Personalized Complementary Product Recommendation

An Yan
University of California San Diego
San Diego, CA, USA
ayan@eng.ucsd.edu

Chaosheng Dong, Yan Gao, Jinmiao Fu,
Tong Zhao, Yi Sun, Julian McAuley
Amazon
Seattle, WA, USA
{chaosd,yanngao,jinmiao,zhaoton,yisun,jumcaule}@amazon.com

ABSTRACT
Complementary product recommendation aims at providing product suggestions that are often bought together to serve a joint demand. Existing work mainly focuses on modeling product relationships at a population level, but does not consider personalized preferences of different customers. In this paper, we propose a framework for personalized complementary product recommendation capable of recommending products that fit the demand and preferences of the customers. Specifically, we model product relations and user preferences with a graph attention network and a sequential behavior transformer, respectively. Two networks are cast together through personalized re-ranking and contrastive learning, in which the user and product embedding are learned jointly in an end-to-end fashion. The system recognizes different customer interests by learning from their purchase history and the correlations among customers and products. Experimental results demonstrate that our model benefits from learning personalized information and outperforms non-personalized methods on real production data.

CCS CONCEPTS
• Computing methodologies → Neural networks.

KEYWORDS
Complementary Product Recommendation, Personalization, Graph Neural Networks

ACM Reference Format:

1 INTRODUCTION
Complementary product recommendation (CPR) aims at suggesting products that are frequently purchased together. In general, CPR tries to find a set of recommendations to optimize the ultimate goal, which could be one or a combination of revenue, diversity, customer click rate, etc. A typical first step in recent work [11, 23] is to form a massive product graph, where the products are represented as nodes and their relationships are represented as edges. Subsequently, a graph neural network (GNN) is leveraged to exploit such graph structure and obtain a product embedding for prediction.

However, a pivotal piece of information from customers’ personalized preferences is neglected. As shown in Figure 1, when a query product (computer) is given, a vanilla complementary product recommendation model without personalization will suggest the same list of products for all users. If two users have different preferences on colors, or have purchased specific products before, this would not be ideal as it wastes resources on products that the customers are not going to buy. Hence it inevitably limits the total number of relevant candidate products we would recommend for individual customers. Inspired by the fact that for the same query product, different users have different complementary co-purchase choices, we propose to incorporate personalized information into complementary product recommendation.

Modeling personalized information in CPR is non-trivial. The challenges include how to model customer preferences efficiently, how to leverage and benefit from both general recommendations and personalized information, and how to solve the data sparsity issue in practice. Simply setting up some arbitrary rules (e.g., excluding certain products purchased before or recommending products with their favorite color) to model customer preferences is not scalable, since the purchase histories are diverse and noisy, so the co-purchase decisions are intricate and difficult to model explicitly. Hence, we design a framework for personalized complementary product recommendation.

More specifically, we leverage a graph attention network (GAT) [34] to capture product relations, and a transformer [33] to capture signals from users’ past behaviors. We combine these two models...
We present our framework in Figure 2. We use a graph attention network to model product relations, and transformer to model user purchase histories. These two modules are learned jointly via a re-ranking scheme into the framework. By including personalization into the framework, we further introduce a contrastive learning scheme into the framework. By including personalization, it could recommend different complementary products for different customers given the same query, as well as improving the overall recommendation performance.

Overall, our contributions are three-fold:

- We combine GAT and Transformer model to capture the sequential signals underlying users’ behavior sequences.
- We propose a re-ranking module with contrastive learning for personalized complementary product recommendation to enable efficient end-to-end training.
- We conduct comprehensive experiments to show the effectiveness of our method, which outperforms the non-personalized method in terms of hit rates.

2 METHOD

We present our framework in Figure 2. We use a graph attention network to model product relations, and transformer to model user histories. These two modules are learned jointly via a re-ranking loss, and enhanced with contrastive learning.

2.1 Graph Attention Network

Our baseline model for non-personalized complementary product recommendation is based on a graph attention network, inspired by previous work [11]. It starts with taking each product’s title and review features $T_i$ as input and applies an universal embedding module $\Phi(\cdot)$, which consists of two feed-forward layers with a batch normalization layer, to all products identically to obtain initial k-dimensional embeddings $\hat{\theta}_i$, as shown in Eq. (1):

$$\hat{\theta}_i = \Phi(T_i) = \sigma((T_i W_1 + b_1) W_2 + b_2)$$

where $\sigma$ is the batch normalization layer, $W_1 \in R^{d \times k}$ and $W_2 \in R^{k \times k}$ are weight matrices. This embedding $\hat{\theta}_i$ is then fed into a GAT layer, which selectively aggregates the neighbors from all local connections, to fine-tune parameters in the feed-forward network. More specifically, given a product $i$ and the set of neighbor products $\mathcal{N}_i$, an attention vector $a_{i,j}$ is calculated based on $\hat{\theta}_i$ and $\hat{\theta}_j$ normalized on the softmax function, which can adaptively capture the similarities when summarizing over products $j \in \mathcal{N}_i$.

$$a_{i,j} = \text{softmax}(\theta_1^T \hat{\theta}_j)$$

(2)

We could learn the product-relation embeddings from GAT as $\hat{\theta}_i = \sum_{j \in \mathcal{N}_i} a_{i,j} \hat{\theta}_j$, and use them for predictions. Given a query product $i$, we obtain its embedding $\hat{\theta}_i$ via the GAT module, and compute the distance between $\hat{\theta}_i$ and all other product embeddings $\hat{\theta}_j$. The top-k list is used for complementary recommendation.

The objective function for learning product-level features is to minimize a hinge loss:

$$\mathcal{L}_G = \sum_{i \in \mathcal{I}} \sum_{y \in \{\pm 1\}} \{\max(0, y(\lambda + \|\hat{\theta}_i - \theta_{yi}\|)^2 - \epsilon)\}$$

(3)

where the loss intends to pull the distance between $\hat{\theta}_i$ and $\theta_{yi}$ less than $\lambda - \epsilon$ while push $\hat{\theta}_i$ away from a random negative sample $\theta_{yi}$ with distance greater than $\lambda + \epsilon$. $\mathcal{I}$ is the product set. $\theta_{yi} = \{\theta_{i+1}, \theta_{i-1}\}$ for $y = \{+1, -1\}$, respectively. $\lambda$ and $\epsilon$ are hyperparameters to control the distance. We refer readers to [11] for more details.

2.2 User Behavior Modeling

To learn user historical activities, we leverage a transformer with positional encoding to model purchase histories in a sequential manner following the design of [33]. Given a user’s recent purchased products $P = \{p_1, ..., p_n\}$, we first encode the products with title features to obtain a sequence of features $T = \{t_1, t_2, ..., t_n\}$, then feed the sequence into a transformer encoder. We use the first embedding from the hidden states of the transformer as a contextual...
user embedding $u$:

$$u = \text{transformer}(t_1, t_2, ..., t_n)$$  \hspace{1cm} (4)

### 2.3 Personalized Re-Ranking

After obtaining both the product-relation embeddings and user embeddings, we introduce personalized via a re-ranking module.

The objective function for learning personalized re-ranking is to optimize the following hinge loss:

$$L_{PR} = \sum_{i \in I} \sum_{y \in \{\pm 1\}} \{\max(0, \epsilon - y(\lambda - \| \theta_i - \theta_{\pm c} \| u \|) \} \}$$  \hspace{1cm} (5)

The loss is similar to Eq. (3), with the same $\lambda$ and $\epsilon$. $+c$ is a co-purchased product with $i$ by the same customer in one session, $-c$ is a random negative sample. The distance between two features are weighted by a user preference embedding $u$ learned from historical purchases. During inference, given top-k recommendations $S_k$ from the non-personalized model, we re-rank this top-k list based on vector multiplication $\theta \odot u$.

### 2.4 Contrastive Learning

Despite the success of applying transformer for sequential user behavior modeling, this sequential prediction task needs to optimize the model with sparse training data, which makes it difficult to learn high-quality user representations. To tackle this challenge, we leverage contrastive learning for sequential modeling, to learn self-supervised signals from the original behavior sequence.

Given a sequence of purchased products $P = \{p_1, ..., p_m\}$ from a user, we apply random cropping and reordering on this product sequence to create two views (i.e., two augmented sequences) $P_1$ and $P_2$. These two sequences are passed through the behavior transformer to obtain two augmented user embeddings $u_1$ and $u_2$. We treat $(u_1, u_2)$ as a positive pair and treat other examples within the same mini-batch as negative samples $S^-$. Our contrastive loss is:

$$L_{CL} = \sum_{u \in U} \log \frac{\exp(\text{sim}(u_1, u_2))}{\exp(\text{sim}(u_1, u_2)) + \sum_{s^- \in S^-} \exp(\text{sim}(u_1, s^-))}$$  \hspace{1cm} (6)

Overall, the model is optimized with a mixture of a graph-level loss, personalized re-ranking loss and contrastive learning loss:

$$L_{loss} = L_G + \lambda_1 L_{PR} + \lambda_2 L_{CL}$$  \hspace{1cm} (7)

where $\lambda_1$ and $\lambda_2$ are hyperparameters that weigh the two losses.

### 3 EXPERIMENTS

#### 3.1 Experimental Setup

**Dataset.** We conduct experiments on one e-commerce dataset. We randomly sampled 10,000 customers with co-purchase sessions from June 2020 to November 2020, and collected their corresponding purchase history from the last month for user behavior modeling, i.e., in May 2020. We randomly split customers into 8K/1K/1K for train/val/test.

**Evaluation Metrics.** A standard measurement for ranking tasks is the Hit@k score. Given a session of products (query product, co-purchased products $J$) in co-purchase test data and the top-k recommendations $S_k$ from the model, the Hit@k score for each

<table>
<thead>
<tr>
<th>Method</th>
<th>Product level</th>
<th>Category level</th>
<th>Product type level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hit@1</td>
<td>Hit@3</td>
<td>Hit@10</td>
</tr>
<tr>
<td>GAT</td>
<td>0.0065</td>
<td>0.014</td>
<td>0.041</td>
</tr>
<tr>
<td>Projection</td>
<td>0.0015</td>
<td>0.005</td>
<td>0.011</td>
</tr>
<tr>
<td>GAT+Avg</td>
<td>0.0056</td>
<td>0.012</td>
<td>0.034</td>
</tr>
<tr>
<td>GAT+Trans</td>
<td>0.0069</td>
<td>0.017</td>
<td>0.044</td>
</tr>
</tbody>
</table>

*Our method 0.0081 0.019 0.046 0.279 0.551*

**Table 1: Performance comparison on the e-commerce dataset.**

GAT is a non-personalized baseline using only product features with similar modules in [11]. Projection uses the direct concatenation of user and product embeddings instead of multiplication in Eq. (5). GAT+Avg is to average user purchase history without sequential modeling. GAT+Trans is to use transformer without contrastive learning. Our final method is GAT+Trans with contrastive learning.

**Figure 3: Performance with different re-ranking sizes.**

The re-ranking size is defined as:

$$\text{Hit@k} = \begin{cases} 1 & (S_k \cap J) \geq 1 \\ 0 & \text{otherwise} \end{cases}, k = 1, 3, 5, 10...$$

We report Hit@k on Product, Category and Product type level.

#### 3.2 Performance Comparison

We compare our personalized model with different methods. As shown in Table 1, our personalized model outperforms non-personalized baseline on hit-rate at all levels, demonstrating the necessity of personalization for complementary recommendation. GAT+Avg underperforms GAT possibly due to the lack of parameter learning for user embeddings. The comparison between GAT+Avg and GAT+Trans validates the importance of sequential user modeling. Finally, adding the contrastive loss to learn robust user sequential representations can further improve performance.

#### 3.3 Ablation Studies

**Re-ranking size.** We test how the sizes of the re-ranking list make impacts on the performance, shown in Figure 3. Empirically, we found a smaller size of list leads to slightly better performance for top-k recommendation with a small $k$ (e.g., $k=1,3$), but overall the results are quite robust for list sizes from 20 to 200.

1Category: It can be viewed as a subset under each product type.
Table 2: Ablation study on different features as input for product or user modeling. Results are reported on our personalized model w/o contrastive learning.

<table>
<thead>
<tr>
<th>Metric</th>
<th>One-hot</th>
<th>BERT+R</th>
<th>BERT</th>
<th>Demo</th>
<th>Pur</th>
<th>Demo + Pur</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit@1</td>
<td>0.0030</td>
<td>0.0058</td>
<td><strong>0.0069</strong></td>
<td>0.0056</td>
<td>0.0065</td>
<td><strong>0.0069</strong></td>
</tr>
<tr>
<td>Hit@3</td>
<td>0.0070</td>
<td>0.0159</td>
<td><strong>0.0172</strong></td>
<td>0.0143</td>
<td>0.0159</td>
<td><strong>0.0172</strong></td>
</tr>
<tr>
<td>Hit@10</td>
<td>0.0260</td>
<td>0.0383</td>
<td><strong>0.0411</strong></td>
<td>0.0386</td>
<td>0.0435</td>
<td><strong>0.0411</strong></td>
</tr>
</tbody>
</table>

**Different features.** To investigate the contribution of different feature options, we conduct ablation studies on the input features of GAT and transformer model in Table 2. First, for learning product relations via the graph neural network, we test three product inputs. One-hot is one-hot encoding based on Category. BERT is to extract representations from product titles through a pretrained BERT [7] model. BERT+R is to further add product-related features such as review rating, price, brand.

Then, we test which features help user modeling. Demo is to build users’ static demographic feature based on their time using the online shopping website, purchase frequency, etc. Pur is to model user purchase histories with a transformer. Demo+Pur is to fuse two features together. Overall, we found richer features (e.g., using BERT representations for product embeddings, and purchase history with demographic features) as input lead to better performance.

**3.4 Case Study**

To have a better understanding of our model, we provide case studies in Figure 4. When given the correction tape as the query product, a common recommendation could be some type of papers, like the top recommended products from the non-personalized model. However, since the user has already purchased many papers in the past month, a reasonable result would be avoiding recommending papers for this purchase. Our personalized model can take this historical pattern into consideration, and papers are not shown in the top-5 recommendation from the personalized model.
REFERENCES


[12] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momen-
tional conference on machine learning. PMLR, 10532–10541.


