Interest-Evolution Framework with Delta Generative Network for Session-based Recommendation

Zhihao Han
Beijing University of Posts and Telecommunications
Beijing, China
Email: hanzhihao@bupt.edu.cn

***Abstract*—Session-based recommendation aims to predict users' next clicks only based on anonymous sessions rather than accessing historical user-item interactions. Recent work on session-based recommendation focused on users' general interests or relationship of items with short term memory methods or collaborative filtering, which neglect the influence of items' order on interest drift. However, from empirical analysis, we considered that each item have multiple concepts meanwhile user interests usually evolve over time dynamically in behaviour sequences. Moreover, none of the existing approaches explicitly captures interest evolution in single session. To address this issue, we proposed Interest-Evolution Framework with Delta Generative Network(IEF-DGN). Specifically, we design two intention generators for producing virtual items utilising multiple concepts. Taking into account of item embeddings in interaction sequences, we propose the interest evolving matrix to capture interest evolving process. Extensive experiments conducted on four public real-world datasets demonstrate IEF-DGN can outperform state-of-the-art models for session-based recommendation. The code is available at: https://github.com/HanZhhao/IEF-DGN**

***Index Terms*—multiple interest evolution, concept drift, session-based recommendation**

I. Introduction

With the rapid emergence of e-commerce, the role of session-based recommendation(SBR) has received increased attention in recent years[1]. SBR is a challenging task to predict users behaviour within anonymous sessions. Traditional recommendation methods[2,3,4] usually adopt collaborative filtering to predict users' next clicks by assuming similar users have the same interests. With the development of deep learning, many SBR studies[5,6,7,8] are based on the neural network. These prior studies allowed for a better understanding of user-item interaction.

A massive number of existing recommendation approaches[9,10] have highlighted the importance of general interests and focused on designing neural network encoders in a wide variety of recommender systems, such as Recurrent Neural Network (RNN)[11] and Transformer[12]. Although these encoders have been proven effective in learning users' general interests, it's still hard to solve the problem of capturing the interest evolution within single session due to the limited information. Hence, some recent works[13,14,15] introduced the generative adversarial network to recommendation system. The training procedure of it is a min-max game, which is trying to minimise the loss function meanwhile generating realistic users. Despite the fake users can confuse the discriminator, there are a few clues toward that these inexistent user embeddings reflect real interests.

However, existing methologies still have certain limitations. First of all, when user behavior in one session is insufficient, it is difficult to estimate the user representation. Generally, the hidden vectors of RNN models are taken as the user representation, and then the global recommender of NARM[16] can be used to generate recommendations. In session-based recommendation systems, however, sessions are usually anonymous and numerous, and the user behavior implied in the session clicks is often limited, making it difficult to accurately estimate the user's representation from each session. Besides, previous studies[17] have shown that the pattern of item transitions is important and can be used as a local factor in session-based recommendation, but these methods[18,19] only model the single-way transition between consecutive items, ignoring the transition among the contexts, i.e. other items in the session. As a result, complex transitions between distant items are often overlooked by these methods.

Extracting multiple embedding vectors from user behavior sequences in industry-level data presents several challenges. Initially, items may not be conceptually clustered well in real systems, and the category information of items may not be available or reliable due to annotation noise. After that, inferring a set of interested concepts for a user from a large concept pool involves a selection operation, which is a discrete optimization problem and difficult to train end-to-end. Moreover, determining which interest is likely to be activated for next-item predictions requires the model to predict a user's next intention adaptively, as the inference stage has no labels from the next predicted item as is available during training.

In line with the view that users' interests drift with time and items have multiple concepts, we proposed Interest-Evolution Framework with Delta Generative Network(IEF-DGN), which enables to capture users' interest evolution and multiple concepts of items. As shown in Figure 1, we design two intention generators for producing virtual items utilising multiple concepts. After that, the processed embeddings in interaction sequences would be matched with some concepts in intention pool. Specifically, we propose a interest evolving matrix to capture interest evolving process by virtue that item concepts drift as the order of sequences. For example, Andy clicked commodity links of red shoes and blue watch, then recommendation system would learn the interest drift from

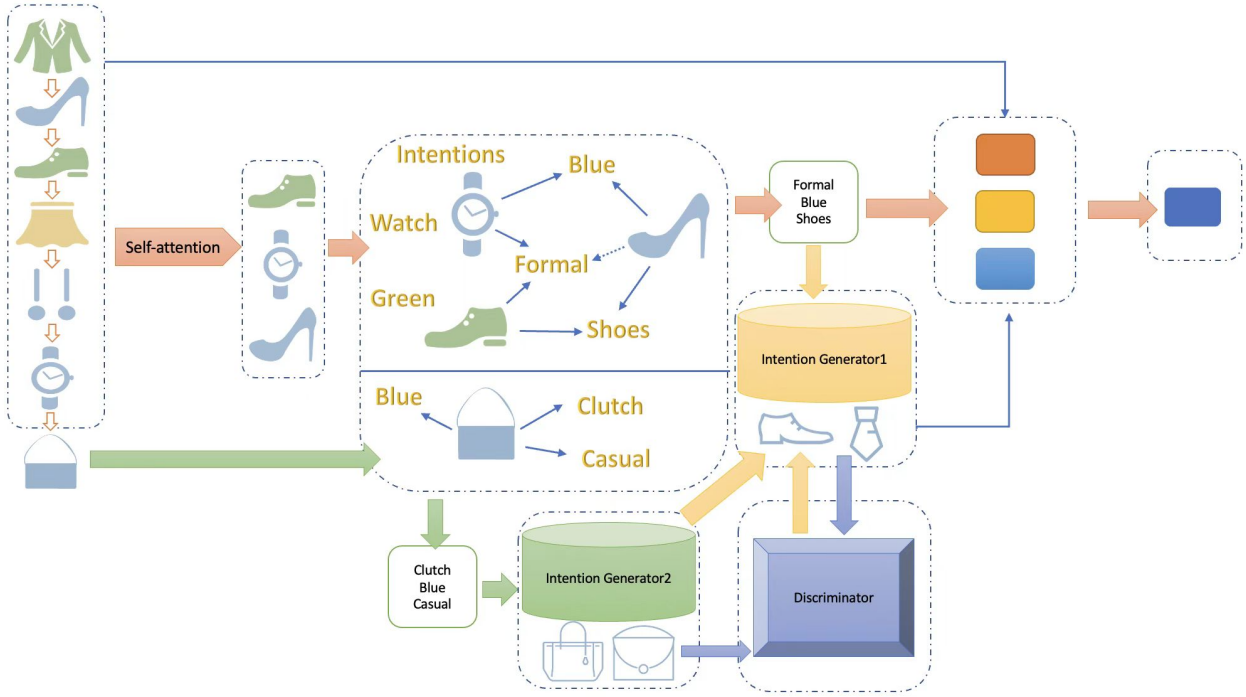


Fig. 1. The architecture of IEF-DGN.The self-attention module takes a user's behavior sequence as input and adaptively activates his/her interests from the large interest pool, outputting multiple interest embeddings. Then, the historical intention generator will produce fake items with historical items meanwhile current intention capturator will produce fake items with the concepts of next item. IEF-DGN provides the capacity to cluster concepts and deduce the user's evolving interests.

shoes to watch and from red to blue. To summarize, the main contributions of this paper are:

- We propose an interest-evolution framework that captures users' interest drift through analysing the evolution of item concepts in a recommender system.
- We investigate an intention pool which contains a number of concepts and a drift matrix which reflects the evolution between concepts.
- Our model is evaluated on four real world datasets, the Tmall dataset, the Yoochoose dataset from RecSys 2015, the RetailRocket dataset and the Diginetica dataset from CIKM Cup 2016

II. Related Work

In the past few years, there has been a rise in research dedicated to the challenge of Session-based Recommendation Systems (SRS), which is a sub-task of Recommender Systems[20,21]. This task is complex as users are usually anonymous and their preferences are not expressed directly, instead, only positive observations (such as purchases or clicks) are available to the decision makers. SRS approaches can be divided into two categories: global models which focus on identifying general user interests, and localized models which emphasize temporal user interests.

Collaborative filtering methods[22,23] can be used to identify user interests by factorizing a user-item matrix constructed from the entire historical data. Neighborhood methods rely on item similarities, which are calculated from the co-occurrences of items in sessions. Lastly, Markov chain based models make predictions based on the sequential connections between user actions.

Various approaches[24] have been proposed to capture a user's general interests. For example, Matrix Factorization leverages latent vectors to characterize general interests by factorizing a user-item matrix composed of the whole historical transaction data. Neighborhood methods attempt to make recommendations based on item similarities calculated from the co-occurrences of items in sessions.

Additionally, Markov chain[25] based models utilize sequential connections between the user actions to make predictions. To differentiate the importance of items, attention mechanisms are applied. Liu[26] applies an attention mechanism to obtain a user's general preference and recent interest using long-term and short-term memories in the current session, respectively. Pan[27] measures item importance using an importance extraction module and considers global preference and recent interest to make item prediction. Chen[28] utilizes a co-attention mechanism to consider the influence of items in a user's long-term and short-term interactions.

However, attention-based methods only focus on the relative importance of items in a session, without considering the intricate transition relationship between items in an ongoing

session.

III. Methology

In this section, we first formulate the task of session-based recommendation(see Section 3.1). Then we detail the IEF-DGN model, which consists of three main components, i.e., multi-concepts capturing module (see Section 3.2), interest evolution module (see Section 3.3), and interest generating module (see Section 3.4).

A. Notations and problem formulation

Assume $\{\mathbf{i}^{(u)}\}_{u=1}^{N}$ be the behavior dataset consists of the interactions between N users and M items. $\mathbf{i}^{(u)} = [i_1^{(u)}, i_2^{(u)}, \cdots, i_n^{(u)}]$ is the ordered sequence of items clicked by user u, where n is the number of clicks made by user u. Each element $i_t^{(u)} \in \{1, 2, \cdots, M\}$ in the sequence is the index of the item being clicked. Then we let $\mathbf{Z} = [Z_1, Z_2, \cdots, Z_n]$ to denote all concepts matrix in intention pool. D is concepts's dimension. And $\mathbf{Z_k^u} \in \mathbb{R}^{\mathbb{K}\times\mathbb{D}}$ denotes user u historical intention about k concepts.

B. conceptual embeddings aggregation

Initially, the proposed our interest-evolution framework learns a $\mathbf{d}$-dimensional real-valued embedding $\mathbf{i}^{(u)} \in \mathbb{R}^{\mathbb{N}\times\mathbb{D}}$ for user u interaction sequence. Given $\mathbf{i}^u \in \mathbb{R}^{n\times D}$, the self-attentive method is first applied to aggregate the input sequence selectively.

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

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

$$\begin{aligned} \mathbf{s}^u &= \langle \mathbf{Z}, \mathbf{y}_u \rangle, \\ \text{idx} &= \text{rank}(\mathbf{s}^u, K), \\ \mathbf{Z}_k^u &= \mathbf{Z}(\text{idx}, :), \end{aligned} \tag{2}$$

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

C. History intention generation

After aggregating the historical concepts $\mathbf{Z}_k^u$, we can utilise historical intention generator to produce the user historical intention related with each item in his/her behaviour sequence according to their distance to the concepts.

$$C_{his}^u = \sum_{j=1}^{K} \frac{\exp(\mathbf{W_3} \cdot Z_t^u)}{\sum_{t=1}^{K} \exp(\mathbf{W_3} \cdot Z_t^u)}, \tag{3}$$

where $C_{his}^u \in \mathbb{R}^{K\times D}$ denotes users' current intention, and $W_3 \in \mathbb{R}^{D\times D}$ are the parameters that can be optimized. Note that we are using soft-max method instead of the total addition here, due to distribution of concepts. This choice is motivated by the fact that soft-max method is better indicate the concepts' value differentials and improve the weight of more precise concepts.

After inferring the historical conceptual prototypes C_{his}^u , we can estimate the user intention related with each item in his/her behavior sequence according to their distance to

Till now, we have introduced the whole process of the interest-evolution network. Given a user's behavior sequence, we first activate his/her preferred concepts from the intention pool. The intention assignment is then performed to estimate the user intention related with each item in the input sequence. After that, the historical intention generator is applied to calculate users' interests in historical sequences.

D. Current intention capturator

After the interest-evolution network, we obtain multiple interest embeddings for each user. A natural follow-up question is how to leverage various interest for practical inference. An intuitive solution is to use the next predicted item as a target label to select different interest embeddings for training as in MIND[29]. Despite its simplicity, the main drawback is that there are no target labels during inference, which leads to a gap between training and testing and may result in performance degeneration.

To address this issue, Then we introduce a transfer matrix denoted as $T^u \in \mathbb{R}^{D\times D}$ to drift historical intention to current intention. The motivation here is that it is easier to predict a user's temporal preference-based next intentions instead of finding the ideal labels. Specifically, based on the intention evolution matrix T^u, we can obtain current intention distribution matrix, denoted by $\mathbf{C}_{cur}^u \in \mathbb{R}^D$, for all items in the behavior sequence. Then, the next behavior's can be reformulated from the intention perspective denoted by $\widehat{\mathbf{X}^u}$. With $\widehat{\mathbf{X}^u}$, the user's current intention $\mathbf{C}_{cur}^u$ is computed as

$$\mathbf{C}_{cur}^u = \text{LayerNorm}\left((\text{softmax}(\tanh(\widehat{\mathbf{X}^u}\mathbf{W}_3)\mathbf{W}_4))^\top \widehat{\mathbf{X}^u}\right)^\top, \tag{4}$$

where $\mathbf{C}_{cur}^u \in \mathbb{R}^{K\times D}$ is the predicted intention of user u for next item. $\mathbf{W}_3 \in \mathbb{R}^{D\times D}$ and $\mathbf{W}_4 \in \mathbb{R}^D$ are trainable parameters. Given intention transfer matrix T^u and user's historical intention $\mathbf{C}_{his}^u$, the predicted weights of current intention are calculated as

$$\widehat{C_{cur}^u} = \frac{\exp((\mathbf{C}_{his}^u)^\top (\mathbf{T_k}^u)/\tau)}{\sum_{k=1}^{D} \exp((\mathbf{C}_{his}^u)^\top (\mathbf{T_k}^u)/\tau)}. \tag{5}$$

Where $T^u = [T_1^u, T_2^u, \cdots, T_D^u]^T \in \mathbb{R}^{D\times D}$ is the transfer vector for diverse interests. τ is a temperature parameter

to tune. When τ is large ($\tau \to \infty$), C^u_{cur} approximates a uniformly distributed vector. When τ is small ($\tau \to 0^+$), C^u_{cur} approximates a one-hot vector. In experiments, we use $\tau = 0.1$ to enforce the aggregator select the most preferred intention for inference. The final user representation $\mathbf{v}^u \in \mathbb{R}^D$ is computed as

$$\mathbf{v}^u = \sum_{k=1}^{K} \widehat{C^u_{cur}} \cdot \mathbf{Z}^u_k. \tag{6}$$

E. Discrimination and model training

We follow the common practice[12] to train our model by recovering the next click $x_t^{(u)}$ based on the truncated sequence prior to the click, i.e., $[x_1^{(u)}, x_1^{(u)}, \cdots, x_{t-1}^{(u)}]$. Given a training sample (u, t) with the user embedding vector $\mathbf{v}^u$ and item embedding $\mathbf{i}_t$, we aim to minimize the following negative log-likelihood

$$\begin{aligned} \mathcal{L}_{like}(\theta) &= -\sum_u \sum_t \log P(x_t^{(u)} | x_1^{(u)}, x_2^{(u)}, \cdots, x_{t-1}^{(u)}) \\ &= -\sum_u \sum_t \log \frac{\exp(\mathbf{i}_t^\top \mathbf{v}^u)}{\sum_{j \in \{1,2,\cdots,M\}} \exp(\mathbf{i}_j^\top \mathbf{v}^u))} \end{aligned} \tag{7}$$

Equation (7) is usually intractable in practice, because the sum operation of the denominator is computationally prohibitive. Therefore, we leverage a sampled softmax technique[14] to train our model. Besides, we also introduce a transfer loss to optimize the weights in intention transfer matrix whose parameters are denoted as ϕ. The transfer loss $\mathcal{L}_t$ to be calculated as

$$\mathcal{L}_t(\phi) = (\mathbf{C}^u_{his}\mathbf{T}^u - \mathbf{C}^u_{cur})^2 \tag{8}$$

There is a variety of model structures for discriminating the fake and real current intention, we empirically find that towered multilayer perceptron (MLP). Let N be the number of hidden layers, the discriminator D is a mapping function that operates in a towered manner:

$$\mathcal{L}_d(\sigma) = f_{out}\Big(f_N\big(\cdots f_2\big(f_1(\mathbf{x})\big)\cdots\big)\Big). \tag{9}$$

$f_l(\cdot)$ with $l = 1, 2, \cdots, N$ denotes the mapping function for the l-th hidden layer. $f_l(\mathbf{x}) = \sigma(\mathbf{W}_l\mathbf{x} + \mathbf{b}_l)$, where $\mathbf{W}_l$ and $\mathbf{b}_l$ are learnable weight matrix and bias vector for layer l. The activation function σ for each layer is sigmoid. We set the size of layers (i.e., dimensionality of $\mathbf{x}$) as one third of the previous layers. The output layer $f_{out}(\cdot)$ is the cross-entropy loss function. σ denotes the whole parameters of discriminator.

Combine the three losses above, the final loss function of our model is

$$\mathcal{L} = \min_\theta \max_\sigma \min_\phi \mathcal{L}_{like}(\theta) + \lambda\mathcal{L}_t(\phi) - \beta\mathcal{L}_d(\sigma) \tag{10}$$

where λ and β is the trade-off parameter to balance the two losses.

F. Connections with Existing Models

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

Implicit approach. This type of method relies on powerful neural networks to implicitly cluster historical behaviors and extract diverse interests. For example, MIND[29] utilizes Capsule network[30] to adaptively aggregate user's behaviors into interest embedding vectors. SASRec[31] adopts the multi-head self-attention mechanism[32] to output multiple representation for a user. Compared with these methods, our model belongs to an explicit approach that explicitly detects intentions from the user's behavior sequence based on latent conceptual prototypes.

Explicit approach. Methods that belong to this type maintain a set of conceptual prototypes to explicitly determine the intentions of items in the user's behavior sequence. MCPRN[18] is a recent representative work for extracting multiple interests from the session for the next-item recommendation. DisenRec[33] utilizes latent prototypes to help learn disentangled representations for recommendation. Compared with them, we also follow the explicit approach, but our model scales to a large-scale dataset. Specifically, they require the number of diverse interest embeddings equals to the number of conceptual prototypes. However, the number of latent concepts depends on applications and can be easily scaled up to hundreds or even thousands in industrial recommender systems, which hinders their application in practice. In contrast, our intention evolution network offers the ability to infer a set of preferred intentions from the large concept pool automatically.

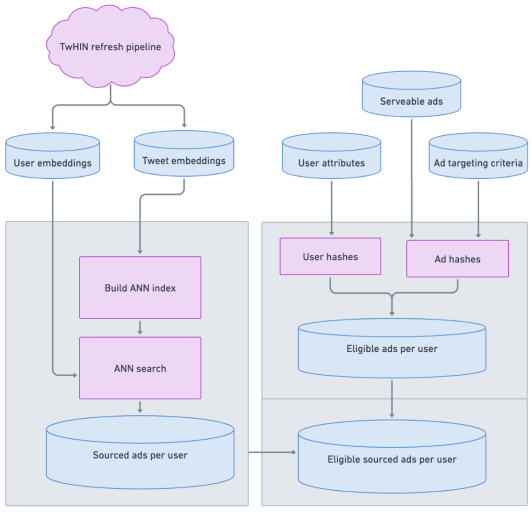


Fig. 2. Graph similarity based candidate generation pipeline.

G. Metrics

The choice of metrics is both a deeply interesting question and critical to the success of any project because they will

guide the overall progress of research and development. It is important to find offline metrics that correlate with our various online metrics and product concerns.

Recall The most obvious and easiest to define metric is recall, or hit-rate, defined as:

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{11}$$

where TP is the number of true positives, or hits, and FN is the number of false negatives. We choose to define a hit as an engagement that was correctly identified in the test set. This provides a method to characterize the ability to predict user engagements. While this is valuable, optimizing for pure recall can be sub-optimal because it neglects the role of the auction and the true positive signals can be sparse and has the potential to underweight brand ads.

Auction Recall To address these issues, we introduce the concept of an auction recall. Specifically, we consider true positives to be those ads that win the auction for their slot. Optimizing for auction recall introduces multiple possible issues, including constructing a system that learns the biases and pathological behaviours of the ranking stack that is downstream. While this can lead to short term metric gains both online and offline, it can easily lead to long-term product decay. As a result, we typically examine both auction recall and engagement recall metrics.

Rankscore NCG Rankscore normalized cumulative gain (NCG) compares a given candidate generation algorithm to a hypothetical algorithm that always selects the top ads by *rankscore*. The metric that is computed is the ratio of the sum of the rankscores of the ads selected by the algorithm and the potential rankscore sum that would be found by the hypothetical algorithm. It is defined as:

$$rNCG_m = \frac{\sum_{i=1}^{n} \sum_{j=1}^{min(m,|C_i|)} C_{i,(|C_i|-j+1)}}{\sum_{i=1}^{n} \sum_{j=1}^{min(m,|R_i|)} R_{i,(|R_i|-j+1)}}, \tag{12}$$

where R_i is the set of rankscores of all eligible ads for the ith request after targeting filters are applied, C_i is a the subset of ad candidates selected by the candidate generation algorithm from R_i, n is the number of ad requests, and m is set to match the number of ads that would typically make it to auction. Lower rankscore NCG ratios indicate a larger headroom for showing more highly ranked ads by making improvements to candidate generation.

Inequality Finally, we track the top 1 percent share (T1PS) of the advertisers to evaluate whether our changes are making the overall ads ecosystem more fair. This is important to improving the advertising experience on the Twitter platform because it improves the experience of small and medium sized businesses, which is a long standing product goal for all online advertising platforms. The T1PS coefficient is defined as:

$$T1PS = \frac{\sum_{S_i \geq S_{(0.99|S|)}} S_i}{\sum_i S_i}, \tag{13}$$

where S_i is the number of served ads among all users from advertiser i over a given period. Thus the condition, $S_i \geq S_{(0.99n)}$ restricts to advertisers in the 99th percentile or top 1 percent of advertisers with regards to served ads.

TABLE I
STATISTICS OF THE DATASETS.

Dataset	# Interactions	# Items	# Sessions	Avg. Length
Diginetica	786,582	42,862	204,532	4.12
Nowplaying	1,085,410	59,593	145,612	9.21
RetailRocket	871,637	51,428	321,032	6.40
Tmall	427,797	37,367	66,909	10.62
Yoochoose	1,434,349	19,690	470,477	4.64

Ads Value It is generally difficult to estimate the average value an advertiser realizes from running an ad campaign on the platform. While we cannot estimate that exact value because we do not have access to the per-advertiser exchange rate for engagements, we can use the average cost per conversion as a natural proxy for that value. Using this intuition, we derive a measure of proxy ads value:

$$\text{AdsValue}(j) = \sum_i \frac{R_i}{C_i} C_{ij} \tag{14}$$

where j indicates the experiment bucket, i indicates the campaign index, R_i indicates the revenue for that campaign, C_i indicates the number of conversions for that campaign, and C_{ij} is the number of conversions for the i^{th} campaign in the j^{th} experiment bucket.

An increase in ads value corresponds to improvements to our ads ecosystem on the demand side. Tracking the ads value allows us to understand if changes to our recommendation ecosystem are improving the overall long-term value that we provide to our advertisers in an a/b test setting.

IV. EXPERIMENTS

Datasets and evaluation metrics. We conduct experiments on five public datasets collected from real-world platforms: *Diginetica*, *Nowplaying*, *RetailRocket*, *Tmall* and *Yoochoose* with their statistics shown in Table I. We filter out sessions of length 1 and items appearing less than 5 times across all datasets[34], and split the sessions in each dataset into train/validation/test set in temporal order in a ratio of 8:1:1[35]. To evaluate the performance of different methods, we employ two widely-used metrics, top-20 Recall (R@20) and top-20 Mean Reciprocal Rank (M@20)[36].

Baselines. To evaluate the performance of the proposed method, we compare it with the following representative baselines: (1) Matrix factorization based methods: **FPMC**[27]; (2) RNN-based methods: **GRU4Rec**[28] and **NARM**[29]; (3) GNN-based methods: **SR-GNN**[30], **NISER+**[31], **LESSR**[32] and **SGNN-HN**[33]. (4) Transformer-based methods: **SASRec**[34], **GC-SAN**[35] and **CL4Rec**[36]. Note that we don't take methods that introduce additional collaborative filtering information[37] or other side features[38] as baselines.

TABLE II

OVERALL PERFORMANCE COMPARISON ON FIVE DATASETS. "$*$" INDICATES THE STATISTICAL SIGNIFICANCE FOR $p < 0.01$ COMPARED TO THE BEST BASELINE METHOD WITH PAIRED t-TEST. SESSIONS ARE SPLIT INTO TRAIN/VALIDATION/TEST SET IN A RATIO OF 8:1:1 FOR FAIR EVALUATION. WE INDICATE PERFORMANCES OF FPMC ON YOOCHOOSE AS "$-$" DUE TO THE OOM ISSUE.

Dataset	Metric	FPMC	GRU4Rec	NARM	SR-GNN	NISER+	LESSR	SGNN-HN	SASRec	GC-SAN	CL4Rec	IEF-DGN	Improv.
Diginetica	R@20	31.83	45.43	47.68	48.76	51.23	48.80	50.89	49.86	50.95	50.03	**52.89***	+3.24%
	M@20	8.79	14.77	15.58	16.93	18.32	16.96	17.25	17.19	17.84	17.26	**18.58***	+1.42%
Nowplaying	R@20	10.18	13.80	14.17	15.28	16.55	17.60	16.75	20.69	18.30	20.59	**21.81***	+5.41%
	M@20	4.51	5.83	6.11	6.10	7.14	7.13	6.13	**8.14**	8.13	8.21	7.35	–
RetailRocket	R@20	46.04	55.32	58.65	58.71	60.36	56.22	58.82	59.81	60.18	59.69	**61.85***	+2.47%
	M@20	21.95	33.18	34.69	36.42	37.43	37.11	35.72	36.03	36.85	35.95	**38.76***	+3.55%
Tmall	R@20	20.30	23.25	31.67	33.65	35.97	32.45	39.14	35.82	35.32	44.48*	**44.67***	+14.13%
	M@20	13.07	15.78	21.83	25.27	27.06	23.96	23.46	25.10	23.48	25.07	**31.85***	+17.70%
Yoochoose	R@20	–	60.78	61.67	61.84	62.99	62.89	62.49	63.55	63.24	63.61	**64.61***	+1.57%
	M@20	–	27.27	27.82	28.15	28.98	28.59	28.24	28.63	**29.00**	28.73	25.05	–

Implementation details The proposed models and all the baselines are implemented based on a popular open-source recommendation library RecBole and its extension RecBole-GNN for easy development and reproduction. The dimension of the latent vectors is fixed to 100, and each session is truncated within a maximum length of 50. We optimized all the compared methods using Adam optimizer[39] with a learning rate of 0.001, and adopted early-stop training if the M@20 performance on the validation set decreased for 5 consecutive epochs. We use a batch size of 2048 for all methods (except TAGNN, for which we use 100 due to the large memory consumption). Other hyper-parameters of baselines are carefully tuned following the suggestions from the original papers and we report each performance under its optimal settings. For IEF-DGN, we apply a grid search for controllable margin τ among $\{0.01, 0.05, 0.07, 0.1, 1\}$ and dropout ratio ρ among $\{0, 0.1, 0.2\}$. We finally report metrics on the test set with models that gain the highest performance on the validation set.

Overall comparison. From the experimental results in Table II, we can observe: IEF-DGN outperforms all the baselines significantly over 8 of 10 metrics on the adopted five datasets. Different from these baselines, IEF-DGN doesn't apply non-linear layers over item embeddings to encode sessions, but learns weights for each item embedding and adopts a weighted sum for encoding session embeddings and item embeddings in consistent representation space. Besides, notice that IEF-DGN considers neither item order in a session, nor item importances. However, it still outperforms all the baselines on 3 metrics, while gaining comparable results with several strong baselines on other metrics. Without carefully designed encoder architecture, IEF-DGN can achieve impressive performance, which further confirms the importance of encoding session embeddings and item embeddings in consistent representation space.

Analysis 1: Ablation study. IEF-DGN involves several components and we now analyze how each part contributes to the performance. As SASRec and IEF-DGN shares the same Transformer architecture, we select SASRec as the base model to compare. We mainly consider the following variants: **IEF-DGN w/o IEF** means replacing interest evolution framework of IEF-DGN to SASRec's encoder; **IEF-DGN w/o DGN** means replacing the delta generation network techniques to the traditional dot product distance; In Table IV, we can observe that the performance order can be summarized as IEF-DGN $>$ IEF-DGN w/o DGN $>$ IEF-DGN w/o IEF $\simeq$ SASRec. These results indicate that all the parts are useful to improve the final performance.

TABLE III

PERFORMANCE COMPARISON OF DIFFERENT METHODS AND THEIR IMPROVED VARIANTS ON TWO DATASETS.

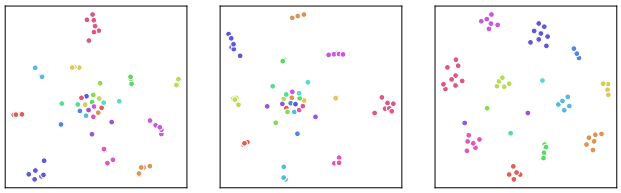
Method	Diginetica		RetailRocket	
	R@20	M@20	R@20	M@20
NARM	47.68	15.58	58.65	34.69
+ IEF	51.86	18.27	60.77	37.01
+ DGN	51.62	17.79	61.33	37.11
+ All	**52.51**	**18.58**	**62.19**	**38.84**
SR-GNN	48.76	16.93	58.71	36.42
+ IEF	49.51	17.53	57.05	35.70
+ DGN	51.36	18.57	61.41	38.27
+ All	**52.38**	**18.95**	**61.43**	**38.38**

TABLE IV

ABLATION STUDY OF IEF-DGN'S VARIANTS ON DIGINETICA AND RETAILROCKET.

Method	Diginetica		RetailRocket	
	R@20	M@20	R@20	M@20
IEF-DGN	**52.89**	**18.58**	**61.85**	**38.76**
w/o IEF	49.82	17.41	59.59	36.27
w/o DGN	52.31	18.38	60.93	37.72
SASRec	49.86	17.19	59.81	36.03

Analysis 2: Improving existing methods with IEF & DGN. Here we slightly modify several popular existing session-based models, showing how the proposed components IEF and DGN



(a) GRU4Rec (b) SASRec (c) IEF-DGN (ours)

Fig. 3. Visualization of learned session embeddings.

improve the performances of existing methods.

Take *SR-GNN[30] + IEF* as an example, the original encoder can be formulated as $\boldsymbol{h}_s = \boldsymbol{W}[\boldsymbol{h}_{s,n}; \boldsymbol{h}_g]$, where $\boldsymbol{W} \in \mathbb{R}^{d \times 2d}$ are learnable parameters and $\boldsymbol{h}_g$ can be seen as the linear combination of item embeddings within a session. Then we remove $\boldsymbol{W}$ and change the encoder to $\boldsymbol{h}_s = (\boldsymbol{h}_{s,n} + \boldsymbol{h}_g)/2$ and the modified SR-GNN encoder is a variant of the proposed IEF, where the session embedding is linear combination of item embeddings.

We can see that generally the combination of IEF and DGN can gain dramatic performance improvements compared to the original models. As for only adding one of the proposed techniques, DGN consistently improve the performance, while IEF has a positive effect in most cases.

Analysis 3: Visualization of session embeddings. To show the effectiveness of the proposed model IEF-DGN, we visualize the learned session embeddings using t-SNE[20] algorithm in Figure 3. Detailed, sessions with the same next item are viewed as in the same class, and are marked the same color. We randomly sample 15 items as ground truth next items. Then we extract all the corresponding sessions from our test set in Diginetica. We can see that different classes' session embeddings learned by IEF-DGN are well separated from each other compared to those learned by GRU4Rec and SASRec.

V. Conclusion

In this work, we described IEF-DGN– a candidate generation system that was designed as part of our ads multi-stage ranking system. Firstly, we proposed Interest-Evolution Framework with Delta Generative Network(IEF-DGN). Next, we design two intention generators for producing virtual items utilising multiple concepts. Finally, taking into account of item embeddings in interaction sequences, we propose the interest evolving matrix to capture interest evolving process. For future work, we will consider studying the expressive ability of the proposed representation-consistent encoder both theoretically and empirically. Besides, we will explore how to introduce side features and useful inductive biases to the proposed framework.

References

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling Popularity Bias in Learning-to-Rank Recommendation. In RecSys'17. 42–46.

[2] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible. Extensions. IEEE Trans. Knowl. Data Eng. 17, 6 (2005), 734–749.

[3] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2019. A Dynamic Co-attention Network for Session-based Recommendation. In CIKM'19. 1461–1470.

[4] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2019. Joint Neural Collaborative Filtering for Recommender Systems. ACM Trans. Inf. Syst. 37, 4 (2019), 39:1–39:30.

[5] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam M. Shroff. 2019. Sequence and Time Aware Neighborhood for Session-based Recommendations: STAN. In SIGIR'19. 1069–1072.

[6] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-Transformer. In NAACL'19. 1315– 1325.

[7] Priyanka Gupta, Diksha Garg, Pankaj Malhotra, Lovekesh Vig, and Gautam M.Shroff. 2019. NISER: Normalized Item and Session Representations with Graph Neural Networks. arXiv preprint arXiv: 1909.04276 (2019).

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In WWW'17. 173–182.

[9] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. 2004. Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. 22, 1 (2004), 5–53.

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

[11] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In RecSys'17. 306– 310.

[12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR'17.

[13] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, MinYen Kan, and TatSeng Chua. 2020. Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems. In WSDM'20. 304–312.

[14] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In CIKM'17. 1419–1428.

[15] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In ICLR'16.

[16] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In KDD'18. 1831–1839.

[17] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. An Intent-guided Collaborative Machine for Session-based Recommendation. In SIGIR'20. 1833–1836.

[18] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. Rethinking Item Importance in Session-based Recommendation. In SIGIR'20. 1837–1840.

[19] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the Item Order in Session-based Recommendation with Graph Neural Networks. In CIKM'19. 579–588.

[20] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In WWW'10. 811–820.

[21] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. Nature 323, 6088 (1986), 533–536.

[22] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In WWW'01. 285–295.

[23] Weiping Song, Zhiping Xiao, Yifan Wang, LaurentChar lin,Ming Zhang, and Jian Tang. 2019. Session-Based Social Recommendation via Dynamic Graph Attention Networks. In WSDM'19. 555–563.

[24] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway Networks. arXiv preprint arXiv:1505.00387 (2015).

[25] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In NeurIPS'15. 2440– 2448.

[26] Yueming Liu and Yi Zhang. 2018. Conversational Recommender System. In SIGIR'18. 235–244.

[27] Ashish Pan, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In NeurIPS'17. 5998–6008.

[28] Petar Chen, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In ICLR'18.

[29] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. 2017. NormFace: L2 Hypersphere Embedding for Face Verification. In MM'17. 1041–1049.

[30] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maartende Rijke. 2019. A Collaborative Session-based Recommendation Approach with Parallel Memory Modules. In SIGIR'19. 345–354.

[31] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for NextBasket Recommendation. In SIGIR'15. 403–412.

[32] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In SIGIR'19. 165–174.

[33] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory Networks. In ICLR'15.

[34] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. In AAAI'19. 346–353.

[35] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self- Attention Network for Session-based Recommendation. In IJCAI'19. 3940–3946.

[36] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In SIGIR'16. 729–732.

[37] Hainan Zhang, Yanyan Lan, Liang Pang, Jiafeng Guo, and Xueqi Cheng. 2019. ReCoSa: Detecting the Relevant Contexts with Self-Attention for Multi-turn Dialogue Generation. In ACL'19. 3721–3730.

[38] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. ACM Comput. Surv. 52, 1 (2019), 5:1–5:38.

[39] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential Recommender System based on Hierarchical Attention Networks. In IJCAI'18. 3926–3932.