CASE: Predict User Behaviors via Collaborative Assistant Sequence Embedding Model

Fei He[1,2], Canghong Jin[1], and Minghui Wu[1(✉)]

[1] Zhejiang University City College, Hangzhou 310015, China
{jinch,mhwu}@zucc.edu.cn
[2] Zhejiang University, Hangzhou 310027, China
fei.he@zju.edu.cn

Abstract. The order of behaviors implies that sequential patterns play an important role of the user-behavior prediction problem. Traditional behavior-prediction models use large-scale static matrices which ignore sequential information. Moreover, although Markov chains and deep learning methods consider sequential information, they still suffer the problems of the behavior uncertainty and data sparsity in real life scenarios. In this paper, we propose a collaborative-assistant sequence embedding prediction (named CASE) model as a solution to address these shortcomings. The idea is to mine sequential behavior patterns with strong intention-expressing ability based on a collaborative selector, and construct original behavior graph and intent determination graph (IDG), following which we predict user behavior based on graph embedding and recurrent neural networks. The experiments on three public datasets demonstrate that CASE outperforms many advanced methods based on a variety of common evaluation metrics.

Keywords: User behavior prediction · Collaborative selector · Graph pattern mining

1 Introduction

Massive user-behavior data are produced on the Internet, such as video viewing and rating, click and purchase of goods, posts on social networking sites and thumb-up behavior. These data contain rich behavior information, and better understanding them should help solve the problem of "information overload", thereby improving the efficiency of demand matching, and providing better services to users. Behavioral data have huge application prospects in the fields of business scenarios [2], social governance [16], etc., and are currently a topic of significant research interest.

H. Gao and X. Wang (Eds.): CollaborateCom 2021, LNICST 407, pp. 175–189, 2021.
https://doi.org/10.1007/978-3-030-92638-0_11

1.1 Top-N Behavior Prediction

This problem can be summarized as follows: Given a set of users $U = \{u_1, u_2, \ldots, u_{|U|}\}$ and a set of all possible behaviors $A = \{act_1, act_2, \ldots, act_{|A|}\}$. Each user u has a historical behavior sequence $actseq^{(u)} = \langle act_{t_1}^{(u)}, act_{t_2}^{(u)}, \ldots, act_{t_{n-1}}^{(u)}, act_{t_n}^{(u)} \rangle$, where $act_{t_i}^{(u)} \in A$. The index t_i for $act_{t_i}^{(u)}$ denotes the chronological order of each behavior in a sequence. The behavior sequence database $D = \{actseq^{(1)}, actseq^{(2)}, \ldots, actseq^{(|U|)}\}$ contains the behavior sequences of all users. Given all user-behavior sequence $actseq^{(u)}$, the goal is to return a behavior list for each user that contains the user's N most likely future behaviors.

1.2 Limitations of Previous Work

Traditional user-behavior modeling methods focus on the long-term static behavior patterns of users. For example, based on matrix decomposition, the singular value decomposition algorithm [3] considers each behavior record to be independent [1,3,7]. They lost the sequential information in the history behaviors and ignore the dynamic variation in the short-term behavior patterns of users. In fact, user behaviors are mutually influenced and context dependent, and prior behaviors affect subsequent behaviors. Therefore, user behaviors over a certain period should be studied as a sequence structure.

Significant research has focused on behavior-sequence modeling. In terms of machine learning, Liu et al. [10] found explicit association rules through statistics in the mining of sequence behavior patterns, but ignored unobservable behavior patterns. Jarboui et al. [8] used the Markov decision process to predict the behavior of MOOC users. Shi et al. [12] proposed the state-sharing sparse hidden Markov model based on the hidden Markov model to establish a separated transition probability matrix for each user. In recent years, deep learning methods based on neural networks have also been widely used in sequential-behavior modeling. For example, recurrent neural networks (RNNs) have achieved great success in treating sequential data such as speech and text and have been introduced into the study of sequential behavior. The GRU4Rec model proposed by Hidasi et al. [6] used gated recursive units (GRUs) to predict sequential behavior. Yu et al. [17] added a time-aware controller and a content-aware controller to the long short-term memory model, so that contextual information could be fully considered when the status was updated, and the attention mechanism was used to make the model adaptively combine the long- and short-term preferences of users according to specific contextual states. Although they use the sequential information of behaviors, they still ignore the problem of uncertain user behaviors. The behavior sequence of users is not strictly sequential. Behaviors can be inserted or missing from the sequence because they do not necessarily occur to provoke subsequent behaviors. A L-order Markov chain or model containing only RNNs do not explicitly model such behavior patterns because they assume that the previous steps have an influence on the immediate next step. Tang et al. [13]

proposed a Caser model, which models recent behaviors as an "image" in time and latent dimensions and learns sequential patterns using convolutional filters. Based on the horizontal and vertical convolution, Caser can capture additional behaviors patterns that can be skipped, and can solve the behavior uncertainty problem to some extent. However, its ability to learn complex uncertain behavior relationships is affected by the size of the convolution kernel.

To express more complex behavioral relationships, the graph-based neural network method is applied to learn the behavior representation. The global-context-enhanced graph neural networks model proposed by Wang et al. [14] can learn two types of granularity information from local behavior sequence graph and global behavior sequence graph. In the global sequence graph, the session-aware attention mechanism recursively integrates the information of surrounding neighbors, and, in the local graph, the behavior information within the sequence is learned by the graph neural network. However, the graph-based approach ignores the sparsity of user behavior. Since many types of behaviors exist, and the recorded behavior of each user only accounts for a small fraction thereof, the obtained behavior graph is very sparse.

To solve these problems (i.e., behavior uncertainty and sparse behavior data), we proposes a collaborative-assistant sequence embedding prediction (CASE) model. The novelty of the CASE model is that we proposed a collaborative selector to find the effective sequential behavior patterns with strong intention-expressing ability and then construct an intent determination graph (IDG). The IDG encodes new relationships and weights between behaviors from intention-expressing ability and uncertainty perspectives, which helps alleviate the problems. The model contribution is as follows:

i. CASE uses a collaborative selector based on collaborative prediction and information entropy to get the strong intention-expressing sequential behavior patterns, and then constructs the IDG that can alleviates the uncertainty and sparsity involved with predicting user behavior.
ii. CASE proposes a behavior prediction model based on graph embedding and RNNs, which allows the model to learn the relationships between behaviors in graph structure and sequence structure.
iii. CASE outperforms many advanced methods for predicting top-N sequential behavior from real life datasets.

2 Proposed Model

In this section, we introduce the CASE model, which is a method to learn user intent from a temporally ordered sequence of behaviors and predict the future behaviors. The goal of the CASE model is to reduce behavior uncertainty and sparsity problems. Figure 1 shows the network architecture of the CASE model, which can be divided into three components: constructing graphs (original behavior graph and IDG), learning behavior embedding and predicting future behavior. Each component of the model is described in detail below.

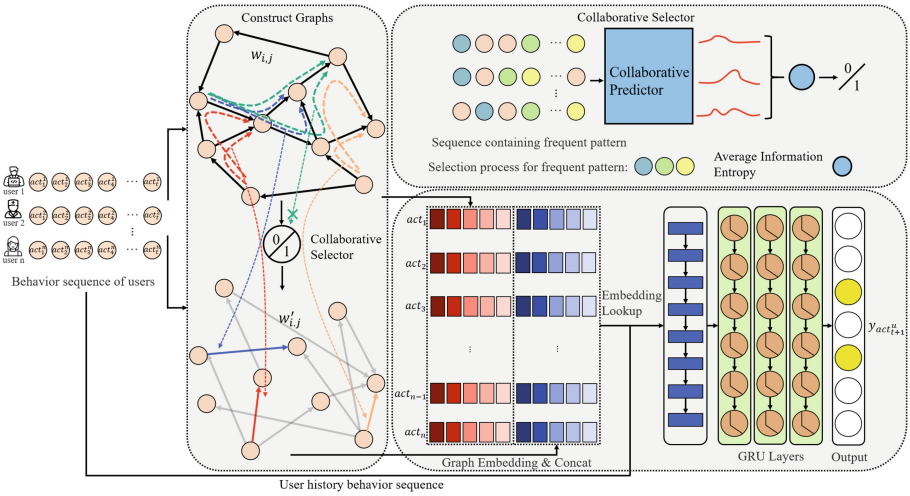

Fig. 1. Network architecture of CASE model. The box on the left contains the original behavior graph, the collaborative selector, and the IDG. The box in the upper-right corner illustrates the process of collaborative selector. The red lines in the box represent the probability distribution of the next behavior predicted by the collaborative predictor. The box in the lower-right corner represents the graph embedding and prediction of future behaviors.

2.1 Constructing Graphs

As shown in the dashed box to the left of Fig. 1, we first build the original behavior graph from the historical behavior sequences of all users. And then frequent sequential pattern mining algorithm is applied to the behavior database D to get the initial sequential behavior patterns. However, not all these sequential behavior patterns are useful because of the behavior uncertainty, and building a behavior graph from these initial patterns will introduces unreliable behavior relationships. Therefore, we propose to use a collaborative selector based on user collaborative prediction and information entropy to further filter these sequential behavior patterns, and then finally construct an intent determination graph (IDG) from the rest.

Original Behavior Graph. According to the behavior-sequence database D, we construct a user original behavior graph $G(V, E)$, where V is the set of nodes in the graph, with every node representing a type of behavior, and E is the set of edges in the graph. For each behavior sequence in D, we add an edge between node i and node j if the user performs behavior i and then behavior j. Refer to previous work [15], the weight $w_{i,j}$ of each edge $e = (i, j) \in E$ is equal to the number of occurrences of the behavior pair in the D.

Because many types of behaviors are possible and users exhibit only a small fraction thereof, the user-behavior co-occurrence matrix is characterized by high

dimensionality and sparsity (as shown in Table 1). This prevents the model from learning the relationship between behaviors. Besides, because of behavior uncertainty, the behavior sequences are also not strictly sequential. For example, some intermediate behaviors may be skipped or inserted in sequences, and completely random invalid behaviors may occur in the sequences, which introduce unreliable relationships between behaviors. To alleviate these problems, the IDG is constructed.

Collaborative Selector. The collaborative selector is based on the following idea: the initial sequential behavior patterns obtained by frequent sequential pattern mining algorithm are those that occur frequently in all user behavior sequences, but not all of them are useful. The really effective sequential behavior patterns should have a strong ability to express users' intention. And a concentrated future behavior distribution of sequences containing a sequential behavior pattern indicates that this sequential behavior pattern plays a significant role in determine the sequences' intention. Moreover, the future behavior distribution of a single user may be contingent, so we need to collaborate with all users to conduct a more accurate analysis. We use a modified PrefixSpan algorithm [4] to get the initial sequential behavior patterns on behavior sequence database D. In order to ensure the timeliness of behaviors in sequential behavior patterns, we add a maximum interval limit W between behaviors of frequent sequential patterns in PrefixSpan algorithm. That is to say, during the step of PrefixSpan algorithm which gets a longer sequential pattern $fp = \langle act_1, act_2, \cdots, act_k, act_{k+1} \rangle$ based on the prefix sequential pattern $\langle act_1, act_2, \cdots, act_k \rangle$, where the subscript k represents the order of the behavior act in the pattern, we check that whether the interval between behavior act_{k+1} and acr_k in the behavior sequence is not greater than W. If the condition is met, we take fp as a valid sequential pattern; otherwise, it is discarded. Additional details of the PrefixSpan algorithm are referred to [4].

The dashed box in the upper-right corner of Fig. 1 shows the selection process of the collaborative selector for a three-order sequential behavior pattern. Given a sequential behavior pattern, we first select all behavior sequences containing it from D to form a sub-database D_{sub}. We then regard each sequence in D_{sub} as a query q and input it into the collaborative predictor. The predictor returns the probability distribution P of the ten most likely future behaviors $actset = \{act_i\}_{i=1,\ldots,10}$, and we calculate the information entropy $entropy_q$ of this probability distribution P:

$$entropy_q = -\sum_{act \in actset} P(act) \log\big(P(act)\big). \tag{1}$$

The information entropy corresponding to each query in D_{sub} is averaged to obtain the average information entropy $avg\_entropy$,

$$avg\_entropy = \frac{1}{N} \sum_{i=1}^{N} entropy_{q_i} \tag{2}$$

where N is the number of queries. The smaller the *avg_entropy* value is, the more centralized is the distribution of future behaviors, and the stronger the ability of the sequential behavior pattern to determine intention is. If *avg_entropy* is less than or equal to the threshold λ, it is recognized as an effective behavior pattern, otherwise, it is recognized as an invalid behavior pattern:

$$y = \begin{cases} 1 & avg\_entropy \leq \lambda \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

then we get the filtered sequential behavior patterns with their corresponding *avg_entropy*.

We implement the collaborative predictor based on the classical collaborative filtering algorithm matrix factorization (MF). MF decomposes the user-behavior co-occurrence matrix $R_{M\times N}$ into a product of two matrices,

$$R_{M\times N} \approx U_{M\times K} \times V_{K\times N} \tag{4}$$

where M is the number of users, and K is the representation dimension of users and behaviors. Each column of the matrix V is a representation of a behavior, which is what we really want. For each query q, we get the representation vector $\{v_i\}_{i=1,\ldots,|q|}$ for each behavior act_i in the query q, where $|q|$ is the behavior number of q. We then calculate the average value of these representation vectors as the query representation. Next, the predictor returns the top ten behaviors $\{act_i\}_{i=1,\ldots,10}$ most similar to that query representation among all the behaviors, and the similarity is defined as cosine similarity. Then we apply an exponential normalization of these cosine similarities $\{sim_i\}_{i=1,\ldots,10}$, get the probability distribution P of the behaviors corresponding to the query q:

$$P(act_i) = \frac{e^{sim_i}}{\sum_j e^{sim_j}} \tag{5}$$

Intent Determination Graph. Based on all sequential behavior patterns that have been filtered by the collaborative selector, we build the IDG graph $G^{'}(V', E')$, where $V^{'}$ is the set of nodes, and $E^{'}$ is the set of edges. For each strong intention-expressing sequential behavior pattern, we add an edge between node i and node j if behavior i appears before behavior j. The weight $w^{'}_{i,j}$ of each edge $e = (i, j) \in E^{'}$ is

$$w^{'}_{i,j} = \frac{1}{|FP|} \sum_{fp\in FP} \frac{1}{avg\_entropy_{fp}} \tag{6}$$

where FP is the set of behavior patterns that contain behavior pair (i, j), $|FP|$ is the number of elements in the FP, $avg\_entropy_{fp}$ is the corresponding *avg_entropy* value for each behavior pattern fp in FP. The greater $w^{'}_{i,j}$ is, the stronger the relationship between the behaviors.

First of all, the nodes on the IDG are behaviors with strong intention-expressing ability filtered by collaborative selector, which reduces the interference of uncertain behaviors. Secondly, the weight of the edge designed based on information entropy further describes the degree of uncertainty between the behaviors. Finally, the IDG contains many edges that never appear on the original behavior graph (as show in Table 2), which also alleviates the problem of data sparsity to some extent.

2.2 Learn Behavior Embedding

The goal of the Learn Behavior Embedding component is to learn a low dimensional embedding representation for each behavior and map the relationship that the behavior satisfies on the graph from the original space to a low dimensional space. We learn low dimensional embedding representations of behaviors on both the original behavior graph and the frequent pattern behavior graph, and then concatenate them.

Take the original behavior graph as an example. The edge weight on graph $G(V, E)$ represents the strength of the connection between two behaviors. We first map the edge set E from the original graph space to a probability space. The weight between behavior i and j is $w_{i,j}$, and the sum of weights of graph G is

$$W = \sum_{(i,j)\in E} w_{i,j} \tag{7}$$

The probability of connection for edge (i, j) is

$$p(i,j) = \frac{w_{i,j}}{W} \tag{8}$$

Defining the target space R^d, the low dimensional embedding representations of behaviors i and j are $u_i, u_j \in R^d$. In the target space, the probability of the connection for edge (i, j) is

$$\hat{p}(i,j) = \sigma(\overrightarrow{u}_i^T, \overrightarrow{u}_j) = \frac{1}{1 + exp(-\overrightarrow{u}_i^T \cdot \overrightarrow{u}_j)} \tag{9}$$

When the edge probability distribution in the low dimensional embedding target space tends to be the same as that in the probability source space, the target space retains the behavior relationship in the source space. To learn the effective behavior embedding representation, we minimize the distance between the probability distributions $p(i, j)$ and $\hat{p}(i, j)$. In other words, we minimize the object function,

$$O = d\big(p(\cdot,\cdot), \hat{p}(\cdot,\cdot)\big), \tag{10}$$

where $d(\cdot, \cdot)$ is the distance between two probability distributions. This paper uses the KL divergence to represent the distance between probability distributions, and the final objective function is obtained after eliminating the constant term,

$$O = -\sum_{(i,j)\in E} w_{i,j} \log \hat{p}(i,j). \tag{11}$$

By finding the $\{\overrightarrow{u}_i\}_{i=1,\ldots,|V|}$ that minimizes the objective in Eq. (6), we can represent every vertex in the d-dimensional space.

2.3 Predict the Future Behavior

RNNs are devised to model variable-length sequence data. The behaviors of a user over a certain time period can be taken as a sequence and the future behaviors can be predicted by using the recurrent neural network. RNNs have an internal hidden state that encodes the historical information up to time t of the sequence in all units that compose the network. Standard RNNs update their hidden state h by using the following update function:

$$h_t = g(Wx_t + Uh_{t-1}) \tag{12}$$

where g is an activation function, x_t is the input of the unit at time t, and h_{t-1} is the hidden state of the previous time step. In this paper, we get the embedding representations of each behavior in the user's behavior sequence through embedding look-up and take them as input x_t for each time step.

In this paper, we apply GRUs, which constitute a more elaborate model of RNN units. A GRU uses the update gate and reset gate to learn when and by how much to update the hidden state of the unit. The activation of the GRU is

$$h_t = (1 - z_t)h_{t-1} + z_t\hat{h}_t \tag{13}$$

where the update gate z_t is calculated by

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{14}$$

while the candidate state $\hat{h}_t$ is calculated by

$$\hat{h}_t = \tanh\big(Wx_t + U(r_t \odot h_{t-1})\big). \tag{15}$$

Finally, the reset gate r_t is given by

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{16}$$

Given the hidden state $h_t \in R^d$ of the current time step t, the RNN outputs the probability distribution over the possible future behavior of the sequence of user u:

$$y_{act^u_{t+1}} = \sigma(Wh_t + b), \tag{17}$$

where $b \in R^n$, $W \in R^{n \times d}$ are the bias and weight matrix of the output layer, and $y_{act^u_{t+1}}$ represents the probability distribution of future possible behavior.

The task of predicting the user's future behavior can be regarded as a classification task or a ranking task. Generally, ranking-task training produces better results [6]. Therefore, we use the pairwise-based ranking loss function in this paper to minimize the loss function L by comparing the predicted probabilities

of positive samples and negative samples so that the ranking of positive samples keeps coming forward and the ranking of negative samples keeps going back:

$$L = \frac{1}{N} \sum_{j=1}^{N_s} \sigma(\hat{r}_{u,j} - \hat{r}_{u,i}) + \sigma(\hat{r}_{u,j}^2), \tag{18}$$

where N_s is the number of negative samples, $\hat{r}_{u,i}$ is the probability of positive samples, and $\hat{r}_{u,j}$ is the probability of negative samples.

3 Experiments

3.1 Datasets

We evaluate the proposed model on three real-world representative datasets which vary in domains and sparsity.

MovieLens 1M(**ML**)[1]: This is a popular benchmark dataset for evaluating behavior-prediction algorithms, which contains the user's rating behavior for the movie. In this work, we adopt a well-established versions, MovieLens 1M [5].

RecSys15-Buy[2]: This dataset contains user-purchase data obtained from the RecSys 2015 competition; it aims to predict which items a user intends to purchase. User behavior is divided into purchase behavior and click behavior, and we use the purchase behavior data part of it.

Amazon **Beauty**[3]: This is a series of product review datasets crawled from Amazon.com by McAuley et al. [11]. They split the data into separate datasets according to the top-level product categories on Amazon. In this work, we adopt the "Beauty" category. This dataset is the most sparse of the three.

According to the general practice in the literature [18,19], we converted all numeric ratings to implicit feedbacks of 1. We also removed cold-start users and items of having less than five feedbacks because dealing with cold-start recommendations is usually treated as a separate issue. We use the first 70% of actions in each user' sequence as the training set, and the remaining 30% of actions in each user' sequence serve as the test set for evaluating the model's performance. Table 1 shows the statistical information of the preprocessed dataset. Sparsity refers to the sparsity of the user-behavior matrix.

Table 1. Statistics of the datasets

Dataset	No. users	No. behavior types	Avg seq. length	No. behavior record	Sparsity
ML	6040	3377	165.47	999416	95.10%
RecSys	45520	4519	6.60	300105	99.85%
Beauty	40037	13951	6.52	261205	99.95%

[1] https://grouplens.org/datasets/movielens/1m/.
[2] https://2015.recsyschallenge.com/.
[3] http://jmcauley.ucsd.edu/data/amazon/.

3.2 Evaluation Metric

We evaluate a model by Precision@N, Recall@N, and F1@N. Given a list of top-N predicted items for a user, denoted $\hat{R}_{1:N}$, and the last 30% of behaviors in her or his sequence (denoted R for the test set), Precision@N and Recall@N are computed by

$$Prec@N = \frac{\left|R \cap \hat{R}_{1:N}\right|}{N}, \tag{19}$$

$$Recall@N = \frac{\left|R \cap \hat{R}_{1:N}\right|}{|R|}. \tag{20}$$

We report the average of these values of all users. $N \in \{5, 10\}$. The F1@N is defined by

$$F1@N = \frac{2 \times Prec@N \times Recall@N}{Prec@N + Recall@N}. \tag{21}$$

3.3 Experiment Design

In this paper, three different strategies from different perspectives are used to evaluate the effectiveness of the proposed CASE model:

1. We compare the CASE model with other six common and advanced models. The code for the methods involved is publicly available on GitHub.
   i. Random[4]: Randomly select a behavior to predict.
   ii. MostPopular(See Footnote 4): This is the simplest baseline that ranks items according to their popularity as determined by the number of interactions.
   iii. ItemKNN (See Footnote 4): Use behavior similar to the current behavior as a prediction result. The similarity between behaviors is defined as the cosine similarity between the behavior vectors in the user-behavior matrix. The num of nearest neighbors k is set to 80.
   iv. BPRMF (See Footnote 4): Uses pairwise ranking loss to optimize the matrix factorization with implicit feedback. And as like in previous work [9], we use the average latent factors of behaviors to represent the sequence. The parameters for this method are set as follows: num of factors is 10, L2 regularization is 0.0025 and learn rate is 0.05.
   v. GRU4Rec[5]: Uses recurrent neural network GRU and ranking loss function to capture the dependencies between user behavior sequences. The parameters for GRU4Rec method are set as follows: latent dimensions is 128, num of negative samples is 10, num of GRU layers is 2, learning rate is 1e-3 and L2 regularization is 1e-6.

[4] https://github.com/zenogantner/MyMediaLite .
[5] https://github.com/slientGe/Sequential_Recommendation_Tensorflow.

vi. Caser[6]: This employs CNNs both horizontally and vertically to model high-order MCs for sequential prediction. By following the previous work [9], we also ignore the user latent representations part of Caser when predicting future behaviors. The parameters for Caser method are set as follows: latent dimensions is 128, num of channels produced by horizontal convolution is 16, horizontal convolution kernel size is (i, 128), where i is from $\{1, \cdots, 5\}$, num of channels produced by vertical convolution is 4, vertical convolution kernel size is (5,1), learning rate is 1e-3, L2 regularization is 1e-6 and num of negative samples is 10.

2. We remove the constructing IDG part from the CASE model and keep the total behavior embedding dimension size, obtaining a new model and denoting it as CASE$^-$. We compare these two models to verify the effectiveness of the IDG.
3. Similar to what is done in the literature [13], we study how embedding dimension affects model performance.

The details of the parameters of the CASE model proposed in this paper are as follows: on all datasets, the maximum interval limit W and the minimum frequent pattern length of sequential frequent pattern mining are set to ten and three, respectively. The maximum frequent pattern length is set to three for ML dataset, and no limit for the RecSys and Beauty datasets. The threshold λ used to determine whether a sequential behavior pattern is selected are set to 1.4, 0.8 and 0.7 for ML, RecSys and Beauty, and the K for collaborative selector is set to 100. The total dimension of behavior embedding is 128 and the dimension of behavior embedding learned from original graph and IDG are both set to half of the total dimension.

3.4 Result and Analysis

Table 2 shows the results of the constructing graphs component. No. New edges refers to edges that appear on the IDG $G^{'}$, but not appear on the original behavior graph G. In ML, RecSys and Beauty datasets, the number of newly added edges increased by 18.60%, 7.05% and 3.06% respectively compared with edge numbers on the original behavior graph, which alleviate the problem of data sparsity to a certain extent. There are many strong intention-expressing behaviors on IDG, and the edge weight depicts the uncertain relationship. The model can encode these information in the behavior embedding representation by learning $G^{'}$.

Table 3 summarizes the best results of the six baselines, the CASE model, and the CASE$^-$ model. The best and second best performers on each column are highlighted in bold face and underline, respectively. The CASE model proposed herein performs the best on the RecSys and Beauty datasets according to all evaluation metrics. On ML dataset, CASE achieves the best performance on $Prec@5$, $Prec@10$, as well as on the overall metrics $F1@5$ and $F1@10$. For the

[6] https://github.com/graytowne/caser_pytorch.

Table 2. Results of constructed graphs

Dataset	No. nodes	No. edges of G	No. edges of $G^{'}$	No. new edges
ML	3377	375039	384953	69748
RecSys	4519	92901	18261	6547
Beauty	13951	133135	27265	4084

$F1@10$ metric, the CASE method has the best performance on all datasets, and has improved 1.32%, 6.80% and 16.91% compared with the second best method on ML, RecSys and Beauty datasets, respectively. These three datasets are increasingly sparse, which proves that the more sparse the original dataset is, the effect of the proposed method CASE is more significant. However, we can also find that the precision values of RecSys and Beauty datasets are very small, which may mean that these two datasets are difficult to predict and there is no good algorithm yet for it.

Table 3. Comparison of performance of the three datasets.

	Method	Prec@5	Recall@5	F1@5	Prec@10	Recall@10	F1@10
ML	Random	0.00993	0.00146	0.00255	0.01061	0.00313	0.00483
	MostPopular	0.11129	0.02134	0.03581	0.10109	0.03754	0.05475
	ItemKNN	0.09152	0.02785	0.04271	0.08851	0.05254	0.06594
	BPRMF	0.12543	0.03066	0.04928	0.11334	0.05379	0.07296
	GRU4Rec	0.29106	0.02958	0.05370	0.26023	0.05290	0.08793
	Caser	0.11815	**0.03201**	0.05037	0.12210	**0.06371**	0.08373
	$CASE^{-}$	0.26613	0.02705	0.04911	0.23864	0.048509	0.080629
	CASE	**0.29305**	0.02978	**0.05407**	**0.26369**	0.05360	**0.08909**
RecSys	Random	0.00011	0.00045	0.00018	0.00013	0.00098	0.00023
	MostPopular	0.00308	0.01165	0.00487	0.00298	0.02280	0.00527
	ItemKNN	0.02421	0.09515	0.03860	0.02495	0.19414	0.04422
	BPRMF	0.00323	0.01256	0.00514	0.00327	0.02499	0.00578
	GRU4Rec	0.02086	0.06909	0.03205	0.01666	0.11033	0.02895
	Caser	0.03139	0.10862	0.04871	0.02804	0.19390	0.04899
	$CASE^{-}$	0.02716	0.08995	0.04172	0.01990	0.13182	0.03458
	CASE	**0.04249**	**0.14074**	**0.06528**	**0.03011**	**0.19944**	**0.05232**
Beauty	Random	0.00006	0.00014	0.00008	0.00005	0.00030	0.00009
	MostPopular	0.00207	0.00520	0.00296	0.00182	0.00919	0.00304
	ItemKNN	0.00182	0.00422	0.00254	0.00202	0.00906	0.00330
	BPRMF	0.00205	0.00527	0.00295	0.00174	0.00875	0.00290
	GRU4Rec	0.00338	0.00837	0.00482	0.00312	0.01545	0.00519
	Caser	0.00440	0.00930	0.00597	0.00380	0.01610	0.00615
	$CASE^{-}$	0.00344	0.00852	0.00490	0.00276	0.01366	0.00459
	CASE	**0.00505**	**0.01249**	**0.00719**	**0.00432**	**0.02137**	**0.00719**

The results in Table 3 show that the sequential prediction methods such as GRU4Rec, Caser, CASE, and CASE$^-$ perform better than the method without considering sequence characteristics. The results prove the importance of sequential information for predicting behavior.

By comparing the performance of CASE and CASE$^-$ on all datasets, we find that constructing the IDG in the CASE model significantly improves the performance of the model. For example, for the RecSys dataset, the performance of the CASE$^-$ model is inferior to that of Caser, but the performance of the CASE model clearly exceeds that of the Caser model and produces the best performance according to all evaluation metrics.

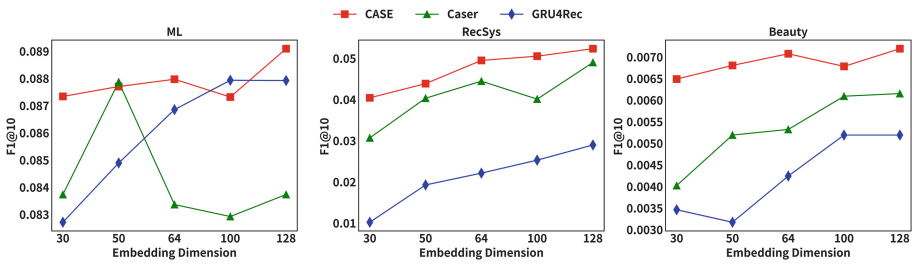

Fig. 2. F1@10 vs. the number of behavior embedding dimensions

Figure 2 shows $F1@10$ for various behavior embedding dimensions while keeping the other optimal hyperparameters unchanged. We can found that the CASE method can provide more stable performance for different behavior representation dimensions. It also works well in smaller representation dimensions, which means the almost good performance can be achieved with less computation.

4 Conclusion

To solve the problems of behavior sparsity and uncertainty in predicting user-behavior sequences, this paper proposes the CASE user-behavior prediction model.

The CASE model is based on frequent-pattern mining with a collaborative selector and graph neural networks to encode user behavior and uses recurrent neural networks to learn the dependencies between behaviors and thereby predict the future behaviors. Experiments show that the proposed CASE model outperforms many advanced methods when applied to various behavioral data sets with different vector dimensions.

In the next step, we can mine the semantic relationship between different behaviors or further study how to find causation as opposed to just correlation between behaviors, and thereby improve the robustness of the model.

Acknowledgements. Our research is supported by the Natural Science Foundation of Zhejiang Province of China under Grant (No. LY21F020003), Zhejiang Provincial Key Research and Development Program of China (NO. 2021C01164, NO.2021C02060).

References

1. Ekstrand, M.D., Riedl, J.T., Konstan, J.A.: Collaborative Filtering Recommender Systems. Now Publishers Inc, Norwell (2011)
2. Feng, X., Zeng, Y.: Joint deep modeling of rating matrix and reviews for recommendation. Chin. J. Comput. **43**(5), 884–900 (2020)
3. Gu, Y., Yang, X., Peng, M., Lin, G.: Robust weighted svd-type latent factor models for rating prediction. Expert Syst. Appl. **141**, 112885 (2020)
4. Han, J., et al.: Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the 17th International Conference on Data Engineering, pp. 215–224. Citeseer (2001)
5. Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. ACM Trans. Interact. Intell. Syst. (TIIS) **5**(4), 1–19 (2015)
6. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015)
7. Huang, L., Lin, C., He, J., Liu, H., Du, X.: Diversified mobile app recommendation combining topic model and collaborative filtering. J. Soft. **28**(3), 708–720 (2017)
8. Jarboui, F., et al.: Markov decision process for MOOC users behavioral inference. In: European MOOCs Stakeholders Summit, pp. 70–80. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19875-6_9
9. Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., Ma, J.: Neural attentive session-based recommendation. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1419–1428 (2017)
10. Liu, D.R., Lai, C.H., Lee, W.J.: A hybrid of sequential rules and collaborative filtering for product recommendation. Inf. Sci. **179**(20), 3505–3519 (2009)
11. McAuley, J., Targett, C., Shi, Q., Van Den Hengel, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 43–52 (2015)
12. Shi, H., Zhang, C., Yao, Q., Li, Y., Sun, F., Jin, D.: State-sharing sparse hidden markov models for personalized sequences. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1549–1559 (2019)
13. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 565–573 (2018)
14. Wang, Z., Wei, W., Cong, G., Li, X.L., Mao, X.L., Qiu, M.: Global context enhanced graph neural networks for session-based recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 169–178 (2020)
15. Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., Tan, T.: Session-based recommendation with graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 346–353 (2019)
16. Yanmin, C., Hao, W., Jianhui, M., Dongfang, D., Hongke, Z.: A hierarchical attention mechanism framework for internet credit evaluation. J. Comput. Res. Dev. **57**(8), 1755 (2020)

17. Yu, Z., Lian, J., Mahmoody, A., Liu, G., Xie, X.: Adaptive user modeling with long and short-term preferences for personalized recommendation. In: IJCAI, pp. 4213–4219 (2019)
18. Yuan, Q., Cong, G., Sun, A.: Graph-based point-of-interest recommendation with geographical and temporal influences. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 659–668 (2014)
19. Zhao, S., Zhao, T., Yang, H., Lyu, M.R., King, I.: Stellar: spatial-temporal latent ranking for successive point-of-interest recommendation. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)