ABSTRACT
In recommender systems, modeling user-item behaviors is essential for user representation learning. Existing sequential recommenders consider the sequential correlations between historically interacted items for capturing users’ historical preferences. However, since users’ preferences are by nature time-evolving and diversified, solely modeling the historical preference (without being aware of the time-evolving trends of preferences) can be inferior for recommending complementary or fresh items and thus hurt the effectiveness of recommender systems. In this paper, we bridge the gap between the past preference and potential future preference by proposing the future-aware diverse trends (FAT) framework. By future-aware, for each inspected user, we construct the future sequences from other similar users, which comprise of behaviors that happen after the last behavior of the inspected user, based on a proposed neighbor behavior extractor. By diverse trends, supposing the future preferences can be diversified, we propose the diverse trends extractor and the time-aware mechanism to represent the possible trends of preferences for a given user with multiple vectors. We leverage both the representations of historical preference and possible future trends to obtain the final recommendation. The quantitative and qualitative results from relatively extensive experiments on real-world datasets demonstrate the proposed framework not only outperforms the state-of-the-art sequential recommendation methods across various metrics, but also makes complementary and fresh recommendations.

CCS CONCEPTS
• Information systems → Recommender systems.

KEYWORDS
recommendation, user modeling, future-aware, diverse trends

ACM Reference Format:
Yujie Lu, Shengyu Zhang, Yingxuan Huang, Luyao Wang, Xinyao Yu, Zhou Zhao, and Fei Wu. 2021. Future-Aware Diverse Trends Framework for Recommendation. In Proceedings of the Web Conference 2021 (WWW ’21), April 19–23, 2021, Ljubljana, Slovenia. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3449791
1 INTRODUCTION

Recommender systems assume a central part of many real-world applications (e.g., e-commerce platforms [43, 57, 58]) with the prevalence of the Internet and information technology. On online platforms, users interact with a series of items in a chronological order, implying continuous and temporary correlation between each item. In this scenario, the sequential recommenders have become indispensable techniques in the recommendation area, which aim at predicting the next item that the user may interact with by modeling users’ preferences on the basis of sequential dependencies among the users’ historical interactions.

Existing sequential recommendation algorithms model and represent user preferences in various manners. Most conventional models such as Markov chain-based [24, 25, 38] and factorization-based [31, 39, 60] ones have successfully captured users’ short-term and long-term interests by adopting Markov chains and matrix factorization respectively, but either failed to model intricate dynamics or ignored the time dependency. In contrast, deep learning based techniques typically represent user preferences with low-dimensional embedding vectors. For instance, the deep neural network proposed for YouTube video recommendation [YouTubeDNN] [11] represents each user by one fixed-length vector transformed from the past practices of users, while not appropriate for modeling various interests because of the dimension explosion. To tackle this issue, Deep Interest Network (DIN) [61] makes the user representation vary over different items with attention mechanisms to capture the diversity of user interests. More recent works [8, 32] propose to encode users’ historical behaviors into users’ varying interests by leveraging capsule routing mechanism. Nevertheless, all of these methods model user preferences only taking into account the past behaviors of users, ignoring the potential future preference and failing to capture the time-evolving trends of user diversified preferences.

We argue that preferences changing over time of similar users is an important factor to model future diverse preference trends in addition to historical preference. Such trends can be summarized from the relative future behaviors of users with similar interests. Specifically, for an inspected user, other users with similar interests tend to have common behaviors (e.g. click the same items) with the inspected user, and the behaviors that happen after the common behavior can be viewed as the relative future behaviors. As shown in 1, models that solely focus on users’ historical interests tend to recommend similar and complementary items. In contrast, the proposed future-aware diverse trends framework is capable of recommending fresh items that may seem irrelevant to the user’s historical preference, while in reality are consistent with one of the preference trends that can be captured from the future behaviors of users with similar interest (or at least similar behaviors).

In this work, we focus on the problem of modeling diverse trends of users for sequential recommendation. In order to overcome the limitations of existing methods, we propose the future-aware diverse trends (FAT) framework for learning user representations that reflect diverse trends of users preferences. To infer the user representation vectors, we design an implicit neighbor behaviors extractor (INBE) layer and a novel diverse trends capture layer. To construct neighborhoods implicitly, the INBE module utilizes Pearson Correlation Coefficient [6] and an interaction-based users filter. The diverse trends capture layer applies dynamic routing and time-aware mechanism to adaptively aggregate neighbor user’s relative future behaviors as user trend representation. The user representation is then computed by concatenating the user historical behaviors embedding from traditional sequence modeling and the user trends embedding. The process of dynamic routing can be viewed as soft-clustering, which groups similar users’ relative future behaviors into several clusters. Each cluster of future behaviors is further used to infer the user trend representation vector according to the time-varying attention of each trend corresponding to the specific items. In this way, for a particular user, FAT outputs the final user preference representations considering both the user past preference and potential future preference. To summarize, the main contributions of this work are as follows:

- To better infer the dynamics of user behaviors, we design a FAT framework, which leverage the future information and capture diverse trends of user preference.
- We first design a neighbor behavior module to extract relative future behaviors from similar users implicitly. We design the diverse trends capture module, which utilizes dynamic routing to adaptively aggregate neighbor’s future behaviors into trend representation vectors. We then leverage time-aware mechanism over trends to better model time-varying user potential preferences.
- Compared with existing methods, FAT shows superior performance on several public datasets over metrics such as Recall and NDCG. In addition, we conduct experiments to show that FAT can bring diversity of retrieved items better than other baselines.

The remainder of this paper is organized as follows: related works are reviewed in Section 2; Section 3 formulates the sequential recommendation task and elaborates the technical details of FAT; In Section 4, we detail the experiments for comparing FAT with existing methods on several public benchmarks; The last section gives conclusion and future work of this paper.

2 RELATED WORK

2.1 Sequential Recommendation

Conventional sequential recommendation popular models usually use matrix factorization and Markov chains to capture long-term and short-term interests of users, respectively. The Markov chain-based sequential recommendation algorithms use functions obtained from past transactions to predict the user’s next interaction. Personalized Markov Chain Factorization (PFCM) [38] combines the advantages of Markov Chain and Matrix Factorization. Since the operation used is linear, FPMC cannot capture the interaction between multiple factors, because each component independently affects the user’s next interaction. Hierarchical Representation Model (HRM) solves this problem by summarizing multiple interaction factors through nonlinear maximum pooling operations [44]. HRM uses continuous value representations of users and items, and builds a mixed representation on users and items based on previous interactions. Both FPMC and HRM only model local interactions between successive transactions.
Due to the strong representation learning capability [56], deep learning techniques have also been adopted in the sequential recommendation in recent years. DREAM [51], based on Recurrent Neural Network (RNN), learns the user’s dynamic representation to reveal the user’s dynamic interest. DIN [61] designs a local activation unit to adaptively learn the representation of user interests from past behaviors with respect to a certain ad. [52] proposes a encoder-decoder networks to integrate future data into model training. [21] propose to model user intention from both ordered and unordered facets simultaneously. Contextualized Temporal Attention Mechanism proposed in [46] learns to weigh historical actions’ influence considering different contexts.

2.2 User Modelling
Representing users as vectors is commonly used in recommender system. Traditional methods assemble user preference as vectors composed of interested items [2, 26, 41], keywords [7, 17] and topics [56]. As the emergence of distributed representation learning, user embeddings obtained by neural networks are widely used. [9] employs RNN-GRU to learn user embeddings from the temporal ordered review documents. [37] utilizes Stacked Recurrent Neural Networks to capture the evolution of contexts and temporal gaps. [18] proposes the framework GraphRec to jointly capture interactions and opinions in the user-item graph.

GRU4Rec [28] introduces recurrent neural networks for the recommender systems firstly. [4, 27, 42, 49] models behavior sequence. [42] applies data augmentation to enhance training of GRU4Rec. [4] considers the dwell time. [27] provides impressive top-k gains for recurrent neural networks for session-based recommendation with a proposed new class of loss functions coupled with an additional sampling (combination of uniform sampling and popularity sampling) for negative sampling in GRU4Rec. [29] considers additional item information other than IDs(parallel RNN). [30] combines the session-based KNNs with GRU4Rec using the methods of switching, cascading, and weight hybrid.

[59] proposes a RNN-based framework for click-through rate (CTR) prediction in sponsor search. RRN [45] is the first recurrent recommender network that attempts to capture the dynamics of both user and item representation. [3] further improves the RRN’s interpretability by devising a time-varying neighborhood style explanation scheme. [10] proposes a memory-augmented neural network for the sequential recommendation, with analogous gains observed in other domains [15, 16, 55]. [14, 36] use GRU to model users and sessions. [12] uses RNN for the collaborative filtering task and considered two different objective functions in the RNN model. [40] deploys a multi-layer GRU network to capture sequential dependencies and user interest from both the inter-session and intra-session levels. HNVM [47] models different levels of user preferences via a unified hierarchical generative process.

NextItNet [53] is a generative CNN model with the residual block structure for the sequential recommendation. RCNN proposed in [48] utilizes the recurrent architecture of RNN and the convolutional operation of CNN to extract long-term and short-term patterns respectively.

3 METHODOLOGY
In this section, we first formulate the sequential recommendation problem, then introduce the proposed framework in detail. We lastly discuss the prediction and network training procedure of FAT.

3.1 Problem Formulation
In a typical recommendation scenario, we have a set of users and a set of items which can be denoted as $U = \{u_1, u_2, ..., u_{|U|}\}$ and $V = \{v_1, v_2, ..., v_{|V|}\}$, respectively. Let $X_u = \{x_{u1}^1, x_{u2}^2, ..., x_{un_u}^n\}$ denote the sequence of interacted items from user $u \in U$ sorted in a chronological order: $x_{ui}$ denotes the item that the user $u$ has interacted with at time step $t$. Given the user historical behaviors, the goal of the sequential recommendation task considered in this paper is to retrieve a subset of items from the pool $V$ for each user in $U$ such that the user is most likely to interact with the recommended items. Notations are summarized in 1.

Table 1: Notations.

| Notation | Description |
|----------|-------------|
| $u$      | a user      |
| $v$      | an item     |
| $x$      | an interaction |
| $d$      | the dimension of user/item embeddings |
| $t$      | the number of trends |
| $U$      | the set of users |
| $V$      | the set of items |
| $X$      | the set of interactions |
| $T$      | the trends set |
| $N$      | the number of retrieved items |

Specifically, each instance is represented by a tuple $(X_u, T_u, A_i)$, where $X_u$ denotes the set of items interacted by user $u$, $T_u$ denotes the relative future sequence set extracted from similar users, detail will be illustrated in the Section 3.4. $A_i$ the features of target item $i$ including the information of interaction time and item ID.

To model diverse user preferences dynamically, FAT learns a function $f$ for mapping user’s corresponding interactions $X_u$ and trend set $T_u$ into user representations, which can be formulated as

$$\overrightarrow{e_u} = f(X_u, T_u)$$ (1)

where $\overrightarrow{e_u} \in \mathbb{R}^{d \times 1}$ denotes the representation vector of user $u$, $d$ the dimension. Besides, the representation vector of target item $i$ is obtained by an embedding function $g$ as

$$\overrightarrow{e_i} = g(A_i)$$ (2)

where $\overrightarrow{e_i} \in \mathbb{R}^{d \times 1}$ denotes the representation vector of item $i$, and the detail of $g$ will be illustrated in the Section 3.3.

When user representation vector and item representation vector are learned, top-$N$ items are recommended according to the likelihood function $p$ as

$$p(i|U, V, X) = P(\overrightarrow{e_u}, \overrightarrow{e_i}, \overrightarrow{e_x})$$ (3)
where \( N \) is the predefined number of items to be retrieved. \( \bar{e}_v \) is the embedding of item \( v \) from set of items \( V \). Our framework outputs the probabilities for all the items, which represent how likely the specific user will engage with the items, and retrieves top-\( N \) candidate items.

The objective function for training our model is to maximize the following log-likelihood:

\[
I = \sum_{(i,U,V,X) \in S} \log p(i|U,V,X)
\]  

(4)

We use the Adam optimizer to train our method.

### 3.2 Framework

The overall structure of our proposed framework FAT is illustrated in Figure 2, which is composed of a sequence modeling layer, an implicit neighbor behavior extractor, a diverse trends capture module, a time-aware attention layer and a final prediction layer. As the relative future sequence for current user is actually the history sequence for the neighbors, the future and history sequences can be modeled using shared parameters. Thus, we applied the sequential model for neighbors and users in the same manner. The framework takes the user historical interactions set \( X \) and item features set \( F \) as input. As for the extremely high dimension of item ID features, we adopt the widely-used embedding technique to embed these ID features into low-dimensional dense vectors. Here, we use \( s_u \) and \( S_u \) represent testing and training data of interactions sequence of user \( u \) respectively. For target item IDs from the set of \( S_u \), embeddings are presented as \( e_v \).

The implicit neighbor behavior extractor constructs neighbors set for each user by filtering out the users that have interactions with the target items in the past, and then select the relative future sequences of each neighbor user. The target items can be selected from the user historical interaction \( X_u \), for simplicity, we only choose the last one in the list, details are illustrated in Section 4.4. The relative future behaviors are defined as the interacted items following the target item in the chronological order, aiming at representing dynamic preference for the user based on the intuition that the user tend to have similar preference trend as users with similar historical behaviors, and that the user can have diverse trends of preferences.

The diverse trends capture module is developed to obtain the neighbor centroids according to diverse motivation of specific interactions of the items. Then we learn high dimensional embeddings for the historical behaviors and future-aware diverse trends behaviors separately. Furthermore, the future sequence representation acquired by time-aware attention layer is concatenated with the historical behavior representation to generate the dynamic user preference representation vector. Finally we compute the user’s preferences over different items from the pool by the prediction decoder. Each part will be elaborated in the following.

#### 3.3 Sequence Modeling

We first computes user embeddings from user historical behaviors and then the diverse trends embeddings from the implicit neighbor relative future behaviors. To capture the dynamics of interaction sequences, we apply RNN to compute embeddings for users. The input of our sequence modeling module is the user historical behavior sequence or the relative future behavior sequences from the extracted neighbors, which contains a list of item IDs representing the user’s interactions with items in time order. The number of item IDs is about billions, thus we adopt the widely-used embedding technique to embed these ID features into low-dimensional dense vectors, which significantly reduces the number of parameters and eases the learning process. The item IDs are fed into an embedding layer and transformed into item embeddings. To capture time-saving preferences of users, we then apply Recurrent Neural Network(RNN) to model the variable-length sequence data to compute the user embeddings. Particularly, we use Long Short-Term Memory cell as the basic RNN unit, which captures temporal dynamics. Each LSTM unit at time \( t \) consists of a memory cell \( c_t \), an input gate \( i_t \), a forget gate \( f_t \), and an output gate \( o_t \). These gates are computed from previous hidden state \( h_{t-1} \) and the current input \( x_t \):

\[
[f_t, i_t, o_t] = \text{sigmoid}(W[h_{t-1}, x_t])
\]

(5)

The memory cell \( c_t \) is updated by partially forgetting the existing memory and adding a new memory content \( l_t \):

\[
l_t = \tanh(V[h_{t-1}, x_t]) c_t = f_t \odot c_{t-1} + i_t \cdot l_t
\]

(6)

Once the memory content of the LSTM unit is updated, the hidden state at time step \( t \) is given by:

\[
h_t = o_t \odot \tanh(c_t)
\]

(7)

At time step \( t \), the new states of the user can be inferred as:

\[
h^u_{t,t} = \text{LSTM}(h^u_{t,t-1}, x^u_t)
\]

(8)

where \( h^u_{t,t} \) denote hidden states for the user. Note that LSTM can be replaced by other options, such as a Gated Recurrent Unit. In this paper, we select LSTM as it is a popular and general choice in previous works [45, 48, 54, 62].

Specifically, item embedding and user historical behavior embedding can be represented as \( E_V = \{e_v, v \in V\} \) and \( E_X = \{h^u, x \in X_u\} \), respectively. \( E_N \) denotes the relative future sequence of neighbors \( n \in N, x \in X \), \( E_N = \{e_v, v \in V, x \in E_N\} \) represent the implicit similar users future behavior embedding, which is the input of the future-aware diverse trends capture module, output diverse motivations behind behaviors representing diverse trends of user dynamic preferences, and then through the attention layer, we obtain a single aggregated trend representation vector of a specific user.

Lastly, corresponding user historical behavior embedding and the trend behavior embedding are concatenated to form the user preference embedding \( E_u \).

#### 3.4 Implicit Neighbor Behavior Extractor

Inspired by some works [19, 20], which extract social relationships in absence of explicit social networks [34], we compare the similarity among users via collaborative filtering to extract neighbor behaviors implicitly based on the historical interactions.

We adopt Pearson Correlation Coefficient [6] as:

\[
s_{ij} = \frac{\sum_{k \in I(i) \cap I(j)} (r_{ik} - \bar{r}_i) \cdot (r_{jk} - \bar{r}_j)}{\sqrt{\sum_{k \in I(i) \cap I(j)} (r_{ik} - \bar{r}_i)^2} \cdot \sqrt{\sum_{k \in I(i) \cap I(j)} (r_{jk} - \bar{r}_j)^2}}
\]

(9)
Future-Aware Diverse Trends Framework for Recommendation

Figure 2: Network Architecture of FAT. The left part of FAT illustrates the network of our base model. The Base model takes user historical behaviors as inputs, and outputs user preference representation vectors $h_u$ for prediction decoder. The right part of FAT consists of an Implicit Neighbor Behavior Extractor, a sequence modeling module same as the left part which outputs relative future preference $F_u$, a diverse trends capture layer and a time-aware attention layer.

where $I(i)$ is a set of items that interacted/rated by user $i$, $r_{ik}$ and $T_i$ represents the rate of user $i$ over item $k$ and the average rate of user $i$. The user similarity $s_i$ is ranging from $[-1, 1]$, and the similarity between users $i$ and $j$ is proportional to the value according to this definition. Following [33], we employ a mapping function $f(x) = (x + 1)/2$ to bound the range of PCC similarities into $[0, 1]$.

In the case of users with only one common item in history, PCC similarity gets 1 when the users’ preferences over the common item are similar and −1 when not, which encourages diversity of neighbors while damaging the fairness of similarity calculation. To tackle this issue, we only kept less than 20% of such neighbors to seek the balance.

In addition to the PCC method, we also design a filter with simple schema to extract similar users. For each user, if the historical interactions $I_u$ is split into two pieces, $\{s_{i}^{u} | t < |I_u|\}$ for training data, and $\{s_{i}^{u} | t+1:|I_u|\}$ for testing data, the item $s_t^{u}$ is defined as the last $K$ target items, $K$ could be any value less than or equal to $|S^u|$, while in practice $K = 1$ can achieve good enough performance with simplicity, details would be illustrated in the Section 4.4 where we do an ablation of $K$. We extracted a list of users $N = \{n_1, n_2, ..., n_N\}$ from the item-user map using the target item as key, which stands for all the users who have interacted with the target item. Furthermore, we constructed the future sequence of each neighbor user $u$ relative to the target item $s_t$ as:

$$F_u = \{s_i, s_j \in I_u, R(s_i) \geq R(s_t)\}$$  \hspace{1cm} (10)

where Timestamp is denoted as $R$ and $s_t$ is the same item as $s_t^u$.

3.5 Future-Aware Diverse Trends

We argue that representing user neighbors by one representation vector can be a bottleneck for capturing diverse neighbors of users, because we have to compress all information related with diverse neighbors of users into one representation vector. Thus, all information about diverse neighbors of users is mixed together, causing inaccurate neighbor retrieval and then the inaccurate item retrieval for the matching stage. Instead, we adopt multiple representation vectors to express distinct neighbors of users separately. By this way, diverse neighbors of users are considered separately in the matching stage, enabling more accurate neighbor retrieval as well as the item retrieval for every aspect of reasons.

We utilize clustering process to group neighbors(represented by historical behaviors of user’s diverse) extracted via previous multi-hop filter into several clusters. Neighbors from one cluster are expected to be closely related and collectively represent one particular aspect of user behaviors. Here, we design the multi-neighbor extractor layer for clustering historical behaviors and inferring representation vectors for resulted clusters.
Since the design of multi-neighbor extractor layer is inspired by MIND [8], which has already revisited essential basics of dynamic routing for representation learning in capsule network, we'll explain how our designed multi-neighbor extractor layer work based on it.

The objective of the multi-neighbor extractor layer is to learn representations for expressing properties of user behaviors as well as whether corresponding behaviors exist. The semantic connection between capsules and neighbor representations motivates us to regard the neighbor representations as neighbor capsules and employ dynamic routing to learn interest capsules from neighbor capsules. Nevertheless, the original routing algorithm proposed for image data is not directly applicable for processing user neighbor data. So, we propose Neighbor-to-interest dynamic routing for adaptively aggregating user’s neighbors into interest representation vectors, and it differs from original routing algorithm in three aspects.

Let $e_i$ be the capsule of the primary layer. We then give the computation of the capsule $j$ of the next layer based on primary capsules. We first compute the prediction vector as:

$$
\hat{e}_{ji} = W_{ij}e_i
$$

where $W_{ij}$ is a transformation matrix. Then the total input to the capsule $j$ is the weighted sum over all prediction vectors $\hat{e}_{ji}$ as:

$$
s_j = \sum_i c_{ij}\hat{e}_{ji}
$$

where $c_{ij}$ are the coupling coefficients that are determined by the iterative dynamic routing process.

We use "routing softmax" to calculate the coupling coefficients using initial logits $b_{ij}$ as:

$$
c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}
$$

where $b_{ij}$ represents the log prior probability that capsule $i$ should be coupled to capsule $j$. To ensure short vectors and long vectors to get shrunk to almost zero length and a length slightly below 1. Then the vector of capsule $j$ is computed by:

$$
v_j = squash(s_j) = \frac{\|s_j\|^2 s_j}{1 + \|s_j\|^2}
$$

where $s_j$ is the total input of capsule $j$.

The output trend capsules of the user $u$ are then formed as a matrix $V_u = [v_1, ..., v_K] \in \mathbb{R}^{d \times K}$ for downstream task.

### 3.6 Time-Aware Attention Layer

For history sequence representation, we simply use the output of the sequence modeling layer given the input of user’s history interactions list, which contains $K$ future potential sequence representations. Then we utilize the time-aware attention to activate the weight of diverse trends to capture the timeliness of each trend. Specifically, the attention function takes the interaction time of item $i$, the interaction time of trends and trend embeddings as the query, key and value respectively. We compute the final future sequence representation of user $u$ as:

$$
HF_u = Attention(T_i, T_r, \overrightarrow{t_u}) = \overrightarrow{t_u}softmax(pow(T_i, T_r))
$$

where Attention denotes the attention function, $T_i$ represents the interaction time of item $i$, $T_r$ represents the average interaction time of items related to the trend, $\overrightarrow{t_u}$ represents the embedding of the trend.

### 3.7 Prediction

After computing the trend embeddings from activated trends through time-aware attention layer, we concatenate it with the user historical behavior embedding to form a user preference embedding. Given a training sample $u, i$ with the user preference embedding and item embedding, we can predict the possibility of the user interacting with the item as

$$
p(i|u, V, X) = \frac{\exp(e_u^T e_i)}{\sum_{v \in V} \exp(e_u^T e_v)}
$$

### 4 EXPERIMENTS

In this section, we first cover the dataset and experimental settings. And then we conduct extensive experiments and in-depth analysis to verify the performance of FAT for recommendation.

#### 4.1 Dataset

We used three large benchmark datasets. The statistics of the three datasets are shown in Table 2.

- **Amazon Books**: This dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014. It includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs).
- **Steam**: This dataset contains more than 40k games from Steam shop with detailed data including reviews and information about which games were bundled together.
- **MovieLens-1M [22]**: One of the currently released MovieLens datasets, which contains 1,000,209 movie ratings from 6,040 users across 3,900 movies.

### Table 2: Statistics of the Datasets.

| Dataset           | users | items | interactions |
|-------------------|-------|-------|--------------|
| Amazon Books      | 459,133 | 313,966 | 8,898,041 |
| Steam             | 2,567,538 | 15,474  | 7,793,069  |
| MovieLens-1M      | 6,040  | 3,416  | 999,611     |

In each dataset, we partition user’s interactions into training, validation and test set by the proportion of 8 : 1 : 1. To avoid data sparsity, we filter out the users and items with only few interactions in our experiment. In the Movielens dataset, we keep users and items with at least 10 and 3 records respectively. In the Amazon Books dataset, we select users and items with at least 10 records each. In detail, we adopt a common setting of training sequential recommendation models. Let the behavior sequence of user $u$ be $X_u = \{s_{u1}, s_{u2}, ..., s_{u|X_u|}\}$. Each training sample uses the first $k$ behaviors of $u$ to predict the $(k + 1)$th behavior, where $k = 1, 2, ..., |X_u|$. 
To evaluate, we randomly select an interacted item by the user as target item for each user, while the items interacted before the target item are collected as the user behaviors.

4.2 Evaluation Metrics
To compare the performance of different models, we use Recall@N and NDCG@N (Normalized Discounted Cumulative Gain), where N is set to 20, 50 respectively as metrics for evaluation. In all these three metrics, a larger value implies better performance. Besides, we adopt per-user average for each metric.

- Recall@N: Number of correctly recommended items divided by the total number of all recommended items.
  \[
  \text{Recall}@N = \frac{1}{|U|} \sum_{u \in U} \frac{|\hat{I}_{u,N} \cap I_u|}{|I_u|}
  \]
  where \( \hat{I}_{u,N} \) denotes the set of top-N recommended items for user u and \( I_u \) is the set of testing items for user u.
- Normalized Discounted Cumulative Gain (NDCG): NDCG not only measures the percentage of correct recommended items, but takes the positions of correct recommended items into consideration.
  \[
  \text{DCG}@N = \frac{1}{|U|} \sum_{u \in U} \sum_{r \in R} \delta_{N}(r) \frac{G(r)}{\log_2(1 + i_r)},
  \]
  \[
  \text{NDCG}@N = \frac{\text{DCG}@N}{\text{IDCG}@N}
  \]
  where \( G \) denotes the ground-truth list, \( i_r \) is the index of r in R. \( \delta_{N}(\cdot) \) is an indicator function which returns 1 if item r is in top-N recommendation, otherwise 0. IDCG is the DCG of ideal ground-truth list which refers to the descending ranking of ground-truth list in terms of predicted scores.

4.3 Competitors
- GRU4Rec[28]: A typical sequential recommendation baseline being the first to propose the usage of recurrent neural networks in recommendation systems.
- YouTube DNN[11]: One of predominant deep learning models based on collaborative filtering systems incorporating with text and image information which have been successfully applied under industrial scenario.
- MIND[32]: A novel industrial applicable recommendation model to capture users’ multi-interest.
- ComiRec[8]: A novel controllable multi-interest framework which can be used in sequential recommendation.
- Base Model: We construct a base model of FAT by ignoring the diverse trends capture module and simply modeling user preferences from historical behaviors.

4.4 Results
We have trained our model with Adam utilizing the TensorFlow distributed machine learning system using 4 replicas on a Nvidia GPU. The model performance for the sequential recommendation is shown in Table 3. We run experiments to dissect the effectiveness of our recommendation model. We compare the performance of FAT with a baseline model of FAT and four state-of-the-art models: GPU4Rec, YouTube DNN, MIND and ComiRec. All these models are running on the three datasets introduced above: Amazon Books, Steam and MovieLens. According to the results shown in Table 3, our model FAT obtain better performance on all evaluation metrics of all the tasks than other models.

As shown in Table 3, Recall and NDCG for the dataset MovieLens of all the models are higher than that for dataset Amazon Books and Steam. It’s caused by the unbalanced size between the datasets that size of the MovieLens is much smaller than the other two datasets.

Table 4 reports the performance of our model FAT in different parameter setting by changing number of trends T. We list the performance result of our model for the three datasets setting T to 2, 4, 6 and 8. Our model achieves improvements on T = 6 over T = 4, which may caused by insufficient trends for the dataset. However, it did not show much improvement when we change T from 6 to 8 showing clustering sequences to 8 trends in these datasets is redundant and to 6 trends is just suitable.

Table 5 compares the result of setting target item from the first to last item, the third to last item and the fifth to last one. The largest improvements appear on increasing K from 1 to 3. This demonstrates that by adding the number of target items, our model can capture more trend information and be more powerful to predict future sequences. Increasing target item number from 3 to 5 does not gain much improvement. This implies our model is efficient to capture much trend information by few historical items of the user.

Table 6 summarizes the recommend diversity performance of baseline models and our models on the three datasets.

The computational complexity of sequence layer modeling user and neighbors is \( O(kn^2d^2) \), where \( k \) denotes the number of extracted neighbors, \( n \) denotes the average sequence length and \( d \) denotes the dimension of item’s representation. Capsule layer’s computational complexity depends on kernel size and number of trends. Average time complexity of capsule layer scales \( O(nTr^2) \), where \( r \) denotes kernel size of capsule layer and \( T \) denotes the number of trends. For large-scale applications, our proposed model could reduce computational complexity by two measures: (1) encode neighbors with a momentum encoder[23], (2) adopt a light-weight Capsule network.

![Figure 3: Heatmap of coupling factors for items recalled by each trend. Each item has the coupling factor on the corresponding trend. The color depth is proportional to the numerical value of the coupling factor.](image-url)
Table 3: Model Performance on public datasets: Amazon books, Steam and MovieLens. FAT is our model and Base model is FAT without diverse trends. Here, we set \( K = 1 \) and \( T = 6 \)

| Model     | Amazon Books | Steam | MovieLens |
|-----------|--------------|-------|-----------|
|           | Metrics@20   | Metrics@50 | Metrics@20 | Metrics@50 | Metrics@20 | Metrics@50 |
|           | Recall       | NDCG   | Recall       | NDCG   | Recall       | NDCG   | Recall       | NDCG   | Recall       | NDCG   | Recall       | NDCG   |
| GRU4Rec   | 3.670        | 5.575  | 5.328        | 7.075  | 2.771        | 3.601  | 3.415        | 4.224  | 23.028       | 23.875 | 33.233       | 33.636 |
| YouTube DNN | 3.933        | 5.703  | 6.612        | 7.623  | 2.812        | 3.711  | 3.667        | 4.373  | 23.676       | 24.102 | 33.592       | 33.847 |
| MIND      | 4.102        | 5.933  | 6.638        | 7.830  | 3.213        | 4.331  | 3.671        | 4.591  | 24.750       | 25.853 | 33.685       | 34.783 |
| ComiRec   | 4.853        | 6.185  | 7.203        | 8.120  | 3.464        | 4.142  | 3.977        | 4.792  | 24.883       | 25.896 | 33.955       | 34.984 |
| Base Model | 3.126        | 4.912  | 4.872        | 6.721  | 2.481        | 3.812  | 3.217        | 4.123  | 23.101       | 23.879 | 33.315       | 33.132 |
| FAT       | 4.923        | 6.612  | 7.882        | 8.882  | 3.428        | 4.543  | 4.017        | 4.954  | 25.016       | 26.502 | 34.288       | 35.166 |

Table 4: Model Performance of parameter sensitivity. \( T \) denotes the number of trends. \( T = 2 \) means relative future sequence from neighbors are clustered to two trends

| Model     | Amazon Books | Steam | MovieLens |
|-----------|--------------|-------|-----------|
|           | Metrics@50   | Recall | NDCG | Recall | NDCG | Recall | NDCG | Recall | NDCG |
| FAT(\( T = 2 \)) | 4.042 | 4.428 | 2.232 | 2.623 | 20.329 | 22.981 |
| FAT(\( T = 4 \)) | 5.993 | 6.728 | 3.017 | 3.693 | 27.981 | 28.989 |
| FAT(\( T = 6 \)) | 7.882 | 8.882 | 4.017 | 4.954 | 34.288 | 35.166 |
| FAT(\( T = 8 \)) | 6.982 | 7.842 | 3.107 | 4.125 | 31.973 | 32.887 |

Table 5: Model Performance of Implicit Neighbor Behavior Extractor with ablation of \( K \) for the target items(setting \( T = 6 \)). \( K = 3 \) means the last item, the second to last time and the third to last time are taken into consideration

| Model     | Amazon Books | Steam | MovieLens |
|-----------|--------------|-------|-----------|
|           | Metrics@50   | Recall | NDCG | Recall | NDCG | Recall | NDCG | Recall | NDCG |
| FAT(\( K = 1 \)) | 7.882 | 8.882 | 4.017 | 4.954 | 34.288 | 35.166 |
| FAT(\( K = 3 \)) | 8.512 | 9.343 | 5.431 | 5.735 | 35.890 | 36.192 |
| FAT(\( K = 5 \)) | 8.482 | 8.912 | 5.117 | 5.654 | 35.888 | 35.894 |

Table 6: Model Recommendation Diversity with \( K = 4 \)

| Model     | Amazon Books | Steam | MovieLens |
|-----------|--------------|-------|-----------|
|           | Diversity    | Diversity | Diversity | Diversity |
| GRU4Rec   | 36.783       | 40.648 | 20.875 |
| YouTube DNN | 38.604       | 42.831 | 23.654 |
| MIND      | 39.967       | 44.984 | 27.502 |
| ComiRec   | 42.915       | 45.947 | 28.961 |
| Base Model | 33.946       | 35.484 | 15.751 |
| FAT       | 43.591       | 46.653 | 29.274 |

Figure 4: The distribution of items recalled by each trend (trend 0,1,2,3) and base model without trend (no trend). The coordinates indicate the recommendation level of items, where 1 signifies the corresponding item is most recommended while 0 matches to the least recommended one. The radius of each circle is proportional to the number of items.

systems are trained to help users to select items which would be interesting to them without much historical interactions between the user and the items. Recommender systems tracks the interaction between the users and their selected items. This information is then processed to train the recommendation model which can not only recommend similar items but also recommend items of similar hidden connection.
Many authors have undertaken research developing new diversification algorithms [1, 5, 13, 35]. Our proposed module can learn the diverse trends of user preference and provide recommendation with diversity. Following [8], we use the following definition of individual diversity:

\[
\text{Diversity}_{@N} = \frac{\sum_{j=1}^{N} \sum_{k=j+1}^{N} \delta(CATE(i_{u,j}) \neq CATE(i_{u,k}))}{N \times (N - 1)/2}
\]  

(20)

where \( CATE \) represents the category of the item, \( i_{u,j} \) denotes item recommended for user \( u \), \( j \) and \( k \) represents the order of the recommended items. \( \delta(\cdot) \) is an indicator function.

Table 6 shows the diversity of models on different datasets when we control the factor \( K = 4 \). From the table, our module FAT achieve the optimum diversity indicating the recommendation it provide can effectively take neighbors’ interests into account.

4.6 Case Study

4.6.1 Coupling Factors. The coupling factors between trends and items are proportional to the correspondence between them. In this section, we visualize these factors to show that the trend capture process is interpretable.

As shown in Figure 3, the coupling factors associated to the user randomly selected from Amazon Book dataset, where each row corresponds to one trend capsule and each column corresponds to one interaction after the selected target item. It shows that user X has interacted with 3 kinds of books (history, science, art) after interacting with the books of history category. Each of the future interactions has the max coupling factors on one trend capsule and forms the corresponding trend.

4.6.2 Distribution. We draw a trend distribution Figure 4 of recommended items recalled by each trend interest based on their similarity to the corresponding interest. Figure 4 shows the recommended item distribution for a user. X axis is the similarity and images are recommended item. The size of the circle demonstrate the recommended rate. As shown, the items recalled are correlated with trend interests.

5 CONCLUSIONS

In this paper, we propose a novel Future-Aware Diverse Trend (FAT) framework to capture diverse trends of user preference dynamically. Our frame work leverages a neighbor behavior extractor to generate relative future interactions from similar users implicitly and utilizes diverse trends module to capture intrinsic varying dynamics of user preferences. To improve the expressive ability of trend representation, we utilize time-aware attention layer to make the duration between prediction time and target item interaction time choose which trend is more relative. Experimental results demonstrate that our models can achieve significant improvements over state-of-the-art models on three challenging datasets. For the future, we plan to leverage multi-hop user-item graphs to address limited interaction issues and incorporate multi-behavior data into neighbors extraction to better model potential trends.

REFERENCES

[1] G. Adomavicius and Y. Kwon. 2012. Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. IEEE Transactions on Knowledge and Data Engineering 24, 5 (2012), 896–911.
[2] Robert Bell and Yehuda Koren. 2007. Improved neighborhood-based collaborative filtering. (09 2007).
[3] Homanga Bharadwaj and Shruti Joshi. 2018. Explanations for Temporal Recommendations. arXiv:1807.06616 [cs:AI]
[4] Veronika Bogina and Tsvi Kuflik. 2017. Incorporating Dwell Time in Session-Based Recommendations with Recurrent Neural Networks. In RecTempRecSys.
[5] Rubí Boim, Tova Milo, and Slava Novgorodov. 2011. Diversification and Refinement in Collaborative Filtering Recommender. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management (Glasgow, Scotland, UK) CIKM ’11. Association for Computing Machinery, New York, NY, USA, 739–744. https://doi.org/10.1145/2063576.2063684
[6] John S. Breese, David Heckerman, and Carl Kadie. 2013. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. arXiv:1301.7363 [cs:IR]
[7] Iván Cantador, Alejandro Bellogin, and David Vallet. 2010. Content-based recommendation in social tagging systems. RecSys ’10 - Proceedings of the 4th ACM Conference on Recommender Systems, 237–240. https://doi.org/10.1145/1864708.1864756
[8] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable Multi-Interest Framework for Recommendation. arXiv:2005.09347 [cs:IR]
[9] T. Chen, R. Xu, Y. He, Y. Xia, and X. Wang. 2016. Learning User and Product Distributed Representations Using a Sequence Model for Sentiment Analysis. IEEE Computational Intelligence Magazine 11, 3 (2016), 34–44.
[10] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jianguo Tang, Yuxin Cao, Zheng Qin, and Hongyan Zha. 2018. Sequential Recommendation with User Memory Networks (WSDM ’18). Association for Computing Machinery, New York, NY, USA, 108–116. https://doi.org/10.1145/3159652.3159668
[11] Paul Cowling, Jay Adams, and Emze Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, New York, NY, USA.
[12] Robin Devosd and Hugues Bersini. 2017. Long and Short-Term Recommendations with Recurrent Neural Networks (UMAP ’17). Association for Computing Machinery, New York, NY, USA, 13–21. https://doi.org/10.1145/3079628.3079670
[13] Tommaso Di Noia, Vito Claudia Ostuni, Jessica Rosati, Paolo Tomese, and Eugenio Di Sciascio. 2014. An Analysis of Users’ Propensity toward Diversity in Recommendations. In Proceedings of the 8th ACM Conference on Recommender Systems (Foster City, Silicon Valley, California, USA) (RecSys ’14). Association for Computing Machinery, New York, NY, USA, 285–288. https://doi.org/10.1145/2645710.2645774
[14] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential User-Based Recurrent Neural Network Recommendations. In Proceedings of the Eleventh ACM Conference on Recommender Systems (Como, Italy) (RecSys ’17). Association for Computing Machinery, New York, NY, USA, 152–160. https://doi.org/10.1145/3109859.3109877
[15] Xinyu Duan, Siliang Tang, Sheng-yu Zhang, Yin Zhang, Zhou Zhao, Jian-rui Xue, Yueying Zhuang, and Fei Wu. 2018. Temporality-enhanced knowledge memory network for factoid question answering. Frontiers Inf. Technol. Electron. Eng. (2018).
[16] Xinyu Duan, Sheng-yu Zhang, Zhou Zhao, Fei Wu, and Yueying Zhuang. 2018. Multi-Label Community-Based Question Classification via Personalized Sequence Memory Network Learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018.
[17] Ali Mammadov Elkahky, Yang Song, and Xiaodong He. 2015. A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems. In Proceedings of the 24th International Conference on World Wide Web (Florence, Italy) (WWW ’15). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 278–288. https://doi.org/10.1145/2736277.2741567
[18] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In The World Wide Web Conference (San Francisco, CA, USA) (WWW ’19). Association for Computing Machinery, New York, NY, USA, 417–426. https://doi.org/10.1145/3308558.3313488
[19] Cricia Felicio, Klérisson Pásao, Guilherme Alves, Sandra Amo, and Philippe Preux. 2016. Exploiting Social Information in Pairwise Preference Recommender. In Proceedings of the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), Florence, Italy

WWW ’21, April 19–23, 2021, Ljubljana, Slovenia
