Learning from Substitutable and Complementary Relations for Graph-based Sequential Product Recommendation

WEI ZHANG and ZEYUAN CHEN, East China Normal University, China
HONGYUAN ZHA, The Chinese University of Hong Kong (Shenzhen), China
JIANYONG WANG, Tsinghua University, China

Sequential product recommendation, aiming at predicting the products that a target user will interact with soon, has become a hotspot topic. Most of the sequential recommendation models focus on learning from users' interacted product sequences in a purely data-driven manner. However, they largely overlook the knowledgeable substitutable and complementary relations between products. To address this issue, we propose a novel Substitutable and Complementary Graph-based Sequential Product Recommendation model, namely, SCG-SPRe. The innovations of SCG-SPRe lie in its two main modules: (1) The module of interactive graph neural networks jointly encodes the high-order product correlations in the substitutable graph and the complementary graph into two types of relation-specific product representations. (2) The module of kernel-enhanced transformer networks adaptively fuses multiple temporal kernels to characterize the unique temporal patterns between a candidate product to be recommended and any interacted product in a target behavior sequence. Thanks to the seamless integration of the two modules, SCG-SPRe obtains candidate-dependent user representations for different candidate products to compute the corresponding ranking scores. We conduct extensive experiments on three public datasets, demonstrating SCG-SPRe is superior to competitive sequential recommendation baselines and validating the benefits of explicitly modeling the product-product relations.

CCS Concepts: • **Information systems** → **Recommender systems**; *Personalization*; Temporal data;

Additional Key Words and Phrases: Sequential recommendation, graph neural networks, attention mechanism, substitutable and complementary relations

ACM Reference format:
Wei Zhang, Zeyuan Chen, Hongyuan Zha, and Jianyong Wang. 2021. Learning from Substitutable and Complementary Relations for Graph-based Sequential Product Recommendation. *ACM Trans. Inf. Syst.* 40, 2, Article 26 (September 2021), 28 pages.
https://doi.org/10.1145/3464302

This article was partially supported by the National Natural Science Foundation of China (Grants No. 62072182, No. 61532010, and No. 61521002), the Fundamental Research Funds for the Central Universities, and Beijing Academy of Artificial Intelligence (BAAI).
Authors' addresses: W. Zhang (corresponding author) and Z. Chen, East China Normal University, No. 3663, North Zhongshan Road, Shanghai, 200062, China; emails: zhangwei.thu2011@gmail.com, chenzyfm@outlook.com; H. Zha, The Chinese University of Hong Kong (Shenzhen), 2001 Longxiang Boulevard, Longgang District, Shenzhen, China; email: zhahy@cuhk.edu.cn; J. Wang, Tsinghua University, Haidian District, Beijing, 100084, China; email: jianyong@tsinghua.edu.cn.

1046-8188/2021/09-ART26 $15.00
https://doi.org/10.1145/3464302

1 INTRODUCTION

User behavior data, one of the most fundamental and important types of user-generated data, is largely accumulated with the rapid development of online services, such as user shopping behavior in e-commerce services. Because of the intrinsic sequentiality in rich behavior data, modern recommender systems tend to organize them as user-level sequential data and perform sequential recommendation [25]. Compared to conventional rating prediction [15], sequential recommendation naturally considers modeling dynamic user interests, which is more accordant with real situations. The behavior sequences utilized in sequential recommendation are usually represented by the interacted items. There are rich types of user-item interactions, for example, clicking or buying preferred products. Considering the techniques used in sequential recommendation, the past decade has witnessed their development from sequential pattern mining [23], to Markov chain-based factorization models [28], to recent deep neural networks [10].

In the domain of deep sequential recommendation approaches, much technical progress has been made to promote recommendation performance. This originates from recurrent neural networks [10], and consequently, different attention mechanisms [12, 16] are developed. In the past two years, graph neural networks [51] have also been successfully applied to this domain [24, 36, 43]. In summary, many efforts have been devoted to how to learn user interest representations from the sequentiality of interacted items, which is realized in a purely data-driven manner. However, despite the temporal relations between items, there exists some prior knowledge about item relations. It can facilitate user representation learning from a knowledge-driven way. In this article, we concentrate on the crucial substitutable and complementary relations between products in e-commerce platforms [21] and aim at leveraging these explicit product-product relations to boost sequential product recommendation. Then a key research problem is naturally raised: how to effectively incorporate knowledgeable substitutable and complementary relations into sequential product recommendation models?

The substitutable and complementary relations between products are ubiquitous in e-commerce platforms. On the one hand, the substitutable relation means two products have similar functions and could be used interchangeably. For example, the cellphones "Iphone 12" and "HUAWEI Mate 40" are substitutes for each other. Intuitively, a consumer normally purchases at most one of them each time in terms of practical usage. On the other hand, the complementary relation denotes that two products could be used in conjunction with each other to pursue high overall utilities. For instance, cellphones are often used in conjunction with cases. As such, they are usually brought within a short time interval. Due to the significance of these two relation types for business marketing, more and more research attention has been paid to this domain, which could be roughly categorized into two aspects: (1) inferring substitutable and complementary relations between products [20, 21, 40, 45] and (2) substitutable and complementary product recommendation [6, 11, 49, 50]. The former aspect focuses on estimating the missing substitutable and complementary relations between products to enrich the existing ones. Thus, it could be regarded as a prerequisite for our study. Although the latter aspect is also towards recommendation, its task setting is to recommend complementary or substitutable products for a given query product, which is fundamentally different from sequential recommendation.

We argue that it is necessary to consider the substitutable and complementary relations for sequential recommendation due to the following reasons. First, and most importantly, characterizing the explicit relation between a given candidate product and one of the interacted items in a user behavior sequence could benefit recommendation. Figure 1 provides a toy example to illustrate the motivation. In this figure, the blue panel contains a behavior sequence for a target user. On the right, two candidate products for recommendation are presented. For the upper candidate (i.e.,

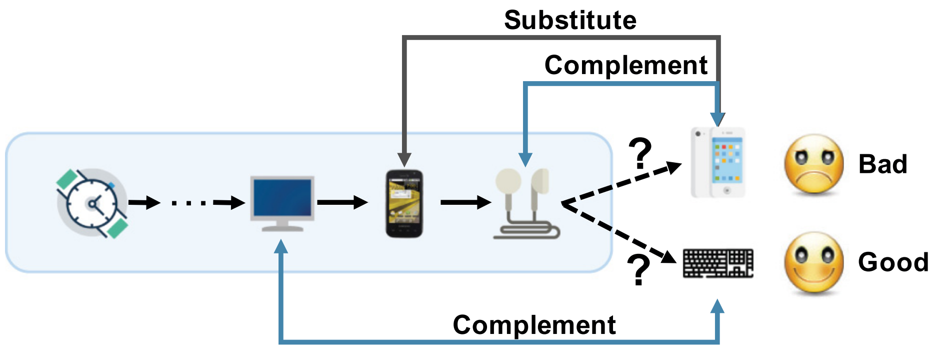


Fig. 1. Illustration of the substitutable and complementary relations for sequential recommendation through a toy example.

cellphone), although it is complementary to the most recent product (i.e., earphone), it is substitutable to another cellphone in the near past. From a whole perspective of the behavior sequence, it is not suitable to take the cellphone as the recommendation, which yet might be adopted by the aforementioned complementary recommendation methods. By contrast, although the lower candidate (i.e., keyboard) has no direct relations with the most recent product, it is complementary to the computer in the sequence. As such, it might be a good choice to recommend the keyboard, which is not easy to be handled by conventional complementary recommendation as well. Second, considering the substitutable and complementary relations between each pair of all the products in a user behavior sequence could promote user representation learning. This is attributed to the fact that these relations enrich product-product correlations from a different view of temporal patterns. Therefore, they lead to better product representations and further facilitate user behavior sequence representation learning.

To our knowledge, the pioneering work [34] is the most relevant study in the literature of leveraging the substitutable and complementary relations for sequential recommendation. Its proposed model first applies TransE [2] to obtain relation-aware embeddings for items in behavior sequences. Afterwards, temporal kernel functions are adopted to characterize the substitutable relation and complementary relations in candidate product recommendation. However, there are some limitations still remained in the model: (1) The substitutable and complementary relations sometimes have a certain degree of transitivity. For example, given two pairs of substitutes, i.e., ("Iphone 12," "HUAWEI Mate 40") and ("Iphone 12," "Galaxy Note20"), we could also find that "HUAWEI Mate 40" and "Galaxy Note20" are substitutable to each other. Capturing this high-order relation in a reasonable manner is critical, because having all pairs with substitutable and complementary relations is unrealistic. However, the model overlooks the high-order relation modeling. (2) The kernel functions used in this model have simple and fixed forms. They could only differentiate the temporal patterns in the product relation and category level. Hence it lacks a strong expressive ability to characterize the unique temporal patterns w.r.t. a specific product-product pair. (3) The study concentrates on enhancing the candidate product representations through relational modeling. But on the user side, the user interests are only associated with the dense embeddings mapped from user IDs. Consequently, the solution is not consistent with the existing sequential recommendation models that are tailored for user dynamics, although the temporal patterns might be interleaved in user and item representations.

To address the above limitations, in this article, we propose a novel model named SCG-SPRe, which stands for the Substitutable and Complementary Graph-based Sequential Product

Recommendation model. Specifically, SCG-SPRe contains two key computational modules: interactive **graph neural networks (GNNs)** and kernel-enhanced transformer networks. First, according to a constructed substitutable product-product graph and a complementary product-product graph, the module of interactive graph neural networks encodes high-order correlations indicated in the graphs into substitute-based product representations and complement-based product representations, respectively. Thanks to the multi-layer representation propagation, relation transitivity could be captured in the obtained product representations to some extent. To grasp more complex relation dependencies, the representation propagation in each layer of the two graphs is coupled with each other through a gating mechanism. Second, based on the obtained two types of product representations, the module of kernel-enhanced transformer networks learns candidate-dependent user representations that are tailored for different candidate products. In particular, each user representation is decomposed into a sequence-aware representation and a kernel-based representation. While the former is obtained over the whole behavior sequence, the latter involves the temporal patterns w.r.t. some interacted products that have the substitutable or complementary relations with a given candidate product. To enhance the expressive ability to learn the patterns, the module introduces multiple types of kernels and adaptively fuses them by attention computation. Based on the seamless integration of the two modules, the candidate-dependent user representations are finally exploited to compute the ranking score for the candidate product.

To sum up, the contributions of this article are as follows:

- We emphasize the importance of learning from substitutable and complementary relations for sequential product recommendation, which is largely neglected by relevant studies. We, therefore, devise the novel model SCG-SPRe to encode these relations into product representations and further generate candidate-dependent user representations.
- We develop interactive graph neural networks to flexibly capture substitutable and complementary relation transitivity by representation propagation, and encode these relations into product representations. To our best knowledge, this is the first study to apply GNNs to model substitutable and complementary relations for sequential recommendation.
- We design kernel-enhanced transformer networks, which are capable of not only adaptively fusing multiple different types of temporal kernel functions to capture unique temporal patterns w.r.t. different specific product pairs but also deriving candidate-dependent user representations.
- Experimental results include an ablation study demonstrate the effectiveness of the proposed SCG-SPRe and its key components, including the interactive graph neural networks and the adaptive fusion of temporal kernels. In particular, our model significantly outperforms the knowledge- and time-aware item modeling approach [34] and the well-performed deep sequential recommendation model [17].

2 RELATED WORK

Broadly speaking, our work is relevant to deep learning for sequential recommendation, substitutable and complementary relation modeling, and graph neural networks for general recommendation. In what follows, we discuss each part to highlight the relations and differences with ours.

2.1 Deep Learning for Sequential Recommendation

The success of deep learning has been widely observed in recommender systems in recent years [46]. With regard to sequential recommendation, **recurrent neural networks (RNNs)** are first leveraged by the seminar work [10] to obtain the representations of whole user behavior sessions for sequential recommendation. To further connect different sessions belonging to the same user, Quadrana et al. [26] investigated the hierarchical version of recurrent neural networks. Moreover, RRN [42] considers the temporal dynamics of both user and item sides by coupling two recurrent neural networks.

Due to the limitations of RNNs in modeling long-range dependency and parameter optimization, there are roughly two technical aspects to boost recommendation performance. On the one hand, attention mechanisms [1] are employed to better fuse the representations of items in a behavior sequence. For example, Li et al. [16] proposed to add a layer of attention computation after the hidden outputs of RNNs, which aims to quantify the importance of each interacted item to represent user interests. To model the latent interactions between items in a behavior sequence, Kang et al. [12] utilized the idea of transformer networks [31] to calculate the attention weights of all the past items w.r.t. the current item at each time step. Besides, nonlinear transformation is adopted in transformer-like networks to enhance model capacities. Inspired by BERT's [3] abilities of bidirectional modeling and pre-training, BERT4Rec [29] is tailored for sequential recommendation by considering the influence from the interacted items in the future on the current item representations. The recent transformer-like recommendation model [17] further incorporates time intervals into attention computation. It could better measure item-item latent relations.

On the other hand, graph neural networks rely on the constructed item-item relational graphs, which is in contrast to the above technical direction. Specifically, Wu et al. [43] built local graphs w.r.t. different user sessions, where the edges between two items denote their co-occurrence within a sliding window for the considered session. Based on the graphs, GNNs are employed to update item representations, which are used to further derive user preference. To extend the item-item relation modeling across different sessions, the two studies [24, 36] are conducted in the same period to build global graphs. Both of them demonstrate the additional gain brought by the global graphs. The study [41] also investigates the fusion of a global item-item graph and each user session for the task.

Although much progress has been made in deep learning for sequential recommendation, only a very few efforts are devoted to leveraging the substitutable and complementary relations for this scenario, which motivates our study.

2.2 Substitutable and Complementary Relation Modeling

As aforementioned, the substitutable and complementary relations between products are critical for recommendation and advertisement in e-commerce services. The relevant studies are categorized into two categories. The first category is to infer the missing substitutable and complementary relations. Reference [21] is the pioneering study in this regard. It proposes to combine review text and observable substitutable and complementary relations into a probabilistic modeling framework. Then the expectation-maximization algorithm is utilized to learn the model. References [40, 47] extend the objective functions to improve the performance of relation inference. Zhang et al. [45] also considered the review text of given two products, along with author set intersection and rating score difference, to judge the relations between the two products. Lately, graph neural networks have been applied by deeply investigating the nature of heterogeneous relation modeling [39] in this task [20]. All of the above studies are in parallel to our work and their progress could benefit the downstream tasks like the studied one.

The second category is substitutable and complementary product recommendation. Zhao et al. [50] constructed new objective functions based on the two types of relations to complement their recommendation loss. A majority of the studies in this field consider recommending complementary items for a given item or item package, which is practical in fashion compatibility. For example, McAuley et al. [21] first exploited image features to calculate the Mahalanobis distance between a query image and a candidate image. Through introducing a user-dependent matrix into the distance computation, personalization is considered to weigh each dimension of images. Reference [49] further expands the single image modality to multiple modalities (e.g., image, text, and user rating). Since it is hard to model the image objects only by the coarse-grained representations of whole images, fine-grained image modeling approaches are adopted to integrate multiple regions of images into relevance computation by attention mechanisms as well [11]. In addition, the study [18] learns the compositional visual coherence of an item collection. To sum up, all the above studies are attributed to the static relevance recommendation setting and incapable of learning user dynamic interests based on behavior sequences.

As discussed in Section 1, the most relevant study of this article is the seminal work [34]. However, due to the summarized limitations, there is still room to improve the techniques and performance, which are empirically demonstrated in the experiments later.

2.3 Graph Neural Networks for General Recommendation

Graph neural networks [5, 14, 32] become one of the milestones in the course of development for deep learning techniques. They are especially good at modeling high-order complex relations between graph nodes and converting them into low-dimensional dense representations. Since user-item interactions in recommender systems can be regarded as a user-item bipartite graph, graph neural networks could be naturally adapted to this scenario. For example, Wang et al. [38] developed GNNs for high-order propagation and adopted the **Bayesian personalized ranking (BPR)** loss [27]. To simplify the model architecture of GNNs, He et al. [8] removed the feature transformation and nonlinear activation in graph convolutional neural networks. Surprisingly, the simpler model even performs better than heavyweight GNN models.

Besides user-item interactions, item relations in **knowledge graphs (KG)** have also been explored. Based on users' interacted items, RippleNet [35] propagates user interests along the links in a knowledge graph. The nodes activated by the propagation process are used as recommendation candidates. Wang et al. [37] combined the conventional knowledge graph embedding approach, i.e., TransR [19], with graph neural networks for multi-task learning. In this hybrid model, knowledge-aware attention is devised to determine the type-specific importance of nodes in propagation. Moreover, KGs are utilized to provide explainable recommendation as well. For example, Xian et al. [44] leveraged reinforcement learning to generate interpretable reasoning paths in KGs. Another relation type commonly used by GNNs is social relations among users. Through representation propagation in social graphs, social-aware user representations could be obtained to benefit personalized prediction tasks [4, 30].

However, none of the above studies, including the aforementioned substitutable and complementary recommendation work, have considered leveraging the power of GNNs for modeling the substitutable and complementary relations for recommendation. In this article, we propose to utilize GNNs to learn from the substitutable and complementary relations for sequential recommendation.

3 PROBLEM FORMULATION AND NOTATIONS

In this section, we first formulate our problem of sequential product recommendation. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ denote a user set where N is the number of users and $\mathcal{P} = \{p_1, p_2, \ldots, p_M\}$ denote a product set where M is the number of products. For user u_n ($n \in \{1, 2, \ldots, N\}$), we define the interacted product sequence by $S_{u_n} = \{(p_1, t_1), (p_2, t_2), \ldots, (p_{\ell_n}, t_{\ell_n})\}$, where ℓ_n is the behavior sequence length. Since our work is based on the substitutable and complementary relations in e-commerce platforms, we assume there is a substitutable product-product graph $\mathcal{G}^s = \{\mathcal{P}, \mathcal{E}^s\}$ and a complementary product-product graph $\mathcal{G}^c = \{\mathcal{P}, \mathcal{E}^c\}$. The nodes (i.e., products) in the two graphs are allowed to have no edges. For ease of simplicity, we sometimes omit the subscripts and

Table 1. Some Key Mathematical Notations

Notation	Description
N	the total number of users
M	the total number of products
d	the dimension of latent representations
K	the total number of layers adopted by GNNs
$\boldsymbol{E}$	the product embedding matrix
$\mathcal{G}^s$ ($\mathcal{G}^c$)	the substitutable (complementary) product-product graph
$\boldsymbol{A}_s$ ($\boldsymbol{A}_c$)	the adjacency matrix of $\mathcal{G}^s$ ($\mathcal{G}^c$)
$\boldsymbol{h}$	the general form of a representation gotten from graph propagation
$\boldsymbol{U}_u$	the final representation of user u
$\boldsymbol{P}_p$	the final representation of product p
$\alpha^k_{s,p}$ ($\alpha^k_{c,p}$)	the gating weight w.r.t. product p and the kth propagation layer in $\mathcal{G}^s$ ($\mathcal{G}^c$)
β_s (β_c)	the attention weights of temporal kernel functions for the substitutable (complementary) relation
Θ	the trainable parameters of SCG-SPRe

superscripts related to specific users and items if not otherwise specified. Without loss of generality, we formulate the studied research problem as follows:

Problem 1 (Sequential Product Recommendation). *Given a target user u and a candidate product p, the aim of this problem is to learn a model f that computes the ranking score denoting the possibility $\hat{y}$ of their interaction shortly, which is defined as*

$$\hat{y} = f(u, p | S_u, \mathcal{G}^s, \mathcal{G}^c; \Theta),$$

where Θ contains all the trainable parameters.

Before proceeding, we briefly introduce the rules for notations throughout this article. We use bold upper case letters to denote matrices (e.g., $\boldsymbol{A}$), bold lower case letters to represent vectors (e.g., $\boldsymbol{h}$), and normal lower case letters to indicate scalars (e.g., α). We summarize some key notations in Table 1 and all the left symbols are defined when they first appear.

4 COMPUTATIONAL METHODOLOGY

In this section, we first give an overview of the proposed model. Then we describe the model's computational details, especially for the two main modules. Finally, we show how to train the model end-to-end.

4.1 Model Overview

Figure 2 depicts the architecture of the proposed SCG-SPRe using a toy example. Overall, SCG-SPRe contains two key modules: interactive graph neural networks and kernel-enhanced transformer networks. SCG-SPRe first projects products into low-dimensional dense representations and then propagates them in the substitutable product-product graph and the complementary product-product graph through the interactive graph neural networks. After propagation, the two types of relation-specific product representations are fed into the kernel-enhanced transformer networks. For each relation type, the module calculates sequence-aware representations and kernel-based representations for a given candidate product, and further composes them to get relation-specific user representations. At last, the two types of relation-specific representations are fused together to denote user interests and calculate ranking scores. In the rest of this section, we mainly

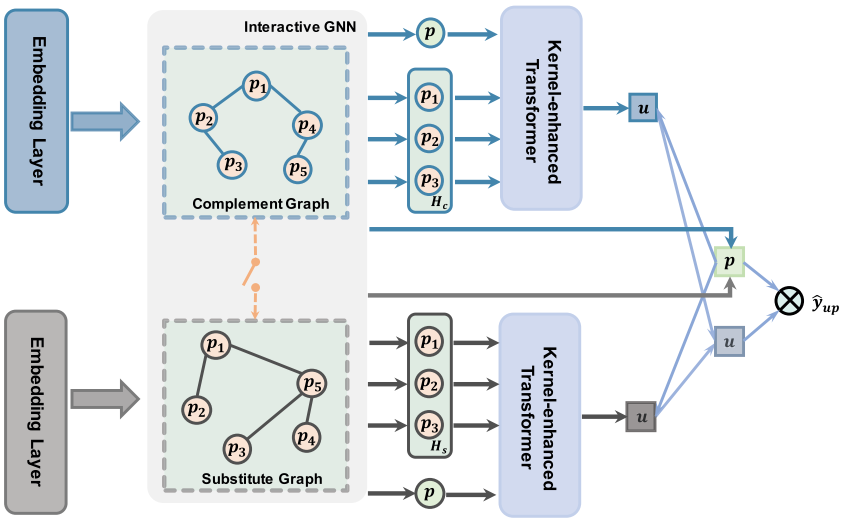


Fig. 2. Illustration of the architecture of SCG-SPRe for a toy example, which includes five products in the graphs and sets the length of the user behavior sequence to 3. The switch between the substitutable product-product graph and the complementary product-product graph means their representation propagation is coupled.

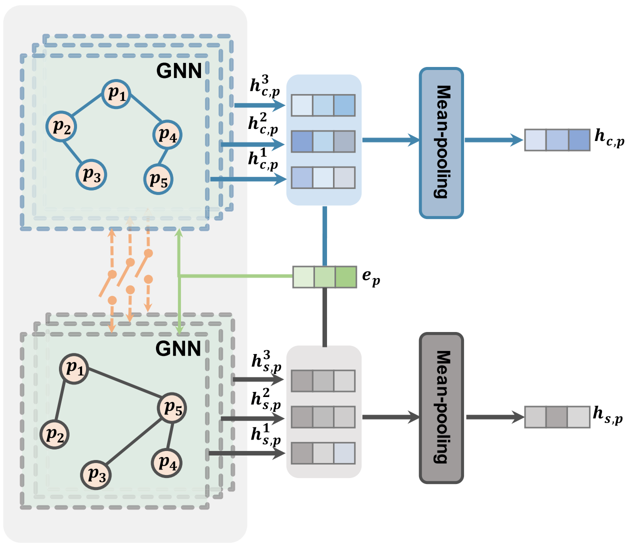


Fig. 3. Illustration of the interactive graph neural networks, where the propagation layer number is supposed to be 3.

exploit user u as a target user and product p as a candidate product to specify the computational details of the model.

4.2 Interactive Graph Neural Networks

Figure 3 provides the main workflow of the interactive graph neural networks. Before using it to propagate product representations, we first get the low-dimensional dense representations by converting the one-hot representations of products. Specifically, we define a trainable embedding matrix $\boldsymbol{E} \in \mathbb{R}^{d \times M}$ and adopt the following mapping function:

$$\boldsymbol{e}_p = \boldsymbol{E}\boldsymbol{o}_p \qquad\qquad p \in \{1, 2, \ldots, M\}, \tag{1}$$

where $\boldsymbol{o}_p$ is the one-hot representation of product p.

Based on the known substitutable and complementary relations, we can easily construct the two graphs, i.e., $\mathcal{G}^s$ and $\mathcal{G}^c$. The motivation behind the usage of the two graphs instead of constructing a heterogeneous graph is that we want to obtain the disentangled relation-specific representations for each product, which are more suitable for different types of kernels in the second module. For the two graphs, we define the corresponding adjacency matrices, i.e., $\boldsymbol{A}_s$ and $\boldsymbol{A}_c$. An entry in the two matrices takes a value of 1 if the corresponding relation holds, and otherwise 0.

Then, we describe the computational details of representation propagation. We assume the product representations of product p after propagating k layers are $\boldsymbol{h}_{s,p}^k$ and $\boldsymbol{h}_{c,p}^k$ for the two graphs, respectively. The initializations of the substitutable relation-specific representation $\boldsymbol{h}_{s,p}^k$ and the complementary relation-specific representation $\boldsymbol{h}_{c,p}^k$ share the same embedding by setting $\boldsymbol{h}_{s,p}^0 = \boldsymbol{h}_{c,p}^0 = \boldsymbol{e}_p$. The representation propagation in each layer is decomposed into two procedures: representation aggregation and representation updating. Since the procedures for the two types of representations are consistent with each other, we only take $\boldsymbol{h}_{s,p}^k$ for illustration due to space consideration. The formulas of $\boldsymbol{h}_{c,p}^k$ could be derived similarly.

First, we formulate the formula for representation aggregation as follows:

$$\bar{\boldsymbol{h}}_{s,p}^k = \sum_{j \in \mathcal{N}_s(p)} \hat{\boldsymbol{A}}_s[p,j] \cdot \boldsymbol{h}_{s,j}^{k-1} + \tanh\left(\hat{\boldsymbol{A}}_s[p,j] \cdot \boldsymbol{h}_{s,j}^{k-1}\right) \odot \left(\boldsymbol{h}_{s,j}^{k-1} + \boldsymbol{r}_s\right) + \left(\hat{\boldsymbol{A}}_s[p,j] \cdot \boldsymbol{h}_{s,j}^{k-1} + \boldsymbol{r}_s\right) \odot \tanh\left(\boldsymbol{h}_{s,j}^{k-1}\right), \tag{2}$$

where $\hat{\boldsymbol{A}}_s = \boldsymbol{D}_s^{-\frac{1}{2}} \boldsymbol{A}_s \boldsymbol{D}_s^{-\frac{1}{2}}$ is the row-wise normalized adjacency matrix without self-loops, and $\boldsymbol{D}_s$ is a diagonal matrix with $\boldsymbol{D}_s[i,i]$ denoting the summation of the ith row of $\boldsymbol{A}_s$. $\boldsymbol{r}_s$ is the trainable embedding of the substitutable relation. $\mathcal{N}_s(p)$ is the neighbor set of product p. To reveal the motivation of the formula, we go deeper into its right part. The first term $\hat{\boldsymbol{A}}_s[p,j] \cdot \boldsymbol{h}_{s,j}^{k-1}$ denotes the common idea of combining neighbor node representations. The second and third terms encode the affinities between the translation-based representations (e.g., $\boldsymbol{h}_{s,j}^{k-1} + \boldsymbol{r}_s$ and $\hat{\boldsymbol{A}}_s[p,j] \cdot \boldsymbol{h}_{s,j}^{k-1} + \boldsymbol{r}_s$) and the neighbor representations (e.g., $\hat{\boldsymbol{A}}_s[p,j] \cdot \boldsymbol{h}_{s,j}^{k-1}$ and $\boldsymbol{h}_{s,j}^{k-1}$). And larger affinity values enable more message passing from neighbors. This conforms to the idea of TransE [2] that the translated head representation should be similar to the corresponding tail representation and could boost the performance to a certain extent (shown in Section 5.4). Since we assume the relations to be undirectional, we use the two terms together to achieve this.

Second, we perform representation updating based on the aggregated representations by Equation (2). We attempt to encode some complex dependencies across the substitutable and complementary relations. Therefore, we define the following formula, which enables the updating to rely on two relations:

$$\boldsymbol{h}_{s,p}^k = \left(1 - \alpha_{s,p}^k\right) \cdot \bar{\boldsymbol{h}}_{s,p}^k + \alpha_{s,p}^k \cdot T_{c->s}\left(\bar{\boldsymbol{h}}_{c,p}^k; \theta_s^T\right), \tag{3}$$

where $T_{c->s}(\cdot; \theta_s^T)$ is a specified transformation function. It maps the representations w.r.t. the complementary relation to the same latent semantic space of the substitutable relation. θ_s^T is the trainable parameters of the function. $\alpha_{s,p}^k$ is the gating weight to flexibly control the information flow between these two relation types, which is given as follows:

$$\alpha_{s,p}^k = \sigma\left(\boldsymbol{W}_s^k \left[\bar{\boldsymbol{h}}_{s,p}^k \oplus T_{c->s}\left(\bar{\boldsymbol{h}}_{c,p}^k; \theta_s^T\right)\right]\right), \tag{4}$$

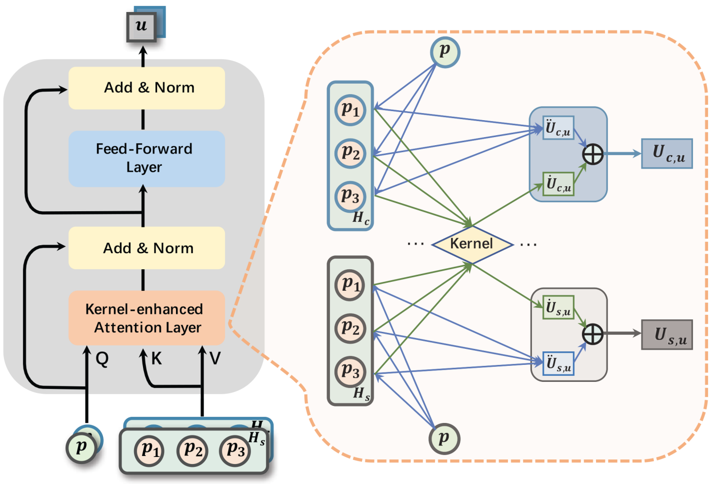


Fig. 4. Illustration of kernel-enhanced transformer networks. K_1, K_2, and K_3 denote different types of kernel functions, where we omit their superscripts w.r.t. different relations.

where σ is the sigmoid function, $\oplus$ denotes the concatenation of the corresponding representations, and $\boldsymbol{W}_s^k$ is a parameter matrix to be learned. Although this gating mechanism is simple, it allows to selectively transfer useful information from one relation to another one.

After propagating representations with K layers, we collect multi-layer representations for each product. Since the representations obtained in different layers emphasize the message passed over different connections, each of them might contain some different information. Consequently, we perform the mean-pooling operation to combine them, which is defined as follows:

$$\boldsymbol{h}_{s,p} = \text{Mean-pooling}\Big([\boldsymbol{h}_{s,p}^0 \oplus \boldsymbol{h}_{s,p}^1 \oplus \cdots \oplus \boldsymbol{h}_{s,p}^K]\Big), \tag{5}$$

where $\boldsymbol{h}_{s,p}$ is the output by the module, as shown in Figure 3. It is worth noting that the product representation $\boldsymbol{h}_{c,p}$ w.r.t. the complementary relation could be similarly obtained according to the above formulas. In all, the trainable parameters involved in the first module are summarized as $\Theta_1 = \{\boldsymbol{E}, \boldsymbol{r}_s, \boldsymbol{r}_c, \boldsymbol{W}_s^k, \boldsymbol{W}_c^k, \theta_s^T, \theta_c^T\}$.

4.3 Kernel-enhanced Transformer Networks

Figure 4 illustrates the main workflow of the module, i.e., kernel-enhanced transformer networks, which takes the candidate product p and the behavior sequence $S_u = \{(p_1, t_1), (p_2, t_2), \ldots, (p_{\ell_u}, t_{\ell_u})\}$ of user u as input. Since we have two relation-specific representations for each product by the previous module, we get two sequences of product representations, i.e., $\boldsymbol{H}_s = [\boldsymbol{h}_{s,1}, \boldsymbol{h}_{s,2}, \ldots, \boldsymbol{h}_{s,\ell_u}] \in \mathbb{R}^{d \times \ell_u}$ and $\boldsymbol{H}_c = [\boldsymbol{h}_{c,1}, \boldsymbol{h}_{c,2}, \ldots, \boldsymbol{h}_{c,\ell_u}] \in \mathbb{R}^{d \times \ell_u}$. They correspond to the substitutable relation and the complementary relation, respectively.

For the candidate product p, we first perform the embedding lookup operation, similar to Equation (1), to have the basic representation $\boldsymbol{e}_p$. To facilitate the learning of candidate-dependent user representations based on the above two types of representation sequences, we add the relation embeddings to the candidate product representation, which is given by

$$\boldsymbol{e}_{s,p} = \boldsymbol{r}_s + \boldsymbol{e}_p \qquad \boldsymbol{e}_{c,p} = \boldsymbol{r}_c + \boldsymbol{e}_p. \tag{6}$$

According to the commonly used transformation operations for the input representations in transformer-like approaches, we define the shared transformation matrices, i.e., $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and

$\boldsymbol{W}^V$, for the two relations. To learn candidate-dependent user representations, we regard candidate product representations as queries and product representations in user sequences as keys and values. After performing the transformation, we get query embeddings $\boldsymbol{q}_{s,p}$ and $\boldsymbol{q}_{c,p}$ for the product, key matrices $\boldsymbol{K}_s$ and $\boldsymbol{K}_c$ for the behavior sequence, as well as the corresponding value matrices $\boldsymbol{V}_s$ and $\boldsymbol{V}_c$. Besides, we also add position embeddings to the two representation sequences to capture the order information.

As shown in the right part of Figure 4, the kernel-enhanced attention layer in the transformer networks learns two categories of user representations w.r.t. different relations: kernel-based representations (shown with the color of light green) and sequence-aware representations (shown with the color of light blue). In what follows, we elaborate each category.

4.3.1 Kernel-based Representations. As discussed in Section 1, the substitutable relation and the complementary relation between two products exhibit different temporal patterns. This is because a consumer tends to purchase only one product from a pair of substitutes in a short period. On the contrary, the consumer probably buys one product in conjunction with its complementary product. To enable our model to learn such temporal patterns, we resort to temporal kernel functions to measure the time difference from a continuous value space.

Compared to the work [34], we aim to enhance the expressive ability of characterizing temporal dynamics by (1) providing multiple types of temporal kernel functions and (2) developing an adaptive mechanism to fuse them. Without loss of generality, we consider three types of kernel instances, i.e., Gaussian kernel, exponential kernel, and logarithmic-decay kernel, which are given as follows:

$$\mathrm{K}_1(\mu,\sigma) = N(\Delta_t|\mu,\sigma) \qquad \mathrm{K}_2(\lambda) = Exp(\Delta_t|\lambda) \qquad \mathrm{K}_3(a,b) = a\log(1+\Delta_t) + b, \tag{7}$$

where Δ_t denotes the continuous-time difference. $N(\Delta_t|\mu,\sigma)$ is a normal distribution with the mean of μ and the standard deviation of σ. $Exp(\Delta_t|\lambda)$ represents the exponential distribution with the parameter of λ. a and b are the parameters of the logarithmic-decay kernel.

Based on the three general forms of temporal kernels, we first define the concrete kernels for the complementary relation as follows:

$$\mathrm{K}_1^c = \mathrm{K}_1(0,\sigma_c) \qquad \mathrm{K}_2^c = \mathrm{K}_2(\lambda_c) \qquad \mathrm{K}_3^c = \mathrm{K}_3(a_c, b_c), \tag{8}$$

where σ_c, λ_c, a_c, and b_c are trainable parameters, which could be readily extended when the categories of products are given. For the substitutable relation, we follow the work [34] to emphasize its inhibitory effect by giving the following combination forms:

$$\mathrm{K}_1^s = -\mathrm{K}_1(0,\sigma_{s1}) + \mathrm{K}_1(\mu_s,\sigma_{s2}) \qquad \mathrm{K}_2^s = -\mathrm{K}_2(\lambda_{s1}) + \mathrm{K}_2(\lambda_{s2}) \qquad \mathrm{K}_3^s = -\mathrm{K}_3(a_{s1},b_{s1}) + \mathrm{K}_3(a_{s2},b_{s2}). \tag{9}$$

Take the kernel-based representation computation w.r.t. the substitutable relation for illustration. Now, we describe how to gain the kernel-based user representation by the following formula:

$$\dot{\boldsymbol{U}}_{s,u} = \sum_{(p',t')\in S_u} \boldsymbol{A}_s[p,p'] \left(\sum_{z=1}^{3} \beta_s[z] \mathrm{K}_z^s(t-t') \right) \cdot \boldsymbol{V}_s[p'], \tag{10}$$

where t is the time when the recommendation generation will happen. $\boldsymbol{V}_s[p']$ denotes the representation of product p' in the value matrix. $\beta_s[z]$ represents the importance weight of the zth kernel function dependent on product p'. To be specific, we compute the attention distribution β_s over different kernel functions by

$$\beta_s = \mathrm{Softmax}\Big(\boldsymbol{W}_s(\boldsymbol{q}_{s,p} + \boldsymbol{K}_s[p'])\Big), \tag{11}$$

where $\boldsymbol{W}_s$ is a trainable parameter matrix with the row size equal to the number of the kernel functions. $\boldsymbol{K}_s[p']$ corresponds to the representation of the key w.r.t. product p'. Through the above manner, the impact of different temporal kernel functions is associated with specific product-product pairs, which is more flexible and adaptive. In a similar fashion, we can get the kernel-based user representation $\dot{\boldsymbol{U}}_{c,u}$ w.r.t. the complementary relation. To sum up, the above computational manner encodes the explicit knowledge of the substitutable and complementary relations into the kernel-based representations. Moreover, the fused kernel functions could partially explain why the candidate product recommendation is generated.

4.3.2 Sequence-aware Representations. Although kernel-based representations provide a knowledge-driven perspective to characterize user interests, they cannot fully utilize all the interacted products in a user sequence. This is because many pairs of products do not have substitutable and complementary relations, which is verified in Table 2. As shown in Equation (10), when $\boldsymbol{A}_s[p,p']$ is equal to 0, the kernel-based representations cannot leverage the representation of product p'. Therefore, we aim to learn sequence-aware user representations to work together with the kernel-based representations.

To achieve this, we adopt attention computation over all the interacted products in a behavior sequence, which is verified to be effective in deep sequential recommendation models [12, 16]. Consequently, the sequence-aware user representation $\ddot{\boldsymbol{U}}_{s,u}$ w.r.t. the substitutable relation is given by

$$\ddot{\boldsymbol{U}}_{s,u} = \mathrm{Softmax}\left(\frac{\boldsymbol{q}_{s,p}^\top \boldsymbol{K}_s}{\sqrt{d}}\right)\boldsymbol{V}_s. \tag{12}$$

Similarly, we have the sequence-aware user representation $\ddot{\boldsymbol{U}}_{c,u}$ w.r.t. the complementary relation.

Based on the obtained kernel-based representations and sequence-aware representations, we combine them together to form two relation-specific user representations, which are given by

$$\boldsymbol{U}_{s,u} = \dot{\boldsymbol{U}}_{s,u} + \ddot{\boldsymbol{U}}_{s,u} \qquad \boldsymbol{U}_{c,u} = \dot{\boldsymbol{U}}_{c,u} + \ddot{\boldsymbol{U}}_{c,u}. \tag{13}$$

We further add **layer normalization (LayerNorm)** and **feed-forward neural (FFN)** networks, as shown in Figure 4. Take $\boldsymbol{U}_{s,u}$ as an example and the computational formulas are represented as follows:

$$\bar{\boldsymbol{U}}_{s,u} = \mathrm{LayerNorm}(\boldsymbol{e}_{s,p} + \boldsymbol{U}_{s,u}), \tag{14}$$

$$\hat{\boldsymbol{U}}_{s,u} = \mathrm{LayerNorm}(\bar{\boldsymbol{U}}_{s,u} + \mathrm{FFN}(\bar{\boldsymbol{U}}_{s,u})). \tag{15}$$

And analogously, we can get $\hat{\boldsymbol{U}}_{c,u}$. At last, we fuse the updated two relation-specific user representations by relational attention computation. As such, the full candidate-dependent user representation is obtained, which is defined as follows:

$$\boldsymbol{U}_u = \mathrm{Softmax}\left(\frac{[\hat{\boldsymbol{U}}_{s,u}; \hat{\boldsymbol{U}}_{c,u}]^\top \boldsymbol{P}_p}{\sqrt{d}}\right)[\hat{\boldsymbol{U}}_{s,u}; \hat{\boldsymbol{U}}_{c,u}], \tag{16}$$

where $\boldsymbol{P}_p$ is the updated representation of product p represented as $\boldsymbol{P}_p = (\boldsymbol{e}_{s,p} + \boldsymbol{e}_{c,p})/2$. It encodes the substitutable and complementary relations. For convenience, we use Θ_2 to cover all the trainable parameters involved in this module.

4.4 Model Prediction and Training

Once the user representation $\boldsymbol{U}_u$ and the product representation $\boldsymbol{P}_p$ are obtained, we use them to generate the ranking score $\hat{y}_{up}$ for the user-product pair, which is given by

$$\hat{y}_{up} = \boldsymbol{U}_u^\top \boldsymbol{P}_p + b_u + b_p, \tag{17}$$

ALGORITHM 1: End-to-end learning of SCG-SPRe.

```
   Input: User set $\mathcal{U}$, product set $\mathcal{P}$, user behavior sequences, and product-product graphs $\mathcal{G}^s$ and $\mathcal{G}^c$.
   Output: Optimized $f(\Theta)$ of the model SCG-SPRe.
1  Initialize the model $f(\Theta)$;
2  for number of training steps do
3      Sample a minibatch $B = \{(u, p, \mathcal{N}_{S_u})\}$ wherein each tuple corresponds to a target user, a candidate
       product (ground-truth for training), and a set of negative products for this interaction;
4      for $(u, p, \mathcal{N}_{S_u})$ in minibatch $B$ do
5          # Interactive graph neural networks
6          Gain dense product representations by look-up table (Equation (1));
7          Perform representation propagation to obtain candidate product representations, and other
             negative product representations (e.g., $\boldsymbol{h}_{s,p}$ and $\boldsymbol{h}_{c,p}$ by Equation (5));
8          # Kernel-enhanced transformer networks
9          Compute the kernel-based user representations for the products involved in the tuple (e.g.,
             $\dot{\boldsymbol{U}}_{s,u}$ and $\dot{\boldsymbol{U}}_{c,u}$ by Equation (10));
10         Calculate the sequence-aware user representations for the products involved in the tuple (e.g.,
             $\ddot{\boldsymbol{U}}_{s,u}$ and $\ddot{\boldsymbol{U}}_{c,u}$ by Equation (12));
11         Obtain the candidate-dependent user representations (e.g., $\boldsymbol{U}_u$ by Equation (16)) and product
             representation $\boldsymbol{P}_p$;
12         # Model loss
13         Calculate $\mathcal{L}(\Theta)$ (Equation (18)) and accumulate the gradients w.r.t. $\Theta$;
14     end
15     Update parameter weights $\Theta$ by Adam with the accumulated gradients;
16 end
```

where b_u and b_p correspond the ranking bias of the user and the product, respectively. To generate the top-K recommendation list each time, we just need to choose the products that have the largest K scores.

The target of model training is to optimize the trainable parameters of our model, which can be represented by $\Theta = \{\Theta_1, \Theta_2, (b_u)_{u=1}^{N}, (b_p)_{p=1}^{M}\}$. Since we know the ground-truth product for a given user sequence in training datasets, we utilize the BPR loss [27] by sampling a set of products as negative ones to the ground-truth product. To be specific, we assume the candidate product p to be the ground-truth and $\mathcal{N}_{S_u}$ to be the sampled negative product set. Then, we define the following loss function w.r.t. the specific user behavior sequence:

$$\mathcal{L}(\Theta) = -\sum_{p' \in \mathcal{N}_{S_u}} \log \sigma(\hat{y}_{up} - \hat{y}_{up'}). \tag{18}$$

The above loss function is readily extended to the real training scenario that considers all the behavior sequences for a given training dataset.

In the end, we adopt Adam, a gradient-based optimization method, to learn model parameters. The concise training procedure is summarized in Algorithm 1. The computational cost in one training step mainly derives from the two key modules. For the module of interactive graph neural networks, the dominant computation consumption in each layer lies in representation aggregation and representation updating for all products. On the one hand, the time complexity for representation aggregation is $O(M^2 d)$. Actually, the substitutable product-product graph and complementary product-product graph are sparse, and efficient sparse matrix multiplication could be exploited. Therefore the complexity is reduced to $O(|\mathcal{E}^s|d + |\mathcal{E}^c|d)$. On the other hand, the time complexity

Table 2. Statistics of the Three Experimental Datasets

Dataset	#Users	#Items	#Interactions	Ratio of substitute pairs	Ratio of complement pairs
Grocery	14,681	8,457	145,832	0.174%	0.346%
Baby	19,446	6,947	159,628	0.233%	0.433%
Cellphones	27,879	10,330	193,228	0.012%	0.220%

of representation updating is $O(Md^2)$. By considering the total number of layers, the time complexity of this module is $O(K(|\mathcal{E}^s|d + |\mathcal{E}^c|d + Md^2))$. For the module of kernel-enhanced transformer networks, the dominant time cost is composed up by the computation of the kernel-based representation and sequence-aware representation, which is summarized as $O(|\mathcal{D}|(\ell_u d + d^2))$. $|\mathcal{D}|$ denotes the total number of training examples, including the sampled negative ones. In total, the **time complexity** of SCG-SPRe is $O(K(|\mathcal{E}^s|d + |\mathcal{E}^c|d + Md^2) + |\mathcal{D}|(\ell_u d + d^2))$.

5 EXPERIMENTS

In this section, we conduct extensive experiments to validate the effectiveness of our proposed SCG-SPRe by answering the following pivotal research questions:

RQ1. How are the results of SCG-SPRe compared with other competitive sequential recommendation models?
RQ2. How do the main model components of SCG-SPRe affect its recommendation performance?

Besides, we investigate the effect of different hyper-parameter settings (e.g., the layer number for representation propagation, the dimension of representations, and the maximal length of behavior sequences). The results on fine-grained partitions of test sets are presented as well to show how our model performs on sequences with different length, and users and items with different numbers of known interactions. Finally, we conduct case studies to show the curves of the fused kernels, visualize the attention weight distributions in the interactive graph neural networks, and analyze the training time cost and the performance of different temporal kernels.

5.1 Experimental Datasets

To make the performance evaluation of our model and competitors reliable, we choose the publicly accessible and widely used Amazon dataset [22]. The user-item interaction records generated by the user rating behavior are taken into consideration. We further extract the data examples that belong to the categories of *Grocery*, *Baby*, and *Cellphones* to build three experimental datasets, wherein the products that have no interaction records are removed. The statistics of the three experimental datasets are presented in Table 2. Following previous studies [21, 34], we construct the substitutable and complementary product-product pairs based on the lists w.r.t. “also_view” and “also_buy.” As shown in Table 2, the numbers of substitute pairs and complement pairs are relatively small compared to the total numbers of product-product pairs.

5.2 Experimental Setups

5.2.1 Baseline Methods. We compare our SCG-SPRe model with representative non-sequential recommendation methods and a variety of deep sequential recommendation models.

- **BPR** [27]: This method endows matrix factorization with the BPR loss, which is suitable for the ranking situation of recommender systems.
- **GMF** [9]: This is a pioneering model that combines deep learning with collaborative filtering for non-sequential recommendation.
- **Tensor** [13]: The model extends user-item matrix factorization used in the above two methods to user-item-time tensor factorization.

- **GRU4Rec** [10]: It is a deep sequential model that applies recurrent neural networks to sequential recommendation. User behavior sequences consisting of interacted items are taken as input.
- **NARM** [16]: An attention mechanism is combined with recurrent neural networks in this model to differentiate the importance of different interacted items.
- **SASRec** [12]: This model leverages the power of transformer networks to generate user representations from behavior sequences.
- **TiSASRec** [17]: Time interval information between two interacted items in a behavior sequence is incorporated into its transformer-based recommendation model.
- **SLRC** [33]: This model utilizes the intensity function of Hawkes process [7] to capture the temporal dynamics for computing the ranking scores of user-item pairs.
- **CFKG** [48]: This method utilizes knowledge base embeddings for personalized recommendation, which could provide another perspective of modeling the substitutable and complementary relations with knowledge graph techniques.
- **Chorus** [34]: This model designs temporal kernel functions for different relations to explicitly characterize the complex dependencies between interacted items and candidate items.

5.2.2 Evaluation Protocols. To keep the same setting as related sequential recommendation models [9, 34], we adopt the leave-one-out evaluation strategy as well. It is implemented by taking the most recent product of each user behavior sequence for testing, the second recent product for validation, and the remaining interacted products for training. Since it is a heavy burden to enumerate all products to evaluate one recommendation of a given user sequence, we exploit the product sampling strategy [9, 34]. In particular, it samples 99 products that a target user does not interact with, together with the ground-truth product, for evaluation.

We evaluate the recommendation performance of all the adopted models by **Hit Ratio (HR@K)** and **Normalized Discounted Cumulative Gain (NDCG@K)**, two popular ranking metrics extensively used in existing sequential recommendation work [34, 36].

- **HR@K:** A recommended product is counted as one hit if it satisfies not only to be the ground-truth product but also to be ranked within the top k position of the corresponding ranking list. Since in the leave-one-out evaluation setting, the number of ground-truth product is equal to the number N_{te} of tested users, we define the computational formula of HR@K as follows:
$$HR@K = \frac{n_{hit-k}}{N_{te}},$$
where n_{hit-k} denotes the count of hits.
- **NDCG@K:** This is a position-aware metric that assigns more weights to the correct products that rank higher in a recommendation list, which is given by
$$NDCG@K = Z \cdot \sum_{u=1}^{N_{te}} \sum_{j=1}^{K} \frac{2^{r_j} - 1}{\log_2 (j+1)},$$
where Z is a normalization factor. r_j takes value of 1 if the jth recommendation is correct, and otherwise 0.

For detailed comparisons, HR@5, HR@10, NDCG@5, and NDCG@10 are used in the experiments. We report the average results by repeating each experiment five times.

5.2.3 Model Implementation. We implement our model by Pytorch and deploy it on a Linux server with GPUs of Nvidia GeForce GTX 1080 Ti (11G memory). The model is learned in a mini-batch fashion with a size of 256. For the adopted Adam optimizer, the learning rate is set to 5e-4

Table 3. Performance Comparison between Different Models

Dataset	Method	HR@5	HR@10	NDCG@5	NDCG@10
Grocery	BPR [27]	0.3591	0.4689	0.2483	0.2840
	GMF [9]	0.3195	0.4269	0.2202	0.2551
	Tensor [13]	0.3544	0.4498	0.2669	0.2977
	GRU4Rec [10]	0.3641	0.4693	0.2579	0.2920
	NARM [16]	0.3627	0.4669	0.2592	0.2928
	SASRec [12]	0.4249	0.5312	0.3046	0.3391
	TiSASRec [17]	0.4290	0.5327	0.3085	0.3421
	SLRC [33]	0.4516	0.5645	0.3333	0.3700
	CFKG [48]	0.4227	0.5482	0.3008	0.3415
	Chorus [34]	<u>0.4773</u>	<u>0.6028</u>	<u>0.3475</u>	<u>0.3882</u>
	Ours (SCG-SPRe)	**0.5145**	**0.6141**	**0.3767**	**0.4091**
	Improv.	7.79%	1.87%	8.40%	5.38%
Baby	BPR [27]	0.2256	0.3399	0.1501	0.1869
	GMF [9]	0.2179	0.3283	0.1434	0.1789
	Tensor [13]	0.2578	0.3889	0.1719	0.2141
	GRU4Rec [10]	0.2926	0.4192	0.1988	0.2397
	NARM [16]	0.2899	0.4156	0.1968	0.2373
	SASRec [12]	0.3053	0.4313	0.2092	0.2498
	TiSASRec [17]	<u>0.3162</u>	<u>0.4451</u>	<u>0.2170</u>	<u>0.2586</u>
	SLRC [33]	0.2980	0.4156	0.2067	0.2444
	CFKG [48]	0.2732	0.3972	0.1842	0.2242
	Chorus [34]	0.2805	0.4060	0.1932	0.2336
	Ours (SCG-SPRe)	**0.3569**	**0.4857**	**0.2504**	**0.2920**
	Improv.	12.87%	9.12%	15.39%	12.92%
Cellphones	BPR [27]	0.3618	0.4716	0.2639	0.2994
	GMF [9]	0.2879	0.3889	0.2060	0.2385
	Tensor [13]	0.3641	0.4965	0.2549	0.2976
	GRU4Rec [10]	0.4119	0.5465	0.2946	0.3381
	NARM [16]	0.4105	0.5471	0.2943	0.3384
	SASRec [12]	0.4663	0.5893	0.3462	0.3860
	TiSASRec [17]	<u>0.4705</u>	<u>0.5939</u>	<u>0.3504</u>	<u>0.3903</u>
	SLRC [33]	0.4445	0.5416	0.3436	0.3749
	CFKG [48]	0.4446	0.5667	0.3257	0.3653
	Chorus [34]	0.4575	0.5774	0.3417	0.3805
	Ours (SCG-SPRe)	**0.5413**	**0.6588**	**0.4162**	**0.4542**
	Improv.	15.05%	10.93%	18.78%	16.37%

Improv. denotes the relative improvements over the second-best results, which are underlined.

and the other hyper-parameters are kept by default. Unless otherwise stated, we set the embedding dimension of all the relevant models to 64 for fair comparisons. We let the number of representation propagation layers to be 4 by grid search in $\{1, 2, \ldots, 5\}$. The maximal length of user behavior sequences is set to 20 by searching in the range from 1 to 20. We discuss the effect of some key hyper-parameter settings later.

5.3 Model Comparison (RQ1)

To demonstrate the effectiveness of the proposed SCG-SPRe, we first compare it with the adopted baselines. The overall results are shown in Table 3, from which we have the following observations:

- ⋄ Compared to other deep sequential recommendation models, the first three non-sequential recommendation models, i.e., BPR, GMF, and Tensor, achieve relatively poor performance on the three datasets. This phenomenon conforms to our expectation, since they do not aim at capturing user dynamic preference based on behavior sequences. Instead, they formulate the recommendation task as a user-item matrix completion problem. Although Tensor additionally considers the temporal dimension, each user-item pair is still modeled independently. Therefore, they are not very suitable for the sequential product recommendation setting.
- ⋄ GRU4Rec and NARM obtain better results than the non-sequential recommendation models, showing that sequential modeling is indeed beneficial for the studied task. By further comparing the two models themselves, we find that NARM only reaches the same performance level as GRU4Rec, although NARM models users' recent interests through a local encoder with attention computation. The reason might be that the local encoder also heavily relies on the current output representation of RNNs like the used global encoder. As such, there is a need to consider more advanced attention computation to promote performance.
- ⋄ SASRec and TiSASRec are transformer-like recommendation models. Due to the strong power of attention computation used in transformer, they perform significantly better than the aforementioned models. In particular, TiSASRec even outperforms SASRec across all the datasets. It makes sense, because TiSASRec is an enhanced version of SASRec by incorporating time interval information between interacted items into attention computation. By contrast, SLRC considers the temporal difference between the timestamp of an interaction and the time when generating recommendations. As such, their performance on multiple datasets is very different.
- ⋄ CFKG behaves not well on the experimental datasets, indicating that simply utilizing knowledge base techniques to model the substitutable and complementary relations is not satisfied. Chorus yields better performance than CFKG. Such improvement might be attributed to the designed kernel functions for different relations and better representations. However, as introduced in Section 1, Chorus still has several limitations that restrict its performance. As we can see, Chorus underperforms TiSASRec on the Baby and Cellphones datasets, while performing better than TiSASRec on Grocery.
- ⋄ The proposed SCG-SPRe significantly outperforms all the baselines on the three datasets. In particular, SCG-SPRe achieves 0.5145 in terms of HR@5 on Grocery, 7.8% higher than the best baseline Chorus. On the Baby and Cellphones datasets, SCG-SPRe improves the strongest baseline TiSASRec by 15.4% and 18.8% w.r.t. NDCG@5, respectively. The success of SCG-SPRe could be attributed to its key model components, including the interactive graph neural networks, the adaptive fusion of multiple kernels, and the combination of sequence-aware and kernel-based representations, and so on. In the next part, the positive contributions of these components will be validated.

5.4 Ablation Study (RQ2)

We conduct an ablation study to demonstrate the rationality and effectiveness of the main components involved in SCG-SPRe by the following model variants, each of which has only one component removed or changed and retains the other components to be the same as the full model.

- **w/o interactive GNN:** We remove the module of the interactive GNNs by directly feeding the embeddings gotten from Equation (1) into the module of the kernel-enhanced transformer networks.
- **r/ adaptive kernels -> Chorus kernels:** We replace the adaptive fusion of the multiple temporal kernels (from Equation (7) to (11)) with the same kernel learning method used by Chorus [34].

Table 4. Ablation study of GDSM-SRe

Dataset	Method	HR@5	HR@10	NDCG@5	NDCG@10
Grocery	w/o interactive GNN	0.4513	0.5442	0.3285	0.3587
	r/ adaptive kernels -> Chorus kernels	0.4724	0.5759	0.3424	0.3760
	w/o kernel-based representations	0.4280	0.5412	0.3003	0.3369
	w/o graph interaction	0.5039	0.6083	0.3731	0.4072
	r/ graph gate -> addition	0.5057	0.6124	0.3697	0.4043
	r/ graph gate -> co-attention	0.4892	0.5920	0.3554	0.3889
	w/o graph translation affinity	0.5028	0.6054	0.3726	0.4060
	r/ relation attention -> addition	0.5093	0.6051	0.3730	0.4054
	Ours (SCG-SPRe)	**0.5145**	**0.6141**	**0.3767**	**0.4091**
Baby	w/o interactive GNN	0.3013	0.4227	0.2056	0.2463
	r/ adaptive kernels -> Chorus kernels	0.3483	0.4750	0.2452	0.2861
	w/o kernel-based representations	0.3335	0.4642	0.2324	0.2745
	w/o graph interaction	0.3528	0.4802	0.2468	0.2879
	r/ graph gate -> addition	0.3525	0.4840	0.2473	0.2903
	r/ graph gate -> co-attention	0.3444	0.4699	0.2435	0.2839
	w/o graph translation affinity	0.3478	0.4721	0.2451	0.2852
	r/ relation attention -> addition	0.3283	0.4612	0.2300	0.2730
	Ours (SCG-SPRe)	**0.3569**	**0.4857**	**0.2504**	**0.2920**
Cellphones	w/o interactive GNN	0.4760	0.5900	0.3580	0.3949
	r/ adaptive kernels -> Chorus kernels	0.5227	0.6441	0.3963	0.4357
	w/o kernel-based representations	0.4867	0.6243	0.3560	0.4005
	w/o graph interaction	0.5354	0.6568	0.4051	0.4451
	r/ graph gate -> addition	0.5396	0.6518	0.4147	0.4522
	r/ graph gate -> co-attention	0.5220	0.6386	0.3990	0.4368
	w/o graph translation affinity	0.5387	0.6500	0.4111	0.4472
	r/ relation attention -> addition	0.5216	0.6489	0.3913	0.4325
	Ours (SCG-SPRe)	**0.5413**	**0.6588**	**0.4162**	**0.4542**

- **w/o kernel-based representations:** We remove the kernel-based representations (e.g., $\dot{U}_{s,u}$) in kernel-enhanced transformer networks and only keep sequence-aware representations.
- **w/o graph interaction:** We remove the interaction between the sutstitutable and complementary relations in graph representation propagation by setting $\alpha_{s,p}^{k} = 0$ (in Equation (3)) and $\alpha_{c,p}^{k} = 0$.
- **r/ graph gate -> addition:** We replace the gating computation in Equation (4) with a simple addition operation for graph representation updating. It is equivalent to equally treating the right two parts of Equation (3).
- **r/ graph gate -> co-attention:** We change the representation updating in Equation (3) by modeling the interactions of the target node from one graph with all the nodes (instead of only with the same node) from another graph.
- **w/o graph translation affinity:** We remove the right two parts in Equation (2). This means the model variant does not consider the affinities between the translation-based representations and the neighbor representations in graph representation aggregation.
- **r/ relation attention -> addition:** We replace the attention computation in Equation (16) by a simple addition operation to get the full candidate-dependent user representation. Under this situation, the variant does not differentiate the different importance of the two relation-specific representations.

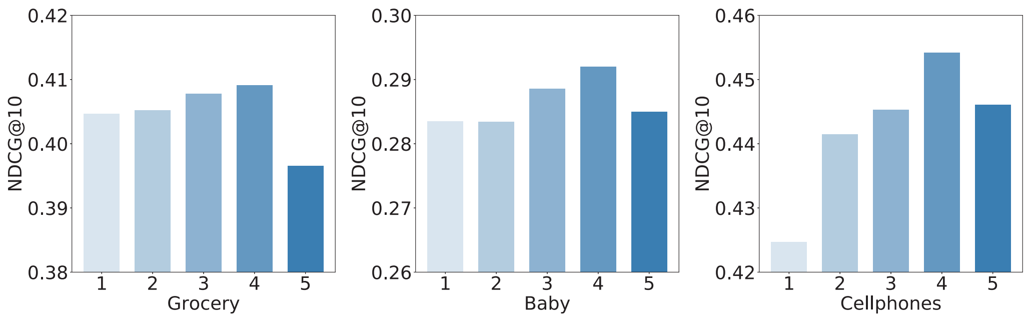


Fig. 5. Result variation when taking different numbers of propagation layers.

We show the results of the ablation study in Table 4 and find that:

- ◇ Compared to the full model SCG-SPRe, both "w/o interactive GNN" and "w/o kernel-based representations" experience significant performance degradation. This demonstrates that: (1) Leveraging graph neural networks to learn from high-order substitutable and complementary relations can lead to better product representations that are beneficial for sequential recommendation. (2) The kernel-based representations obtained by adaptively fusing multiple types of the temporal kernels indeed largely enhance sequence-aware user representations.
- ◇ "r/ adaptive kernels -> Chorus kernels" consistently performs worse than SCG-SPRe on all the datasets. This validates that the devised method of adaptively fusing multiple types of temporal kernels brings additional gains compared to the fixed-form kernels used by Chorus. It makes sense, because the hybrid kernels ensure the learned temporal patterns are distinct for different product-product pairs and thus improve the expressive ability.
- ◇ The first four model variants in the second part of Method for each dataset enable a deeper understanding of the interactive graph neural networks. In particular, the results of "w/o graph interaction" and "r/ graph gate -> addition" jointly demonstrate that modeling dependencies across the substitutable and complementary relations through gating computation marginally improves the performance. Although the variant "r/ graph gate -> co-attention" also captures interactions between two graphs, it does not bring performance gains. This shows the interaction modeling should be well treated. Moreover, the performance drop of "w/o graph translation affinity" is a little larger than the first two models in most cases. This reveals that utilizing the affinities to control representation propagation is profitable. Besides, "r/ relation attention -> addition" is obviously worse than SCG-SPRe in most cases, which shows the necessity of differentiating the importance of the kernel-based representations and the sequence-aware representations for different users.

5.5 Hyper-parameter Analysis

In this section, we further analyze the impact of some key hyper-parameter settings of our model, including the propagation layer number of the interactive graph neural networks, the dimension of the main representations used in SCG-SPRe, and the maximal length of behavior sequences. We report their results in terms of NDCG@10 for simplicity and the trend for the other metrics is similar.

5.5.1 Effect of Propagation Layer Number. We first investigate the effect of the layer number by varying it in the range of $\{1, 2, \ldots, 5\}$. Figure 5 shows the corresponding results, wherein the x-axis corresponds to the propagation layer number. We observe that when increasing the layer

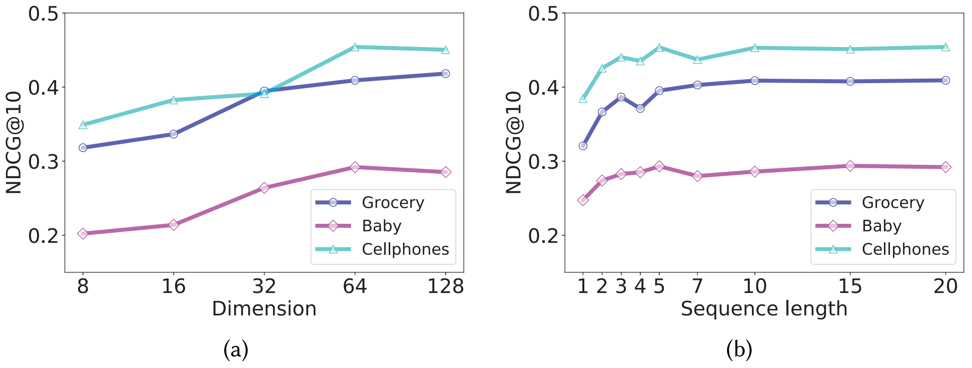


Fig. 6. (a) Performance variation w.r.t. different representation dimensions. (b) Performance variation w.r.t. different maximum length settings for training behavior sequences.

number from 1 to 4, the performance consistently takes an ascend trend across the three datasets and reaches the peak when the layer number is equal to 4. This phenomenon reveals that modeling high-order substitutable and complementary relations is promising for pursuing better sequential recommendation performance. When further increasing the layer number (e.g., 5 layers), the performance is damaged. The observation is consistent with previous studies [36, 38] on employing GNNs for recommendation.

5.5.2 Effect of Representation Dimension. Figure 6(a) depicts the performance variation w.r.t. the change of our model's representation dimension in $\{8, 16, 32, 64, 128\}$. We can see that there is a clear trend that when the dimension is increased from small integers, the results are continuously boosted on the three datasets. This is attributed to the fact that the model capacity is enlarged with the increase of the dimension. Moreover, when the dimension reaches a certain extent, the results tend to be stable or become even slightly worse. The reason is that introducing too many parameters might cause the overfitting issue.

5.5.3 Effect of Maximal Behavior Sequence Length. As for the practical implementation, we should set the maximal length of user behavior sequences to train the sequential recommendation models. Figure 6(b) presents the change of our model performance w.r.t. different sequence length, which is searched in $\{1, 2, 3, 4, 5, 7, 10, 15, 20\}$. As expected, the results tend to be better when the length gets longer. Such improvement might come from more user information used for modeling. After reaching the peak, the performance remains stable.

5.6 Fine-grained Test Results Analysis

To go deeper into the analysis of test results, we partition the test sets into different groups and see how our model behaves on these fine-grained subsets.

5.6.1 Results w.r.t. Different Relations. We first partition the test sets into different groups according to relations, aiming to investigate (1) whether SCG-SPRe promotes the recommendation performance of the products that have the substitutable or complementary relations with the interacted products in a test sequence, and (2) how our model behaves on those test examples that have no relations. Specifically, for a behavior sequence in a test set, if its ground-truth product (to be predicted) has neither substitutable nor complementary relation with any product in the sequence, it is assigned to the normal test group. Otherwise, it is assigned to the complement group or the

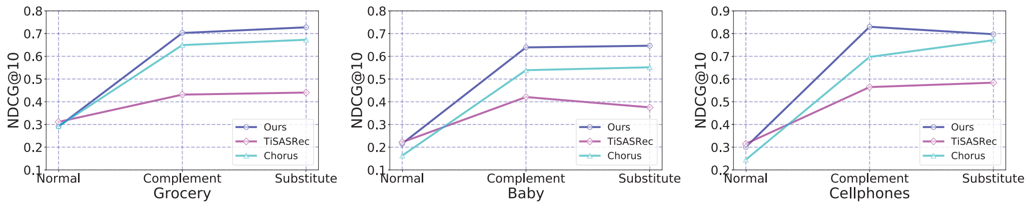


Fig. 7. Fine-grained performance on three groups with different relations.

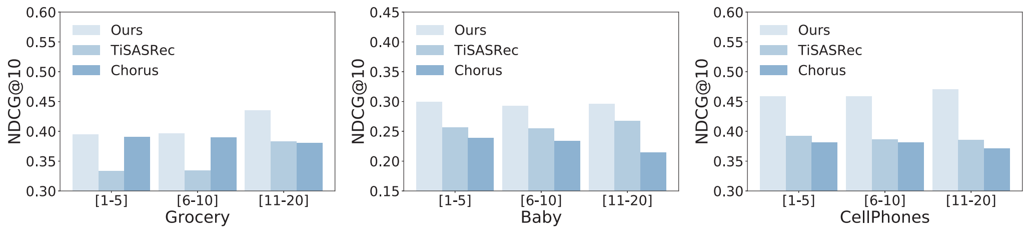


Fig. 8. Fine-grained performance on three groups with different length intervals.

substitute group based on the existence of relations. It is worth noting that if both relations are available for a test sequence, then we assign it to the two groups simultaneously.

Here, we choose the well-performed baseline TiSASRec and the most relevant model Chorus as comparisons. Figure 7 reports the results of our model and the compared two models in different groups on all the datasets, from which we have two key findings:

◇ SCG-SPRe yields much better performance than TiSASRec in the complement and substitute groups, whereas the improvement in the normal group is smaller. Such a finding verifies that our model effectively utilizes the two types of relations to benefit sequential recommendation.
◇ SCG-SPRe not only performs better than Chorus in the complement and substitute groups but also in the normal group. These results reveal that our model could behave well in a more general recommendation scenario than Chorus.

5.6.2 Results w.r.t. Different Length of Sequences. We then analyze the performance of our model and the two representative baselines TiSASRec and Chorus on test sequences with different length. To achieve this, we divide the sequences into different test groups based on the following length intervals: $[1-5]$, $[6-10]$, and $[11-20]$, where 20 is the maximal length of a user behavior sequence. Figure 8 presents the results of the three models on different groups. It is obvious that SCG-SPRe is superior for all the groups and the performance improvement is significant in most cases. This finding indicates SCG-SPRe is robust to behavior sequences with varied length and is good in practical usage.

5.6.3 Results w.r.t. Different User and Item Groups. To further verify the performance of the proposed model on cold-start users and items, we group them in the test sets according to the numbers of their existing interactions with counterparts. Specifically, we set the three intervals $[1-5]$, $[6-10]$, and $[11+]$ for user and item groups, respectively. As such, the users and items in the groups of $[1-5]$ could be regarded as cold-start users and items. Figures 9 and 10 present the detailed results of our model, TiSASRec, and Chorus on different groups, based on which, we have two key observations:

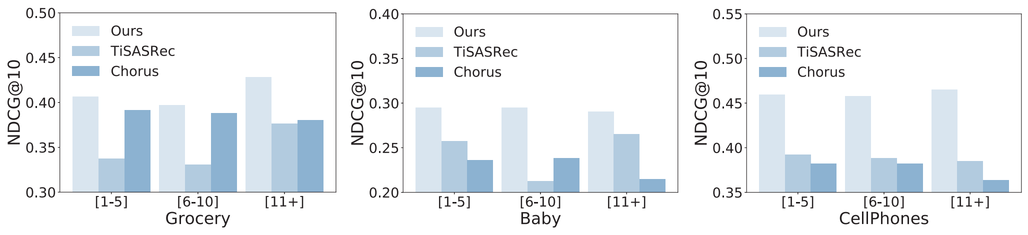


Fig. 9. Fine-grained performance on three user groups with different numbers of known interactions with the item side.

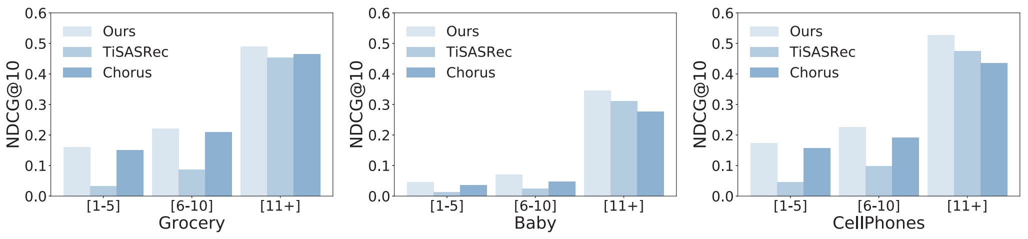


Fig. 10. Fine-grained performance on three item groups with different numbers of known interactions with the user side.

- ⋄ The performance of our model is consistently better than the two competitive baselines among different user and item groups. This demonstrates our model is relatively profitable, even facing cold-start users and items.
- ⋄ The performance of different groups in the item side varies considerably, while in the user side, the results vary smoothly. This might be attributed to the fact that the three models heavily rely on item representations as basic input to not only characterize items but also build user representations.

5.7 More Result Analysis

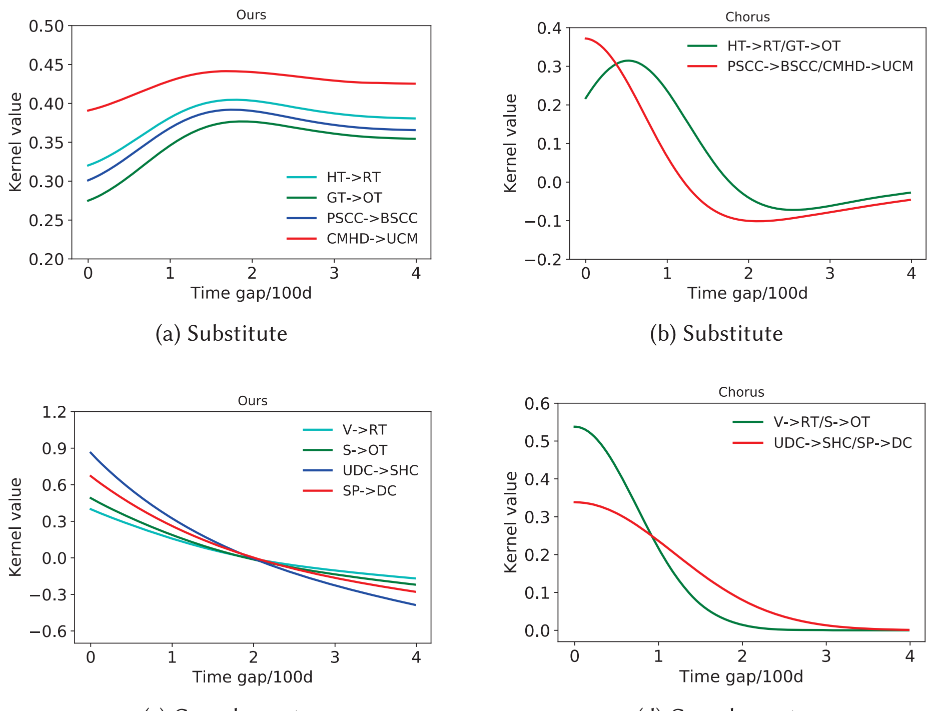
This section adds some discussions about the case study of the learned kernel functions, the gating weights in the interactive graph neural networks, the training time cost, and the performance of different temporal kernels.

5.7.1 Case Study of Temporal Kernels. Since the adaptive fusion of multiple kernel functions plays a pivotal role in characterizing the temporal patterns and building user representations, we conduct some case studies to show the curves of the fused kernels. For each of the Grocery and Cellphones datasets, we select four pairs of products that have either the substitutable relation or the complementary relation. The details of the selected products are shown in Table 5. Specifically, the column of Interacted Product contains the interacted products in a user behavior sequence. Their timestamps are regarded as the start time (i.e., t' in Equation (10)) in kernel functions. The column of Candidate Product corresponds to the products to be predicted and the recommendation (interaction) time is set to t in Equation (10).

For the proposed SCG-SPRe and the baseline Chorus, we visualize the curves of their learned kernels w.r.t. the temporal difference (i.e., $t - t'$) in Figure 11, wherein we use -> (e.g., V->RT) to represent the direction from interacted products (e.g., Vanilla) to candidate products (e.g., Red tea). First, due to the intrinsic mechanism of the kernel function learning adopted in Chorus, it cannot

Table 5. Product Pairs Selected for Visualizing the Learned Kernels by SCG-SPRe and Chorus

Dataset	Relation	Interacted Product	Candidate Product
Grocery	Substitute	Herb tea (HT) Green tea (GT)	Red tea (RT) Oolong tea (OT)
	Complement	Vanilla (V) Sweetener (S)	Red tea (RT) Oolong tea (OT)
Cellphones	Substitute	Pink skin case cover (PSCC) Car mount holder desk (CMHD)	Blue skin case cover (BSCC) Universal car mount (UCM)
	Complement	USB data charge (UDC) Screen protector (SP)	Shell holster combo (SHC) Defender case (DC)



(a) Substitute (b) Substitute

(c) Complement (d) Complement

Fig. 11. Visualization of the learned temporal kernel functions for the cases.

differentiate the temporal patterns of different product pairs that belong to the same category. As a result, the curves for any two product pairs coming from either Grocery or Cellphones coincide completely, as shown in Figures 11(b) and 11(d). Second, in contrast to Chorus, our model is capable of providing fine-grained kernel functions that could capture the unique temporal patterns for different product pairs. As can be seen in Figures 11(a) and 11(c), the curves of the fused kernel functions for different product pairs exhibit not exactly the same temporal patterns, although their shapes are somewhat similar. The advantages of the adaptive fusion of temporal kernels over the fixed-form kernels in Chorus are also proved by the ablation study (see Table 4 for reference).

5.7.2 Analysis of Gating Weights in Interactive GNNs. This part analyzes the rationality of the computed gating weights in the module of the interactive graph neural networks. From the

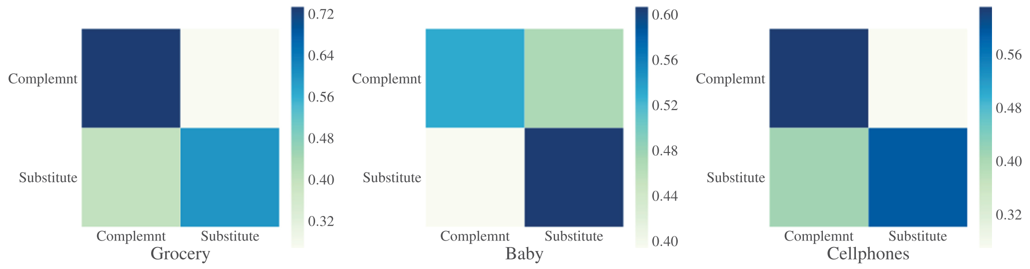


Fig. 12. Visualization of the gate weights calculated in the interactive GNNs.

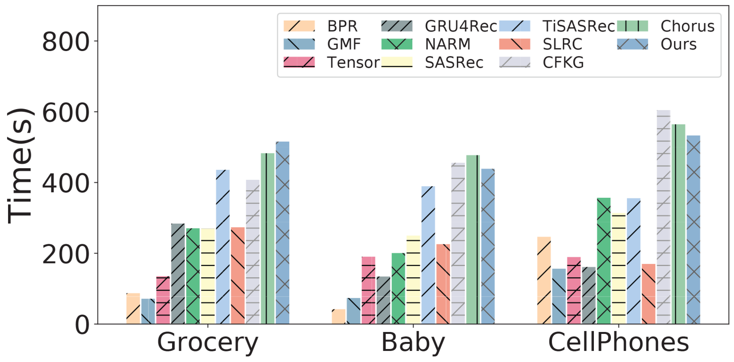


Fig. 13. Training time cost of different models.

relation level, we calculate the average gate weights w.r.t. all the products in different layers of both the substitutable product-product graph and the complementary product-product graph. Figure 12 visualizes the values of gating weights on three datasets through heat maps, where each row corresponds to the specific graph being modeled. It shows the diagonal values are noticeably larger than the other values in each heat map. It makes sense, because the product representation updating should be mainly dependent on the representations from the same relational graph and the representations from another relation graph play an auxiliary role.

5.7.3 Analysis of Training Time Cost. We further present the training time cost of the adopted models in Figure 13. It meets the expectation that the complicated models (e.g., our SCG-SPRe, Chorus, and TiSASRec) consume more training time than the simple models (e.g., BPR, GMF, and Tensor). However, the sacrifice of training efficiency is meaningful, since the complicated models can obtain better recommendation performance, especially for SCG-SPRe, which gains large relative improvements shown in Table 3. Moreover, although our model involves computation w.r.t. graph neural networks and transformer networks, its training efficiency is comparable with Chorus and TiSASRec in most cases. This is because Chorus is a two-stage approach, which requires to first learn knowledge embeddings for different product datasets, and TiSASRec needs heavy self-attention computation. It is worth noting that CFKG costs more time than Chorus on CellPhones, since CFKG requires more training steps to achieve its good performance.

5.7.4 Performance Analysis of Different Temporal Kernels. Finally, we investigate the contributions of different types of temporal kernels used by SCG-SPRe. To realize this, we select kernels in a combinatorial manner and incorporate them into our model to show the detailed performance. For ease of illustration, the Gaussian kernel, exponential kernel, and logarithmic-decay kernel are

Table 6. Performance of Using Different Kernels

Dataset	Kernel	HR@5	HR@10	NDCG@5	NDCG@10
Grocery	exp	0.4295	0.5399	0.3069	0.3427
	norm	0.4724	0.5759	0.3424	0.3760
	log	0.5020	0.6052	0.3656	0.3992
	exp+norm	0.4744	0.5762	0.3466	0.3796
	exp+log	0.5065	0.6098	0.3707	0.4044
	norm+log	0.4933	0.5963	0.3603	0.3937
	Ours (exp+norm+log)	**0.5145**	**0.6141**	**0.3767**	**0.4091**
Baby	exp	0.3493	0.4757	0.2455	0.2864
	norm	0.3483	0.4750	0.2452	0.2861
	log	0.3516	0.4814	0.2438	0.2856
	exp+norm	0.3453	0.4714	0.2455	0.2862
	exp+log	0.3511	0.4761	0.2450	0.2854
	norm+log	0.3522	0.4804	0.2462	0.2875
	Ours (exp+norm+log)	**0.3569**	**0.4857**	**0.2504**	**0.2920**
Cellphones	exp	0.4709	0.6051	0.3442	0.3876
	norm	0.5227	0.6441	0.3963	0.4357
	log	0.5373	0.6554	0.4089	0.4485
	exp+norm	0.5211	0.6441	0.3940	0.4338
	exp+log	0.5395	0.6566	0.4083	0.4472
	norm+log	0.5384	0.6574	0.4098	0.4480
	Ours (exp+norm+log)	**0.5413**	**0.6588**	**0.4162**	**0.4542**

abbreviated as norm, exp, and log, respectively. As shown in Table 6, the fusion of the three types of kernels achieves the best performance. Moreover, different types of kernels are with different importance for the studied problem. This indicates the necessity of conducting attention computation to learn the weights of specific kernel types.

6 CONCLUSION

This article has studied the problem of sequential product recommendation by leveraging the substitutable and complementary relations between products from two perspectives: a graph perspective and a sequence perspective. We have designed SCG-SPRe, a graph-based sequential recommendation model with two novel modules. The module of the interactive graph neural networks learns the relation-specific product representations by capturing the high-order product relations from the graph perspective. Based on the obtained product representations, the module of the kernel-enhanced transformer networks learns candidate-dependent user representations from the sequence perspective. It first obtains kernel-based user representations based on the adaptive fusion of multiple kernel functions and consequently integrates them with sequence-aware user representations to represent user interests for generating recommendation. Comprehensive experiments on three real-world datasets have demonstrated the performance superiority of SCG-SPRe over competitive baselines and validated the effectiveness of its modules and key components.

ACKNOWLEDGMENTS

Wei Zhang thanks the editors and reviewers for their valuable comments.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.

[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS'13)*. 2787–2795.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'19)*. 4171–4186.

[4] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of the World Wide Web Conference (WWW'19)*. 417–426.

[5] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems (NIPS'17)*. 1024–1034.

[6] Junheng Hao, Tong Zhao, Jin Li, Xin Luna Dong, Christos Faloutsos, Yizhou Sun, and Wei Wang. 2020. P-companion: A principled framework for diversified complementary product recommendation. In *Proceedings of the 29th International Conference on Information and Knowledge Management (CIKM'20)*. 2517–2524.

[7] Alan G. Hawkes. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* 1 (1971), 83–90.

[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM Conference on Research and Development in Information Retrieval (SIGIR'20)*. 639–648.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW'17)*. 173–182.

[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. Retrieved from https://arXiv:1511.06939.

[11] Wang-Cheng Kang, Eric Kim, Jure Leskovec, Charles Rosenberg, and Julian J. McAuley. 2019. Complete the look: Scene-based complementary product recommendation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 10532–10541.

[12] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of IEEE International Conference on Data Mining (ICDM'18)*. 197–206.

[13] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the 4th Conference on Recommender Systems (RecSys'10)*. 79–86.

[14] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of 5th International Conference on Learning Representations (ICLR'17)*.

[15] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[16] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 26th Conference on Information and Knowledge Management (CIKM'17)*. 1419–1428.

[17] Jiacheng Li, Yujie Wang, and Julian J. McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM'20)*. 322–330.

[18] Zhi Li, Bo Wu, Qi Liu, Likang Wu, Hongke Zhao, and Tao Mei. 2020. Learning the compositional visual coherence for complementary recommendations. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI'20)*. 3536–3543.

[19] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI'15)*, Blai Bonet and Sven Koenig (Eds.). 2181–2187.

[20] Yiding Liu, Yulong Gu, Zhuoye Ding, Junchao Gao, Ziyi Guo, Yongjun Bao, and Weipeng Yan. 2020. Decoupled graph convolution network for inferring substitutable and complementary items. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*. 2621–2628.

[21] Julian J. McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th International Conference on Knowledge Discovery and Data Mining (SIGKDD'15)*. 785–794.

[22] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM Conference on Research and Development in Information Retrieval (SIGIR'15)*. 43–52.

[23] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. 2002. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'02)*. 669–672.

[24] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting cross-session information for session-based recommendation with graph neural networks. *ACM Trans. Inf. Syst.* 38, 3 (2020), 22:1–22:23.

[25] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *ACM Comput. Surv.* 51, 4 (2018), 66:1–66:36.

[26] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys'17).* 130–137.

[27] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI'09).* 452–461.

[28] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10).* 811–820.

[29] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM'19).* 1441–1450.

[30] Xianfeng Tang, Yozen Liu, Neil Shah, Xiaolin Shi, Prasenjit Mitra, and Suhang Wang. 2020. Knowing your FATE: Friendship, action and temporal explanations for user engagement prediction on social apps. In *Proceedings of the 26th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD'20).* 2269–2279.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'17).* 5998–6008.

[32] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18).*

[33] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2019. Modeling item-specific temporal dynamics of repeat consumption for recommender systems. In *Proceedings of the World Wide Web Conference (WWW'19).* 1977–1987.

[34] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Make it a chorus: Knowledge- and time-aware item modeling for sequential recommendation. In *Proceedings of the 43rd International Conference on Research and Development in Information Retrieval (SIGIR'20).* 109–118.

[35] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM'18).* 417–426.

[36] Wen Wang, Wei Zhang, Shukai Liu, Qi Liu, Bo Zhang, Leyu Lin, and Hongyuan Zha. 2020. Beyond clicks: Modeling multi-relational item graph for session-based target behavior prediction. In *Proceedings of the Web Conference (WWW'20).* 3056–3062.

[37] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM International Conference on Knowledge Discovery & Data Mining (SIGKDD'19).* 950–958.

[38] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM Conference on Research and Development in Information Retrieval (SIGIR'19).* 165–174.

[39] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. AM-GCN: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 25th ACM International Conference on Knowledge Discovery & Data Mining (SIGKDD'20).* 1243–1253.

[40] Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2018. A path-constrained framework for discriminating substitutable and complementary products in e-commerce. In *Proceedings of the 11th International Conference on Web Search and Data Mining (WSDM'18).* 619–627.

[41] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xianling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM Conference on Research and Development in Information Retrieval (SIGIR'20).* 169–178.

[42] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM'17).* 495–503.

[43] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19).* 346–353.

[44] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM Conference on Research and Development in Information Retrieval (SIGIR'19).* 285–294.

[45] Mingyue Zhang, Xuan Wei, Xunhua Guo, Guoqing Chen, and Qiang Wei. 2019. Identifying complements and substitutes of products: A neural network framework based on product embedding. *ACM Trans. Knowl. Discov. Data* 13, 3 (2019), 34:1–34:29.
[46] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.* 52, 1 (2019), 5:1–5:38.
[47] Shijie Zhang, Hongzhi Yin, Qinyong Wang, Tong Chen, Hongxu Chen, and Quoc Viet Hung Nguyen. 2019. Inferring substitutable products with deep network embedding. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*, Sarit Kraus (Ed.). 4306–4312.
[48] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. 2018. Learning over knowledge-base embeddings for recommendation. Retrieved from https://arXiv:1803.06540.
[49] Yin Zhang, Haokai Lu, Wei Niu, and James Caverlee. 2018. Quality-aware neural complementary item recommendation. In *Proceedings of the 12th Conference on Recommender Systems (RecSys'18)*. 77–85.
[50] Tong Zhao, Julian J. McAuley, Mengya Li, and Irwin King. 2017. Improving recommendation accuracy using networks of substitutable and complementary products. In *Proceedings of International Joint Conference on Neural Networks (IJCNN'17)*. 3649–3655.
[51] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. Retrieved from https://arxiv.org/abs/1812.08434.

Received November 2020; revised March 2021; accepted April 2021