Dual Sequential Prediction Models Linking Sequential Recommendation and Information Dissemination

Qitian Wu[1], Yirui Gao[1], Xiaofeng Gao[1*], Paul Weng[2], Guihai Chen[1]
[1]Shanghai Key Laboratory of Scalable Computing and Systems,
Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]UM-SJTU Joint Institute, Shanghai Jiao Tong University
echo740@sjtu.edu.cn,gaoyirui98@gmail.com,gao-xf@cs.sjtu.edu.cn,paul.weng@sjtu.edu.cn,gchen@cs.sjtu.edu.cn

ABSTRACT

Sequential recommendation and information dissemination are two traditional problems for sequential information retrieval. The common goal of the two problems is to predict future user-item interactions based on past observed interactions. The difference is that the former deals with users' histories of clicked items, while the latter focuses on items' histories of infected users. In this paper, we take a fresh view and propose dual sequential prediction models that unify these two thinking paradigms. One user-centered model takes a user's historical sequence of interactions as input, captures the user's dynamic states, and approximates the conditional probability of the next interaction for a given item based on the user's past clicking logs. By contrast, one item-centered model leverages an item's history, captures the item's dynamic states, and approximates the conditional probability of the next interaction for a given user based on the item's past infection records. To take advantage of the dual information, we design a new training mechanism which lets the two models play a game with each other and use the predicted score from the opponent to design a feedback signal to guide the training. We show that the dual models can better distinguish false negative samples and true negative samples compared with single sequential recommendation or information dissemination models. Experiments on four real-world datasets demonstrate the superiority of proposed model over some strong baselines as well as the effectiveness of dual training mechanism between two models.

KEYWORDS

Sequential Recommendation, Information Dissemination, Sequential Prediction Model, Semi-Supervised Learning

*Xiaofeng Gao is the corresponding author. This work was supported by the National Key R&D Program of China [2018YFB1004703]; the National Natural Science Foundation of China [61872238, 61672353]; the Shanghai Science and Technology Fund [17510740200]; the Huawei Innovation Research Program [HO2018085286]; and the State Key Laboratory of Air Traffic Management System and Technology [SKLATM20180X]; the Tencent Social Ads Rhino-Bird Focused Research Program.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330959

ACM Reference Format:
Qitian Wu, Yirui Gao, Xiaofeng Gao, Paul Weng, Guihai Chen. 2019. Dual Sequential Prediction Models Linking Sequential Recommendation and Information Dissemination. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330959

1 INTRODUCTION

Sequential data that record the interactions between users and items are ubiquitous in many real-life scenarios. One well-known research problem in sequential information retrieval is to predict future interactions based on observed observations. In recommender systems, the platform may try to recommend items for each user according to the user's purchasing or browsing records. In social network services, some user-generated items (e.g., tweets, pictures or videos) may generate a huge number of shares or retweets, and one would like to predict who will be influenced next based on the item's history of infected users (borrowing a term from epidemiology given the analogy between information diffusion and viral contagion). These two situations respectively correspond to two classical sequential prediction problems: sequential recommendation [9, 16, 29, 40, 46] and information dissemination [28, 30, 34, 45]. The two problems share a dual structure, as depicted in Fig.1.

Prior studies take distinct views and develop different schools of thinking to tackle these two prediction tasks, respectively. On the one hand, for sequential recommendation (Fig. 1.a), previous works tend to model the conditional probability that a user u would click on (retweet or share depending on the domain) a next item i based on user u's history of clicked items $\mathcal{I}_u$, i.e., $P(\text{click} \mid (u, i), \mathcal{I}_u)$. For instance, they adopt Markov chain [9, 29], time-aware SVD [23], translation model [13, 14], recurrent model [16, 35, 44] as well as self-attention mechanism [20, 46] to approximate this conditional probability. On the other hand, for information dissemination (Fig. 1.b), prior researches generally consider the conditional probability that a next user u would be infected by an item i based on item i's history of infected users $\mathcal{U}_i$, i.e., $P(\text{click} \mid (u, i), \mathcal{U}_i)$. The approaches include epidemic model [30, 33], stochastic point process [2, 32, 45], regression model [25, 26] as well as recurrent neural network [3, 6]. While the two schools of thinking possess respective theoretical foundations and could capture the temporal dependencies in user's behaviors (resp. item's influences), they assume an item's attributes (resp. user's interests) to be static and thus ignore the temporal dependencies of the other side. In fact, on the one hand, for sequential recommendation, an item's attributes tend to vary dynamically when clicked by users from different communities, which may change an item's attractiveness to users as time

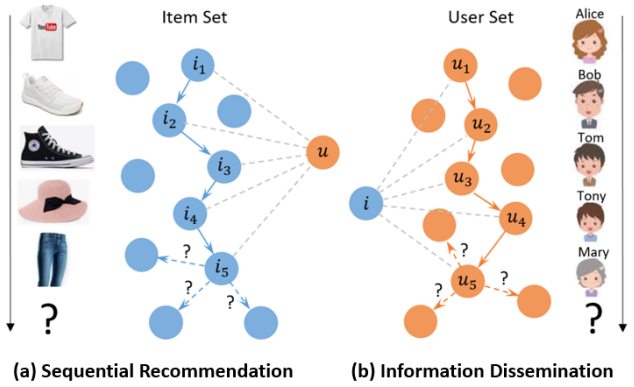

Figure 1: The dual structure between sequential recommendation (left) v.s. information dissemination problems (right). For sequential recommendation, with user u's history of clicked items $i_1 \sim i_5$, the goal is to predict the next item that she would possibly click on. For information dissemination, with item i's history of infected users $u_1 \sim u_5$, the goal is to predict the next user that would possibly be influenced by such item.

goes by. On the other hand, for information dissemination, a user's interests incline to change while exposed to different items, which enlightens us to consider distinct user preferences over time.

However, concurrently modeling the two-side temporal dependencies and targeting the distribution $P(\text{click} \mid (u, i), \mathcal{I}_u, \mathcal{U}_i)$ would pose a series of challenges. First, it introduces a higher computational cost, because: i) we need two sequential units to respectively capture the temporal dynamics in two history sequences, and ii) the elements in the two sequences may pairwise couple to form $O(n^2)$ hidden states. Second, the model would be more likely to get stuck in local optimum or gradient vanishing due to the complexity of the objective function. Third, since we only have observed positive samples, we need to sample enough negative examples from unobserved interactions for training. Such negative samples could contain a number of false negative ones, which would give wrong directions for training.

In this paper, we take a fresh view and propose DEEMS (for Dual Sequential Prediction Models), which takes advantage of the dual structure between sequential prediction and information dissemination problems to address the above-mentioned limitations and challenges in an elegant way. DEEMS consists of dual sequential prediction models, which can be specified by any time-dependent units such as RNN or self-attention modules. One model, which is *user-centered*, takes a user's history of clicked items as input and captures the hidden dynamic states in the user's interest shifts. The other model, which is *item-centered*, leverages an item's history of infected users as input to encode the dynamic patterns in the item's influence variation. The training for the two models is quite different from previous methods: we let them play a game during which each model alternately tries to match the prediction of the other model. More specifically, during a round where the user-centered model is trained, an observed interaction between a user u and an item i is sampled, for which the model predicts a score representing the probability that an interaction occurs based on the history of user u, i.e., $P(\text{click} \mid (u, i), \mathcal{I}_u)$. The same sample is fed to the item-centered model, which provides the score, $P(\text{click} \mid (u, i), \mathcal{U}_i)$. The latter predicted score is treated as 'label' and used to compute the discrepancy between the two scores (called *hedge loss*), which is exploited as a feedback to train the user-centered model. Then, in the next round, they interchange roles, and a similar process is repeated to train the item-centered model. With this scheme, the dual models compensate each other, and exchange the information to reach a consensus.

DEEMS concurrently capture features expressing temporal dynamics of both users' and item' states in an efficient way, and the hedge loss based training makes it possible to get out of their respective local optimums. More importantly, the feedback from the other model can help one model to distinguish false negative samples and true negative ones, which further improves prediction accuracy and model robustness. To verify the approach, we conduct extensive experiments over four real-world datasets from the Amazon product platform, and compare with eight strong baselines for both sequential recommendation and information dissemination. The results show that, DEEMS achieves significant improvements of precision and AUC compared with other existing methods. Also, a series of ablation studies further demonstrate the superiority of this dual mechanism as well as the training approach.

Our contributions can be summarized as follows:

- We propose to unify two schools of thinking for sequential information retrieval—sequential recommendation and information dissemination—to take advantage of the dual information to conduct prediction.
- We design a new training approach that lets two models play a game with each other and use the output of the opponent to define a hedge loss for training. Such mechanism could help distinguish the false negative samples.
- We conduct extensive experiments over four practical datasets and compare with eight strong competitors. The results show that dual models are superior than the single counterpart, and the proposed model achieves great improvement of prediction accuracy under different settings.

2 PRELIMINARY

User-item interaction is classically represented as a matrix $R = \{r_{ui}\}_{M \times N}$, where M and N denote the numbers of users and items, respectively. We focus on implicit feedback, which implies that matrix R consists of 0 and 1, i.e., $r_{ui} = 1$ means that user u has clicked item i, and $r_{ui} = 0$ otherwise. However, for sequential prediction problems, each interaction corresponds to a timestamp t that records when the interaction happened. Therefore, we adopt a triplet (u, i, t) to denote one user-item interaction, and $\mathcal{T} = \{(u, i, t)\}$ denotes all observed interactions. From $\mathcal{T}$ and for a given time t, we define two history sequences as follows.

Definition 2.1. (History of Item Sequence): For user u, we call the item sequence $\mathcal{I}_u^t = \{i \mid (u, i, t') \in \mathcal{T}, t' < t\} = \{i_1, i_2, \cdots\}$ as user u's history of item sequence or set of clicked items.

Definition 2.2. (History of User Sequence): For item i, we define the user sequence $\mathcal{U}_i^t = \{u \mid (u, i, t') \in \mathcal{T}, t' < t\} = \{u_1, u_2, \cdots\}$ as item i's history of user sequence or set of infected users.

We now proceed to formulate the general form of sequential prediction problem.

Definition 2.3. (Sequential Prediction Problem): Given observed user-item interactions in $\mathcal{T}$, the sequential prediction problem consists in predicting the unobserved potential interaction between a user u and an item i at some time t.

The above general problem could be reduced to two specific forms: sequential recommendation and information dissemination problems. For sequential recommendation, we rely on the history of item sequence $\mathcal{I}_u^t$ for each user, and aim at predicting the items that will be clicked. In this situation, the model requires to estimate the conditional probability $P\big(\text{click} \mid (u, i), \mathcal{I}_u^t\big)$. For information dissemination, we consider the history of user sequence $\mathcal{U}_i^t$ for each item, and attempt to predict the potential users that would interact with such item. Here, the model needs to estimate $P\big(\text{click} \mid (u, i), \mathcal{U}_i^t\big)$. Indeed, the two problems share the same goal, i.e., to predict future interactions between users and items, but adopt distinct aspects to tackle the prediction. In this paper, we propose to unify these two schools of thinking and leverage the dual information for prediction via a two-player game.

3 PROPOSED MODEL

In this section, we present our proposed model DEEMS (Dual Sequential Prediction Models), whose framework is shown in Fig. 2. We first introduce the general model that takes advantage of the dual structure between sequential recommendation and information dissemination models. Then we detail the specifications of the sequential prediction units in our model. Finally, we decompose the efficacy of our model to help to understand how DEEMS works.

3.1 Dual Sequential Prediction Models

From interaction triplet set $\mathcal{T}$, we can obtain the history of item sequence $\mathcal{I}_u^t$ and the history of user sequence $\mathcal{U}_i^t$ for each (u, i, t). As shown in Fig. 2, DEEMS contains two models: a user-centered model and an item-centered model. The user-centered model takes $\mathcal{I}_u^t$ as input to capture the dynamic pattern in the user's interest states, and estimates the conditional probability $P\big(\text{click} \mid (u, i), \mathcal{I}_u^t\big)$ that item i would be clicked by user u conditioned on her history of item sequence.

By contrast, the item-centered sequential model takes $\mathcal{U}_i^t$ as input to model the item's time-variant influence, and approximates the conditional probability $P\big(\text{click} \mid (u, i), \mathcal{U}_i^t\big)$ that user u would be infected by item i conditioned on its history of user sequence.

Embedding Layer. To better encode user's and item's features, we represent each user (resp. item) as one low-dimensional vector. We first encode user u as a one-hot column vector $\mathbf{t}_u \in \mathbb{R}^M$, where only the k-th value is 1 and the other values are zero if user u is the k-th user, for $k \in [M]$ (denoting $[M] = \{1, \ldots, M\}$). Similarly, item i is encoded as a one-hot column vector $\mathbf{e}_i \in \mathbb{R}^N$. For users, we define $\mathbf{P} \in \mathbb{R}^{d \times M}$, where d is the embedding dimension, and $\mathbf{p}_u = \mathbf{P}\mathbf{t}_u$ denotes the embedding vector for user u reflecting the user's static preferences and interests. Similarly, for items, we define matrix $\mathbf{Q} \in \mathbb{R}^{d \times N}$ and $\mathbf{q}_i = \mathbf{Q}\mathbf{e}_i$ denotes the embedding vector for item i, characterizing the item's static attribute and influence. In DEEMS, the user-centered model and the item-centered model are two self-contained models, so we adopt independent embedding parameters for them. Specifically, we use $\mathbf{P}^U$ and $\mathbf{Q}^U$ (resp. $\mathbf{P}^I$ and $\mathbf{Q}^I$) for the embedding parameters of the user-centered (resp. item-centered) model, and denote the corresponding embedding vectors as $\mathbf{p}_u^U$ and $\mathbf{q}_i^U$ (resp. $\mathbf{p}_u^I$ and $\mathbf{q}_i^I$). The embedding parameters will be learned together with the following sequential prediction layer in order to achieve end-to-end training.

Sequential Prediction Layer. Then we proceed to capture the temporal dependencies in user's states and item's states through sequential prediction layers in the two models respectively. The user-centered model takes the history of item sequence $\mathcal{I}_u^t$ as input and output a hidden vector $\mathbf{s}_u^t$ that reflects user u's dynamic interest. Concretely, we have

$$\mathbf{s}_u^t = SU_U(\mathcal{I}_u^t, \mathbf{p}_u^U, \mathbf{q}_i^U; \mathbf{w}_U), \tag{1}$$

where $SU_U(\cdot; \mathbf{w}_U)$ denotes a sequential unit (implemented by a Markov Chain, RNN or self-attention module, etc) parametrized with $\mathbf{w}_U$. Similarly, the item-centered model takes the history of user sequence $\mathcal{U}_i^t$ as input and compute $\mathbf{l}_i^t$ that characterizes item i's dynamic influence. Specifically, we have

$$\mathbf{l}_i^t = SU_I(\mathcal{U}_i^t, \mathbf{p}_u^I, \mathbf{q}_i^I; \mathbf{w}_I), \tag{2}$$

where $SU_I(\cdot; \mathbf{w}_I)$ denotes a sequential unit parametrized with $\mathbf{w}_I$.

Then we leverage user embedding $\mathbf{p}_u^U$ (resp. $\mathbf{p}_u^I$), item embedding $\mathbf{q}_i^U$ (resp. $\mathbf{q}_i^I$) and user dynamic interest $\mathbf{s}_u^t$ (resp. item dynamic influence $\mathbf{l}_i^t$) to predict the interaction probability. For the user-centered model, the predicted score is defined by:

$$\hat{r}_{ui}^U = PU_U(\mathbf{p}_u^U, \mathbf{q}_i^U, \mathbf{s}_u^t; \mathbf{y}_U), \tag{3}$$

where $PU_U(\cdot; \mathbf{y}_U)$ denotes a general prediction unit parametrized with $\mathbf{y}_U$. Also, for the item-centered model, the predicted score is computed by:

$$\hat{r}_{ui}^I = PU_I(\mathbf{p}_u^I, \mathbf{q}_i^I, \mathbf{l}_i^t; \mathbf{y}_I). \tag{4}$$

where $PU_I(\cdot; \mathbf{y}_I)$ is a general prediction unit parametrized with $\mathbf{y}_I$. We will further discuss the specifications for $SU_U(\cdot; \mathbf{w}_U)$, $SU_I(\cdot; \mathbf{w}_I)$, $PU_U(\cdot; \mathbf{y}_U)$, and $PU_I(\cdot; \mathbf{y}_I)$ in Section 3.2. To keep the notation clean, we use θ_U and θ_I to denote all parameters in user-centered and item-centered model, respectively, i.e., $\theta_U = [\mathbf{P}^U, \mathbf{Q}^U, w_U, y_U]$, and $\theta_I = [\mathbf{P}^U, \mathbf{Q}^I, w_I, y_I]$.

Output Layer. The final predicted probability that user u will interact with item i is the averaged scores, i.e., $\hat{r}_{ui} = \frac{\hat{r}_{ui}^U + \hat{r}_{ui}^I}{2}$.

Loss and Training. First, we hope that the output predicted scores $\hat{r}_{ui}^U = P_{\theta_U}\big(\text{click} \mid (u, i), \mathcal{I}_u^t\big)$ and $\hat{r}_{ui}^I = P_{\theta_I}\big(\text{click} \mid (u, i), \mathcal{U}_i^t\big)$ could approximate the true conditional probability $P\big(\text{click} \mid (u, i), \mathcal{I}_u^t\big)$ and $P\big(\text{click} \mid (u, i), \mathcal{U}_i^t\big)$ respectively. The cross-entropy loss can be used to measure the discrepancy between the predicted scores $\hat{r}_{ui}^U$ (or $\hat{r}_{ui}^I$) and the ground-truth labels r_{ui}.

$$\mathcal{L}_{sv}^U = -\sum_{(u,i,t)\in\mathcal{T}} r_{ui} \cdot \log \hat{r}_{ui}^U + (1 - r_{ui}) \cdot \log(1 - \hat{r}_{ui}^U). \tag{5}$$

$$\mathcal{L}_{sv}^I = -\sum_{(u,i,t)\in\mathcal{T}} r_{ui} \cdot \log \hat{r}_{ui}^I + (1 - r_{ui}) \cdot \log(1 - \hat{r}_{ui}^I). \tag{6}$$

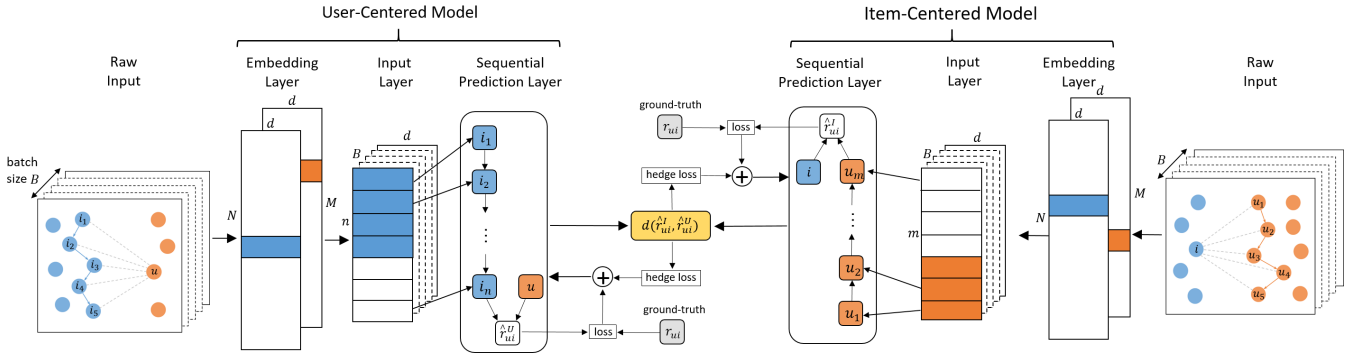

Figure 2: The framework of DEEMS that consists of two sub-models: user-centered (left) and item-centered (right) models. They play a game with each other during training. First, the user-centered model takes a user u's history of item sequence ($i_1 \sim i_5$) as input, and encodes each user and item into low-dimensional vectors through an embedding layer. Then the embedding representations of users and items are fed into the sequential prediction layer (implemented as Markov Chain, RNN or self-attention mechanism, etc.), which will outputs a conditional probability $\hat{r}^U_{ui} = P_{\theta_U}(\textbf{click} \mid (u,i), \mathcal{I}^t_u)$ that item i would be clicked by user u conditioned on her history of item sequence. On the other hand, the item-centered model leverages an item i's history of user sequence ($u_1 \sim u_5$) to estimate another conditional probability $\hat{r}^I_{ui} = P_{\theta_I}(\textbf{click} \mid (u,i), \mathcal{U}^t_i)$ that user u would be infected by item i conditioned on its history of user sequence. Then the dissimilarity between the two probabilities $d(\hat{r}^U_{ui}, \hat{r}^I_{ui})$ defines a hedge loss as feedback for the user-centered model, which will be updated via the combination of supervised signal and the hedge loss. Secondly, their roles interchange and the training process repeats until convergence.

Here, we call $\mathcal{L}^U_{sv}$ and $\mathcal{L}^I_{sv}$ supervised losses for the user-centered and item-centered models, respectively.

Second, in order to take advantage of the dual information and make the two sequential models compensate each other, we let them play a two-player game and treat the discrepancy from one's prediction to the prediction of the other model (denoted as $d(\hat{r}, \hat{r}')$ for two predictions $\hat{r}, \hat{r}'$), as a feedback. Since both predictions are probabilities, one natural candidate for the discrepancy measure is the cross-entropy. Mathematically, for the user-centered model, the discrepancy between $\hat{r}^I_{ui}$ and $\hat{r}^U_{ui}$ would be

$$d(\hat{r}^I_{ui}, \hat{r}^U_{ui}) = -\hat{r}^I_{ui} \cdot \log \hat{r}^U_{ui} - (1 - \hat{r}^I_{ui}) \cdot \log(1 - \hat{r}^U_{ui}), \tag{7}$$

For the item-centered model, we use the dual form:

$$d(\hat{r}^U_{ui}, \hat{r}^I_{ui}) = -\hat{r}^U_{ui} \cdot \log \hat{r}^I_{ui} - (1 - \hat{r}^U_{ui}) \cdot \log(1 - \hat{r}^I_{ui}), \tag{8}$$

to measure the discrepancy between $\hat{r}^U_{ui}$ and $\hat{r}^I_{ui}$. We call (7) and (8) *hedge losses* for the prediction given by the dual models. Interestingly, other (possibly symmetric) distance metrics could be used, like L1 distance $d(x,y) = |x-y|$, L2 distance $d(x,y) = |x-y|^2$, logistic distance $d(x,y) = \frac{1}{1+exp(-|x-y|)}$, Gaussian function $d(x,y) = exp(\frac{|x-y|^2}{c})$, etc. In the experiment part, we will discuss the model performance under different distance metrics.

For one positive interaction (u,i,t), we sample K unobserved interactions (u_k, i_k, t_k) $(k = 1, \cdots, K)$ as negative samples. Then the hedge losses for dual models can be specified as:

$$\mathcal{L}^U_{hd} = \frac{1}{K} \sum_{k=1}^{K} -\hat{r}^I_{u_k i_k} \cdot \log \hat{r}^U_{u_k i_k} - (1 - \hat{r}^I_{u_k i_k}) \cdot \log(1 - \hat{r}^U_{u_k i_k}), \tag{9}$$

$$\mathcal{L}^I_{hd} = \frac{1}{K} \sum_{k=1}^{K} -\hat{r}^U_{u_k i_k} \cdot \log \hat{r}^I_{u_k i_k} - (1 - \hat{r}^U_{u_k i_k}) \cdot \log(1 - \hat{r}^I_{u_k i_k}), \tag{10}$$

where $\mathcal{L}^U_{hd}$ and $\mathcal{L}^I_{hd}$ are averaged hedge losses for user-centered and item-centered models, respectively.

Then we linearly combine the supervised loss and hedge loss together with a regularization term:

$$\mathcal{L}^U = \mathcal{L}^U_{sv} + \alpha \mathcal{L}^U_{hd} + \frac{\lambda}{2} \|\theta_U\|^2, \tag{11}$$

$$\mathcal{L}^I = \mathcal{L}^I_{sv} + \alpha \mathcal{L}^I_{hd} + \frac{\lambda}{2} \|\theta_I\|^2, \tag{12}$$

where α is the hyper-parameter that controls the balance between supervised learning and hedge loss based learning, and λ is the hyper-parameter to balance the optimization and regularization.

The training for the dual sequential models can be conducted alternately. First, the game begins with a sampled user u and generates a set of sampled items i_k $(k = 1, \cdots, K)$ from unobserved items of user u. Then the user-centered model will compute the predicted score $\hat{r}^U_{ui_k}$ and generate the supervised gradient $\nabla_{\theta_U} \mathcal{L}^U_{sv}$. Later on, we let the item-centered model output its predicted score $\hat{r}^I_{ui_k}$, which is used as a 'label', and compute the hedging gradient $\nabla_{\theta_U} \mathcal{L}^U_{hd}$. Finally, we add them together and use the synthetic gradient value $\nabla_{\theta_U} \mathcal{L}^U$ to update the user-centered model. This ends the first round of the game. In the 2nd round, the game will begin with a sampled item i and the roles of two models will interchange. Such process will repeat until convergence or after a number of epochs. We present the algorithm in Alg. 1 (Appendix A).

3.2 Specifications of Sequential Units

In this subsection, we further specify the forms of the sequential units and prediction units with two commonly-used sequential modules: recurrent neural network and self-attention module. To avoid repetition, we only present the specific forms for the user-centered sequential unit $SU_U(\cdot; \mathbf{w}_U)$ and prediction unit $PU_U(\cdot; \mathbf{y}_U)$.

Their counterparts for the item-centered model are presented in Appendix B.

3.2.1 Recurrent Neural Network Model. Recurrent neural networks (RNN) are widely used to model sequential patterns and temporal dependencies. Assume $\mathbf{h}_k$ to be the hidden states and $\mathbf{q}_k^U$ to be the embedding vector for the k-th item in $\mathcal{I}_u^t$ $(k = 1, \cdots, n)$. Then we have:

$$\mathbf{h}_k = \tanh(\mathbf{W}_I \mathbf{q}_k^U + \mathbf{W}_H \mathbf{h}_{k-1} + \mathbf{b}_I), \tag{13}$$

$$\mathbf{s}_u^t = \mathbf{W}_O \mathbf{h}_n + \mathbf{b}_O, \tag{14}$$

where $\mathbf{W}_I, \mathbf{W}_H, \mathbf{W}_O$ are weight matrices and $\mathbf{b}_I, \mathbf{b}_O$ are bias vectors. Here we use basic RNN units as an illustrative example, and one can also adopt other sophisticated structures like LSTM or GRU to capture complex temporal dependencies. We further concatenate the user embedding $\mathbf{p}_u^U$, item embedding $\mathbf{q}_i^U$ and dynamic state $\mathbf{s}_u$, and use a fully-connected layer to give the predicted score:

$$\hat{r}_{ui}^U = \sigma(\mathbf{W}_U \cdot [\mathbf{p}_u^U, \mathbf{q}_i^U, \mathbf{s}_u^t] + \mathbf{b}_U), \tag{15}$$

where σ, $\mathbf{W}_U$, $\mathbf{b}_U$ denote activation function, weight matrix, and bias vector respectively. We call this model *DEEMS-RNN*.

3.2.2 Self-Attention Model. The drawback of RNN model is that it assumes each element in the history sequence has the same importance. In fact, some elements in the sequence, like recently clicked items or similar items, can have more importances. Inspired by machine translation tasks [1], self-attention networks (SAN) for sequential prediction are proposed to address this problem. Specifically, the attention weights of candidate item i on a clicked item j can be

$$e_{ij} = \frac{\exp(\mathbf{W}_A[\mathbf{q}_j^U, \mathbf{q}_i^U] + \mathbf{b}_A)}{\sum_{k=1}^{n} \exp(\mathbf{W}_A[\mathbf{q}_k^U, \mathbf{q}_i^U] + \mathbf{b}_A)}. \tag{16}$$

Then we attentively add all historical clicked items as the dynamic factor:

$$\mathbf{s}_u^t = \sum_{j=1}^{n} e_{ij} \mathbf{q}_j^U. \tag{17}$$

Also, we leverage the fully-connect layer to give the final prediction:

$$\hat{r}_{ui}^U = \sigma(\mathbf{W}_U \cdot [\mathbf{p}_u^U, \mathbf{q}_i^U, \mathbf{s}_u^t] + \mathbf{b}_U). \tag{18}$$

Here, the self-attention model do not consider different positions of clicked items in the sequence. To reserve this information, one can encode the position as an embedding vector and add it to corresponding item embedding, like the works in [20]. We call the attention based model *DEEMS-SAN*.

Also, there are some other specifications for sequential prediction layer, like Markov chain, translation model, or some other advanced architecture, like memory network [4]. We leave it for future study.

3.3 Discussions

In this subsection, we discuss the motivations of the model design and provide some explanations about the efficacy of DEEMS. If we knew the true labels for all possible interactions and train the dual models in a supervised manner, we could exactly fit the true conditional probabilities $P(\text{click} \mid (u, i), \mathcal{I}_u^t)$ and $P(\text{click} \mid (u, i), \mathcal{U}_i^t)$. However, in fact, we only observed a few interactions whose labels are 1, and a large number of possible interactions are unobserved,

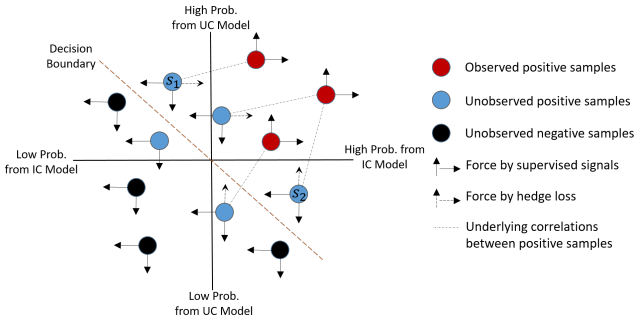

Figure 3: Illustration of training for DEEMS with user-centered (UC) model and item-centered (IC) model.

among which there exist both unobserved positive samples and negative ones. Unobserved positive samples refer to interactions that have not yet happened but are very likely to happen in the future. This is because the user has not been exposed to the item yet. By contrast, unobserved negative samples refer to interactions that have not yet happened and would also not happen in the future. In this situation, the user indeed dislikes the item. We have no prior knowledge to distinguish such unobserved positive and negative samples, since under most circumstances there is no auxiliary information that can help to unveil the true causal effects behind the data. Therefore, the unobserved positive samples will be treated as negative samples with label 0 and provide wrong supervised signals.

Fortunately, in DEEMS, the above problem can be alleviated to a certain degree. Compared with unobserved negative samples, unobserved positive samples are more likely to have correlations with observed positive samples. Such correlations can be further divided in two ways: user-centered correlation and item-centered correlation. For user-centered correlation, the candidate item is similar to one or some of user's clicked items. For item-centered correlation, the target user is related to one or some of item's infected users. If there exists at least one-kind correlation for the unobserved negative sample, it would be given a relatively high predicted score by either the user-centered or item-centered model. Then the hedging gradient would update the other model and push it to give a higher score. Finally, the unobserved positive samples would be ranked before the negative ones.

To make it easier to understand, let us illustrate it by an analogy in Fig. 3. We consider a plane with two orthogonal axes which stand for the predicted probabilities given by the user-centered and item-centered models respectively. Three kinds of samples are denoted by circles with different colors. First, the supervised signals from dual models would push the observed positive samples upward and rightward, and the unobserved samples downward and leftward. Then consider two specific sampled interactions $s_1 = (u_1, i_1, t_1)$ and $s_2 = (u_2, i_2, t_2)$. Assume that i_1 has a strong correlation with one of user u_1's clicked items, and then the user-centered model would give a relatively high probability for s_1. The feedback from user-centered model would further let the item-centered model think s_1 as a 'true' positive sample and update the predicted probability from low to high. Besides, for s_2, assume that u_2 is correlated with

one of item i_2's infected users, and then the item-centered model would give a relatively high probability. Later, the hedge loss from item-centered model would guide the user-centered model to think s_2 as a 'true' positive sample and push the sample upward. Note that s_1 and s_2 are common among all unobserved samples. Hence, once dual models reach a consensus, the unobserved positive samples would be more likely to be located at the right (or upper) side of the decision boundary than the negative ones.

4 EXPERIMENTS

To comprehensively evaluate the proposed model DEEMS[1], we conduct experiments to answer the following research questions:
RQ1 How does DEEMS perform compare with state-of-the-art methods for sequential prediction problems?
RQ2 Are the key components in DEEMS, such as dual models and hedge loss based training, necessary for improving performance?
RQ3 How do hyper-parameters in DEEMS impact the performance?

4.1 Experiment Setup

Datasets. We adopt the Amazon product data [15, 27] to construct the experimental datasets. The data contains the temporal users' purchasing behaviors ranging from the May 1996 to July 2014, with 24 categories of different sizes. Each category is a large tuple consisting of the information of user, item, rating and timestamp. We choose four categories *Clothing*, *Grocery*, *Baby* and *Digital Music* to conduct our experiments. We use two ways to split the training set and test set, and the specific data preparation is presented in Appendix C.

Comparative Methods. We use eight related methods proposed by previous researchers as competitors for comparison with our model: *PMF* [31][18], *FPMC* [29], *TransRec* [14][13], *DREAM* [44], *DIN* [46], *RMTP* [7], *TopoLSTM* [38], *RRN* [41]. Here, PMF is a basic matrix factorization based methods. FPMC combines Markov chain and matrix factorization, while TransRec leverages translation based model. Besides, DREAM and DIN are two strong baselines for sequential recommendation, using RNN and self-attention network to predict next clicked item for users, respectively. RMTP and TopoLSTM, as two strong baselines for information dissemination, leverages RNN and LSTM to predict next infected user for items, respectively. Finally, RRN is another strong baseline that takes both user sequence and item sequence as input of two different recurrent models. For each baseline, we set the hyper-parameters according to their reports in the paper and do some tuning by ourselves to achieve an optimal performance.

We also implement some variants and simplified versions of DEEMS to validate the necessity of some key components as ablation study, including *Dual-RNN*, *Dual-SAN*, *User-RNN*, *Item-RNN*, *User-SAN* and *Item-SAN*, *DEEMS-L1*, *DEEMS-L2*, *DEEMS-Log*, and *DEEMS-Gauss*. The details for implementation of proposed model and the above variants are presented in Appendix D and E.

Evaluation Protocol. We use four metrics to evaluate the model performance: P@k, AUC, GP@k, and GAUC. Here P@k and AUC are recommendation metrics, while GP@k and GAUC measure the global accuracy. The details for computing each metric are shown in Appendix F.

[1]The experiment codes are released at https://github.com/echo740/DEEMS-KDD-19.

4.2 Comparative Results: RQ1

We report experiment results of DEEMS and other comparative methods in Table 1. As we can see, our model DEEMS-RNN and DEEMS-SAN outperform other comparative methods and achieve significant improvements for all evaluation metrics. Also, DEEMS-SAN achieves better performance on recommendation metrics (P@3 and AUC), while DEEMS-RNN performs better on global metrics (GP@3 and GAUC). Besides, there are some findings in these comparative experiments. First, matrix factorization based method (i.e., PMF) gives the worst prediction accuracy since they fail to capture any temporal dependencies among interactions. Second, Markov model and translation based model perform better than matrix factorization methods, but are still not good enough compared with deep models. As for RNN based models and self-attention based models, we find that DREAM performs better than DIN on global metrics while the opposite is true on recommendation metrics. This is possibly because self-attention modules help to select important elements in history sequences for different users (resp. items), but such importances are estimated over clicked items (resp. infected users) for one user (resp. item) rather than all candidate items (reps. users) among the dataset due to the softmax layer. Furthermore, two user-centered models (DREAM, DIN) tend to perform slightly better than the item-centered models (RMTP, TopoLSTM) on recommendation metrics , but their performances are not that different on global metrics.

4.3 Ablation Study: RQ2

We conduct a series of ablation studies to investigate the necessities of some key components in our model.

Effectiveness of Dual Models and Hedge Loss. Fig. 4 (resp. Fig.5) shows the results of DEEMS-RNN (resp. DEEMS-SAN) and its variants. As is depicted by the figures, the dual sequential prediction models improve the prediction accuracy compared with the single user-centered or item-centered model, and achieve averaged 0.6% improvement for AUC. The possible reason is because dual models can take advantage of two-side information from two history sequences. Besides, the hedge loss based training contributes to averagely 0.9% improvement for AUC over dual models with no hedging gradient. Indeed, the hedge loss could merge two-side information and adds enough random noise to the training process which helps to get out of local optimums.

In Fig. 6, we present the learning curves of synthetic prediction in DEEMS as well as the predictions given by a single user-centered and item-centered model. As we can see, at the beginning of training, user-centered model performs the best. As the training epochs increases, the performance of synthetic prediction becomes better and outperforms those of other two predictions.

Different Distance Metrics. We compare the performances under different distance metrics in Table 4. The results show that the prediction performances of DEEMS are sensitive to distinct distance metrics, among which the cross-entropy (i.e., what we use in the paper) contributes to the best performance. Besides, the logistic distance performs slightly worse than the cross-entropy, but outperforms other metrics. The L1 and L2 distances give the worst results. In fact, the distance metrics have a great impact on the ways to reach the equilibrium while training.

Table 1: Experiment results of DEEMS and competitors when we hold out the last two clicked items of each user for test. The bold value marks the best one in one column, while the underlined value corresponds to the best one among all baselines.

	Digital				Baby				Grocery				Clothing			
	P@3	AUC	GP@3	GAUC	P@3	AUC	GP@3	GAUC	P@3	AUC	GP@3	GAUC	P@3	AUC	GP@3	GAUC
PMF	0.511	0.752	0.691	0.830	0.523	0.765	0.631	0.765	0.518	0.754	0.615	0.753	0.481	0.738	0.639	0.775
FPMC	0.515	0.765	0.717	0.845	0.527	0.766	0.639	0.770	0.520	0.776	0.620	0.759	0.492	0.739	0.646	0.770
TransRec	0.518	0.776	0.691	0.842	0.531	0.787	0.617	0.771	0.521	0.765	0.605	0.754	0.495	0.746	0.624	0.771
DREAM	0.529	0.780	0.724	0.855	0.525	0.778	0.648	0.785	0.524	0.775	0.633	0.763	0.507	0.750	0.652	0.789
DIN	0.531	0.779	0.712	0.843	0.529	0.776	0.643	0.782	0.524	0.777	0.630	0.768	0.503	0.751	0.648	0.789
RMTP	0.528	0.771	0.722	0.861	0.521	0.781	0.646	0.784	0.523	0.776	0.634	0.767	0.493	0.745	0.656	0.791
TopoLSTM	0.523	0.777	0.716	0.845	0.524	0.784	0.648	0.782	0.523	0.778	0.634	0.768	0.497	0.747	0.652	0.782
RRN	0.524	0.775	0.716	0.859	0.531	0.789	0.653	0.781	0.521	0.775	0.638	0.775	0.495	0.751	0.660	0.790
DEEMS-RNN	**0.564**	**0.827**	**0.726**	**0.867**	0.533	0.791	**0.659**	**0.793**	**0.529**	**0.783**	0.640	**0.781**	**0.530**	**0.789**	**0.667**	**0.800**
DEEMS-SAN	0.524	0.793	0.716	0.867	**0.553**	**0.809**	0.655	0.792	0.521	0.762	**0.644**	0.780	0.516	0.780	0.663	0.799

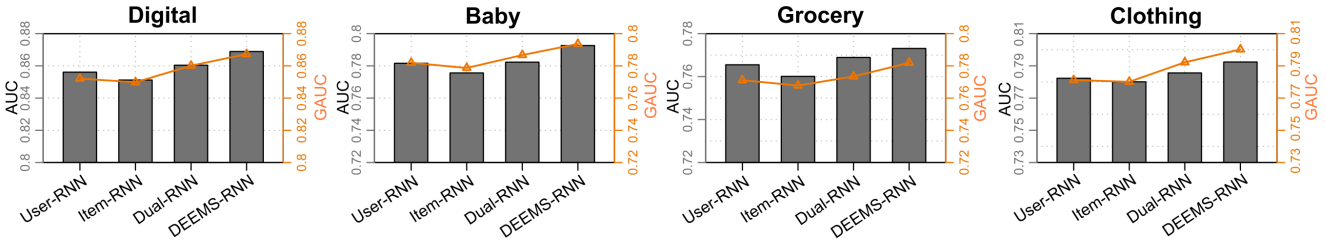

Figure 4: Ablation studies of dual models and hedge loss based training for DEEMS-RNN.

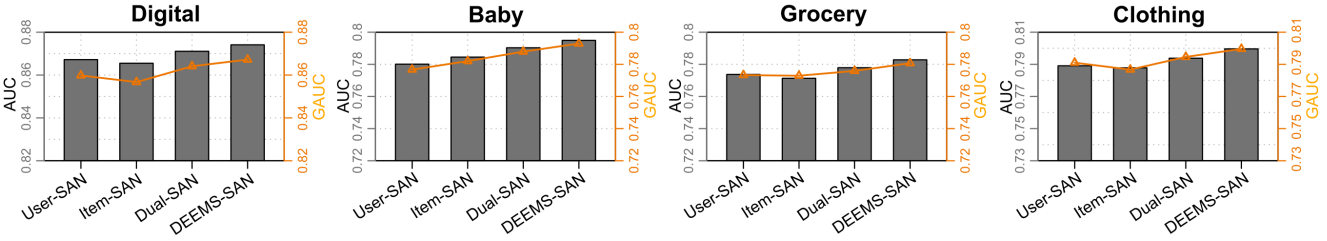

Figure 5: Ablation studies of dual models and hedge loss based training for DEEMS-SAN.

4.4 Parameter Sensitivity: RQ3

We study the performance variation for our model w.r.t some hyper-parameters and the comparative results are presented in Fig. 7.

Impact of Embedding Size. We let the embedding size varies from 5 to 25 with step size 5, and the AUC value improves at first and decays latter. The possible reason is that small embedding size may deprive the model of enough expressiveness, and too large embedding size would make the representation vector too sparse, which leads to performance decline.

Impact of Regularization. We change the regularization coefficient from 0.001 to 10, and the best performance is achieved when it is set to 0.1. In fact, the regularization can help to limit the values of model parameters and avoid over-fitting. A proper trade-off between regularization and accuracy optimization is needed. Too small and too large λ could both break the balance.

Impact of Weight for Hedge Loss. We let α change from 0.001 to 10 to study the performance variation under different weights of hedge loss. The weight coefficient controls the balance between supervised training and hedge loss based training. The figure shows that $\alpha = 0.1$ achieves the best trade-off.

Table 2: Performance comparison under different metrics for hedge loss on Baby dataset.

	P@3	AUC	GP@3	GAUC
DEEMS - RNN - L1	0.5290	0.7841	0.6522	0.7807
DEEMS - RNN - L2	0.5312	0.7830	0.6529	0.7825
DEEMS - RNN - Log	0.5329	**0.7934**	0.6575	0.7902
DEEMS - RNN - Gauss	0.5301	0.7854	0.6510	0.7875
DEEMS - RNN	**0.5332**	0.7914	**0.6595**	**0.7966**
DEEMS - SAN - L1	0.5337	0.8014	0.6441	0.7850
DEEMS - SAN - L2	0.5356	0.8041	0.6414	0.7828
DEEMS - SAN - Log	0.5423	0.8097	0.6541	0.7892
DEEMS - SAN - Gauss	0.5375	0.8015	0.6508	0.7882
DEEMS - SAN	**0.5531**	**0.8091**	**0.6556**	**0.7928**

Impact of Truncation Length. We also study the impact of truncation length for history sequences. When it changes from 5 to 25, the AUC values become larger. Such phenomenon is quite normal since large truncation length contributes to more complete information from temporal sequences. However, large m may lead to heavy computational cost, so we set $m = 20$ in our experiments.

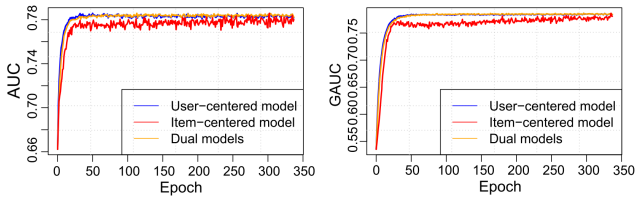

Figure 6: Learning curves of AUC and GAUC for user-centered, item-centered and dual models on Baby dataset.

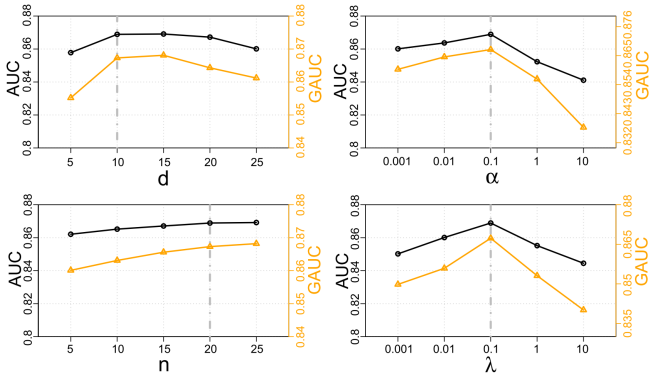

Figure 7: Performance of DEEMS on Baby dataset w.r.t different hyper-parameters. The vertical dotted lines mark the values set in this paper.

5 LINKS TO RELATED WORKS

Sequential Recommendations. In a recommendation system, the sequential behaviors of a user often expose one's interest. Sequential recommendation aims at predicting user's next clicked item based on the historical clicking sequence. One method is by Markov chain, modeling the local sequential patterns between every adjacent pair of user's clicked items [9]. Later on, [29] proposes factorized personalized Markov chains which integrates Markov chains and matrix factorization method. Furthermore, the hierarchical representation model (HRM) [40] takes transaction and user representations into the prediction to model complicated interactions among differenet factors. Different from the above works that only consider the adjacent dependency between clicked items, [44] proposes to use a dynamic recurrent model (DREAM) to capture global sequential patterns by incoporating users' current interests and global sequential features based on reccurent neural network (RNN). Moreover, TransRec [14] takes a different view and models user's dynamic interest as translation among different languages. It embeds items into a transition space modeling a third-order relationship among a user, her previously visited item(s), and the next item to consume. Recently, inspired by machine translation task [1], works in [20, 46] leverage self-attention mechanism to model different importances of past clicked items, and achieve accuracy improvement. Nevertheless, the above-mentioned approaches limit the scope in user-centered aspect, and they ignore the dynamic item's states hidden in different users who click on such item.

Information Dissemination. On many social platforms, information like news, events, pictures or some fashion elements may disseminate in a predictable pattern. With some early-stage observed information about one item, we can predict the future evolution and potential impact [28][5]. Early studies use matrix factorization to predict who will be influenced in the future [19] [18]. Later on, several time series and point process models are proposed to address this problem by modelling the temporal dependencies [45][8]. Recently, [6] proposes to use RNN to capture the hidden dynamics in temporal evolution of item's infection sequence. The work in [39] extends this idea, using RNN to predict the next infected user. Also, recent studies [3] and [17] adopt self-attention mechanism to probe into different importances of historical infected users. Information dissemination models take an item-centered aspect, and the majority of them ignore the dynamic user's states that will change with different exposed items.

Dual Mechanism. Dual phenomena are very common in real-life situations. Based on this, many recent works propose to add dual structures into previous model and achieve great performance improvement. For example, [12] proposes a dual learning method that allows two translation models to train by themselves through the reward from the opponent. Later on, the work in [43] extends such dual design to model level. Also, [47] designs dual Graph Convolutional Networks (GCN) which could capture both global and local consistencies in complicated graphs. Furthermore, dual-attention mechanisms are widely used to handle scene segmentation [11], dialogue act classification [24] ,and question answering [22]. Different from these works, DEEMS takes dual aspects to unify sequential recommendation and information dissemination models, and makes them communicate with each other to reach a consensus. The two sequential prediction models take different input sequences, output the predicted probability based on user-centered (resp. item-centered) temporal dependencies, and use the prediction by the other model as feedback for training.

Model Unification. Recently, there exist several researches that attempt to unify different schools of thinking in one domain. The work in [37] proposes a minimax game which unifies the discriminative and generative models for information retrieval. Also, [42] designs an adversarial framework to take advantage of both feature driven and point process models for popularity prediction. Besides, [21] proposes a dynamic meta-embedding methods for sentence representation and combines a set of weak representation models. [30] constructs a hybrid model that unites the epidemic model and the Hawkes model for information prediction. Compared with existing approaches, DEEMS takes a different view and leverage the dual structure to unify the sequential recommendation and information dissemination models. First, in DEEMS, the user-centered and item-centered models are two self-contained models that can both give prediction, which differs from the case in [30] and [21] where several models with the same goal are combined into a synthetic one. Second, DEEMS makes dual models communicate with each other through the hedge losses to achieve a common goal, which differs from the case in [37] and [42] where two models struggle to defeat each other in an adversarial manner.

6 CONCLUSION

In this paper, we propose dual sequential models DEEMS to unify two schools of thinking (sequential recommendation and information dissemination). Specifically, DEEMS contains two self-contained models: user-centered model and item-centered model. The user-centered model aims at capturing the dynamic patterns in user's

historical clicking behaviors, while the item-centered model attempts to model the dynamic states from item's historical infected users. For model training, we let the two models play a game with each other where one model will use the predicted score of the other model as feedback to guide the training. Such mechanism enables the framework to efficiently take advantage of dual information and more competent in distinguishing the false negative samples. Also, DEEMS is a general framework, it can be applied to arbitrary sequential prediction models, like RNN unit, self-attention module, Memory network, etc. Comprehensive experiments demonstrate the superiority of DEEMS under different settings.

For future study, we believe that one can probe intro the social network or item network relationship by using some graph representation techniques to improve the current embedding layers, if the graph topology information is known. More generally, one could also consider more than one kind of interactions (like browse, click, purchase, review, etc.) and capture more complicated user's behaviors and item's diffusion patterns.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[2] Peng Bao, Hua-Wei Shen, Xiaolong Jin, and Xue-Qi Cheng. 2015. Modeling and Predicting Popularity Dynamics of Microblogs Using Self-Excited Hawkes Processes. In *WWW*.

[3] Qi Cao, Huawei Shen, Keting Cen, Wentao Ouyang, and Xueqi Cheng. 2017. DeepHawkes: Bridging the Gap between Prediction and Understanding of Information Cascades. In *CIKM*. 1149–1158.

[4] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential Recommendation with User Memory Networks. In *WSDM*. 108–116.

[5] Justin Cheng, Lada A. Adamic, P. Alex Dow, Jon M. Kleinberg, and Jure Leskovec. 2014. Can cascades be predicted?. In *WWW*. 925–936.

[6] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *KDD*. 1555–1564.

[7] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *KDD*. 1555–1564.

[8] Mehrdad Farajtabar, Manuel Gomez-Rodriguez, Yichen Wang, Shuang Li, Hongyuan Zha, and Le Song. 2018. COEVOLVE: A Joint Point Process Model for Information Diffusion and Network Co-evolution. In *Companion Proceedings of the The Web Conference 2018 (WWW)*.

[9] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *IJCAI*. 2069–2075.

[10] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *IJCAI*. 2069–2075.

[11] Jun Fu, Jing Liu, Haijie Tian, Zhiwei Fang, and Hanqing Lu. 2018. Dual Attention Network for Scene Segmentation. *CoRR* abs/1809.02983 (2018).

[12] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual Learning for Machine Translation. In *NIPS*. 820–828.

[13] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2018. Translation-based Recommendation: A Scalable Method for Modeling Sequential Behavior. In *IJCAI*. 5264–5268.

[14] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys)*.

[15] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*. 507–517.

[16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

[17] Mohammad Raihanul Islam, Sathappan Muthiah, Bijaya Adhikari, B. Aditya Prakash, and Naren Ramakrishnan. 2018. DeepDiffuse: Predicting the 'Who' and 'When' in Cascades. In *ICDM*. 1055–1060.

[18] Bo Jiang, Jiguang Liang, Ying Sha, Rui Li, Wei Liu, Hongyuan Ma, and Lihong Wang. 2016. Retweeting Behavior Prediction Based on One-Class Collaborative Filtering in Social Networks. In *SIGIR*. 977–980.

[19] Bo Jiang, Jiguang Liang, Ying Sha, and Lihong Wang. 2015. Message Clustering based Matrix Factorization Model for Retweeting Behavior Prediction. In *CIKM*. 1843–1846.

[20] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. 197–206.

[21] Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Dynamic Meta-Embeddings for Improved Sentence Representations. In *EMNLP*. 1466–1477.

[22] Kyung-Min Kim, Seong-Ho Choi, Jin-Hwa Kim, and Byoung-Tak Zhang. 2018. Multimodal Dual Attention Memory for Video Story Question Answering. In *ECCV*. 698–713.

[23] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *KDD*. 447–456.

[24] Ruizhe Li, Chenghua Lin, Matthew Collinson, Xiao Li, and Guanyi Chen. 2018. A Dual-Attention Hierarchical Recurrent Neural Network for Dialogue Act Classification. *CoRR* abs/1810.09154 (2018).

[25] Changsha Ma, Zhisheng Yan, and Chang Wen Chen. 2017. LARM: A Lifetime Aware Regression Model for Predicting YouTube Video Popularity. In *CIKM*. 467–476.

[26] Yasuko Matsubara, Yasushi Sakurai, B. Aditya Prakash, Lei Li, and Christos Faloutsos. 2012. Rise and fall patterns of information diffusion: model and implications. In *KDD*. 6–14.

[27] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.

[28] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2011. RT to Win! Predicting Message Propagation in Twitter. In *ICWSM*.

[29] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *WWW*.

[30] Marian-Andrei Rizoiu, Swapnil Mishra, Quyu Kong, Mark Carman, and Lexing Xie. 2018. SIR-Hawkes: Linking Epidemic Models and Hawkes Processes to Model Diffusions in Finite Populations. In *Proceedings of the 2018 World Wide Web Conference (WWW)*.

[31] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *NIPS*. 1257–1264.

[32] Hua-Wei Shen, Dashun Wang, Chaoming Song, and Albert-László Barabási. 2014. Modeling and Predicting Popularity Dynamics via Reinforced Poisson Processes. In *AAAI*. 291–297.

[33] Karthik Subbian, B. Aditya Prakash, and Lada A. Adamic. 2017. Detecting Large Reshare Cascades in Social Networks. In *WWW*. 597–605.

[34] Bongwon Suh, Lichan Hong, Peter Pirolli, and Ed H. Chi. 2010. Want to be Retweeted? Large Scale Analytics on Factors Impacting Retweet in Twitter Network. In *PASSAT*. 177–184.

[35] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *DLRS*. 17–22.

[36] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*. 565–573.

[37] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *SIGIR*. 515–524.

[38] Jia Wang, Vincent W. Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. 2017. Topological Recurrent Neural Network for Diffusion Prediction. In *ICDM*. 475–484.

[39] Jia Wang, Vincent W. Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. 2017. Topological Recurrent Neural Network for Diffusion Prediction. In *ICDM*. 475–484.

[40] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for NextBasket Recommendation. In *SIGIR*. 403–412.

[41] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In *WSDM*. 495–503.

[42] Qitian Wu, Chaoqi Yang, Hengrui Zhang, Xiaofeng Gao, Paul Weng, and Guihai Chen. 2018. Adversarial Training Model Unifying Feature Driven and Point Process Perspectives for Event Popularity Prediction. In *CIKM*. 517–526.

[43] Yingce Xia, Xu Tan, Fei Tian, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2018. Model-Level Dual Learning. In *ICML*. 5379–5388.

[44] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[45] Qingyuan Zhao, Murat A. Erdogdu, Hera Y. He, Anand Rajaraman, and Jure Leskovec. 2015. SEISMIC: A Self-Exciting Point Process Model for Predicting Tweet Popularity. In *KDD*. 1513–1522.

[46] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *SIGKDD*. 1059–1068.

[47] Chenyi Zhuang and Qiang Ma. 2018. Dual Graph Convolutional Networks for Graph-Based Semi-Supervised Classification. In *WWW*.

A TRAINING ALGORITHM FOR DEEMS

Algorithm 1: Training for Dual Sequential Models

REQUIRE: interaction triple set $\mathcal{T}$.
REQUIRE: θ_U, initial parameters for user-centered model. θ_I, initial parameters for item-centered model. $\eta = 0.1$, learning rate. $\lambda = 0.01$, regularization coefficient. $B = 64$, batch size. $K = 5$, sample size. $\alpha = 0.1$, weight for hedge loss.
while *not converged* **do**
  Sample B user-item interactions $\mathcal{B} = \{(u_b, i_b, t_b)\}_{b=1}^{B}$ from $\mathcal{T}$;
  $\mathcal{L}_{sv}^{U} \leftarrow 0$, $\mathcal{L}_{hd}^{U} \leftarrow 0$, $\mathcal{L}_{sv}^{I} \leftarrow 0$, $\mathcal{L}_{hd}^{I} \leftarrow 0$;
  for *each sampled interaction* $(u, i, t) \in \mathcal{B}$ **do**
    Generate K items i_k uniformly among the unobserved pairs $(u, i') \notin \mathcal{T}$;
    $\mathcal{L}_{sv}^{U}$ &+ $= -\log \hat{r}_{ui}^{U} - \frac{1}{K}\sum_{k=1}^{K} \log(1 - \hat{r}_{ui_k}^{U})$;
    $\mathcal{L}_{hd}^{U}$ + = $-\frac{1}{K}\sum_{k=1}^{K} \hat{r}_{ui_k}^{I} \cdot \log \hat{r}_{ui_k}^{U} + (1 - \hat{r}_{ui_k}^{I}) \cdot \log(1 - \hat{r}_{ui_k}^{U})$;
  $\theta_U \leftarrow \theta_U - \eta \left[\nabla_{\theta_U} \mathcal{L}_{sv}^{U} + \alpha \nabla_{\theta_U} \mathcal{L}_{hd}^{U} + \lambda \|\theta_U\|\right]$;
  for *each sampled interaction* $(u, i, t) \in \mathcal{B}$ **do**
    Generate K user u_k uniformly among the unobserved pairs $(u', i) \notin \mathcal{T}$;
    $\mathcal{L}_{sv}^{I}$ &+ $= -\log \hat{r}_{ui}^{I} - \frac{1}{K}\sum_{k=1}^{K} \log(1 - \hat{r}_{u_k i}^{I})$;
    $\mathcal{L}_{hd}^{I}$ + = $-\frac{1}{K}\sum_{k=1}^{K} \hat{r}_{u_k i}^{U} \cdot \log \hat{r}_{u_k i}^{I} + (1 - \hat{r}_{u_k i}^{U}) \cdot \log(1 - \hat{r}_{u_k i}^{I})$;
  $\theta_I \leftarrow \theta_I - \eta \left[\nabla_{\theta_I} \mathcal{L}_{sv}^{I} + \alpha \nabla_{\theta_I} \mathcal{L}_{hd}^{I} + \lambda \|\theta_I\|\right]$;

B SPECIFICATIONS FOR ITEM-CENTERED SEQUENTIAL UNITS AND PREDICTION UNITS

The sequential units and prediction units in item-centered model possess the dual structures compared with those in user-centered model.

B.1 Recurrent Neural Network Model

Assume $\mathbf{h}_k$ to be the hidden states, $\mathbf{p}_k^I$ to be the embedding vector for the k-th user in $\mathcal{U}_i^t$ ($k = 1, \cdots, m$), and then we have,

$$\mathbf{h}_k = \tanh(\mathbf{W}'_I \mathbf{p}_k^I + \mathbf{W}'_H \mathbf{h}_{k-1} + \mathbf{b}'_I), \tag{19}$$

$$\mathbf{l}_i^t = \mathbf{W}'_O \mathbf{h}_m + \mathbf{b}'_O, \tag{20}$$

where $\mathbf{W}'_I, \mathbf{W}'_H, \mathbf{W}'_O$ are weight matrices and $\mathbf{b}'_I, \mathbf{b}'_O$ are bias vectors. We further concatenate the user embedding $\mathbf{p}_u^I$, item embedding $\mathbf{q}_i^I$ and dynamic item state $\mathbf{l}_i$, and use a fully-connected layer to give the predicted score:

$$\hat{r}_{ui}^I = \sigma(\mathbf{W}'_U \cdot [\mathbf{p}_u^I, \mathbf{q}_i^I, \mathbf{l}_i^t] + \mathbf{b}'_U), \tag{21}$$

where σ, $\mathbf{W}'_U$, $\mathbf{b}'_U$ denote activation function, weight matrix, bias vector, respectively.

B.2 Self-Attention Model

The attention weights of target user u on a past infected user v can be

$$e_{uv} = \frac{\exp(\mathbf{W}'_A[\mathbf{p}_v^I, \mathbf{p}_u^I] + \mathbf{b}'_A)}{\sum_{k=1}^{m} \exp(\mathbf{W}'_A[\mathbf{p}_w^I, \mathbf{p}_u^I] + \mathbf{b}'_A)}. \tag{22}$$

Then we attentively add all historical clicked items as the dynamic factor:

$$\mathbf{l}_i^t = \sum_{v=1}^{n} e_{uv} \mathbf{p}_v^I. \tag{23}$$

Also, we leverage the fully-connect layer to give the final prediction:

$$\hat{r}_{ui}^I = \sigma(\mathbf{W}'_U \cdot [\mathbf{p}_u^I, \mathbf{q}_i^I, \mathbf{l}_i^t] + \mathbf{b}'_U). \tag{24}$$

C DATA PREPARATION

Since we focus on implicit feedback scenario, we consider all existing rating as observed positive interactions whose label is 1. Followed by other related works [10, 14, 29, 36], we filter out users and items with less than five interactions. For each dataset, similar to related works [29, 44, 46], we use the last two clicked items of each user for test, and the remaining interactions for training. The statistics of each dataset is given in Table 3.

Table 3: Statistics of four datasets.

Datasets	#Users	#Items	#Interactions	Density
Clothing	184050	174484	1068972	0.003%
Grocery	32126	39264	275256	0.022%
Baby	27619	18749	216310	0.042%
Digital	20165	20356	132595	0.032%

D IMPLEMENTATION DETAILS FOR DEEMS

We use Tensorflow to implement our model and deploy it on GTX 1080 GPU with 8G memory. For parameter tuning, we adopt coordinate descendant method to search for a sub-optimal settings. Some hyper-parameter setting is provided in Alg. 1, and other parameters are set as follows. The embedding size $d = 10$ for *Digital*, *Baby*, *Grocery*, and $d = 20$ for *Clothing*. Since the history sequence for different users (resp. items) are quite different, we truncate the recent $n = 20$ clicked items as user's history item sequence and $m = 20$ infected users as item's history user sequence. For optimization, we adopt stochastic gradient descendant to conduct training and clip the gradient within step size $c = 20$ to avoid gradient explosion.

E IMPLEMENTATION DETAILS FOR VARIANTS OF DEEMS

For *Dual-RNN*, we remove the hedge loss term from the loss function in (11) and (12),

$$\mathcal{L}^U = \mathcal{L}^U_{sv} + \frac{\lambda}{2}\left\|\theta_U\right\|^2, \tag{25}$$

$$\mathcal{L}^I = \mathcal{L}^I_{sv} + \frac{\lambda}{2}\left\|\theta_U\right\|^2. \tag{26}$$

For *User-RNN* and *User-SAN*, we directly use the prediction given by user-centered model as the final prediction:

$$\hat{r}_{ui} = r^U_{ui} = \sigma(\mathbf{W}_U \cdot [\mathbf{p}^U_u, \mathbf{q}^U_i, \mathbf{s}^t_u] + \mathbf{b}_U). \tag{27}$$

The difference is that in *User-RNN* the dynamic state $\mathbf{s}^t_u$ is computed by (13) and (14), and in *User-SAN* the dynamic state $\mathbf{s}^t_u$ is given by (16) and (17). Then for *Item-RNN* and *Item-SAN*, we directly use the prediction given by item-centered model as the final prediction:

$$\hat{r}_{ui} = r^{(2)}_{ui} = \sigma(\mathbf{W}'_U \cdot [\mathbf{p}^{(2)}_u, \mathbf{q}^{(2)}_i, \mathbf{l}^t_i] + \mathbf{b}'_U). \tag{28}$$

Also, for *Item-RNN*, the dynamic state $\mathbf{l}^t_i$ will be given by (19) and (20), while for *Item-SAN*, it will be given by (22) and (23).

We also probe into different discrepancy metrics to construct the hedge loss. We list the metrics we adopt in Table 4.

Table 4: Different discrepancy metrics for hedge loss.

Method	$d(r^I_{ui}, d^U_{ui})$	$d(r^U_{ui}, d^I_{ui})$
DEEMS-L1	$\lvert r^I_{ui} - r^U_{ui}\rvert$	$\lvert r^U_{ui} - r^I_{ui}\rvert$
DEEMS-L2	$\lvert r^I_{ui} - r^U_{ui}\rvert^2$	$\lvert r^U_{ui} - r^I_{ui}\rvert^2$
DEEMS-Log	$\frac{1}{1+\exp(-\lvert r^I_{ui} - r^U_{ui}\rvert)}$	$\frac{1}{1+\exp(-\lvert r^U_{ui} - r^I_{ui}\rvert)}$
DEEMS-Gauss	$\exp(\lvert r^I_{ui} - r^U_{ui}\rvert^2)$	$\exp(\lvert r^U_{ui} - r^I_{ui}\rvert^2)$

F EVALUATION METRICS

For each model, we define $E_u = \{\hat{r}_{ui_1}, \hat{r}_{ui_2}, ..., \hat{r}_{ui_k}\}$ as user u's top k recommendation list, and $\hat{r}_{ui_x}$ represents the predicted score at the x-th position of the list E_u. Define G_u to be the set of user u's interacted items in the test set. Assume there are M users in our system. Then two recommendation metrics can be calculated as follows.

- *Precision@k (P@k)* counts the per-user average top k precision:

$$P@k = \frac{1}{M}\sum_u P_u@k = \frac{1}{M}\sum_u \frac{E_u \cap G_u}{\hat{E}_u}. \tag{29}$$

- *AUC* counts the per-user average area under the curve:

$$AUC = \frac{\sum_{i\in\mathcal{I}^+_u}\sum_{j\in\mathcal{I}^-_u}\delta(\hat{y}_{u,i,j} > 0)}{|\mathcal{I}^+_u||\mathcal{I}^-_u|}, \tag{30}$$

  where $\mathcal{I}^+_u = \{i|r_{ui} = 1\}$ and $\mathcal{I}^-_u = \{j|r_{uj} = 0\}$ denote the sets of clicked items and not clicked items for user u respectively. The counting function $\delta(\hat{r}_{ui} > \hat{r}_{uj})$ returns 1 when $\hat{r}_{ui} > \hat{r}_{uj}$ and 0 otherwise.

Furthermore, we define E as the list of samples with top $k\%$ predicted scores over all test samples, and G as the list of all positive samples in test set. Then we define another two global metrics.

- *GP@k* counts the precision of top $k\%$ samples in test set:

$$GP@k = \frac{E \cap G}{\hat{E}}. \tag{31}$$

- *GAUC* counts the area under the curve over all test samples:

$$GAUC = \frac{\sum_{(u,i)\in\mathcal{S}^+}\sum_{(v,j)\in\mathcal{S}^-}\delta(\hat{r}_{ui} > \hat{r}_{vj})}{|\mathcal{S}^+||\mathcal{S}^-|}, \tag{32}$$

  where $\mathcal{S}^+ = \{(u,i)|r_{ui} = 1\}$ and $\mathcal{S}^- = \{(v,j)|r_{vj} = 0\}$ denote the sets of positive samples and negative samples in test set, respectively.

G PARAMETER SEARCHING SPACE

We present the searching spaces for each hyper-parameter in our model in Table 5.

Table 5: Parameter searching space.

Hyper-parameter	Definition	Search space
η	learning rate	[10, 1, 0.1, 0.01, 0.001]
B	batch size	[16, 32, 64, 128, 256]
λ	regularization	[1, 0.1, 0.01, 0.001]
m, n	truncation length	[30, 25, 20, 15, 10, 5]
d	embedding size	[25, 20, 15, 10, 5]
c	gradient clipping	[25, 20, 15, 10, 5]