# Sparse-Interest Network for Sequential Recommendation

Qiaoyu Tan[1], Jianwei Zhang[2], Jiangchao Yao[2], Ninghao Liu[1]
Jingren Zhou[2], Hongxia Yang[2], Xia Hu[1]
[1] Department of Computer Science and Engineering, Texas A&M University, TX, USA
[2] Alibaba Group
{qytan,nhliu43,xiahu}@tamu.edu
{zhangjianwei.zjw,jiangchao.yjc,jingren.zhou,yang.yhx}@alibaba-inc.com



## ABSTRACT

Recent methods in sequential recommendation focus on learning an overall embedding vector from a user's behavior sequence for the next-item recommendation. However, from empirical analysis, we discovered that a user's behavior sequence often contains multiple conceptually distinct items, while a unified embedding vector is primarily affected by one's most recent frequent actions. Thus, it may fail to infer the next preferred item if conceptually similar items are not dominant in recent interactions. To this end, an alternative solution is to represent each user with multiple embedding vectors encoding different aspects of the user's intentions. Nevertheless, recent work on multi-interest embedding usually considers a small number of concepts discovered via clustering, which may not be comparable to the large pool of item categories in real systems. It is a non-trivial task to effectively model a large number of diverse conceptual prototypes, as items are often not conceptually well clustered in fine granularity. Besides, an individual usually interacts with only a sparse set of concepts. In light of this, we propose a novel **S**parse **I**nterest **NE**twork (SINE) for sequential recommendation. Our sparse-interest module can adaptively infer a sparse set of concepts for each user from the large concept pool and output multiple embeddings accordingly. Given multiple interest embeddings, we develop an interest aggregation module to actively predict the user's current intention and then use it to explicitly model multiple interests for next-item prediction. Empirical results on several public benchmark datasets and one large-scale industrial dataset demonstrate that SINE can achieve substantial improvement over state-of-the-art methods.

## CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

## KEYWORDS

Recommender system, Sequential recommendation, Sparse-interest network, Multi-interest extraction


*WSDM '21, March 8–12, 2021, Virtual Event, Israel*

ACM ISBN 978-1-4503-8297-7/21/03…$15.00
https://doi.org/10.1145/3437963.3441811

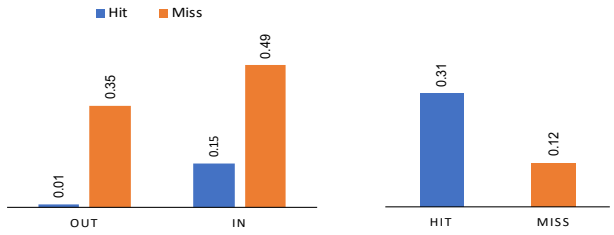


**Figure 1: Hit and Miss analysis in top@100 of single-embedding based SASRec [21] for next-item prediction on Taobao [51]. The left side shows the prediction results over "In" and "Out" settings. "In" means similar items belong to the same category of next predicted item are interacted in recent fifty behaviors, otherwise "Out". The right side shows the frequency of similar items in recent five behaviors. SASRec prefers to correctly predict the next-item if similar items are dominant in past interactions.**

**ACM Reference Format:**
Qiaoyu Tan, Jianwei Zhang, Jiangchao Yao, Ninghao Liu, Jingren Zhou, Hongxia Yang, Xia Hu. 2021. Sparse-Interest Network for Sequential Recommendation. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM '21), March 8–12, 2021, Virtual Event, Israel. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3437963.3441811*

## 1 INTRODUCTION

Recommender systems have been widely applied to many online services such as E-commerce, advertising, and social media to perform personalized information filtering [14, 17, 31, 46]. At its core is to estimate how likely a user will interact with an item based on the past actions, e.g., purchases and clicks. Traditional recommendation methods adopt collaborative filtering approaches [35] to address the problem by assuming that behaviorally similar users would exhibit similar preferences on items. Recently, neural-based deep recommendation models have shown revolutionary performance in many recommendation scenarios, due to the powerful expressive ability of deep learning. For example, NCF [14] extends matrix factorization based models [35] by replacing the interaction function of inner product with nonlinear neural networks. PinSage [46] is built on GraphSage [10], and learns user and item embeddings by conducting convolutional operations on the user-item interaction graph. However, these methods ignore the sequential structure in

user behaviors and thus fail to capture the correlations between adjacent behaviors.

Some recent works formalize recommendation as a sequential problem. The principal idea behind this is to represent each user with an ordered sequence and assume its order matters. With a user's behavior history, the sequential recommendation approach first sorts the past behaviors to obtain the ordered sequence. After that, the sequence will be fed into different neural sequential modules (e.g., recurrent neural network [17], convolutional network [42], and Transformer [21]) to generate an overall user embedding vector, which is then used to predict the next interested item. Since the sequential recommendation approach reflects the real-world recommendation situation, it has attracted much attention in modern recommendation systems.

Despite the recent advances, we argue that existing sequential recommendation models may be sub-optimal for next-item prediction due to the bottleneck of learning a single embedding from the user's behavior sequence. Each user in an E-commerce platform usually interacts with several types of items over time that are conceptually different. For example, we find that the number of categories of items that belong to different categories in a user's recent fifty behaviors is around 10 on Taobao dataset [51]. With multiple user's intentions [1], we also observe that, in Figure 1, an overall user embedding vector learned from a behavior sequence is primarily affected by the recent most frequent actions. Thus, it may fail to extract related information for learning to predict the next item if its conceptually similar items are not dominant in recent interactions. Therefore, a promising alternative solution is to learn multiple embedding vectors from a user's behavior sequence, where each embedding vector encodes one aspect of the user's interests.

However, there are several challenges for effectively extracting multiple embedding vectors from the user's behavior sequence in industry-level data. First, items are often not conceptually well clustered in real systems. Although category information of items can be used as concepts, in many cases, such type of auxiliary information may not be available or reliable due to annotation noise in practice. The second challenge is to adaptively infer a sparse set of interested concepts for a user from the large concept pool. The inference procedure includes a selection operation, which is a discrete optimization problem and hard to train end-to-end. Third, given multiple interest embedding vectors, we need to determine which interest is likely to be activated for next-item predictions. During training, the next predicted item could be used as a label to activate the preferred intention, but the inference stage has no such label. The model has to predict a user's next intention adaptively.

In this paper, we propose a novel Sparse-Interest NEtwork (SINE) for sequential recommendation to address these issues. SINE can learn a large pool of interest groups and capture multiple intentions of users in an end-to-end fashion. Figure 4 shows the overall structure of SINE. Our sparse interest extraction module adaptively infers the interacted interests of a user from a large pool of interest groups and outputs multiple interest embeddings. The aggregation module enables dynamically predicting the user's next intention, which helps to capture multi-interests for top-N item recommendation explicitly. We conduct experiments on several public benchmarks and an industrial dataset. Empirical results show that our framework outperforms state-of-the-art models and produces reasonable item clusters. To summarize, the main contributions of this paper are:

- We propose a comprehensive framework that integrates large-scale item clustering and sparse-interest extraction jointly in a recommender system.
- We investigate an adaptive interest aggregation module to explicitly model users' multiple interests for top-N recommendation in the sequential recommendation scenario.
- Our model not only achieves state-of-the-art performance on several real-world challenging datasets, but also produces reasonable interest groups to assist multi-interest extraction.

## 2 RELATED WORK

### 2.1 General Recommendation

In the conventional recommendation system, researchers focus on extracting users' general tastes from their historical behaviors. The typical examples include collaborative filtering [35, 36], matrix factorization techniques [23], and factorization machines [32]. The critical challenge of them lies in representing users and items with embedding vectors to compute their similarity. Matrix factorization (MF) methods seek to map users and items into joint latent space and estimate user-item interactions through the inner product between their embedding vectors. Factorization machines [32] aim to model all interactions between variables using factorized parameters and can even estimate interactions when facing sparsity problems.

Recently, inspired by the success of deep learning in computer vision and natural language processing [49], much effort has been put into developing deep-learning-based recommender algorithms [9, 14, 40]. One line of work seeks to use neural networks to extract additional features for the content-aware recommendation [22]. Another range of work targets to replace traditional MF. For example, NCF [14] uses multi-layer perceptions to replace the inner product operation in MF for interaction estimation, while AutoRec [37] adopts autoencoders to predict ratings. Moreover, several attempts also tried to apply graph neural networks [7, 19, 39, 48] for recommendation [13, 46].

### 2.2 Sequential Recommendation

Sequential recommendation has become the crucial problem of modern recommender systems, owing to its ability to capture the sequential patterns among successive items. One line of work attempts to model the item-to-item transition matrix based on the Markov Chain (MC). For instance, some works model the sequence using first-order Markov chain [4, 33], which assumes that the next action only relies on the last behavior. To relax this limitation, there are also methods adopting high-order MCs that consider more previous items [11, 12, 45]. A representative work is Caser [42], which treats use's behavior sequence as an "image" and adopts Convolutional Neural Network to extract user representation.

Another line of works seeks to use a sequential neural module to process the user behavior sequence [16, 21, 38, 41]. For example, GRU4Rec [17] first applies Gated Recurrent Units (GRU) to model the whole session for a more accurate recommendation. At the

[1]Through out the paper, we interchangeably use intention and interest to indicate item cluster that consists of conceptually similar items.

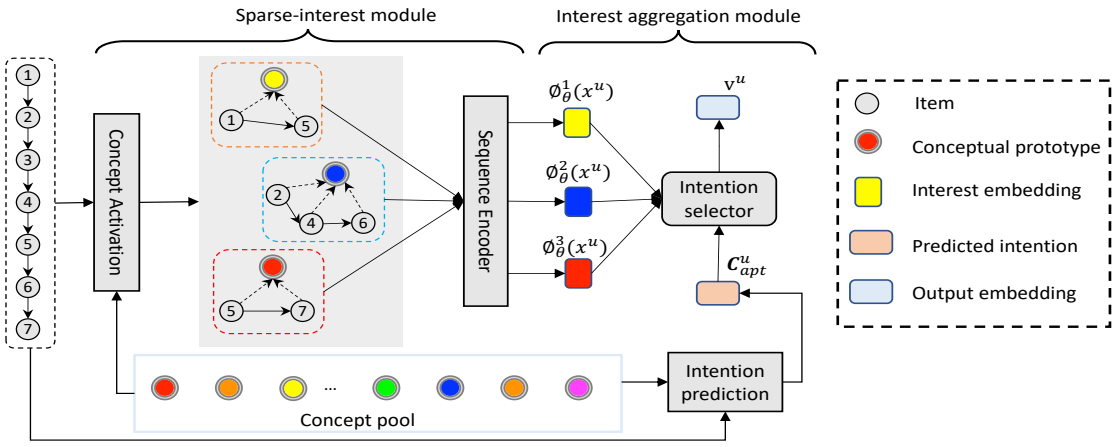


**Figure 2: The architecture of SINE (better viewed in color). Given a user's behavior sequence as input, sparse-interest module aims to adaptively activate his/her interests from the large interest group pool as well as output multi-interest embeddings. Then, the interest aggregation module helps to select the most preferred interest for next-item recommendation by actively predicting user's next intention. SINE offers the ability to cluster items and infer user's sparse set of interests in an end-to-end fashion.**

same time, SASRec [21] explores to use self-attention [43] based sequential model to capture long-term semantics and use an attention mechanism to make its prediction based on relatively few actions. Besides, there are some other works [16, 25, 47] that introduces specific neural modules for particular recommendation scenarios. For instance, DIN [50] develops a local activation unit to adaptively learn the user's representation from past behaviors for a specific ad. RUM [3] introduces a memory-augmented neural network with the insights of collaborative filtering for the recommendation. SDM [28] integrates a multi-head self-attention module with a gated fusion module to capture both short- and long-term user preferences for the next-item prediction.

## 2.3 Attention Mechanism

The attention mechanism is initially proposed in computer vision [2] and only becomes popular in recent years. It is first applied to solve the machine translation problem by [1] and later becomes an outbreaking building block as Transformer [43]. Recently, BERT leverages Transformer to achieve enormous success in the natural language processing filed for pre-training. It has also been successfully applied in many recommendation applications [38] and is rather useful and efficient in real-world application tasks.

# 3 METHODOLOGY

In this section, we first introduce the problem formulation and then discuss the proposed framework in detail. Finally, we discuss the difference between our framework and existing methods.

## 3.1 Notations and Problem Formulation

Assume $\{\mathbf{x}^{(u)}\}_{u=1}^{N}$ be the behavior dataset consists of the interactions between $N$ users and $M$ items. $\mathbf{x}^{(u)} = [x_1^{(u)}, x_2^{(u)}, \cdots, x_n^{(u)}]$ is the ordered sequence of items clicked by user $u$, where $n$ is the number of clicks made by user $u$. Each element $x_t^{(u)} \in \{1, 2, \cdots, M\}$ in the sequence is the index of the item being clicked. Note that, due to the strict requirements of latency and performance, industrial recommender systems consist of two stages, the matching stage and ranking stage [6]. The matching stage aims to retrieve top-$N$ candidate items from a large volume of item pool, while the ranking stage targets to sort the candidate items by more precise scores. We focus on improving the effectiveness of the matching stage, where the task is to retrieve high-quality candidate items that the user might be clicked with based on the observed sequence $\mathbf{x}^{(u)}$.

## 3.2 Sparse-Interest Framework

As the item pools of real-world recommender systems often consist of millions or even billions of items, the matching stage is crucial in modern recommender systems. Specifically, a deep sequential model in the matching stage typically has a sequence encoder $\phi_\theta(\cdot)$ and an item embedding table $\mathbf{H} \in \mathbb{R}^{M \times D}$, where $\theta$ is the set that contains all the trainable parameters including $\mathbf{H}$. The encoder takes the user's historical behavior sequence $\mathbf{x}^{(u)}$ as input and outputs the representation of the sequence $\phi_\theta(\mathbf{x}^{(u)})$, which can be viewed as the representation of the user's intention. The user's intention embedding is then used as a query to generate his/her candidate items from the item pool via a fast K nearest neighbor algorithm (i.e., faiss [20]). Most encoders $\phi_\theta(\cdot)$ in the literature output a single $D$-dimensional embedding vector, while there are also models that output $K$ $D$-dimensional embedding vectors to preserve the user's intentions under $K$ latent categories. We mainly focus on the latter direction and target to capture a user's diverse intentions accurately.

The state-of-art sequence encoders for capturing a user's multiple intentions can be summarized into two categories. The first type of methods resort to powerful sequential encoders to *implicitly* extract the user's multiple intentions, such as models based on multi-head self-attention (aka the Transformer [43]). The other

type of methods rely on the latent prototype to *explicitly* capture a user's multiple intentions. In general, the former approach may limit its ability to capture multiple intentions due to the mixed nature of intention detection and embedding in practice. For example, the empirical results show that the multiple vector representations learned by Transformer do not seem to have a clear advantage over the single-head implementation [21] for recommendation. In contrast, the later can effectively extract a user's diverse interests with the help of concept identified via clustering as empirically proved in [27, 29]. However, these methods scale poor because they require each user has an intention embedding under every concept, which easily scales up to thousands in industrial applications. For instance, millions or even billions of items belong to more than 10 thousand expert-labeled leaf categories [24] in the e-commerce platform of Tmall in China. With a large pool of interest concepts in real systems, a scalable multi-interest extraction module is needed.

Therefore, we propose a sparse-interest network here, which offers the ability to adaptively activate a subset of concepts from the large concept pool for a user. The input of our model is the user's behavior sequence $\mathbf{x}^{(u)}$, which is then fed into an embedding layer and transformed into item embedding matrix $\mathbf{X}^u \in \mathbb{R}^{n \times D}$. Let $\mathbf{C} \in \mathbb{R}^{L \times D}$ denotes the overall conceptual prototype matrix, and $\mathbf{C}^u \in \mathbb{R}^{K \times D}$ indicates the activated prototypical embedding matrix on $K$ latent concepts for user $u$. $L$ is the total number of concepts.

*3.2.1 Concept activation.* Our sparse-interest layer starts by inferring the interested conceptual prototypes $\mathbf{C}^u$ for each user $u$. Given $\mathbf{X}^u \in \mathbb{R}^{n \times D}$, the self-attentive method [26] is first applied to aggregate the input sequence selectively.

$$\mathbf{a} = \text{softmax}(\tanh(\mathbf{X}^u \mathbf{W}_1)\mathbf{W}_2), \tag{1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{D \times D}$ and $\mathbf{W}_2 \in \mathbb{R}^D$ are trainable parameters. The vector $\mathbf{a} \in \mathbb{R}^n$ is the attention weight vector of user behaviors. When we sum up the embeddings of input sequence according to the attention weight, we can obtain a virtual concept vector $\mathbf{z}_u = (\mathbf{a}^\top \mathbf{X}^u)^\top$ for the user. $\mathbf{z}_u \in \mathbb{R}^D$ reflects the user's general intentions and could be used to activate the interested conceptual prototypes as:

$$\begin{aligned} \mathbf{s}^u &= \langle \mathbf{C}, \mathbf{z}_u \rangle, \\ \text{idx} &= \text{rank}(\mathbf{s}^u, K), \\ \mathbf{C}^u &= \mathbf{C}(\text{idx}, :) \odot (\text{Sigmoid}(\mathbf{s}^u(\text{idx}, :)\mathbf{1}^T)), \end{aligned} \tag{2}$$

where $\text{rank}(\mathbf{s}^u, K)$ is the top-K ranking operator, which returns the indices of the $K$-largest values in $\mathbf{s}^u$. The index returned by $\text{rank}(\mathbf{s}^u, K)$ contains the indices of prototypes selected for user $u$. $\mathbf{C}(\text{idx}, :)$ performs the row extraction to form the the sub-prototype matrix, while $\mathbf{s}(\text{idx}, :)$ extracts values in $\mathbf{s}^u$ with indices idx. $\mathbf{1} \in \mathbb{R}^K$ is a vector with all elements being 1. $\odot$ represents Hadamard product and $\langle \cdot, \cdot \rangle$ is inner product. $\mathbf{C}^u \in \mathbb{R}^{K \times D}$ is the final activated $K$ latent concept embedding matrix for user $u$. Equation 2 is a top-$K$ selection trick that enables discrete selection operation differentiable, prior work [8] has found that it is very effective in approximating top-$K$ selection problem.

*3.2.2 Intention assignment.* After inferring the current conceptual prototypes $\mathbf{C}^u$, we can estimate the user intention related with each item in his/her behavior sequence according to their distance to the prototypes.

$$P_{k|t} = \frac{\exp\left(\text{LayerNorm}_1(\mathbf{X}_t^u \mathbf{W}_3) \cdot \text{LayerNorm}_2(\mathbf{C}_k^u)\right)}{\sum_{k'=1}^{K} \exp\left(\text{LayerNorm}_1(\mathbf{X}_t^u \mathbf{W}_3) \cdot \text{LayerNorm}_2(\mathbf{C}_{k'}^u)\right)}, \tag{3}$$

where $P_{k|t}$ measures how likely the primary intention at position $t$ is related with the $k^{th}$ latent concept. $\mathbf{C}_k^u \in \mathbb{R}^D$ is the embedding of the $k^{th}$ activated conceptual prototype of user $u$. $\mathbf{W}_3 \in \mathbb{R}^{D \times D}$ is the trainable weight matrix. $\text{LayerNorm}_l(\cdot)$ represents a layer normalization layer. Note that we are using cosine similarity instead of the inner product here, due to the normalization. This choice is motivated by the fact that cosine is much less vulnerable than dot product when it comes to model collapse [29], e.g., the degeneration situation where the model is ignoring most prototypes.

*3.2.3 Attention weighting.* In addition to the attention weight $P_{k|t}$ calculated from the conceptual perspective, we also consider another attention weight $P_{t|k}$ to estimate how likely the item at position $t$ is essential for predicting the user's next intentions.

$$\begin{aligned} P_{t|k} &= \mathbf{a}_t^k, \\ \mathbf{a}^k &= \text{softmax}(\tanh(\mathbf{X}^u \mathbf{W}_{k,1})\mathbf{W}_{k,2})^T, \end{aligned} \tag{4}$$

$\mathbf{a}^k \in \mathbb{R}^n$ is the attention vector for all positions. The superscript $k$ represents it's the attention layer for the $k^{th}$ activated intention. Similar to Equation 1, the above equation is another self-attentive layer. The primary difference lies in that we try to make use of the order of user sequences here and add extra trainable positional embeddings [43] to the input embeddings. The dimensionality of positional embeddings is the same as that of the item embeddings so that they can be directly summed.

*3.2.4 Interest embedding generation.* We can now generate multiple interest embedding vectors from a user's behavior sequence $\mathbf{X}^u$ according to $P_{k|t}$ and $P_{t|k}$. Specifically, the $k^{th}$ output of our sparse-interest encoder $\phi_\theta^k(\mathbf{x}^{(u)}) \in \mathbb{R}^D$ is computed as follows:

$$\phi_\theta^k(\mathbf{x}^{(u)}) = \text{LayerNorm}_3\left(\sum_{t=1}^{n} P_{k|t} \cdot P_{t|k} \cdot \mathbf{X}_t^u\right). \tag{5}$$

Till now, we have introduced the whole process of the sparse-interest network. Given a user's behavior sequence, we first activate his/her preferred conceptual prototypes from the concept pool. The intention assignment is then performed to estimate the user intention related with each item in the input sequence. After that, the self-attentive layer is applied to calculate all items' attention weights for next-item prediction. Finally, the user's multiple interest embeddings are generated through a weighted sum, according to Equation 5.

## 3.3 Interest Aggregation Module

After the sparse-interest extraction module, we obtain multiple interest embeddings for each user. A natural follow-up question is how to leverage various interest for practical inference. An intuitive solution is to use the next predicted item as a target label to select different interest embeddings for training as in MIND [24]. Despite its simplicity, the main drawback is that there are no target labels

during inference, which leads to a gap between training and testing and may result in performance degeneration.

To address this issue, we propose an adaptive interest aggregation module based on active prediction. The motivation here is that it is easier to predict a user's temporal preference-based next intentions instead of finding the ideal labels. Specifically, based on the intention assignment score $P_{k|t}$ computed in Equation 3, we can obtain an intention distribution matrix, denoted by $\mathbf{P}^u \in \mathbb{R}^{n \times K}$, for all items in the behavior sequence. Then, the input behavior sequence $\mathbf{x}^u$ can be reformulated from the intention perspective denoted by $\widehat{\mathbf{X}^u} = \mathbf{P}^u \mathbf{C}^u$, where $\widehat{\mathbf{X}^u} \in \mathbb{R}^{n \times D}$ is viewed as the intention sequence of user $u$. With $\widehat{\mathbf{X}^u}$, the user's next intention $\mathbf{C}^u_{apt}$ is adaptively computed as

$$\mathbf{C}^u_{apt} = \text{LayerNorm}_4\left((\text{softmax}(\tanh(\widehat{\mathbf{X}^u}\mathbf{W}_3)\mathbf{W}_4))^\top \widehat{\mathbf{X}^u}\right)^\top, \tag{6}$$

where $\mathbf{C}^u_{apt} \in \mathbb{R}^D$ is the predicted intention of user $u$ for next item. $\mathbf{W}_3 \in \mathbb{R}^{D \times D}$ and $\mathbf{W}_4 \in \mathbb{R}^D$ are trainable parameters. Given $\mathbf{C}^u_{apt}$ and multiple interest embeddings $\{\phi^k_\theta(\mathbf{x}^{(u)})\}^K_{k=1}$, the aggregation weights of different interests are calculated as

$$e^u_k = \frac{\exp((\mathbf{C}^u_{apt})^\top \phi^k_\theta(\mathbf{x}^{(u)})/\tau)}{\sum^K_{k'=1} \exp((\mathbf{C}^u_{apt})^\top \phi^{k'}_\theta(\mathbf{x}^{(u)})/\tau)}. \tag{7}$$

Where $e^u = [e^u_1, e^u_2, \cdots, e^u_K]^T \in \mathbb{R}^K$ is the attention vector for diverse interests. $\tau$ is a temperature parameter to tune. When $\tau$ is large ($\tau \to \infty$), $e^u$ approximates a uniformly distributed vector. When $\tau$ is small ($\tau \to 0^+$), $e^u$ approximates a one-hot vector. In experiments, we use $\tau = 0.1$ to enforce the aggregator select the most preferred intention for inference. The final user representation $\mathbf{v}^u \in \mathbb{R}^D$ is computed as

$$\mathbf{v}^u = \sum_{k=1}^{K} e^u_k \cdot \phi^k_\theta(\mathbf{x}^{(u)}. \tag{8}$$

## 3.4 Model Optimization

We follow the common practice [21, 24] to train our model by recovering the next click $x^{(u)}_t$ based on the truncated sequence prior to the click, i.e., $[x^{(u)}_1, x^{(u)}_1, \cdots, x^{(u)}_{t-1}]$. Given a training sample $(u, t)$ with the user embedding vector $\mathbf{v}^u$ and item embedding $\mathbf{H}_t$, we aim to minimize the following negative log-likelihood

$$\begin{aligned}\mathcal{L}_{like} &= -\sum_u \sum_t \log P(x^{(u)}_t | x^{(u)}_1, x^{(u)}_2, \cdots, x^{(u)}_{t-1}) \\ &= -\sum_u \sum_t \log \frac{\exp(\mathbf{H}^\top_t \mathbf{v}^u)}{\sum_{j \in \{1,2,\cdots,M\}} \exp(\mathbf{H}^\top_j \mathbf{v}^u))}.\end{aligned} \tag{9}$$

Equation (9) is usually intractable in practice, because the sum operation of the denominator is computationally prohibitive. We, therefore, leverage a Sampled Softmax technique [6, 18] to train our model. Besides, we also introduce a covariance regularizer following [5] to enforce the learned conceptual prototypes orthogonally. Specifically, denote $\mathbf{M} = \frac{1}{D}(\mathbf{C} - \overline{C})(\mathbf{C} - \overline{C})^\top$ as the covariance matrix of prototype embeddings, where $\overline{\mathbf{C}}$ is the mean matrix of $\mathbf{C}$. The regularization loss $\mathcal{L}_c$ to regularize the covariance is

$$\mathcal{L}_c = \frac{1}{2}(||\mathbf{M}||^2_F - ||\text{diag}(\mathbf{M})||^2_F). \tag{10}$$

Where $||\cdot||_F$ is the Frobenius norm matrix. Combine the two losses above, the final loss function of our model is

$$\mathcal{L} = \mathcal{L}_{like} + \lambda \mathcal{L}_c, \tag{11}$$

where $\lambda$ is the trade-off parameter to balance the two losses.

## 3.5 Connections with Existing Models

We compare our model and existing methods that focus on extracting user's multiple interest embeddings in the matching stage of recommendation. We roughly divided them into two categories and analyzed the difference below.

**Implicit approach**. This type of method relies on powerful neural networks to implicitly cluster historical behaviors and extract diverse interests. For example, MIND [24] utilizes Capsule network [34] to adaptively aggregate user's behaviors into interest embedding vectors. SASRec [21] adopts the multi-head self-attention mechanism [43] to output multiple representation for a user. Compared with these methods, our model belongs to an explicit approach that explicitly detects intentions from the user's behavior sequence based on latent conceptual prototypes.

**Explicit approach**. Methods that belong to this type maintain a set of conceptual prototypes to explicitly determine the intentions of items in the user's behavior sequence. MCPRN [44] is a recent representative work for extracting multiple interests from the session for the next-item recommendation. DisenRec [29] utilizes latent prototypes to help learn disentangled representations for recommendation. Compared with them, we also follow the explicit approach, but our model scales to a large-scale dataset. Specifically, they require the number of diverse interest embeddings equals to the number of conceptual prototypes. However, the number of latent concepts depends on applications and can be easily scaled up to hundreds or even thousands in industrial recommender systems, which hinders their application in practice. In contrast, our sparse-interest network offers the ability to infer a sparse set of preferred intentions from the large concept pool automatically.

# 4 EXPERIMENTS

In this section, we conduct experiments over three benchmark datasets and one billion-scale industrial data to validate the proposed approach. Specifically, we try to answer the following questions:

- How effective is the proposed method compared to other state-of-the-art baselines? **Q1**
- What are the effects of the different modules, sparse-interest module, and interest aggregation module through ablation studies? **Q2**
- How sensitive are the hyper-parameter settings, including the preferred $K$ intentions and the corresponding $L$ conceptual prototypes? **Q3**

## 4.1 Experimental Setup

In this section, we elaborate on the dataset description, evaluation metrics, and comparing methods in our experiments.

**Datasets**. We conduct experiments on three benchmark datasets and one billion-scale industrial data. The statistics of the datasets are shown in Table 2.

**Table 1: Recommendation performance on public datasets. The best results are highlighted with bold fold. All the numbers in the table are percentage numbers with '%' omitted.**

| | MovieLens | | | | Amazon | | | | Taobao | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Metrics@10 | | Metrics@50 | | Metrics@50 | | Metrics@100 | | Metrics@50 | | Metrics@100 | |
| | HR | NDCG | HR | NDCG | HR | NDCG | HR | NDCG | HR | NDCG | HR | NDCG |
| **GRU4Rec** | 14.61 | 5.66 | 41.61 | 10.66 | 1.70 | 0.51 | 2.74 | 0.67 | 9.41 | 3.60 | 12.43 | 4.08 |
| **Caser** | 15.44 | 6.13 | 43.64 | 11.53 | 2.60 | 0.81 | 3.96 | 1.03 | 10.71 | 4.96 | 13.50 | 5.68 |
| **SASRec** | **17.34** | **7.84** | **46.01** | **13.53** | 3.17 | 1.01 | 4.43 | 1.28 | 13.36 | 5.64 | 15.73 | 6.38 |
| **MIND** | 15.62 | 6.58 | 43.98 | 12.30 | 3.85 | 1.29 | 5.35 | 1.56 | 15.35 | 8.35 | 17.49 | 8.72 |
| **MCPRN** | 15.82 | 6.77 | 44.21 | 12.83 | 3.42 | 1.18 | 5.22 | 1.47 | 14.32 | 7.34 | 16.43 | 7.67 |
| **SINE** | 16.34 | 7.06 | **45.79** | **13.50** | **4.57** | **1.61** | **6.26** | **1.88** | **17.69** | **10.41** | **20.64** | **10.89** |

**Table 2: Statistics of the datasets.**

| Dataset | # users | # items | # interactions |
|---|---|---|---|
| MovieLens | 6,040 | 3,952 | 1,000,209 |
| Amazon | 8,026,324 | 2,330,066 | 22,507,155 |
| Taobao | 987,994 | 4,162,024 | 100,150,807 |
| ULarge | 106,527,123 | 25,000,000 | 4,000,000,000 |

- **MovieLens** [2] collects user's rating score for movies. In experiments, we follow [15] to preprocess the dataset.
- **Amazon** [3] consists of product views from Amazon. In experiments, we use the rating only version of Book category behaviors. Note that this version is more challenging than the 5-core version used in [24], due to its large volume and sparsity.
- **Taobao** [4] collects user behaviors from Taobao's recommender system. In experiments, we only use the click behaviors.
- **ULarge** consists of the clicked behaviors collected from the daily logs of an Alibaba company from March 29 to April 4, 2020.

For all datasets, we follow [21] to split the datasets into training/validation/testing sets. Specifically, we split the historical sequence for each user into three parts: (1) the most recent action for testing, (2) the second most recent action for validation, and (3) all remaining actions for training. Note that during testing, the input sequences contain training actions and the validation actions.

**Competitors**. We compare our proposed model SINE with the following state-of-the-art sequential recommendation baselines.

- **Single embedding models**: GRU4Rec [17] is a pioneering work that employs GRU to model user behavior sequences. Caser [42] is a recent CNN-based sequential recommendation benchmark.
- **Multi-embedding models**: MIND [24] and SASRec [21] are recently proposed multi-interest methods based on capsule network [34] and multi-head self-attention [43]. MCPRN is another state-of-the-art multi-interest framework based on latent conceptual prototypes.

**Parameter Configuration**. For a fair comparison, all methods are implemented in Tensorflow and optimized with Adam optimizer with a mini-batch size of 128. The learning rate is fixed as 0.001. We tuned the parameters of comparing methods according to values suggested in original papers and set the embedding size $D$ as 128 and the number of negative samples as 5 and 10 for MovieLens and other datasets. For our method, it has three crucial hyper-parameters: the trade-off parameter $\lambda$, the number of intentions $K$, and latent prototypes $L$. We search $K$ from $\{4, 8, 12, 16\}$, $L$ from $\{50, 100, 500, 1000, 2000, 5000\}$ and $\lambda$ from 0 to 1 with step size 0.1. We found our model performs relative stable when $\lambda$ is around 0.5 and set $\lambda = 0.5$. The configuration of the other two parameters for four datasets are reported in Table 3.

**Table 3: The optimal setting of our hyper-parameters for our model. Other parameters like dimension $D$, sequence length $n$ and $\lambda$ are set as 128, 20 and 0.5, respectively.**

| | # intentions $K$ | # concepts $L$ |
|---|---|---|
| MovieLens | 4 | 50 |
| Amazon | 4 | 500 |
| Taobao | 8 | 1000 |
| ULarge | 8 | 5000 |

**Evaluation Metrics**. For each user in the test set, we treat all the items that the user has not interacted with as negative items. We use two commonly used evaluation criteria [14]: *hit rate* (HR) and *normalized discounted cumulative gain* (NDCG) to evaluate the performance of our model. Besides, we also leverage the widely used *Normalized Mutual Information* (NMI) [30] to quantitative analysis of the effectiveness of the learned conceptual prototypes of our model in clustering items.

## 4.2 Comparisons with SOTA (Q1)

Table 1 summarizes the performance of SINE as well as baselines on three benchmark datasets. Clearly, SINE achieves comparable performance to all of the baselines on all the evaluation criteria in general. Caser obtains the best performance over other models (GRU4Rec) that only single output embedding for each user. It can be observed that employing multiple embedding vectors (SASRec, MIND, MCPRN, SINE) for a user perform generally better than single embedding based methods (Caser and GRU4Rec). Therefore, exploring multiple user-embedding vectors has proved to be an

[2] https://grouplens.org/datasets/movielens/1m/
[3] http://jmcauley.ucsd.edu/data/amazon/
[4] https://tianchi.aliyun.com/dataset/dataDetail?dataId=649

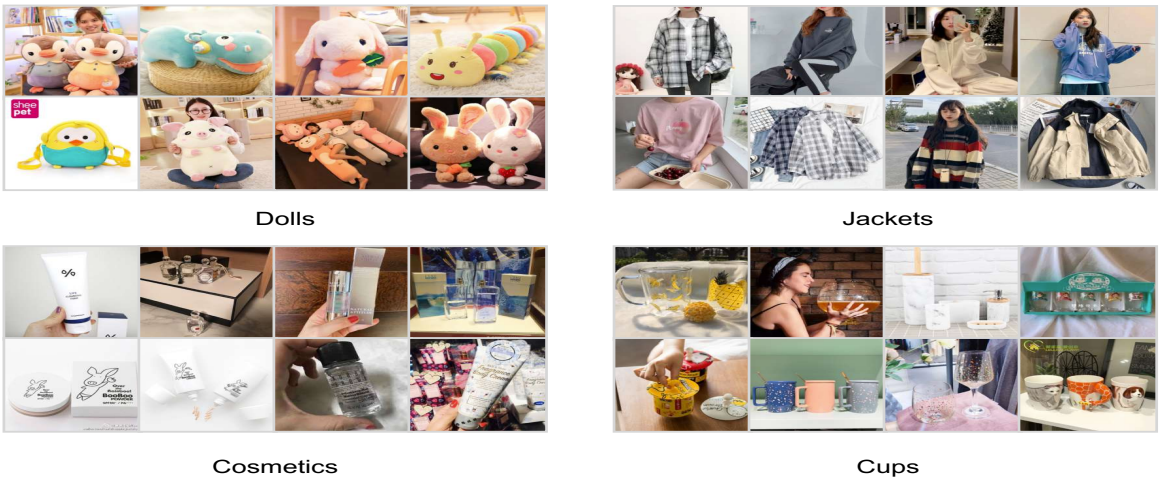


**Figure 3: Concept visualization. We draw four concepts "dolls", "jackets", "cosmetics" and "cups" with the top-8 closest items.**

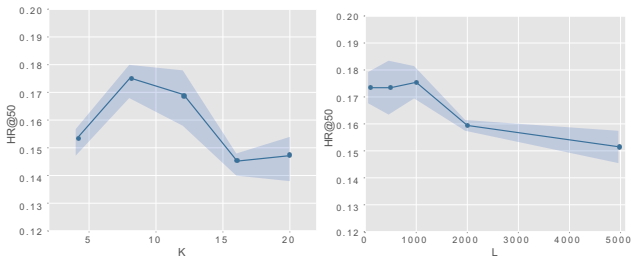


**Figure 4: Sensitivity of SINE towards $K$ and $L$ on Taobao.**

effective way of modeling users' diverse interests and boosting sequential recommendation accuracy. Moreover, we can observe that the improvement introduced by capturing user's various intentions is more significant for Taobao and Amazon datasets. The users of Taobao and Amazon tend to exhibit more diverse interests in online shopping than rating movies. The improvement of MIND over SASRec shows that dynamic routing serves as a better multi-interest extractor than multi-head self-attention. An interesting observation is that MIND beats MCPRN on Amazon and Taobao while losses on MovieLens. It is mainly because MCPRN only supports cluster all items into a small set of prototypes, which is difficult well to cluster millions of items on Amazaon and Taobao. Considering the MIND and SINE results, SINE consistently outperforms MIND on three datasets over all evaluation metrics. This can be attributed to two points: 1) The sparse-interest extractor layer explicitly utilizes a large set of conceptual prototypes to cluster items and automatically infer a subset of preferred intentions for interest embeddings generation, which achieves a more precise representation of a user. 2) Interest aggregation module actively predicts the user's current intention to directly attend over multiple user embedding vectors, enabling modeling multi-interests for top-N recommendation.

**Parameter Sensitivity (Q3).** We also investigate the sensitivity of the number of intentions $K$ and conceptual prototypes $L$. Figure 4 reports the performance of our model in terms of HR. In particular, we randomly select 1 million users for inference, and the average result of 10 runs is reported. Results hold the same for other datasets, and we omit the figure here for more space. From the figure, we can observe that SINE obtains the best performance when $K = 8$ and $L = 1000$. Considering that Taobao has around 9000 different categories in total, it verifies that the learned concepts indeed have a strong connection of categories of items, and the concept could be viewed as a virtual category that consists of several categories.

**Table 4: Recommendation performance on industrial dataset ULarge. Improv. row means the improvement of our model compared with the second-best baseline.**

| | HR@50 | HR@100 | HR@500 |
|---|---|---|---|
| **Caser** | 6.93 | 16.75 | 36.94 |
| **GRU4Rec** | 5.46 | 14.80 | 33.35 |
| **SASRec** | 8.64 | 18.58 | 38.82 |
| **MCPRN** | 7.89 | 17.65 | 37.66 |
| **MIND** | 9.13 | 19.31 | 39.09 |
| **SINE** | **12.24** | **21.12** | **40.81** |
| **Improv.** | 34.06% | 9.37% | 4.40% |

## 4.3 Industrial Results (Q1)

We further conduct an offline experiment to investigate the effectiveness of our model in extracting user's diverse interests in the industrial dataset. We implemented our model and baselines on the Alibaba company's distributed cloud platform, where every two workers share an NVIDIA Tesla P100GPU with 16GB memory.

Table 4 summarizes the performance in terms of Hit Rate. It is clear that SINE significantly outperforms other baselines by a wide

**Table 5: Prototype clustering evaluation compared with the first, second and leaf level category information on ULarge.**

| | Level-1 | Level-2 | Level-leaf |
|---|---|---|---|
| NMI | 0.09 | 0.37 | 0.29 |

**Table 6: Ablation study of SINE.**

| Dataset | Method | HR@50 | HR@100 |
|---|---|---|---|
| Taobao | **SINE-cate** | 12.45 | 15.33 |
| | **SINE-label** | 16.22 | 18.74 |
| | **SINE** | **17.69** | **20.64** |
| ULarge | **SINE-cate** | 7.18 | 17.46 |
| | **SINE-label** | 10.09 | 20.33 |
| | **SINE** | **12.24** | **21.12** |

margin. Another interesting observation is that the gap between SINE and the second-best benchmark (MIND) decreases when the number of recalled items increases. This fact indicates that our sparse-interest network helps capture user's diverse interests and ranks the most preferred items on the top recommendation list.

**Case Study** We also visualize the learned conceptual prototypes of our model. Concretely, for each concept, we leverage its prototypical embedding vector to retrieve the top-8 closest items under their cosine similarity. Figure 3 illustrates four exemplar concepts to show their clustering performance. As can be seen, our model successfully groups some semantic-similar items into a latent concept. More importantly, the items in one concept come from different semantic-close leaf categories. For example, the "cosmetics" concept contains different kinds of skin-nursing products. It indicates that compared to the conventional leaf-category partition, our conceptual prototype is related to the user's high-level intention.

To confirm this point, we compare the learned concepts with the expert-labeled category hierarchy in Alibaba company, where the number of categories in the first, second, and leaf-level are 178, 7,945, and 14874, respectively. Table 5 reports the results in terms of NMI. We can observe that the learned concepts are closest to the second level category, not in the extreme fine-grained granularity (leaf) or the very coarse granularity (first). This result demonstrates that our model can capture the relative high-level semantics for the user's intention modeling.

## 4.4 Ablation Study (Q2)

We introduce two variants (SINE-cate and SINE-label) to validate the effectiveness of the learned new prototypes and the interest aggregation module. Specifically, SINE-cate is obtained by using the category attributes as prototypes, while SINE-label is obtained by adopting label-aware attention in [24] for training. We only conduct experiments on Taobao and ULarge, since other datasets do not have category attributes. Taobao and ULarge have 9439 and 14874 distinct categories, respectively. Note that, similar to MIND [24], SINE-label first independently retrieves $K \cdot$ N candidate items based on $K$ embedding vectors and then outputs the final top-N recommendation list by sorting $K \cdot$ N items. Table 5 reports the results in terms of HR. Obviously, SINE significantly outperforms the other two variants in two datasets. The substantial difference between SINE-cate and SINE shows that the learned concepts are better to cluster items than the original items' categories. It verifies our motivation to cluster items in our model jointly. The improvement of SINE over SINE-label validates that our interest attention module is useful to model multiple interests for next-item recommendation.

## 5 CONCLUSIONS

In this paper, we propose a novel sparse-interest embedding framework for the sequential recommendation. Our model can adaptively activate multiple intentions from a large pool of conceptual prototypes to generate multiple interest embeddings for a user. It also develops an interest aggregation module to capture multi-interests to obtain the overall top-N items actively. Empirical results demonstrate that our model performs better than state-of-the-art baselines on challenging datasets. Results on the billion-scale industrial dataset further confirm our model's effectiveness in terms of recommendation accuracy and producing reasonable item clusters. We plan to leverage lifelong learning to capture users' long-term interests for a more accurate recommendation.

## REFERENCES


[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[2] Peter J Burt. 1988. Attention mechanisms for vision in a dynamic world. In *ICPRAM*. IEEE Computer Society, 977–978.
[3] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*. 108–116.
[4] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. 2013. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*.
[5] Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. 2015. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068* (2015).
[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*. 191–198.
[7] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *WWW*. 417–426.
[8] Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. *arXiv preprint arXiv:1905.05178* (2019).
[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).
[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1024–1034.
[11] Ruining He, Chen Fang, Zhaowen Wang, and Julian McAuley. 2016. Vista: a visually, socially, and temporally-aware model for artistic recommendation. In *RecSys*. 309–316.
[12] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*. IEEE, 191–200.
[13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *arXiv preprint arXiv:2002.02126* (2020).
[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[15] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*. ACM, 549–558.
[16] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*. 843–852.
[17] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[18] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).

[19] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Multi-behavior recommendation with graph convolutional networks. In *SIGIR*. 659–668.

[20] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* (2019).

[21] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. IEEE, 197–206.

[22] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *RecSys*. 233–240.

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

[24] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *CIKM*. 2615–2623.

[25] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*. 322–330.

[26] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* (2017).

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

[28] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *CIKM*. 2635–2643.

[29] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning disentangled representations for recommendation. In *NIPS*. 5711–5722.

[30] Aaron F McDaid, Derek Greene, and Neil Hurley. 2011. Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv:1110.2515* (2011).

[31] Covington Paul, Adams Jay, and Sargin Emre. 2016. Deep neural networks for YouTube Recommendation. In *RecSys*. 191–198.

[32] Steffen Rendle. 2010. Factorization machines. In *ICDM*. IEEE, 995–1000.

[33] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. 811–820.

[34] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *NIPS*. 3856–3866.

[35] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.

[36] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 291–324.

[37] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*. 111–112.

[38] 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 *CIKM*. 1441–1450.

[39] Qiaoyu Tan, Ninghao Liu, and Xia Hu. 2019. Deep Representation Learning for Social Network Analysis. *Frontiers in Big Data* 2 (2019), 2.

[40] Qiaoyu Tan, Ninghao Liu, Xing Zhao, Hongxia Yang, Jingren Zhou, and Xia Hu. 2020. Learning to Hash with Graph Neural Networks for Recommender Systems. In *WWW*. 1988–1998.

[41] Qiaoyu Tan, Jianwei Zhang, Ninghao Liu, Xiao Huang, Hongxia Yang, Jignren Zhou, and Xia Hu. 2021. Dynamic memory based attention network for sequential recommendation. In *AAAI*.

[42] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.

[44] Shoujin Wang, Liang Hu, Yan Wang, Quan Z Sheng, Mehmet A Orgun, and Longbing Cao. 2019. Modeling Multi-Purpose Sessions for Next-Item Recommendations via Mixture-Channel Purpose Routing Networks.. In *IJCAI*. 3771–3777.

[45] An Yan, Shuo Cheng, Wang-Cheng Kang, Mengting Wan, and Julian McAuley. 2019. CosRec: 2D Convolutional Neural Networks for Sequential Recommendation. In *CIKM*. 2173–2176.

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

[47] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *SIGIR*. 729–732.

[48] Wenhui Yu and Zheng Qin. 2020. Graph Convolutional Network for Recommendation with Low-pass Collaborative Filters. In *ICML*. PMLR, 10936–10945.

[49] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[50] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *KDD*. 1059–1068.

[51] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning tree-based deep model for recommender systems. In *SIGKDD*. 1079–1088.