Trending Now: Modeling Trend Recommendations

Hao Ding[1], Branislav Kveton[1], Yifei Ma[1], Youngsuk Park[1], Venkataramana Kini[2], Yupeng Gu[1], Ravi Divvela[2], Fei Wang[2], Anoop Deoras[1], Hao Wang[1]
[1]AWS AI Labs, [2]Amazon
USA
{haodin,bkveton,yifeim,pyoungsu,venkini,yupenggu,rdivvela,feiww,adeoras,howngz}@amazon.com

ABSTRACT

Modern recommender systems usually include separate recommendation carousels such as 'trending now' to list trending items and further boost their popularity, thereby attracting active users. Though widely useful, such 'trending now' carousels typically generate item lists based on simple heuristics, e.g., the number of interactions within a time interval, and therefore still leave much room for improvement. This paper aims to systematically study this under-explored but important problem from the new perspective of time series forecasting. We first provide a set of rigorous definitions related to item trendiness and formulate the trend recommendation task as a one-step time series forecasting problem. We then propose a deep latent variable model, dubbed Trend Recommender (TrendRec), to forecast items' future trends and generate trending item lists. Furthermore, we design associated evaluation protocols for trend recommendation. Experiments on real-world datasets from various domains show that our TrendRec significantly outperforms the baselines, verifying our model's effectiveness.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Recommender Systems, Trend Recommendation, Bayesian Deep Learning

ACM Reference Format:
Hao Ding[1], Branislav Kveton[1], Yifei Ma[1], Youngsuk Park[1], Venkataramana Kini[2],, Yupeng Gu[1], Ravi Divvela[2], Fei Wang[2], Anoop Deoras[1], Hao Wang[1]. 2023. Trending Now: Modeling Trend Recommendations. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3604915.3608810

1 INTRODUCTION

Nowadays, recommender systems (RecSys) have been widely adopted in real-world scenarios [12, 13, 15, 19, 33, 40]. It is common to design a variety of recommendation carousels on the homepages of websites or mobile apps to explore and capture different perspectives of user interests, such as 'recommended for you' [24], 'buy it again' [7], 'frequently bought together' [10, 37], etc. In this work, we study an important yet under-explored class of recommendation carousels called 'trending now', which identify the trends at the current moment and promote trending items to the users.

Recommender systems are known to suffer from the 'rich-get-richer' problem, where popular items are more likely to be recommended [1, 34]. While the global or recent popular items are frequently captured by recommendation carousels such as 'recommended for you' (due to popularity bias [41, 43]) and 'recent popular', we hope the 'trending now' carousel can provide a complementary perspective by promoting items that are not frequently appearing in other carousels, but are *rising fast in popularity* and *have the potential to become popular in the near future.*

Despite its significance in practice, there is surprisingly limited previous research on modeling trend in the recommendation context. Existing related works [2, 4, 23, 26, 42] focus on either modeling trend to improve personalized recommendation or identifying trends in a non-recommendation scenario such as social media; none of them focus on recommending trending items. To bridge this gap, in this paper, we formally define the notion of 'trend', formulate the trend recommendation task for our specific use case, and develop new models tailored for this task.

Specifically, we first define *popularity* as the number of interactions at a certain time interval. We then define 'trend' in the recommendation context as *the change rate of popularity*, or *acceleration*. In line with the definitions above, the primary target of the 'trending now' carousel is to identify and promote items that are receiving increasingly more interactions at the current moment, while these items are not necessarily the global or recent most popular items. This can also be viewed as conducting an effective item exploration without popularity bias. In fact, through our data analysis, we observed a relatively low overlap between the popular items and the trending items based on our definition.

It is desirable to capture the current trend instantly and recommend trending items in real time. However, in reality, a certain amount of time is required to collect sufficient interactions in order to reliably and stably identify the current trends, while the trends are changing dynamically and may drift during data collection period. Given this challenge, we formulate the trend recommendation task as a *one-step forecasting problem*. Fig. 1(left) demonstrates our problem setting: given items' historical accelerations, we aim to predict which items will be trendy in a short future time interval, i.e., the *next time step*. This formulation is closely aligned with online real-world scenarios: once the model predicts trending items at the next time step, we display them to users throughout the next time step while buffering data at the back-end. At the end of the next time step, the model makes new predictions for the time step right after it based on newly accumulated data. We then refresh the recommendations and start another cycle of data collection.

RecSys '23, September 18–22, 2023, Singapore, Singapore

ACM ISBN 979-8-4007-0241-9/23/09.
https://doi.org/10.1145/3604915.3608810

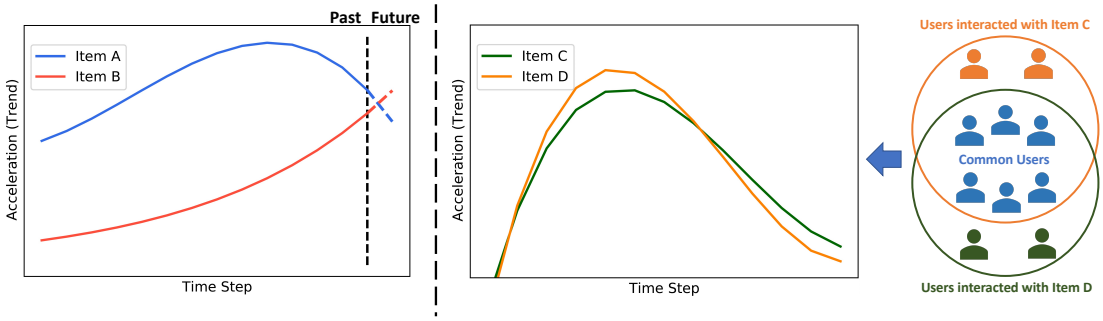


Figure 1: Left: Our problem definition on trend recommendation, where given items' historical accelerations, we predict their accelerations at the next time step and rank them accordingly for trend recommendation. In this example, the rank order would be [item B, item A]. Right: Motivation for TrendRec: two items with large percentage of overlapping users exhibit similar trends.

So, how should we design a trend recommender that is optimized under our specific use case? Naturally, one could build on existing time series forecasting models. There is a rich literature on this topic. Among them, the state of the art is sequential deep learning, which adaptively learns how to effectively ingest historical time series as context and predicts future [18, 20, 22, 28, 31, 35, 39].

On top of time series forecasting models, the unique property of the recommendation context offers interesting new opportunities for adapting these models in our scenario. Specifically, beyond just coarse cumulative numbers of interactions for each item in the time series, there exist more fine-grained user-item interactions in our setting. In other words, for a given item, we not only know how many users have interacted with it, but also know who exactly these users are. This brings additional information to help us discover the underlying correlations between items; items with a large number of overlapping users may share common trend patterns, and therefore improve the trend forecasting accuracy. Fig. 1(right) illustrates our modeling motivation.

In this paper, we design a principled model, dubbed TrendRec, for trend recommendation. Our model can be instantiated with any deep learning based time series forecasting models, and is capable of leveraging user-item interactions as additional context. TrendRec is a two-phase framework and is trained step-by-step on both the auxiliary objective of next item recommendation as well as the primary time series forecasting objective of next-step acceleration prediction. Specifically, through initial training on the next item recommendation objective, TrendRec leverages user-item interactive signals to detect the underlying correlations across items and encode such knowledge as item embeddings. In the second phase training, the learned item embeddings are used to provide additional context on the respective time series for the time series forecasting objective. Our main contributions are summarized as follows:

- We formally define the notion of 'trend' in the recommendation context, and establish corresponding evaluation metrics and evaluation process.
- We identify a bias-variance tradeoff that prevents the model from discovering the trend timely and stably: variance introduced by data sparsity, and bias introduced by temporal drift. Based on this observation, we then formulate the trend recommendation task as a one-step time series forecasting problem.
- Grounded on the unique property of the recommendation scenario, we develop a principled two-phase model, dubbed TrendRec, which leverages the user-item interactive signals to uncover the underlying correlations across items and ingest such knowledge to facilitate trend forecasting.
- Extensive experiments on datasets from a wide range of domains including retail, media, and news demonstrate the robustness and effectiveness of the proposed model.

2 PRELIMINARIES

In this section we first introduce trend-related terms and trend definition, followed by our formal definition of trend recommendation problem, and then we introduce some basic models which will be used as baselines in the experiment section.

2.1 Defining Acceleration as Trend

We first define the following terms that will be used in the rest of the paper:

(1) **Time Step**: The time step is a time interval with a predefined duration Δt (e.g., one hour). We denote the time step with time interval $[t \cdot \Delta t, (t+1) \cdot \Delta t]$ as time step t, where t is the time step index.
(2) **Velocity**: For an item j, we define its number of interactions collected during time step t as its velocity at time step t, and denote it as $\mathbf{W}_{jt} \in \mathbb{R}$. Here $\mathbf{W}_{jt}$ represents item j's popularity per unit time Δt.
(3) **Acceleration**: We denote the acceleration for item j at time step t as $\mathbf{A}_{jt} \in \mathbb{R}$, and compute it as $\mathbf{A}_{jt} = \Delta\mathbf{W}_{jt} = \mathbf{W}_{jt} - \mathbf{W}_{j(t-1)}$. Here $\mathbf{A}_{jt}$ represents that item j's velocity is changing $\Delta\mathbf{W}_{jt}$ per unit time Δt.

Trend Definition. For an item j, we define its trend at time step t to be its acceleration at time step t, which is $\mathbf{A}_{jt}$. An item j is considered as trendy at time step t if its acceleration $\mathbf{A}_{jt}$ is among the highest of all items' accelerations at time step t. In the following part, the terms 'acceleration' and 'trend' will be used interchangeably as they are equivalent in our definition.

Why define acceleration as trend in the recommendation regime? The trending now carousel is expected to recommend a complementary group of items, which rarely overlap with items recommended by other carousels, yet are receiving increasingly more interactions. These trending up items are not necessarily the global or recent most popular items. Recommending items with large accelerations allows us to conduct an effective item exploration on promising trending up items that may not be popular yet, raising their exposure rate to further facilitate their trends. Some exemplar types of targeting items are as follows:

- Recently released cold items of good quality (e.g., a new episode of Game of Thrones).
- Items experiencing sudden changes (e.g., a movie won an Oscar award and abruptly becomes trendy).
- Long lasting items with periodic up-trend (e.g., items with seasonal effect such as winter clothes).

2.2 Problem Definition

Ideally, we promptly detect the current trend and recommend trending items in real time. Nevertheless, in practice, a certain amount of time is necessary for accumulating adequate interactions to ensure reliable identification of the current trend, while trends are subject to dynamic variations and may drift during the course of data collection. In other words, we hope the time interval Δt to be small enough so that $\mathbf{A}_{jt}$ reflects the trend at the current moment, but with a small Δt, $\mathbf{A}_{jt}$ can be noisy due to insufficient data collected during Δt. Therefore, we argue that instead of capturing and surfacing trending items in real-time, it is more feasible and reasonable to predict and display trending items in a short future time interval, while frequently update our recommendations based on the newly collected data.

Based on the observation above, we define the trend recommendation task as a one-step time series forecasting problem. Formally, for each item, given its historical acceleration $[\mathbf{A}_{j0}, \mathbf{A}_{j1}, \ldots, \mathbf{A}_{jt}] := \mathbf{A}_{j,0:t}$ and additional contextual information $[\mathbf{C}_{j0}, \mathbf{C}_{j1}, \ldots, \mathbf{C}_{jt}] := \mathbf{C}_{j,0:t}$ such as covariates [31], we hope to predict its acceleration at the next time step $t+1$

$$\mathbf{P}(\mathbf{A}_{j(t+1)} | \mathbf{A}_{j,0:t}, \mathbf{C}_{j,0:t}), \tag{1}$$

and we recommend the top k items based on the trend prediction.

Hypothesis on correlations between the time step length and the task feasibility. It is worth noting that different lengths of time steps suggest different sets of ground-truth items (e.g., whether to predict the top k items with highest acceleration in the next one hour or in the next one day), and thus will be mapped to different problems of different difficulty levels. The correlations between the task feasibility and the time step length is postulated as the curve shown in Fig. 2, which essentially depicts a bias-variance tradeoff: when the time step length is small (e.g., one hour), variance is introduced due to data sparsity; when the time step length is large (e.g., one day), bias is introduced due to temporal drift. Therefore, we need to find a 'sweet spot' that strikes a good balance between the two.

Later in Sec. 4.3 we propose two evaluation metrics for acceleration which are closely aligned with our trendiness definition and the problem formulation for trend recommendation. The above hypothesis is validated by experimental results based on the proposed metrics (refer to Sec. 4.5 for more details) on a variety of datasets across different domains. We further develop an adaptive mechanism to select the most appropriate time interval Δt for each dataset.

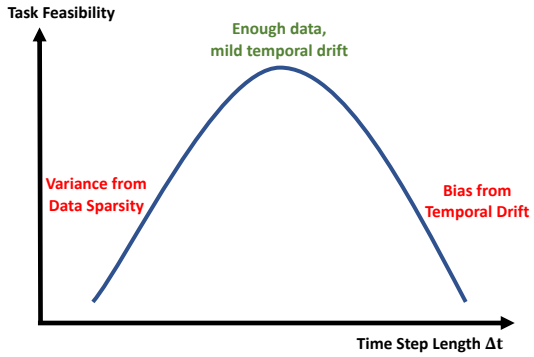


Figure 2: Hypothesis on the correlation between the task feasibility and the time step length Δt. When Δt is small, the trend can be noisy due to insufficient collected data, introducing variance; when Δt is large, the trend may already changed due to the long process of data collection, leading to bias. We therefore propose a bias-variance tradeoff over the choice of Δt with the goal of finding a balance. This hypothesis is verified by experimental results in Sec. 4.5.

2.3 Baseline Models

Here we present various basic models for trend recommendation. Based on the problem definition in Sec. 2.2, it is natural to leverage existing methods designed for the time series forecasting task. We first describe two widely adopted heuristic models and then introduce a generic form of deep learning based probabilistic time series forecasting models. All models introduced here will be used as baselines in the experiment section.

2.3.1 Markov Heuristic Model. we assume an arbitrary item j's acceleration at the next time step $\mathbf{A}_{j(t+1)}$ only depends on its acceleration at the current time step $\mathbf{A}_{jt}$. Since in reality the accelerations are prone to remain the same over short period of time, we define the Markov heuristic model as:

$$\hat{\mathbf{A}}_{j(t+1)} = \mathbf{A}_{jt}, \tag{2}$$

where $\hat{\mathbf{A}}_{j(t+1)}$ is the predicted acceleration at the next time step. Note that the Markov heuristic model is essentially a special case of AR [9].

2.3.2 Exponential Moving Average (EMA) Heuristic Model. The main drawback of the Markov heuristic model is that the item acceleration at the next time step is exclusively influenced by the current time step, which can be noisy due to issues such as data sparsity. We therefore hope to take multiple latest time steps into consideration, assigning more weights to the more recent time steps. This leads us to define the Exponential Moving Average (EMA) heuristic model as below:

$$\hat{\mathbf{A}}_{j(t+1)} = \sum_{k=0}^{T-1} w_k \cdot \mathbf{A}_{j(t-k)}, \tag{3}$$

where $\hat{\mathbf{A}}_{j(t+1)}$ is the predicted acceleration at the next time step, T is the number of recent time steps the model is considering, and w_k is the predefined weight which decays exponentially with the increasing number of steps away from the current time step. Note that the EMA model can be viewed as a special case of ARIMA [8].

2.3.3 Deep Learning based Time Series Forecasting Model. The heuristic models typically encode general assumptions that lack the flexibility to adapt to diverse scenarios. However, the acceleration patterns vary across domains (retail, media, news, etc.). For example, there exist abundant periodical acceleration patterns of different cadences (daily, weekly, seasonal, etc.) in both retail and media domains, such as a new episode of a TV series is released every Wednesday. On the contrary, such regular acceleration patterns are rarely observed in the news domain, as news are time-sensitive and people tend to follow most recent news. Additionally, even within the same domain, various acceleration patterns may coexist. For example, on a specific movie platform, the acceleration curve of newly released action movies may consistently exhibit a steeper increase compared to that of newly released documentary movies due to the preference of the user community on that platform.

A more general solution for trend recommendation is therefore to design a learnable deep learning based time series forecasting model that is adaptive to a variety of scenarios. We formulate the model as:

$$\mathbf{P}(\mathbf{A}_{j(t+1)}|\mathbf{A}_{j,0:t}, \mathbf{C}_{j,0:t}) = f_{seq}(\mathbf{A}_{j,0:t}, \mathbf{C}_{j,0:t}), \tag{4}$$

where $f_{seq}(\cdot)$ is a sequential model that aggregates the historical accelerations and projects the probabilistic distribution of the acceleration at the next time step. There is a rich body of literature under the topic of deep learning based sequential time series forecasting which adopted different sequential models $f_{seq}(\cdot)$, such as DeepAR [31] based on RNN, MQCNN [35] based on CNN, TFT [22] based on transformer, and so on.

3 TRENDREC: COLLABORATIVE TIME SERIES FORECASTING MODEL WITH USER-ITEM INTERACTIONS

In this section, we present our TrendRec, a principled two-phase model which detects item correlations from user-item interactions to facilitate the trend forecasting. It is compatible with any deep learning based time series forecasting models.

3.1 Model Overview

Unlike typical time series forecasting scenarios where the input data is one or multiple series of values (e.g., a time series of daily product consumption of a company) [22, 31, 35], the unique recommendation context promises more fine-grained user-item interactions instead of coarse cumulative number of interactions for each item at each time step. Such user-item interactions provide extra information beyond the number of interactions of an item, as they also reveal the exact group of users who interacted with the item. It is a reasonable assumption that items with lots of overlapping interactive users are likely to have common item properties and targeting users, therefore possess similar trend (acceleration) patterns. For example, 'The Avengers' and 'Justice League' are both superhero movies that appeal to a similar audience, and have a significant number of overlapping viewers. As a result, they have very similar trends: a strong opening weekend as their fan base rushes to watch them, followed by a gradual decline in the following weeks.

To distill knowledge from user-item interactions, we design an auxiliary training objective on next item recommendation task. This objective facilitates representation learning on item properties and yields dense latent item embeddings. The generated latent item embeddings envelop underlying correlations between items which are implied by user-item interactions, and can be employed to facilitate the time series forecasting objective by providing additional context for the corresponding time series.

Overall, TrendRec encompasses two components: a recommendation model and a time series forecasting model. The recommendation model is trained on the next item recommendation objective with user-item interactions. The time series forecasting model is trained on the next-step acceleration prediction objective with time series of items' accelerations. The two components are connected through the shared latent item embeddings which encode item correlations.

Fig. 3 shows the probabilistic graphical model (PGM) for TrendRec, which is partially inspired by collaborative deep learning (CDL) [33] and ZESRec [12]. It integrates two training objectives: (1) next item recommendation (right part), and (2) item acceleration forecasting (left part). Below we explain its rationale in detail:

- Variables $\mathbf{V}_{jt} \in \mathbb{R}^D$, $\mathbf{A}_{j,0:t} \in \mathbb{R}^{N_{jt}}$ represent item j's properties till time step t (both static properties and dynamic properties) and item j's historical acceleration till time step t which is $[\mathbf{A}_{j0}, \mathbf{A}_{j1}, \ldots, \mathbf{A}_{jt}]$, respectively. N_{jt} denotes the number of historical time steps of item j till time step t. D is the hidden dimension of the embedding.
- Variables $\mathbf{U}_{it} \in \mathbb{R}^D$ and $\mathbf{S}_{it} \in \mathbb{R}^{N_{it} \times D}$ represent user i's interests till time step t and user i's historical interaction sequence till time step t, respectively. Here N_{it} is the number of interactions from user i till time step t. $\mathbf{S}_{it}$ is an embedding matrix and each row of it represents an item embedding.
- Variable $\mathbf{R}_{ijt} \in \{0, 1\}$ is the interaction label denoting whether user i interacted with item j at time step t.
- Variable $\mathbf{A}_{j(t+1)} \in \mathbb{R}$ is the acceleration of item j at the next time step $t + 1$.
- Edge $\mathbf{S}_{it} \rightarrow \mathbf{U}_{it}$: User's previous interactions reveal the user interests and affect the user's next action (e.g., a user who purchased a cell phone may want to purchase its accessories next).
- Edge $\{\mathbf{U}_{it}, \mathbf{V}_{jt}\} \rightarrow \mathbf{R}_{ijt}$: Interaction depends on user interests $\mathbf{U}_{it}$ and item properties $\mathbf{V}_{jt}$.
- Edge $\{\mathbf{V}_{jt}, \mathbf{A}_{j,0:t}\} \rightarrow \mathbf{A}_{j(t+1)}$: The acceleration of item i at the next time step $t+1$ is influenced by item's properties (e.g., action movie is more likely to become trendy in the user community of a specific website) and item's historical acceleration (e.g., an item with periodical weekly trend pattern).

Here $\mathbf{U}_{it}, \mathbf{V}_{jt}$ are the learnable embeddings with the same hidden dimension D, and we term them, *latent user embedding* and *latent item embedding*, respectively. λ_u and λ_v, are hyperparameters related to distribution variance. The corresponding conditional probabilities in the PGM are listed in Eqn. 5. In general, TrendRec is an organic combination of a recommender system (RecSys) and a time

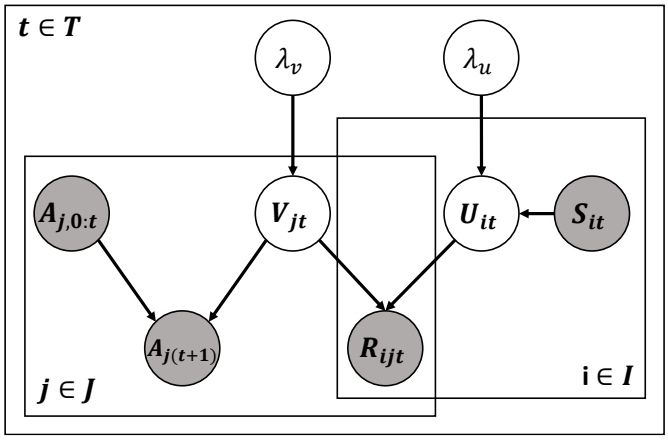


Figure 3: The probabilistic graphical model (PGM) for our TrendRec. Overall, $\mathrm{U}_{it}, \mathrm{V}_{jt}$ are the hidden variables, λ_u and λ_v are the hyperparameters related to the distribution variance of U_{it} and V_{jt}, respectively, while S_{it}, R_{ijt}, $\mathrm{A}_{j(t+1)}$, and $\mathrm{A}_{j,0:t}$ are the observed variables. Specifically, U_{it} and S_{it} represent user i's interests till time step t and user i's historical interaction sequence till time step t, respectively. V_{jt} and $\mathrm{A}_{j,0:t}$ represent item j's properties till time step t and item j's historical accelerations till time step t, respectively. R_{ijt} is the interaction label denoting whether user i interacted with item j at time step t, and $\mathrm{A}_{j(t+1)}$ is the item j's acceleration at the next time step $t+1$. Time step t represents the time interval $[t \cdot \Delta t, (t+1) \cdot \Delta t]$. I, J, T denote number of users, number of items, and number of time steps, respectively. Note that we define $\mathrm{P}(\mathrm{U}_{it}|\mathrm{S}_{it}, \lambda_u) = \mathcal{N}(\mathrm{U}_{it}; f_{\boldsymbol{seq}}(\mathrm{S}_{it}), \lambda_u^{-1}\mathrm{I}_D))$ and $\mathrm{P}(\mathrm{A}_{j(t+1)}|\mathrm{V}_{jt}, \mathrm{A}_{j,0:t}) = f_{\boldsymbol{ts}}(\mathrm{V}_{jt}, \mathrm{A}_{j,0:t})$, and both $f_{\boldsymbol{seq}}(\cdot)$ and $f_{\boldsymbol{ts}}(\cdot)$ are learnable functions. For simplicity we exclude their corresponding learnable parameters in the PGM.

series forecasting model. The PGM for classic sequential RecSys models such as [15, 19, 32] is essentially the subgraph of Fig. 3 with nodes $S_{it}, U_{it}, R_{ijt}, V_{jt}, \lambda_u$, and λ_v. The PGM for classic sequential time series forecasting models such as [22, 31, 35] is essentially the subgraph of Fig. 3 with nodes $\mathbf{A}_{j,0:t}$ and $\mathbf{A}_{j(t+1)}$.

Generative Process. Inspired by CDL [33] and ZESRec [12], the generative process in Fig. 3 is defined as follows:

For each time step $t \in [T]$:
(a) For each item $j \in [J]$:
- Draw a latent item offset vector $\boldsymbol{\epsilon}_{jt}^{v} \sim \mathcal{N}\left(\mathbf{0}, \lambda_v^{-1}\mathbf{I}_D\right)$.
- Adopt the latent item offset vector as the latent item embedding: $\mathbf{V}_{jt} = \boldsymbol{\epsilon}_{jt}^{v}$.
- Compute the acceleration for time step $t+1$: $\mathbf{A}_{j(t+1)} = f_{ts}(\mathbf{A}_{j,0:t}, \mathbf{V}_{jt})$.

(b) For each user $i \in [I]$:
- Draw a latent user offset vector $\boldsymbol{\epsilon}_{it}^{u} \sim \mathcal{N}\left(\mathbf{0}, \lambda_u^{-1}\mathbf{I}_D\right)$.
- Obtain the user embedding: $\mathbf{n}_{it} = f_{seq}(\mathbf{S}_{it})$.
- Compute the latent user embedding: $\mathbf{U}_{it} = \boldsymbol{\epsilon}_{it}^{u} + \mathbf{n}_{it}$.
- Compute the recommendation score $\mathbf{Y}_{ijt}$ for each user-item pair (i, j) as $\mathbf{Y}_{ijt} = f_{softmax}(\mathbf{U}_{it}^{\top}\mathbf{V}_{jt})$. For user i we compute the recommendation scores for all items: $\mathbf{R}_{i*t} \sim Cat([\mathbf{Y}_{ijt}]_{j=1}^{J})$, where '$*$' represents the collection of all elements in a specific dimension, and $Cat(\cdot)$ denotes a categorical distribution.

Here I, J, T denote number of users, number of items, and number of time steps, respectively. $f_{softmax}(\cdot)$ is the softmax function, and $f_{softmax}(\mathbf{U}_{it}^{\top}\mathbf{V}_{jt}) = \exp(\mathbf{U}_{it}^{\top}\mathbf{V}_{jt}) / \sum_{j=1}^{J} \exp(\mathbf{U}_{it}^{\top}\mathbf{V}_{jt})$. $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \lambda^{-1}\mathbf{I}_D)$ denotes the probability density function (PDF) of a Gaussian distribution with mean $\boldsymbol{\mu}$ and diagonal covariance $\lambda^{-1}\mathbf{I}_D$ for the variable $\mathbf{x}$. $f_{ts}(\cdot)$ represents any type of probabilistic time series forecasting model which consumes both item historical accelerations and latent item embeddings to predict the probabilistic distribution of acceleration at the next time step $t+1$. $f_{seq}(\cdot)$ denotes any type of sequential recommender system which generates the user embedding based on user's historical interactions, and in this work we adopt the classic recommender system, GRU4Rec [15].

3.2 Training

Maximum a Posteriori (MAP) Estimation. Given the graphical model in Fig. 3, we hope to estimate the latent variables $\mathbf{U}_{it}$ and $\mathbf{V}_{jt}$ as well as the learnable parameters in functions $f_{seq}(\cdot)$ and $f_{ts}(\cdot)$. Note that for simplicity, we exclude the learnable parameters of $f_{seq}(\cdot)$ and $f_{ts}(\cdot)$ in the PGM. The maximum a posteriori (MAP) estimation can be decomposed as follows:

$$\mathbf{P}(\mathbf{U}_{it}, \mathbf{V}_{jt}|\mathbf{R}_{ijt}, \mathbf{A}_{j(t+1)}, \mathbf{S}_{it}, \mathbf{A}_{j,0:t}, \lambda_u, \lambda_v,) \propto \mathbf{P}(\mathbf{R}_{ijt}|\mathbf{U}_{it}, \mathbf{V}_{jt}) \cdot \mathbf{P}(\mathbf{A}_{j(t+1)}|\mathbf{A}_{j,0:t}, \mathbf{V}_{jt}) \cdot \mathbf{P}(\mathbf{U}_{it}|\mathbf{S}_{it}, \lambda_u) \cdot \mathbf{P}(\mathbf{V}_{jt}|\lambda_v). \tag{5}$$

Below we explain each conditional probability from Eqn. 5 in detail.

We assume Gaussian distributions on latent variables $\mathbf{U}_{it}$ and $\mathbf{V}_{jt}$ in Fig. 3 as follows:

$$\mathbf{P}(\mathbf{U}_{it}|\mathbf{S}_{it}, \lambda_u) = \mathcal{N}(\mathbf{U}_{it}; f_{seq}(\mathbf{S}_{it}), \lambda_u^{-1}\mathbf{I}_D), \tag{6}$$

$$\mathbf{P}(\mathbf{V}_{jt}|\lambda_v) = \mathcal{N}(\mathbf{V}_{jt}; \mathbf{0}, \lambda_v^{-1}\mathbf{I}_D). \tag{7}$$

For the next item recommendation, we define the conditional probability over the observed interactions as:

$$\mathbf{P}(\mathbf{R}_{ijt}|\mathbf{U}_{it}, \mathbf{V}_{jt}) = f_{softmax}(\mathbf{U}_{it}^{T}\mathbf{V}_{jt}). \tag{8}$$

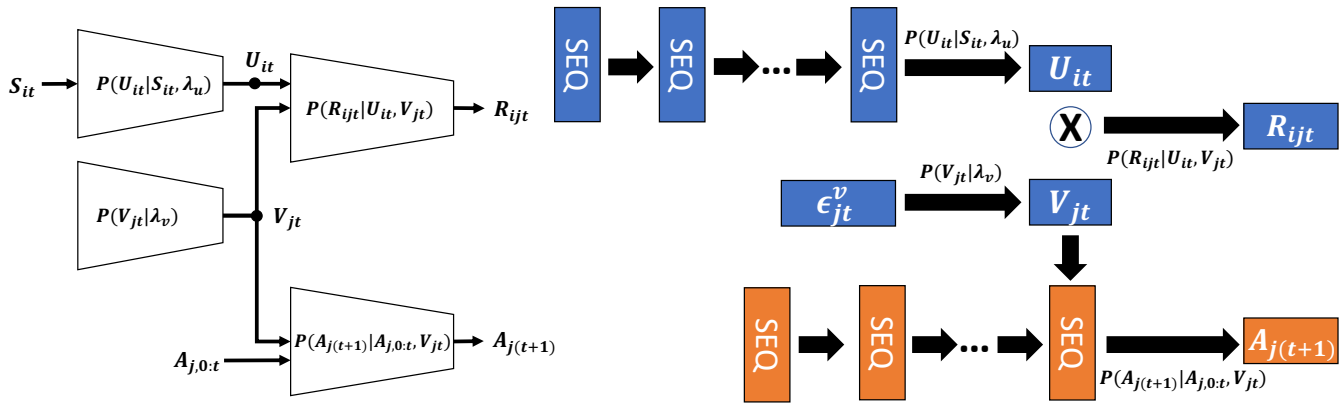


Figure 4: Model architecture for TrendRec from a neural network perspective. The left side figure shows the network structure, and the right side figure illustrates the implementation details of each network component. In the right side figure, blue and orange imply learnable components trained on the next item recommendation objective and learnable components trained on the time series forecasting objective, respectively. Each SEQ block represents a specific time step of a sequential model. Note that both figures are instantiated on a specific user-item pair (user i and item j). Overall, TrendRec encompasses two major components: (1) a sequential recommender, and (2) a collaborative time series forecasting model, and they are connected through the learnable latent item embeddings.

For the time series forecasting, in line with Eqn. 4, we define the conditional probability over the observed item accelerations as:

$$\mathbf{P}(\mathbf{A}_{j(t+1)}|\mathbf{A}_{j,0:t}, \mathbf{V}_{jt}) = f_{ts}(\mathbf{A}_{j,0:t}, \mathbf{V}_{jt}). \tag{9}$$

Negative Log Likelihood (NLL). Maximizing the posterior probability is equivalent to minimizing the negative log likelihood (NLL) of $\mathbf{P}(\mathbf{U}_{it}, \mathbf{V}_{jt}|\mathbf{R}_{ijt}, \mathbf{A}_{j(t+1)}, \mathbf{S}_{it}, \mathbf{A}_{j,0:t}, \lambda_u, \lambda_v,)$. We substitute each terms in Eqn. 5 with our definitions in Eqn. 6, Eqn. 7, Eqn. 8, and Eqn. 9, and compute the NLL as:

$$\mathcal{L} = -\sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J} \mathbf{R}_{ijt} \log(f_{softmax}(\mathbf{U}_{it}^T \mathbf{V}_{jt})) \tag{10}$$

$$-\sum_{t=1}^{T}\sum_{i=1}^{I} \log(\ell(\mathbf{A}_{j(t+1)}|f_{ts}(\mathbf{A}_{j,0:t}, \mathbf{V}_{jt})) \tag{11}$$

$$+\frac{\lambda_v}{2}\sum_{j=1}^{J}||\mathbf{V}_{jt}||^2 + \frac{\lambda_u}{2}\sum_{i=1}^{I}||\mathbf{U}_{it} - f_{seq}(\mathbf{S}_{it})||^2 \tag{12}$$

where $f_{seq}(\mathbf{S}_{it}) = \mathbf{n}_{it}$, and $\ell(\cdot)$ represents a likelihood function adopted by the probabilistic time series forecasting model. Note that we train hidden variables $\mathbf{U_{it}}$ and $\mathbf{V_{jt}}$ as well as the function $f_{seq}(\cdot)$ and $f_{ts}(\cdot)$ based on the NLL training objective above. Below we describe the intuition of each term in $\mathcal{L}$.

(1) **Next Item Recommendation Loss (Eqn. 10).** Minimizing this term improves the next item recommendation performance in the training set.
(2) **Time Series Forecasting Loss (Eqn. 11).** Minimizing this term improves acceleration forecasting in the training set.
(3) **Regularizing Latent Item Embedding $\mathbf{V}_{jt}$ and Latent User Embedding $\mathbf{U}_{it}$ (Eqn. 12).** The first term regularizes the latent item embedding $\mathbf{V}_{jt}$ to be close to the zero-mean Gaussian prior. The second term regularizes the latent user embedding $\mathbf{U}_{it}$ to be close to the computed $f_{seq}(\mathbf{S}_{it})$ as user history implies user interests, while providing flexibility for $\mathbf{U}_{it}$ to deviate from $f_{seq}(\mathbf{S}_{it})$ as users with the same history may still have non-overlapping interests.

3.3 Inference

During inference, we only focus on the time series forecasting task and predict the acceleration as below:

$$\mathbf{P}(\mathbf{A}_{j(t+1)}|\mathbf{A}_{j,0:t}, \mathbf{V}^*_{jt}) = f^*_{ts}(\mathbf{A}_{j,0:t}, \mathbf{V}^*_{jt}), \tag{13}$$

where with a bit overload on $*$, we use $\mathbf{V}^*_{jt}$ to denote the posterior of item j's latent item embedding, and $f^*_{ts}(\cdot)$ to denote the trained sequential time series forecasting model.

3.4 Model Architecture

Fig. 4 shows the model architecture from the neural network point of view. The left side figure demonstrates the overview network structure, while the right side figure visualizes the full details of the TrendRec implementation. In general, the model contains two principal components: (1) a sequential recommender, and (2) a collaborative time series forecasting model. These two components are joined through the learnable latent item embeddings. Below we elaborate on the process in terms of two training objectives: the next item recommendation objective and the time series forecasting objective.

Next Item Recommendation Objective. For the next item recommendation objective, TrendRec generates the latent user embedding from a sequential recommender, and adopts latent item offset vector as the latent item embedding. The recommendation score is calculated based on the inner product between the latent user embedding and the latent item embedding, which is subsequently normalized by the softmax function $f_{softmax}(\cdot)$.

Time Series Forecasting Objective. For the time series forecasting objective, for each item, TrendRec retrieves the latent item

embedding pre-trained on the next item recommendation objective and utilizes it in conjunction with the item historical accelerations to predict the acceleration at the next time step.

In the experiment, for simplicity, we remove the regularization term on the latent user embedding, and decouple the training process by initially training on the next item recommendation (NIR) objective (Eqn. 14), followed by training on the time series forecasting (TSF) objective (Eqn. 15) which is our primary target:

$$\mathcal{L}_{NIR} = -\sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}\mathbf{R}_{ijt}\log(f_{softmax}(\mathbf{U}_{it}^{T}\mathbf{V}_{jt})) + \frac{\lambda_v}{2}\sum_{j=1}^{J}||\mathbf{V}_{jt}||^2, \tag{14}$$

$$\mathcal{L}_{TSF} = -\sum_{t=1}^{T}\sum_{i=1}^{I}\log(\ell(\mathbf{A}_{j(t+1)}|f_{ts}(\mathbf{A}_{j,0:t}, \mathbf{V}_{jt}^{*})), \tag{15}$$

where $\mathbf{V}_{jt}^{*}$ denotes the posterior of item j's latent item embedding after training on the NIR objective.

3.5 Practical Challenges and Solutions

We describe the unique practical impediments on adopting the time series forecasting models in the trend recommendation scenario, and propose various mechanisms to alleviate them.

Skewed Data. A real-world recommendation dataset usually includes hundreds of thousands of items. Due to the well known 'heavy-tail' problem in the recommendation regime, the majority of items are considered as cold items owing to low number of interactions (e.g., less than five interactions). As a result, the time series associated with these items exhibit nearly flat curves, hovering around zero. The time series forecasting models can be biased towards predicting zeros if a substantial proportion of such time series exist in the training data.

Short Active Period. The time series of warm items may still suffer from short active periods due to reasons below:

- *Varying active periods across items*: From data analysis, we found it is pervasive that the item time series contain leading zeros (because the item is not yet released) and trailing zeros (because the item is no longer available). This is primarily caused by the mismatch between the active periods of different items. For example, item A might be introduced to the market just as item B becomes unavailable.
- *Short lifecycle*: Items in certain domains have short lifecycles by their nature. For example, in the news domain, the lifecycle of an item normally spans less than one day as most people only pay attention to today's news.

Solutions. The aforementioned issues essentially pertain to data sparsity, and a principled solution is to filter out noise in the training data to avoid the garbage-in-garbage-out conundrum. In practice, we inspect the acceleration curve for each item and select top η% of items that display the most dynamic changes in acceleration, where η is a tunable hyperparameter. The dynamic nature of acceleration for an item j is measured with the metric $\mathcal{D}_j = \sum_t^T |\mathbf{A}_{jt}|$. We further adopt training masks on the time series of selected items; this enforces the model to focus on the active period and updates gradients based on the non-zero values. Additionally, we implement a range of mechanisms aimed at improving training stability and inference latency, please refer to Appendix A for more details.

Table 1: Datasets Statistics.

Dataset	Domain	#Users	#Items	#Interactions
TaoBao Cat 1	Retail	392K	27K	4.18M
TaoBao Cat 2	Retail	352K	35K	2.96M
TaoBao Cat 3	Retail	343K	31K	2.90M
Netflix	Media	478K	12K	89.33M
MIND	News	749K	19K	3.96M

4 EXPERIMENTS

We evaluate the heuristic models, time series forecasting models, and the proposed TrendRec on three public datasets from a variety of domains including retail, media, and news, with the major goals of addressing the following questions:

- **Q1** Does the proposed hypothesis on correlations between the task feasibility and the time step length in Sec. 2.2 (see Fig. 2) hold for all datasets? How do we select the time step length for each dataset?
- **Q2** Does TrendRec outperform all the baseline models including the heuristic models and the vanilla deep learning based time series forecasting model?

4.1 Datasets

In the experiment, we consider three public datasets: (1) TaoBao User Behavior dataset[1] [44] in Retail domain, (2) Netflix dataset [6] in Media domain, and (3) MIND dataset[2] [36] in News domain.

(1) **TaoBao User Behavior** [44] is a public dataset containing user behavior data from Taobao, one of the largest e-commerce platforms in the world. It contains around 1M users, 4M items, and covers 9 days of interaction data. Due to the large item catalogue, in the experiment, we split the data by item category, and select the top-3 item categories based on number of interactions to construct three separate datasets. We call them **TaoBao Cat 1**, **TaoBao Cat 2**, and **TaoBao Cat 3**, respectively.
(2) **Netflix** [6] is a classical recommendation dataset including 100M user ratings on movies with 17K items and 480K users. We consider the densest time period from 2003 January to 2005 December for our experiment.
(3) **MIND** [36] is a large-scale news recommendation dataset collected from the user interaction logs of Microsoft News. It includes 1-week user-item interactions and 4-week impression logs on 1M users and 160K news articles. In the experiment, we only utilize the 1-week interactions as the impression logs do not include timestamp information.

The statistics of the datasets after pre-processing are shown in Table 1. We adopted a rigorous experimental setup to ensure there is no temporal leakage between the next item recommendation objective and the time series forecasting objective. Specifically, we (1) temporally split the data such that all training interactions must happen before all testing interactions and (2) use the exact same training data for both objectives during the training phase.

4.2 Evaluated Methods

We consider the following methods during evaluation:

[1] https://tianchi.aliyun.com/dataset/649
[2] https://msnews.github.io/

- **Oracle**, which has access to the ground truth future acceleration at the next time step. Therefore, it always accurately predicts the acceleration and recommends the correct top-k trending items.
- **Random**, which recommends items by random selection from the whole item catalogue without replacement.
- **Markov**, which is a rule-based model to predict the acceleration at the next time step to be the same as the acceleration at the current time step. See Sec. 2.3.1 for more details.
- **Exponential Moving Average (EMA)**, which is a rule-based model to predict the acceleration at the next time step based on a weighted sum of accelerations at the latest m time steps (in the experiment we set $m = 8$). The weight decays exponentially with a factor of 0.75 with increasing number of time steps away from the current time step. See Sec. 2.3.2 for more details.
- **DeepAR** [31], which is one of the state-of-the-art time series forecasting models built on an auto-regressive recurrent network (RNN). See Sec. 2.3.3 for more details.
- **TrendRec**, which is our proposed two-phase model. It is instantiated on the DeepAR model for the time series forecasting task. The reason we choose DeepAR is twofold: (1) it is one of the state-of-the-art time series forecasting models and has been widely adopted, and (2) DeepAR is a likelihood method, and therefore it is more suitable for computing MAP. For the next item recommendation objective, we adopt GRU4Rec [15] to learn the latent item embeddings.

4.3 Evaluation Metrics

Instead of adopting evaluation metrics such as RMSE from the time series forecasting setting, we design evaluation metrics that are closely aligned with the goal of trending now recommendation carousel, which is to promote trending items at the next time step. Different from typical time series forecasting settings, for trend recommendation we are only interested in correctly recommending the *top-k* trending items instead of accurately predicting the acceleration for each item. Therefore, we design the metrics with emphasis on measuring the model's ability of identifying the items with large acceleration at the next time step. Similar to the classical evaluation metrics, Recall and NDCG, which are widely adopted in the recommendation domain, we relate acceleration to relevance and propose two evaluation metrics: **Acceleration (Acc)** and **TNDCG**. *Assuming the current time step is $t-1$, and we hope to evaluate on the time step t*, we define the proposed metrics as follows.

4.3.1 Acceleration Metric. We first select the top-k items based on the model's predicted accelerations at the next time step t, and denote them as $\mathcal{P}$. Then we map the selected k items to their corresponding ground truth accelerations at the next time step t as $\{\mathbf{A}_{jt}\}_{j\in\mathcal{P}}$. Since in our definition, acceleration is a quantitative measurement of trendiness, we compute the sum of next time step accelerations of the predicted items $\sum_{j\in\mathcal{P}}\mathbf{A}_{jt}$ and use it as the model's *trendiness score*. The higher the trendiness score, the better the model at predicting the trending items at the next time step.

We hope to scale the metric in the range of $[0, 1]$, and therefore apply the min-max normalization on top of the trendiness score. The upper bound of the trendiness score comes from the *Oracle* model: we denote the ground truth top-k items with largest accelerations at time step t as O, and compute the trendiness score as $T^k_{oracle,t} = \sum_{j\in O}\mathbf{A}_{jt}$. There is no lower bound for the trendiness score as the acceleration can be an arbitrary negative value. We artificially set the lower bound as the trendiness score of the *Random* model. This also helps prevent over-penalization on poor recommendations at a single time step, as the model is evaluated on multiple time steps (see Sec. 4.4 for more details). The trendiness score for the *Random* model is computed as the expectation over the acceleration summation of k items sampled from an uniform distribution $T^k_{random,t} = \frac{T_t}{|J|}\cdot k$, where $T_t = \sum_j^J \mathbf{A}_{jt}$. The final trendiness score for the model is $T^k_{model,t} = max(\sum_{j\in\mathcal{P}}\mathbf{A}_{jt}, T^k_{random,t})$. Finally, we combine different trendiness scores to compute the acceleration metric at time step t:

$$\text{Acc@}k_t = \frac{T^k_{model,t} - T^k_{random,t}}{T^k_{oracle,t} - T^k_{random,t}}, \tag{16}$$

where $Acc@k_t \in [0, 1]$.

4.3.2 TNDCG Metric. Our Trendiness-Normalized-DCG (TNDCG) metric further takes the item's rank position into consideration with a logarithmic reduction factor. We use r to index the rank position, $\mathbf{A}^{\mathcal{P}}_r$ and $\mathbf{A}^{O}_r$ to represent the acceleration of the item ranked at position r based on order from model prediction and ground truth, respectively. Then we compute $\text{TDCG}^k_{model,t} = \sum_{r=1}^k \frac{\mathbf{A}^{\mathcal{P}}_r}{log_2(r+1)}$ and $\text{TDCG}^k_{oracle,t} = \sum_{r=1}^k \frac{\mathbf{A}^{O}_r}{log_2(r+1)}$. The TDCG of *Random* model is calculated as $T^k_{random,t} = \sum_{r=1}^k \frac{T_t}{|J|}\cdot\frac{1}{log_2(r+1)}$. We use the min-max normalization to scale the final TNDCG metric at time step t to the range of $[0, 1]$

$$\text{TNDCG@}k_t = \frac{\text{TDCG}^k_{model,t} - \text{TDCG}^k_{random,t}}{\text{TDCG}^k_{oracle,t} - \text{TDCG}^k_{random,t}}, \tag{17}$$

and $\text{TNDCG@}k_t \in [0, 1]$.

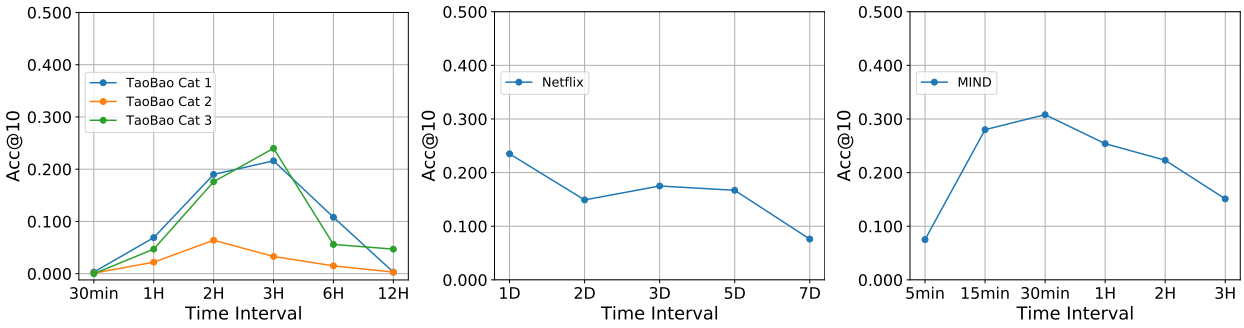
4.4 Evaluation Protocol

During evaluation, we split the training and test sets by timestamp, and leave the latest 20% time span for testing (e.g., eight-hour training window, two-hour testing window). The evaluation is conducted in a rolling fashion with a total of T steps. For instance, the test time span is two days while the time step length Δt is six hours; we therefore have $T = 8$ evaluation steps.

After evaluation, we will have T acceleration metrics for the evaluated model, *Random*, and *Oracle*, respectively. We aggregated them through a weighted sum

$$\text{Acc@}k = \frac{\sum_t^T \text{Acc@}k_t\cdot(T^k_{oracle,t} - T^k_{random,t})}{\sum_t^T (T^k_{oracle,t} - T^k_{random,t})} = \frac{\sum_t^T (T^k_{model,t} - T^k_{random,t})}{\sum_t^T (T^k_{oracle,t} - T^k_{random,t})}, \tag{18}$$

where $\text{Acc@}k \in [0, 1]$. The weight for each time step is $T^k_{oracle,t} - T^k_{random,t}$, and $\sum_t^T (T^k_{oracle,t} - T^k_{random,t})$ is the partition function for normalization. The intuition behind our weight design is that a time step is more important if the performance gap between *Oracle* and *Random* is large at that time step, as there is a large room for improvement.



(a) TaoBao (b) Netflix (c) MIND

Figure 5: Validating our hypothesis on the correlation between the task feasibility and the time interval (i.e., time step length Δt). Datasets from left to right are TaoBao (all three datasets), Netflix, and MIND. Note that the timestamp granularity of Netflix is one day; hence we are only able to show the right-hand side of the full curve, where model has collected sufficient data but is suffering from the temporal drift. Experimental results verified our proposed hypothesis in Sec. 2.2 (see Fig. 2).

Similarly, we calculate the aggregated TNDCG metric as

$$\begin{aligned}\text{TNDCG@}k &= \frac{\sum_t^T \text{TNDCG@}k_t \cdot (\text{TDCG}^k_{oracle,t} - \text{TDCG}^k_{random,t})}{\sum_t^T (\text{TDCG}^k_{oracle,t} - \text{TDCG}^k_{random,t})} \\ &= \frac{\sum_t^T (\text{TDCG}^k_{model,t} - \text{TDCG}^k_{random,t})}{\sum_t^T (\text{TDCG}^k_{oracle,t} - \text{TDCG}^k_{random,t})}, \end{aligned} \tag{19}$$

where $\text{TNDCG@}k \in [0, 1]$. The weight for each time step equals to $\text{TDCG}^k_{oracle,t} - \text{TDCG}^k_{random,t}$, and $\sum_t^T (\text{TDCG}^k_{oracle,t} - \text{TDCG}^k_{random,t})$ is the partition function for normalization.

4.5 Hypothesis Validation (Q1)

We conduct experiments to validate the hypothesis proposed in Fig. 2, which presumes a correlation curve between the task feasibility and the time step length Δt. Specifically, we evaluate the performance of the Markov heuristic model, which simply projects $\mathbf{A}_{j(t+1)} = \mathbf{A}_{jt}$, on a range of time intervals based on the acceleration metric and plot the curve. The Markov heuristic model is a rudimentary model grounded on a simple but generic assumption, and therefore its performance should reflect the task feasibility.

The results are shown in Fig. 5. On TaoBao and MIND datasets, the curves exhibit a clear trajectory that is consistent with our conjecture, where the acceleration metric first improves with increasingly longer time interval due to alleviation on the data sparsity, and then decreases due to temporal drift. On the other hand, the curve on the Netflix dataset keeps decreasing. This is because the timestamp granularity of the Netflix dataset is one day, which is long enough to collect sufficient data but starts to suffer from the temporal drift.

Overall, the results verify our hypothesis. In the following experiments, we select the time interval Δt for each dataset based on the peak of respective curves: three-hour time interval for all three TaoBao datasets for consistency, one-day time interval for Netflix, and thirty-minute time interval for MIND.

4.6 Experimental Results (Q2)

We evaluate our proposed TrendRec against various baselines on datasets from three domains: (1) TaoBao from the Retail domain, (2) Netflix from the Media domain, and (3) MIND from the News domain. The results are summarized in Table 2, from which we can easily conclude that TrendRec demonstrates superior performance. We have the following observations:

- **TrendRec outperforms all baseline models.** Our TrendRec consistently shows better performance compared to the baseline models across all the datasets. Since TrendRec's time series forecasting model is instantiated with DeepAR, the performance gain over DeepAR demonstrates the effectiveness of leveraging pre-trained latent item embeddings to boost the time series forecasting performance.
- **Performance improvement from TrendRec is associated with the quality of latent item embeddings.** While TrendRec demonstrates the best performance on all cases, we observe that the improvement over DeepAR is most significant on TaoBao, while it is marginal on Netflix and MIND. We present TrendRec's results on the auxiliary objective of next item recommendation in Table 3, where TrendRec shows significantly better results on TaoBao compared to Netflix and MIND. Based on this observation, we argue that the performance gain on the time series forecasting objective, which comes from leveraging pre-trained latent item embeddings, depends on the embedding quality. Notably, this quality is positively related to TrendRec's performance on the next item recommendation objective.
- **Deep learning based models significantly outperform heuristic models.** The deep learning based models (both DeepAR and TrendRec) outperform the heuristic models by a large margin, especially on TaoBao and Netflix datasets. The results emphasize the importance of adopting a learnable model for trend recommendation.
- **The EMA model is worse than the Markov model in most cases.** Interestingly, the EMA model underperforms the Markov model in most cases. This shows that the trend is changing dynamically and that the dependency relationship between the trend at the next time step and historical trends is more complicated than a simple weighted sum with recency bias.
- **Performance gain from deep learning based models is relatively small in the News domain.** The performance gap between the deep learning based models (both DeepAR and TrendRec) and the heuristic models is relatively marginal

Table 2: Trend recommendation results on three TaoBao datasets in the retail domain, the Netflix dataset in the media domain, and the MIND dataset in the news domain. Based on the results in Sec. 4.5, in experiments we set the time step length Δt to three hours, one day, and thirty minutes for TaoBao (all three datasets for consistency), Netflix, and MIND, respectively. The best results are shown in bold.

Method	TaoBao Cat 1		TaoBao Cat 2		TaoBao Cat 3		Netflix		MIND	
	Acc@10	TNDCG@10	Acc@10	TNDCG@10	Acc@10	TNDCG@10	Acc@10	TNDCG@10	Acc@10	TNDCG@10
Markov	0.209	0.212	0.057	0.028	0.180	0.155	0.326	0.340	0.319	0.286
EMA	0.239	0.230	0.034	0.024	0.111	0.085	0.163	0.204	0.225	0.193
DeepAR	0.710	0.725	0.471	0.417	0.631	0.598	0.531	0.505	0.365	0.348
TrendRec	**0.758**	**0.769**	**0.540**	**0.484**	**0.651**	**0.616**	**0.534**	**0.513**	**0.383**	**0.374**

Table 3: Next item recommendation results on all datasets. All experiments are conducted with GRU4Rec [15].

Method	TaoBao Cat1	TaoBao Cat2	TaoBao Cat3	Netflix	MIND
Recall@20	0.223	0.230	0.251	0.072	0.024
NDCG@20	0.114	0.116	0.131	0.027	0.009

in the News domain compared to Retail and Media domains. Analyzing item time series from the News domain reveals that items typically reach their maximum acceleration in a short period of time once they are released, which is followed by a sudden drop. This is primarily due to the time-sensitive nature of news, and poses a great challenge for deep learning based models. As we only focus on top items with large accelerations at each time step, the major goal of deep learning based models is to learn how to accurately forecast the items' acceleration when they are trendy given limited context (historical trends).

5 RELATED WORK

Trend Modeling. There are limited previous works have been done on the topic of trend modeling in the recommendation context, while the existing works [3, 23, 25] typically refer to recent popularity as trend, which is *velocity* in our term. We argue that promoting recent popular items in the 'trending now' recommendation carousel will not only amplify the 'rich-get-richer' problem, but also will lead to redundant recommendations compared with other carousels such as 'recent popular'. Moreover, none of these works focus on recommending trending items. For example, [23] leverages the trend information to probe users' potential interests, which have not been shown in their historical interactions, in order to improve next item recommendation for the users. Another line of research [2, 26, 27, 38, 42] studies trends in another scenario: social media (e.g., Twitter), and adopts a distinct problem setting from ours. For instance, [26] tries to detect bursty keywords from recent tweets, and then identify trending topics based on them. Overall, the previous works in trend modeling are either focusing on a different task or set in a distinct scenario. Therefore their proposed methods are not adaptable to our use case of trend recommendation.

Time Series Forecasting. Time series forecasting goes back to several decades in statistical approach including ARIMA [8], ETS [16]. However, neural networks showed little success in the forecasting literature until recently. With the explosive sample of time series data available and advances in neural architectures, deep learning for forecasting has become increasingly popular, which includes the architectures of RNN [30, 31, 35], CNN [20, 35], Transformer [20, 22, 39]. Most of these methods support probabilistic forecasting in either distribution-assumed manner as a likelihood approach [20, 29, 31] or distribution-free manner with quantile regression [18, 22, 28, 35]. Multivariate (or local) modelings like DeepVAR [30] are often preferred due to its capacity to encode the correlation between time series or items, but global modeling like DeepAR [31] or MQ-R(C)NN [35] are still widely used by leveraging data samples efficiently. See [17] for more detailed review on modern deep forecasting.

Adopting Time Series Models in Recommendation. Various prior research [5, 11, 14, 21, 45] has explored adopting time series models to model the temporal information in the recommendation scenario, with the primary goal of improving next item recommendation performance. STEN [21] employs temporal point processes to model the impact of friends' behaviors on users' dynamic interests. LSHP [11] considers Hawkes Process to model the long-term and short-term dependency between users' online interactive behaviors. Different from the aforementioned works, our main focus lies in accurately predicting trending items in the near future. Additionally, we design an auxiliary training objective to distill item correlations from user-item interactions, thereby boosting our forecasting performance.

6 CONCLUSION

In this work, we study an under-explored topic in the recommendation regime: the trend recommender. Since limited previous works have been done on this topic, we start by formally defining the notion of 'trend' within the recommendation context. We then recognize a bias-variance tradeoff hindering the model's ability to identify the trend in a timely and stable manner. This tradeoff comprises variance resulting from data sparsity, and bias emerging from temporal drift. Based on this observation, we formulate the trend recommendation task as a one-step time series forecasting problem. In terms of methodology, we develop a principled two-phase model, dubbed TrendRec, which harnesses the user-item interactive signals to uncover the underlying correlations across items and ingests such knowledge to facilitate the trend forecasting. We further establish corresponding evaluation protocols for trend recommendation. Extensive experiments on datasets from various domains including retail, media, and news demonstrate the effectiveness of our proposed TrendRec.

A SOLUTIONS TO PRACTICAL CHALLENGES

To address the challenges identified in Sec. 3.5, i.e., skewed data and short active period, along with additional empirical problems, we design the following mechanisms in featurization, training, and inference:

A.1 Featurization

- **Dynamic Filter**: To alleviate the problem of skewed data, we examine acceleration patterns of each item and select top η% of items that exhibit the most significant changes in acceleration. Here η is a tunable hyperparameter. We quantify the dynamics of acceleration for an item j based on the metric $\mathcal{D}_j = \sum_t^T |\mathbf{A}_{jt}|$. Note that this mechanism is closely aligned with the objective of trend recommendation, which is to correctly predict top-k trending items instead of accurately forecasting the accelerations of all items at the next time step.

A.2 Training

- **Training Masks**: Due to the problem of short active period, the time series of selected items (i.e., items chosen by the dynamic filter) can remain sparse. To prevent the model from overfitting zeros, we employ training masks which instruct the model to disregard zeros during training and only update gradients based on non-zero values.
- **Different Context Windows for Different Domains**: The context window is a hyperparameter of time series forecasting models which specifies the number of past time steps the model should take into account. Depending on the domain, such as media and retail with longer active periods for items or news with shorter item lifecycles, the context window size is adjusted. For example, a larger context window of 30 time steps may be used for the media and retail domains, while a smaller context window of 2 time steps may be used for the news domain.
- **Training on Velocity**: To ensure training stability, we choose to train the time series forecasting model using velocity time series instead of more dynamic acceleration time series. Empirically, we discovered that the time series forecasting model trained on velocity time series achieved better performance.

A.3 Inference

- **Candidate Filter**: To guarantee the quality of recommendations, we apply a velocity filter to select the top γ% of items as candidates for the acceleration prediction of the time series forecasting model. This filter is based on the velocity of the item at the current time step, and the value of γ is a hyperparameter that depends on the dataset.
- **Active Filter**: We exclude the items with zero velocity at the current time step from the candidate list. This is mostly useful for items in the news domain due to short active period.

Both filtering mechanisms reduce the number of candidates prior to the time series forecasting model's inference, with the goal of improving forecasting accuracy and reducing inference latency.

REFERENCES

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. 42–46.

[2] Luca Maria Aiello, Georgios Petkos, Carlos Martin, David Corney, Symeon Papadopoulos, Ryan Skraba, Ayse Göker, Ioannis Kompatsiaris, and Alejandro Jaimes. 2013. Sensing trending topics in Twitter. *IEEE Transactions on multimedia* 15, 6 (2013), 1268–1282.

[3] Ziad Al-Halah, Rainer Stiefelhagen, and Kristen Grauman. 2017. Fashion forward: Forecasting visual style in fashion. In *Proceedings of the IEEE international conference on computer vision*. 388–397.

[4] Sarah A Alkhodair, Steven HH Ding, Benjamin CM Fung, and Junqiang Liu. 2020. Detecting breaking news rumors of emerging topics in social media. *Information Processing & Management* 57, 2 (2020), 102018.

[5] Ting Bai, Lixin Zou, Wayne Xin Zhao, Pan Du, Weidong Liu, Jian-Yun Nie, and Ji-Rong Wen. 2019. CTrec: A long-short demands evolution model for continuous-time recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 675–684.

[6] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. New York, 35.

[7] Rahul Bhagat, Srevatsan Muralidharan, Alex Lobzhanidze, and Shankar Vishwanath. 2018. Buy it again: Modeling repeat purchase recommendations. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 62–70.

[8] George EP Box and Gwilym M Jenkins. 1968. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 17, 2 (1968), 91–109.

[9] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

[10] Yuri M Brovman. 2019. Complementary Item Recommendations at eBay Scale.

[11] Renqin Cai, Xueying Bai, Zhenrui Wang, Yuling Shi, Parikshit Sondhi, and Hongning Wang. 2018. Modeling sequential online interactive behaviors with temporal point process. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 873–882.

[12] Hao Ding, Yifei Ma, Anoop Deoras, Yuyang Wang, and Hao Wang. 2021. Zero-shot recommender systems. *arXiv preprint arXiv:2105.08318* (2021).

[13] Ziwei Fan, Hao Ding, Anoop Deoras, and Trong Nghia Hoang. 2023. Personalized federated domain adaptation for item-to-item recommendation. In *Proceedings of the UAI 2023*. https://www.amazon.science/publications/personalized-federated-domain-adaptation-for-item-to-item-recommendation

[14] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S Yu. 2021. Continuous-time sequential recommendation with temporal graph collaborative transformer. In *Proceedings of the 30th ACM international conference on information & knowledge management*. 433–442.

[15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[16] Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. 2008. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media.

[17] Tim Januschowski, Jan Gasthaus, Yuyang Wang, Syama Sundar Rangapuram, and Laurent Callot. 2018. Deep Learning for Forecasting: Current Trends and Challenges. *Foresight: The International Journal of Applied Forecasting* 51 (2018).

[18] Kelvin Kan, Francois-Xavier Aubet, Tim Januschowski, Youngsuk Park, Konstantinos Benidis, Lars Ruthotto, and Jan Gasthaus. 2022. Multivariate quantile function forecaster. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 10603–10621.

[19] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[20] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems* 32 (2019).

[21] Yunzhe Li, Yue Ding, Bo Chen, Xin Xin, Yule Wang, Yuxiang Shi, Ruiming Tang, and Dong Wang. 2021. Extracting Attentive Social Temporal Excitation for Sequential Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 998–1007.

[22] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* 37, 4 (2021), 1748–1764.

[23] 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*. 2992–3001.

[24] Yifei Ma, Balakrishnan Narayanaswamy, Haibin Lin, and Hao Ding. 2020. Temporal-contextual recommendation in real-time. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2291–2299.

[25] Utkarsh Mall, Kevin Matzen, Bharath Hariharan, Noah Snavely, and Kavita Bala. 2019. Geostyle: Discovering fashion trends and events. In *Proceedings of the IEEE/CVF international conference on computer vision*. 411–420.

[26] Michael Mathioudakis and Nick Koudas. 2010. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 1155–1158.

[27] Mor Naaman, Hila Becker, and Luis Gravano. 2011. Hip and trendy: Characterizing emerging trends on Twitter. *Journal of the American Society for Information Science and Technology* 62, 5 (2011), 902–918.

[28] Youngsuk Park, Danielle Maddix, Fran**c**cois-Xavier Aubet, Kelvin Kan, Jan Gasthaus, and Yuyang Wang. 2022. Learning quantile functions without quantile crossing for distribution-free time series forecasting. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 8127–8150.

[29] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. 2018. Deep state space models for time series forecasting. *Advances in neural information processing systems* 31 (2018).

[30] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. 2019. High-dimensional multivariate forecasting with low-rank gaussian copula processes. *Advances in neural information processing systems* 32 (2019).

[31] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.

[32] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[33] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1235–1244.

[34] Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. 2021. Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1791–1800.

[35] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053* (2017).

[36] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3597–3606.

[37] An Yan, Chaosheng Dong, Yan Gao, Jinmiao Fu, Tong Zhao, Yi Sun, and Julian McAuley. 2022. Personalized complementary product recommendation. In *Companion Proceedings of the Web Conference 2022*. 146–151.

[38] Louis Yu, Sitaram Asur, and Bernardo A Huberman. 2011. What trends in Chinese social media. *arXiv preprint arXiv:1107.3522* (2011).

[39] Xiyuan Zhang, Xiaoyong Jin, Karthick Gopalswamy, Gaurav Gupta, Youngsuk Park, Xingjian Shi, Hao Wang, Danielle C Maddix, and Yuyang Wang. 2022. First De-Trend then Attend: Rethinking Attention for Time-Series Forecasting. *arXiv preprint arXiv:2212.08151* (2022).

[40] Yuhui Zhang, Hao Ding, Zeren Shui, Yifei Ma, James Zou, Anoop Deoras, and Hao Wang. 2021. Language models as recommender systems: Evaluations and limitations. (2021).

[41] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal intervention for leveraging popularity bias in recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 11–20.

[42] Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th international conference on world wide web*. 1395–1405.

[43] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. 2021. Disentangling user interest and conformity for recommendation with causal embedding. In *Proceedings of the Web Conference 2021*. 2980–2991.

[44] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1079–1088.

[45] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to Do Next: Modeling User Behaviors by Time-LSTM.. In *IJCAI*, Vol. 17. 3602–3608.