \leftarrow (h^t_u)^L, \forall u \in P$
```

and co-view neighbors. The source hidden node representation of u in the l^{th} step is aggregated as a linear combination of two non-linear terms i.e. non-linear aggregate of target representation of its co-purchased out-neighbors and non-linear aggregate of source representation of its co-viewed out-neighbors from the $(l-1)^{th}$ step. W^l corresponds to fully connected layer at step l with non-linear ReLU activation function σ (line 4). We perform a similar neighborhood aggregation to estimate a node's target representation during the l^{th} step (line 5). The source and target node representations of step l is used in the next step. We normalize the embeddings to a unit norm (lines 7, 8). We repeat this process for L steps to generate the final source and target product representation of products $\{\theta^s_u, \theta^t_u\}$ $\forall u \in P$ (line 10, 11). We leverage the generated product embeddings to recommend related products. We also show relevant properties of the embeddings in Lemma 1 and 2.
Related product recommendation: Given a query product q, we use θ^s_q, the source embedding of q, to perform a nearest neighbor lookup in the target embedding space of all the products to recommend top-k set of related products denoted by R^q_k. Specifically, for a query product $q \in P$, we compute a relevance score with respect to a candidate product $v \in P$ as shown in Equation 1.

$$rel(q, v) = (\theta^s_q)^\intercal (\theta^t_v) \tag{1}$$

Observe that $rel(q, v) \neq rel(v, q)$ which helps capturing product asymmetry.
Cold-start related product recommendation: Our model can also be used for *cold-start* product recommendation. Given a cold-start product c, we perform a neighborhood lookup to find similar existing (warm-start) products i.e. $\{c_1, c_2, ..c_k\}$ based on its input product features (X). We augment the edges of the form $\{(c, c_1), (c, c_2), ..(c, c_k)\}$ to the product graph G. Further, c has an input feature X_c from catalog metadata. We pass the subgraph corresponding to

the cold-start product c to the GNN model and generate the source and target embedding of c using Algorithm 1. We use θ_c^s to probe the target embeddings to recommend related products for c using Equation 1.

4.2.2 Learning DAEMON parameters: Backward Pass In order to train the model in an unsupervised manner, we employ a novel asymmetric loss function shown in Equation 2. This helps us capture different aspects as follows:

1. When $(u,v) \in E_{cp}$, the model forces the source embedding of u to be similar to the target embedding of v, and distant from the target embedding of a disparate product z (terms 1 and 2). Negative samples are generated using a uniform distribution P_r. This models product co-purchase likelihood and asymmetry.
2. In order to force product asymmetry, for every one-way directed co-purchase edge i.e. $(u,v) \in E_{cp} \wedge (v,u) \notin E_{cp}$, our model assigns a high score to the edge (u,v) and a low score to the edge (v,u) (terms 3 and 4).
3. For similar products i.e. $(u,v) \in E_{cv}$, our model forces both the source and target embeddings of u and v to be similar (terms 5 and 6). This property is useful in cold-start recommendation and mitigating selection bias.

$$\begin{aligned} loss = &-\left\{ \sum_{(u,v)\in E_{cp}} log(\sigma(\theta_u^s \cdot \theta_v^t)) + \sum_{\substack{n_s=1 \\ z\sim\ P_r(P), u\neq z,}}^{n_k} log(\sigma(1-\theta_u^s \cdot \theta_z^t)) \right\} \\ &-\left\{ \sum_{\substack{(u,v)\in E_{cp}\wedge \\ (v,u)\notin E_{cp}}} log(\sigma(\theta_u^s \cdot \theta_v^t)) + \sum_{\substack{(u,v)\in E_{cp}\wedge \\ (v,u)\notin E_{cp}}} log(\sigma(1-\theta_v^s \cdot \theta_u^t)) \right\} \\ &-\left\{ \sum_{(u,v)\in E_{cv}} log(\sigma(\theta_u^s \cdot \theta_v^s)) + \sum_{(u,v)\in E_{cv}} log(\sigma(\theta_u^t \cdot \theta_v^t)) \right\} \end{aligned} \tag{2}$$

Lemma 1. *The embeddings generated by DAEMON capture product co-purchase likelihood and product asymmetry.*

Proof. When $(a,b) \in E_{cp} \leftrightarrow aR_{cp}b$, our model assigns a high score to $\theta_a^s.\theta_b^t$ compared to $\theta_a^s.\theta_z^t$ i.e. $\theta_a^s.\theta_b^t >> \theta_a^s.\theta_z^t$ (for a random product z). This implies that the co-purchase likelihood between related products a and b is preserved. Further, when $(a,b) \in E_{cp} \wedge (b,a) \notin E_{cp} \leftrightarrow aR_{cp}b \wedge \sim bR_{cp}a$, our model will assign a higher score to $\theta_a^s.\theta_b^t$ compared to $\theta_b^s.\theta_a^t$ i.e. $\theta_a^s.\theta_b^t >> \theta_b^s.\theta_a^t$, thereby capturing product asymmetry.

Lemma 2. *The embeddings generated by DAEMON helps mitigating selection bias.*

Proof. For $a, b, c \in P$, G consists paths of the form $\{a, b, c\}$ where, $(a, b) \in E_{cp} \leftrightarrow aR_{cp}b$ and $(b, c) \in E_{cv} \leftrightarrow bR_{cv}c$. Our model assigns a high score to $\theta^s_a.\theta^t_b$ as $(a, b) \in E_{cp}$. As $(b, c) \in E_{cv}$, our model assigns a high score to $\theta^s_b.\theta^s_c$ and $\theta^t_b.\theta^t_c$. Hence, our model assigns a high score to $\theta^s_a.\theta^t_c = \theta^s_a.\theta^t_b \times \theta^t_b.\theta^t_c$, implying $aR_{cp}c$. We can present a similar argument to show that the embeddings capture $aR_{cv}b \wedge bR_{cp}c \implies aR_{cp}c$. In this manner, our model mitigates selection bias.

In terms of scalability, we employ minibatch sampling both during training and inference procedure to generate product embeddings. We use FAISS [4] to perform efficient nearest neighbor lookup. Our model uses O($|P|$) space to store embeddings corresponding to $|P|$ products. We discuss the results in the next section.

5 Experiments

As previously discussed, RPR problem is equivalent to the node recommendation task in directed graphs. In this section, we evaluate the performance of DAEMON against state-of-the-art models in directed graphs. Specifically, we aim to answer the following evaluation questions:

- **EQ1:** Is DAEMON able to improve the node recommendation performance on real-world e-commerce datasets?
- **EQ2:** How effective is DAEMON in capturing the co-purchase likelihood between products?
- **EQ3:** Is DAEMON able to capture product asymmetry?
- **EQ4:** How effective is DAEMON for cold-start product recommendation?
- **EQ5:** Is DAEMON able to combat selection bias?
- **EQ6:** Semantically, what kind of relationships between products can be extracted using the source and target embeddings generated by DAEMON?

To this end, we introduce the datasets, baselines and follow it up with experiments to answer these questions.

5.1 Experimental Setting

5.1.1 Datasets We extract two real-world datasets sampled from different emerging marketplaces in Amazon. For each dataset, we construct a graph consisting of products as nodes and edges corresponding to either a co-purchase or co-view relationship as explained in Section 4.1. Note that we use anonymized browse logs while creating product co-view pairs. The statistics of the graph datasets is shown in Table 2. These graphs consist 2-5.5M nodes and 14-32M edges and are directed ($\sim$75% of the edges are directed). In both the datasets, we represent each product using input features of size 384 and 512 respectively. These datasets are sampled, and not reflective of production traffic in terms of scale.

Table 2. Summary of graph datasets employed

Dataset	# Nodes	# Edges	Average Degree	%Directed Edges	# Co-purchase pairs	# Co-view pairs	Input Product Feature Dimension
G1	1.98M	14.1M	7.3	76.33	7M	7.1M	384
G2	5.5M	31.7M	5.76	79.97	13.2M	18.5M	512

5.1.2 Implementation Details We implemented DAEMON using DGL and PyTorch. We vary the learning rate in $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ and observed 10^{-4} using Adam's optimizer to work the best. We use $L = 3$ layer GNN model for DAEMON. We use minibatches of size 1024 during model training (Section 4.2.2) and inference (Section 4.2.1). We also use layer-wise neighborhood sampling, i.e. 20 neighbors for the first layer and 10 random neighbors for the subsequent layers. After generating the product embeddings, we perform nearest neighbor lookup to suggest top-k related products for k values in $\{5, 10, 20\}$. All the experiments were conducted on a 64-core machine with a 488 GB RAM running Linux. For all models, we learn 64 dimensional product embeddings trained for a maximum of 30 epochs. We repeat all the experiments 10 times and report the average value across the runs.

5.1.3 Baselines We previously discussed state-of-the-art models in directed graphs (Section 2). In order to evaluate our proposed model, DAEMON, we choose the most competitive baselines as follows:

1. We choose APP [21] and NERD [5] as they deliver superior performance compared to deepwalk[10], node2vec [2], LINE [14], HOPE[9] and VERSE [17].
2. Gravity GAE [11] and DGGAN [22] outperform deepwalk [10], node2vec [2], LINE [14], APP[21], HOPE [9] and VGAE [6].
3. MagNet [20] is the latest state-of-the-art GNN model which delivers best results compared to GraphSage [3], GAT [18], DGCN [16] and APPNP [8].

In summary, we choose APP [21], NERD [5], Gravity GAE [11], DGGAN [22] and MagNet [20] as the most competitive baselines. We use publicly released code repositories for all the employed baselines. Further, we employ parameter tuning to choose the best parameters for each baseline.

However, all the baselines are suitable only for homogeneous directed graphs. For fair comparison, we restrict the evaluation to co-purchase directed graphs (edges corresponding to E_{cp}), and evaluate DAEMON against the baselines (Section 5.2, Section 5.3). For the case of complete data ($E_{cp} \cup E_{cv}$), we compare DAEMON against state-of-the-art R-GCN [12] model in Section 5.4.

5.1.4 Experiment Setup In order to answer the evaluation questions, we setup the experiments (similar to [21], [20], [3]) as follows:

– [EQ1]. Node Recommendation Task (Section 5.2): For each graph dataset, we use 75%, 5% and 20% non-overlapping edges for training, validation and testing respectively. For a ground-truth recommendation (u, v) (from the test data), where u is the query node, we retrieve top-k node recommendations (R_k^q) suggested by each model. In order to evaluate the quality of recommendations, we use HitRate@k and MRR@k for k in $\{5, 10, 20\}$. This helps answering EQ1.

– [EQ2]. Existential link prediction Task (Section 5.3): For each graph dataset, we use 75%, 5% and 20% non-overlapping edges for training, validation and testing respectively. In this task, we want to capture the edge score predicted by each model; a good model assigns higher scores to existing links compared to non-existing links. We evaluate the performance of various models using AUC for this task. This helps answering EQ2.
– [EQ3]. Directed link prediction Task (Section 5.3): For each graph dataset, we use 75%, 5% and 20% non-overlapping edges for training, validation and testing respectively. In this task, we consider only one-way directed edges $((u, v) \in G \wedge (v, u) \notin G)$ as the test edges. We reverse the direction of the test edges to create negative links for testing. We want to evaluate how different models capture the edge direction; a good model assigns a high score to the correct edge (positive links in the test set) and a low score to the reverse edge (negative links in the test set). We evaluate the model performance using AUC for this task. This helps answering EQ3.
– [EQ4]. Cold-start Recommendation Task (Section 5.4): For each graph dataset, we use subgraphs corresponding 75%, 5% and 20% non-overlapping nodes for training, validation and testing respectively. For evaluation, we take the same approach as employed in the node recommendation task. This helps answering EQ4.
– [EQ5]. Selection-bias Recommendation Task (Section 5.4): For each graph dataset, we use edges corresponding 75%, 5% and 20% non-overlapping nodes for training, validation and testing respectively. *Further, we add edges corresponding to transitive relationships $aR_{cp}b \wedge bR_{cv}c \implies aR_{cp}c$ to the test set.* In order to mitigate selection bias, a model needs to recommend c as a related product for a as these transitive relationships are present in the training graph. For evaluation, we take the same approach as employed in the node recommendation task. This helps answering EQ5.

5.2 EQ1. Node Recommendation Task on Co-purchase Data

The results[3] for the node recommendation task is shown in Table 3. We see that DAEMON performs the best for this task in terms of both HitRate and MRR. DAEMON yields more gains on the bigger $G2$ dataset compared to the $G1$ datset. Further, DGGAN fails to complete model training in 48 hours. It does not scale to real-world datasets; the paper show their model efficacy on smaller datasets (the biggest graph has 15K nodes). Further, Gravity GAE runs out of memory (488 GB RAM) during model training. This is due to one-hot feature

[3] Results are relative to the co-purchase baseline and absolute numbers are not presented due to confidentiality.

encoding employed by the model to represent node input features i.e. each node is represented as a million valued embedding initially. APP delivers the second best performance. MagNet is not applicable for node recommendation task, and we compare against it for the link prediction tasks.

Table 3. Node recommendation. Best results are in **bold** and second are underlined

Dataset	Model	HitRate@k			MRR@k		
		5	10	20	5	10	20
G1	APP	2.14x	4.01x	6.95x	1.02x	1.26x	1.46x
	NERD	0.62x	1.94x	0.0314	0.61x	0.71x	0.79x
	DAEMON	**3.07x**	**5.54x**	**8.98x**	**1.45x**	**1.77x**	**2.01x**
	%Gain	43.4	38.15	29.2	42.15	40.47	37.6
G2	APP	1.23x	1.99x	3.31x	0.67x	0.77x	0.85x
	NERD	1.28x	1.85x	2.63x	0.71x	0.78x	0.84x
	DAEMON	**3.51x**	**5.87x**	**8.6x**	**1.71x**	**2.02x**	**2.21x**
	%Gain	174	194	159.8	155.2	162.3	160

5.3 [EQ2,EQ3.] Link Prediction Tasks on Co-purchase Data

Table 4. Link prediction (%). Best results are in **bold** and second are underlined

Dataset	Model	Existence Prediction (AUC %)	Direction Prediction (AUC %)
G1	APP	34.9x	0.1x
	NERD	23.3x	7.1x
	MagNet	36.1x	12.65x
	DAEMON	**40.31x**	**14.72x**
	Δ Gain	4.2	2.25
G2	APP	5.27x	0.15x
	NERD	14.45x	2.48x
	MagNet	18.26x	6.62x
	DAEMON	**34.28x**	**12.53x**
	Δ Gain	16	6.6

We present the performance[4] of different models in Table 4 for two link prediction tasks. We see that DAEMON outperforms all baselines on both these tasks. MagNet displays the second best performance. We make a note that predicting existence of links is a simpler task when compared to predicting the correct edge direction; this is also observed in the results. As explained previously, DGGAN fails to complete model training in 48 hours and Gravity GAE runs out of memory (488GB RAM) during model training. These results indicate that the proposed model is able to jointly model co-purchase likelihood and product asymmetry compared to the baselines which answers **EQ2** and **EQ3**.

5.4 [EQ4, EQ5, EQ6.] Ablation Study on G1 graph

We evaluate[5] DAEMON on the complete graph in this section to analyze its performance for the case of cold-start product recommendation task (Table 5) and the case of mitigating selection bias (Table 6). Observe from Table 5 that DAEMON delivers significant performance gains over R-GCN for cold-start recommendation evaluation. Although, both models were fed with the same cold-start graphs, observe that R-GCN performs poorly. Results indicate how leveraging catalog product information to create subgraphs pertaining to cold-start products aid in boosting the performance of our model. This answers **EQ4**.

Table 5. Cold-start evaluation on G1 graph

Model	HitRate@k			MRR@k		
	5	10	20	5	10	20
R-GCN	x	x	x	x	x	x
DAEMON	**34.14x**	**22.02x**	**15.52x**	**98.05x**	**86.75x**	**51.4x**

Table 6 shows the results pertaining to selection bias. Observe that when the model is trained only the co-purchase edges, it results in a performance dip. This is because the transitive relationships crafted in the test set cannot be captured when trained only on the co-purchase edges. However, R-GCN trained on both co-purchase and co-view edges delivers a poor performance which confirms our claim that we need to provide differential treatments to co-purchase and co-view edges during feature aggregation. DAEMON delivers the best performance compared to R-GCN model. These results demonstrate how DAEMON is able to mitigate selection and this answers **EQ5**.

In order to answer **EQ6**, we assume a query product q. We generate a related product v, when $\theta_q^s.\theta_v^t$ is the highest $\forall v \in P$. Similarly, given a query product q,

[4] Results are relative to the co-purchase baseline and absolute numbers are not presented due to confidentiality.

[5] Results are relative to the R-GCN baseline and absolute numbers are not presented due to confidentiality.

Table 6. Selection bias evaluation on G1 graph

Model	HitRate@k			MRR@k		
	5	10	20	5	10	20
R-GCN (co-purchase+co-view)	x	x	x	x	x	x
DAEMON (co-purchase)	3.1x	3.26x	3.5x	4.26x	4.02x	3.94x
DAEMON (co-purchase+co-view)	**3.43x**	**3.5x**	**3.59x**	**4.58x**	**4.32x**	**4.21x**

we generate a similar product v, when $\theta_q^s.\theta_v^s$ is the highest $\forall v \in P$. Observe from Table 7 the kind of recommendations generated. This shows that we can leverage both the source and target embeddings generated by DAEMON to recommend not only related products, but also similar items as a byproduct.

Table 7. Sample product recommendations generated using DAEMON

Sl. No	**query product**	**related product recommendation**	**similar product recommendation**
1	amway attitude sunscreen cream 100 g	amway attitude face wash for dry skin - 100 ml	amway attitudes sunscreen cream 100 g x 2 = 200 gm
2	oppo reno phone 6 pro 5g stellar black 12gb ram 256gb storage	lustree oppo reno 6 pro 5g bumper back cover case	oppo reno phone 6 5g stellar black 8gb ram 128gb storage
3	paseo tissues pocket hanky 6 packs 3 ply	origami wet wipes wet tissue wet facial tissue	inkulture pocket hanky tissue paper white pack of 10
4	yiwoo 5 pieces false eyelashes curler	ardell duo individual lash adhesive white 7 g	vega premium eye lash curler
5	skinn by titan steele fragrance for men 100 ml	blue nectar natural vitamin c face cream for glowing skin	skinn by titan fragrance for men 20ml

5.5 Online Platform Performance

We further evaluate the performance of DAEMON in production environment by conducting an A/B test in two different marketplaces. For the control group, we use an incumbent approach based on product co-purchases, while for the treatment group, we show the recommendations generated from DAEMON. We run the experiments for 4 weeks and observe +170% improvement on product sales and +190% improvement on profit gain. All the results are statistically significant with p-value < 0.05. These results show the product recommendations generated from DAEMON can significantly improve customer shopping experience in discovering potentially related products of interest.

6 Conclusion and Future Work

In this paper, we propose DAEMON, a novel Graph Neural Network based framework for related product recommendation, wherein the problem is formulated as a node recommendation task on a directed product graph. In order to train the model, we employ an asymmetric loss function by modelling product co-purchase likelihood, capturing product asymmetry and mitigating selection bias. We leverage the trained GNN model to learn dual embeddings for each product, by appropriately aggregating features from its neighborhood. Extensive offline experiments show that DAEMON outperforms state-of-the-art baselines for the node recommendation and link prediction tasks against state-of-the-art baselines. In the future, we will explore the possibility of representing products using complex valued embeddings and extend DAEMON to the case of complex embeddings to improve the performance.

References

1. Chen, J., Dong, H., Wang, X., Feng, F., Wang, M., He, X.: Bias and debias in recommender system: A survey and future directions. arXiv preprint (2020)
2. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: SIGKDD. pp. 855–864 (2016)
3. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. NIPS **30** (2017)
4. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. IEEE Transactions on Big Data **7**(3), 535–547 (2021)
5. Khosla, M., Leonhardt, J., Nejdl, W., Anand, A.: Node representation learning for directed graphs. In: ECML PKDD. pp. 395–411 (2019)
6. Kipf, T.N., Welling, M.: Variational graph auto-encoders. NIPS Workshop on Bayesian Deep Learning (2016)
7. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. ICLR (2017)
8. Klicpera, J., Bojchevski, A., Günnemann, S.: Predict then propagate: Graph neural networks meet personalized pagerank. ICLR (2019)
9. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD (2016)
10. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD. pp. 701–710 (2014)
11. Salha, G., Limnios, S., Hennequin, R., Tran, V.A., Vazirgiannis, M.: Gravity-inspired graph autoencoders for directed link prediction. In: CIKM. pp. 589–598 (2019)
12. Schlichtkrull, M., Kipf, T.N., Bloem, P., Berg, R.v.d., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: ESWC. pp. 593–607 (2018)
13. Sun, J., Bandyopadhyay, B., Bashizade, A., Liang, J., Sadayappan, P., Parthasarathy, S.: Atp: Directed graph embedding with asymmetric transitivity preservation. In: AAAI (2019)
14. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW. pp. 1067–1077 (2015)

15. Tong, Z., Liang, Y., Sun, C., Li, X., Rosenblum, D., Lim, A.: Digraph inception convolutional networks. NIPS **33**, 17907–17918 (2020)
16. Tong, Z., Liang, Y., Sun, C., Rosenblum, D.S., Lim, A.: Directed graph convolutional network. arXiv preprint arXiv:2004.13970 (2020)
17. Tsitsulin, A., Mottin, D., Karras, P., Müller, E.: Verse: Versatile graph embeddings from similarity measures. In: WWW 2018. pp. 539–548 (2018)
18. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. ICLR (2018)
19. Wang, X., Zhang, R., Sun, Y., Qi, J.: Combating selection biases in recommender systems with a few unbiased ratings. In: WSDM. pp. 427–435 (2021)
20. Zhang, X., He, Y., Brugnone, N., Perlmutter, M., Hirn, M.: Magnet: A neural network for directed graphs. NIPS **34** (2021)
21. Zhou, C., Liu, Y., Liu, X., Liu, Z., Gao, J.: Scalable graph embedding for asymmetric proximity. In: AAAI (2017)
22. Zhu, S., Li, J., Peng, H., Wang, S., Yu, P.S., He, L.: Adversarial directed graph embedding. AAAI (2021)