Co-Attentive Graph Learning for Session-based Recommendation

Fei Cai
Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology
China
caifei08@nudt.edu.cn

Zhiqiang Pan [*]
Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology
China
panzhiqiang@nudt.edu.cn

***Abstract*—The goal of session-based recommendation is to predict user's action at the next timestamp by modeling his recent limited behaviors. Existing methods mainly concentrate on the sequential signal or pairwise transition relation between items using recurrent neural networks (RNN) or graph neural networks (GNN). However, the sequential methods fail to consider the transition relation between items while the GNN-based models face a serious over-smoothing problem. Thus, we propose a Co-Attentive Graph Learning (CAGL) method for session-based recommendation, which introduces a co-attention network to prevent over-smoothing, thus learning accurate representations for items. Extensive experiments are conducted on two public datasets, i.e., Diginetica and Gowalla. The results show that CAGL can beat the state-of-the-art baselines in terms of Recall and MRR.**

Keywords-recommender systems; session-based recommendation; graph neural networks; co-attention network

I. INTRODUCTION

With the information rapidly increasing on the Internet, users face a serious information overload problem when acquiring personalized needs [1]. As an effective method to filter information for users, recommender systems can detect the user intent from user's historical interactions, thus to generate recommendations for maintaining user's information needs [2]. However, in some realistic scenarios, such as the user is newly resisted or not logined in, the long-term interactions are unavailable [3], making it difficult to capture user's inherent preference. Thus, session-based recommendation (SBR) is proposed, which aims to predict user's action in the next timestamp by modeling the user preference from limited interactions in a recent short period, such as in 24 hours.

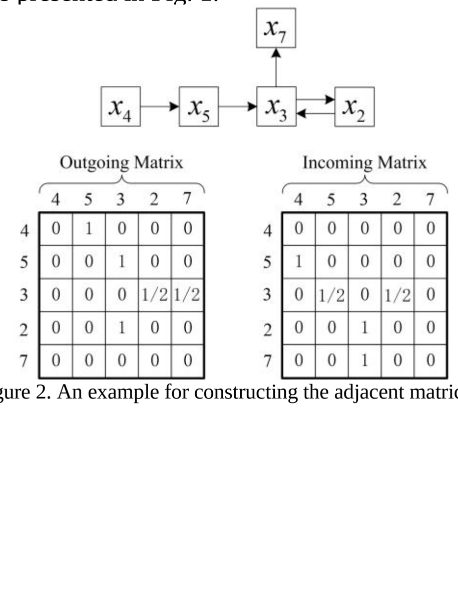
Existing methods mainly model the sequential signal or the pairwise transition relation in the item sequence via RNNs [3, 4] or GNNs [5-7]. For instance, Hidasi et al. propose GRU4REC, which adopts the gated recurrent unit (GRU) to capture the sequential dependencies between items [3]. Moreover, Wu et al. first apply the gated graph neural networks (GGNN) [8] to model the item transitions by transforming the item sequence into a session graph [5]. In addition, attention mechanism is adopted independently [9] or together with other methods like RNNs [10] and GNNs [7, 11] to capture user's main intent by assigning different importance to items.

Though the above-mentioned methods have achieved remarkable performance, there still remains several problems: (1) The sequential methods merely consider the chronological order of items, failing to consider the complicated transition relation between items; (2) The GNN-based methods achieve competitive performance on the SBR task, however the GNNs easily lead to a serious over-smoothing problem [5], making the learned representations of items hard to distinguish.

Thus, to handle the above problem, we propose a Co-Attentive Graph Learning (CAGL) method for SBR to alleviate the over-smoothing problem in GNNs. Specifically, the item sequence is first converted into a session graph represented by an incoming and an outgoing matrices. Then, we perform information propagation on the constructed graph to learn accurate representations of items in the session. After that, we introduce a co-attention network to dynamically combine the input (i.e., item embeddings) and the output of GNNs for generating the user preference. Finally, the cross-entropy is adopted as the training loss for model optimization. In addition, extensive experiments are conducted on the Diginetica and Gowalla datasets, and the experimental results validate the effectiveness of our proposed CAGL.

II. APPROACH

Given a session $S = \{v_1, v_2, ..., v_n\}$ consisting of user's n historical interactions, SBR aims to predict the next item that user may interact with, i.e., v_{n+1}. The overall framework of our proposed Co-Attentive Graph Learning (CAGL) method is presented in Fig. 1.



$$\begin{array}{c|ccccc} & 4 & 5 & 3 & 2 & 7 \\ \hline 4 & 0 & 1 & 0 & 0 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1/2 & 1/2 \\ 2 & 0 & 0 & 1 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 \end{array} \qquad \begin{array}{c|ccccc} & 4 & 5 & 3 & 2 & 7 \\ \hline 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1/2 & 0 & 1/2 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 \\ 7 & 0 & 0 & 1 & 0 & 0 \end{array}$$

Figure 2. An example for constructing the adjacent matrices.

*Corresponding author.

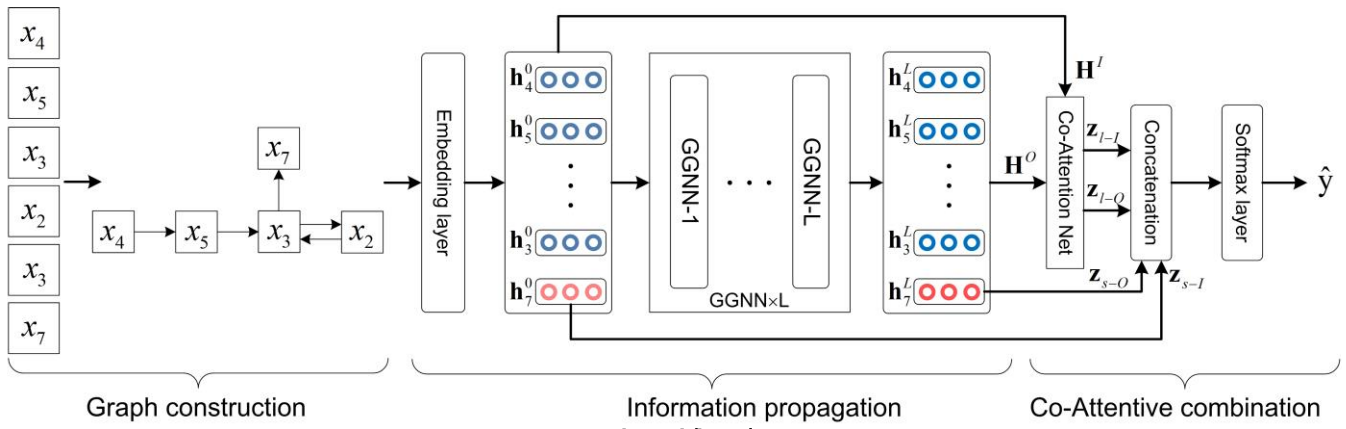


Figure 1. The workflow of CAGL.

A. *Graph construction*

Given a session S, we first convert it into a session graph where the transition patterns between items can be reflected. In specific, we construct graph $G_s = \{\mathcal{V}, \mathcal{E}\}$ for session $S = \{v_1, v_2, ..., v_n\}$, where $\mathcal{V} = \{x_1, x_2, ..., x_m\}$ denotes all the unique items in S. Note that $m \le n$ since user may repeatedly interact with an item in the session. And $\mathcal{E}$ is the edges in G_s, where each edge $(x_i, x_j) \in \mathcal{E}$ denotes that user clicks item x_j after clicking x_i. Then, we adopt the adjacent matrices $\mathbf{A}_s = \{\mathbf{A}_s^{out}, \mathbf{A}_s^{in}\}$ to represent the connection relation between items in session S, including an outgoing matrix $\mathbf{A}_s^{out}$ and an incoming matrix $\mathbf{A}_s^{in}$, which represent the outgoing and incoming connections in the session graph, respectively. For example, giving a session $S = \{v_4, v_5, v_3, v_2, v_3, v_7\}$, we can construct the outgoing and incoming matrices as in Fig. 2.

B. *Information propagation*

After constructing the adjacent matrices, for each node x_i in the l-th GNN layer, we first obtain the neighbor information from the neighbor nodes using the outgoing and incoming matrices as follows:

$$\begin{aligned}\mathbf{a}_i^l = Concat(&\mathbf{A}_i^I[\mathbf{x}_1^{l-1}, \mathbf{x}_2^{l-1}, ..., \mathbf{x}_1^{l-1}]^{\mathrm{T}}\mathbf{W}^I + \mathbf{b}^I), \\ &\mathbf{A}_i^O[\mathbf{x}_1^{l-1}, \mathbf{x}_2^{l-1}, ..., \mathbf{x}_1^{l-1}]^{\mathrm{T}}\mathbf{W}^O + \mathbf{b}^O)\end{aligned} \tag{1}$$

where $\mathbf{x}_i^0 \in R^d$ is the item embeddings of x_i generated by the embedding layer, $\mathbf{A}_i^O, \mathbf{A}_i^I \in R^{1\times m}$ are the outgoing and incoming weights corresponding to node x_i, i.e., the i-th row of the outgoing and incoming matrices, respectively. $\mathbf{W}^O, \mathbf{W}^I \in R^{d\times d}$ are the trainable parameters for outgoing and incoming edges, and $\mathbf{b}^O, \mathbf{b}^I \in R^d$ are the bias vectors. Thus, we can obtain $\mathbf{a}_i^l \in R^{1\times 2d}$ to represent the neighbor information propagated to node x_i.

After obtaining the neighbor information $\mathbf{a}_i^l \in R^{1\times 2d}$, we utilize GGNN to conduct the information propagation. Specifically, given the hidden state in the last GNN layer $\mathbf{h}_i^{l-1}$ of node x_i and the neighbor information $\mathbf{a}_i^l \in R^{1\times 2d}$, we update the representation of x_i as follows:

$$\begin{aligned}\mathbf{z}_i^l &= \sigma(\mathbf{W}_z\mathbf{a}_i^l + \mathbf{U}_z\mathbf{h}_i^{l-1}) \\ \mathbf{r}_i^l &= \sigma(\mathbf{W}_r\mathbf{a}_i^l + \mathbf{U}_r\mathbf{h}_i^{l-1}) \\ \tilde{\mathbf{h}}_i^l &= \tanh(\mathbf{W}_h\mathbf{a}_i^l + \mathbf{U}_h(\mathbf{r}_i^l \odot \mathbf{h}_i^{l-1})) \\ \mathbf{h}_i^l &= (1-\mathbf{z}_i^l)\odot\mathbf{h}_i^{l-1} + \mathbf{z}_i^l \odot \tilde{\mathbf{h}}_l\end{aligned} \tag{2}$$

where $\mathbf{W}_z, \mathbf{W}_h \in R^{d\times 2d}$ and $\mathbf{V}_z, \mathbf{V}_h, \mathbf{U}_z, \mathbf{U}_h \in R^{d\times d}$ are the learnable parameters, σ denotes the sigmoid function, and $\odot$ is the element-wise multiplication. $\mathbf{z}_i^l$ and $\mathbf{r}_i^l$ are the update gate and reset gate, which control how much information in the former state $\mathbf{h}_i^{l-1}$ should be preserved and written into the candidate activation state $\tilde{\mathbf{h}}_i^l$, respectively. In this way, the neighbor information can be propagated to learn accurate representation of x_i.

C. *Co-attentive combination*

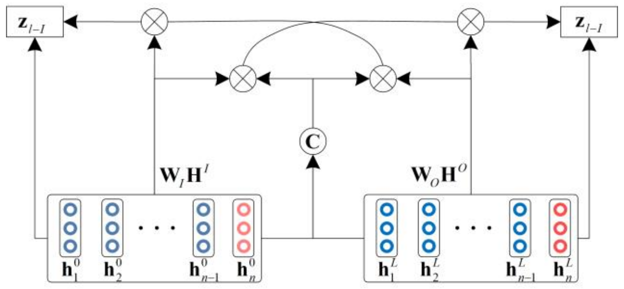
In order to exploit high-order transitions between items, multi-layer GGNNs can be stacked. After adopting L-layer GGNNs, we can obtain the representations of sequential items in session S as $\{\mathbf{h}_1^L, \mathbf{h}_2^L, ..., \mathbf{h}_n^L\}$. However, deep GGNNs may lead to a serious over-smoothing problem, which results in that the representations of items in the session learned by GGNNs are highly similar, losing their original features. Thus, we propose to utilize the co-attention mechanism for dynamically determine the item importance in the initial item embeddings (i.e., the input of GGNNs) and the output of GGNNs (i.e., the item representations learned by GGNNs) for generating the user preference.

Specifically, as shown in Fig. 3, given the input $\mathbf{H}^I = \{\mathbf{h}_1^0, \mathbf{h}_2^0, ..., \mathbf{h}_n^0\}$ and output $\mathbf{H}^O = \{\mathbf{h}_1^L, \mathbf{h}_2^L, ..., \mathbf{h}_n^L\}$ of GGNNs, we first calculate the affinity matrix $\mathbf{C}$ between the input and output as follows:

$$\mathbf{C} = \tanh(\mathbf{H}^{I\,\mathrm{T}}\mathbf{W}_c\mathbf{H}^O) \tag{3}$$

where $\mathbf{W}_c \in R^{d\times d}$ is the learnable weights. After calculating the affinity matrix, we adopt it to transform the input representation space into the output representation space:

$$\tilde{\mathbf{H}}^I = \tanh(\mathbf{W}^I\mathbf{H}^I + (\mathbf{W}^O\mathbf{H}^O)C^{\mathrm{T}}) \tag{4}$$



$\mathbf{z}_{l-I}$
$\mathbf{z}_{l-I}$
C
$\mathbf{W}_I\mathbf{H}^I$
$\mathbf{W}_O\mathbf{H}^O$
$\mathbf{h}_1^0$ $\mathbf{h}_2^0$ $\mathbf{h}_{n-1}^0$ $\mathbf{h}_n^0$
$\mathbf{h}_1^L$ $\mathbf{h}_2^L$ $\mathbf{h}_{n-1}^L$ $\mathbf{h}_n^L$

Figure 3. Structure of the co-attention network.

$$\alpha^I = \mathrm{softmax}(\mathbf{w}_I^T \tilde{\mathbf{H}}^I) \quad (5)$$

and vice versa:

$$\tilde{\mathbf{H}}^O = \tanh(\mathbf{W}^O\mathbf{H}^O + (\mathbf{W}^I\mathbf{H}^I)\mathbf{C}) \quad (6)$$

$$\alpha^O = \mathrm{softmax}(\mathbf{w}_O^T \tilde{\mathbf{H}}^O) \quad (7)$$

Then, the item representations in the input and output are combined using the above calculated attention scores:

$$\mathbf{z}_{l-I} = \sum\nolimits_{i=1}^{n} \alpha_i^I \mathbf{h}_i^I \quad (8)$$

$$\mathbf{z}_{l-O} = \sum\nolimits_{i=1}^{n} \alpha_i^O \mathbf{h}_i^O \quad (9)$$

where $\mathbf{z}_{l-I}$ and $\mathbf{z}_{l-O}$ are the long-term preference reflected in the input and output, respectively. Moreover, considering that the last interacted item reflects user's instant interest, we directly adopt the representation of the last item in the input and output as the corresponding short-term preferences, i.e., $\mathbf{z}_{s-I} = \mathbf{h}_n^I$ and $\mathbf{z}_{s-O} = \mathbf{h}_n^O$. Then, we concatenate the long- and short-term preferences in the input and output as the final user preference:

$$\mathbf{z} = \mathbf{W}[\mathbf{z}_{l-I}, \mathbf{z}_{l-O}, \mathbf{z}_{s-I}, \mathbf{z}_{s-O}] \quad (10)$$

where $\mathbf{W} \in R^{d\times 4d}$ is the trainable matrix converting the representation dimension R^{4d} from to R^d .

After obtaining the user preference, we calculate the prediction score on each candidate item:

$$\tilde{\mathbf{y}} = \mathbf{z}^{\mathrm{T}}\mathbf{v}_i \quad (11)$$

where $\mathbf{v}_i$ is the embeddings of item x_i . Then, the scores are normalized by a softmax function:

$$\hat{\mathbf{y}} = \mathrm{softmax}(\tilde{\mathbf{y}}) \quad (12)$$

Finally, we adopt the cross-entropy loss as the optimization objective to train the model:

$$L(\hat{\mathbf{y}}) = -\sum_{i=1}^{|V|} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1-\mathbf{y}_i)\log(1-\hat{\mathbf{y}}_i) \quad (13)$$

where $\mathbf{y}$ and $\hat{\mathbf{y}}$ are the true and predicted preference distributions, respectively. Finally, the Back-Propagation Through Time (BPTT) [12] algorithm is applied to train our CAGL.

III. Experiments

A. Research questions

To demonstrate the effectiveness of our proposal, we address the following three research questions: (1) Can our proposed CAGL beat the competitive baselines on SBR? (2) How does each component in CAGL contribute to the recommendation performance? (3) How is the performance of CAGL on sessions of different lengths?

TABLE I. Data Statistic

Statistics	Diginetica	Gowalla
# clicks	981,620	1,122,788
# training sessions	716,835	675,561
# test sessions	60,194	155,332
# items	42,596	29,510
Average session length	4.80	3.85

B. Datasets

We adopt two public datasets to evaluate the performance of CAGL, including an e-commerce dataset Diginetica and a check-in dataset Gowalla. Following [7], sessions of length 1 and items appearing less than 5 times are removed. Moreover, sessions of the last week in Diginetica and the last 20% sessions in Gowalla are separated as their corresponding test sets. Detailed statistics are presented in Table 1.

C. Model summary

We include nine baselines for comparison to prove the effectiveness of CAGL, including: (1) Two traditional methods, i.e., Item-KNN [13] and FPMC [14]; (2) A CNN-based method NextItNet [2] and an RNN-based model NARM [10]; (3) Five GNN-based methods, i.e., FGNN [6], SR-GNN [5], GC-SAN [11], LESSR [7] and GCE-GNN [15].

D. Experimental setup

We tune the hyper-parameters on the validation set which is the last 20% part of the training set. Specifically, the batch size and embedding dimension are both set to 100, the Adam optimizer is adopted for training the model with an initial learning rate of $1e^{-3}$ which decays by 0.1 every 2 epochs. Moreover, following [5], the GNN layer number is set to 1 to avoid overfitting. Moreover, we adopt Recall@K and MRR@K as the evaluation metrics.

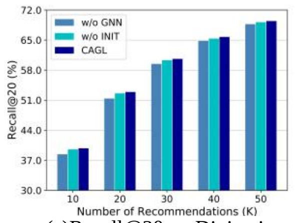
IV. Results and Discussion

TABLE II. Model Performance

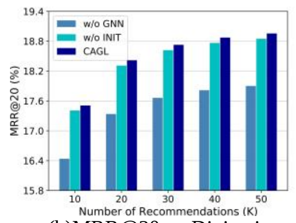
Method	Diginetica		Gowalla	
	Recall@20	*MRR@20*	*Recall@20*	*MRR@20*
Item-KNN	39.51	11.22	38.60	16.66
FPMC	28.50	7.67	29.91	11.45
NextItNet	45.41	15.19	45.15	21.26
NARM	49.80	16.57	50.07	23.92
FGNN	50.03	17.01	50.06	24.12
SR-GNN	50.81	17.31	50.32	24.25
GC-SAN	50.90	17.63	50.68	24.67
LESSR	51.71	18.15	51.34	25.49
GCE-GNN	51.66	17.53	51.53	23.52
CAGL	**52.91**	**18.42**	**52.26**	**26.21**

A. Overall performance

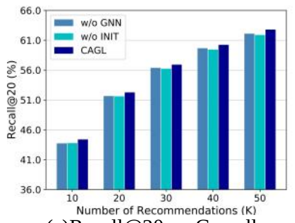
To answer RQ1, we evaluate the performance of CAGL and the baselines in terms of Recall@20 and MRR@20. Table 2 present the results. First, from Table 2, we can see that neural models all outperform the traditional ones including Item-KNN and FPMC. Moreover, among the neural models, the GNN-based methods achieve the state-of-the-art performance, generally showing better performance than the CNN-based NextItNet and the RNN-based NARM. This indicates the effectiveness of GNNs for modeling the transition relation between items in the session. In addition, LESSR performs best in most cases on two datasets, except underperforming GCE-GNN on the Recall@2 metric on Gowalla. This is due to that compared to other GNN-based methods, LESSR effectively solves the lossy session encoding problem and GCE-GNN incorporates the session-level item transitions for item representation learning. Thus, in the following experiments, LESSR and GCE-GNN are adopted as the baselines. Moreover, for the proposed CAGL, we can observe that CAGL can achieve the best performance in all cases, validating the effectiveness of CAGL on generating accurate recommendations on the SBR task.



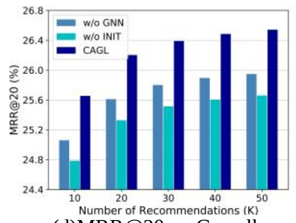
(a)Recall@20 on Diginetica



(b)MRR@20 on Diginetica



(c)Recall@20 on Gowalla



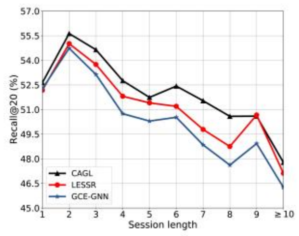
(d)MRR@20 on Gowalla

Figure 3. Ablation study.

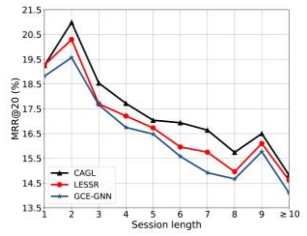
B. *Ablation study*

To answer RQ2, we compare CAGL with its two variants to see the contribution of each component in CAGL, where the variants include: (1) w/o GNN, which abandons the GNN component and generates the user preference by combining the long- and short-term preferences in the input of GNNs; (2) w/o INIT, which merely adopts the long- and short-term preferences in the output of GNNs as the user preference. The results are presented in Fig. 4, where we range the recommendation number from 10 to 50 to provide a comprehensive comparison.

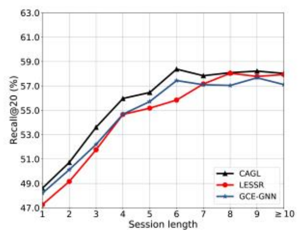
From Fig. 4, we can see that CAGL can outperform two variants in terms of both metrics on two datasets, validating the effectiveness of our designed co-attentive user preference generation, which can dynamically combine the input and output of GNNs for obtaining the user preference. Moreover, it can be observed that, w/o GNN underperforms w/o INIT in terms of Recall@20 and MRR@20 on Diginetica while outperforms w/o INIT in terms of both metrics on Gowalla. This may be due to that the over-smoothing problem caused by GNNs is more serious on Gowalla, thus resulting in a performance decrease after adopting GNNs for learning item representations. However, CAGL can effectively generate the user preference by dynamically combine the item representations in the input and output of GNNs for generating accurate recommendations.



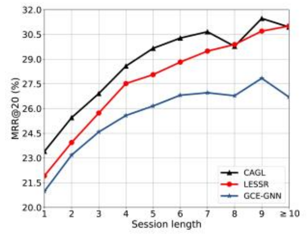
(a)Recall@20 on Diginetica



(b)MRR@20 on Diginetica



(c)Recall@20 on Gowalla



(d)MRR@20 on Gowalla

Figure 4. Performance on different lengths.

C. *Impact of the session length*

To answer RQ3, we examine the performance of CAGL and the baselines including LESSR and GCE-GNN on sessions of different lengths. The results are plotted in Fig. 5.

From Fig. 5, we can see that CAGL achieves the best performance in most cases on two datasets. Moreover, from Fig. 5 (a) and 5 (b), we can see that with the increasing of session length, the performance of three models first increases, and then shows a generally decreasing trend. This could be due to that the non-relevant items interacted by the user in the e-commerce scenario may disturb the detection of user's main intent. Differently on the Gowalla dataset, when the session length increases, the performance of all models is continuously increasing. This could be due to the fact that in the check-in scenario, users intend to visit similar places, thus relatively more interactions can help to generate the user preference accurately. In addition, we can see that CAGL shows the largest performance improvement from length 4 to 7 in terms of both metrics on two datasets. This may be due to that short sessions contain relatively few interactions to reflect the user intent while long sessions may include many non-relevant items, which both increase the difficulty for CAGL of generating right recommendations.

V. CONCLUSION

In this paper, we propose a Co-Attentive Graph Learning (CAGL) method for session-based recommendation, aiming to alleviate the serious over-smoothing problem existing in GNNs. Specifically, we introduce a co-attention network to dynamically combine the input and output of GNNs, thus to generate accurate item representations. Comprehensive experiments conducted on two public datasets validate the effectiveness of our proposal. As for future work, we would like to prevent overfitting for achieving more effective model optimization in SBR.

Acknowledgment

This work was partially supported by the Postgraduate Scientific Research Innovation Project of Hunan Province under No. CX20200055.

References

[1] 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.

[2] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'19). ACM, 582–590.

[3] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In Proceedings of the International Conference on Learning Representations (ICLR'16).

[4] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A collaborative session-based recommendation approach with parallel memory modules. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19). ACM, 345–354.

[5] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'19). AAAI Press, 346–353.

[6] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the item order in session-based recommendation with graph neural networks. In Proceedings of the International Conference on Information and Knowledge Management (CIKM'19). ACM, 579–588.

[7] Tianwen Chen and Raymond Chi-Wing Wong. 2020. Handling information loss of graph neural networks for session-based recommendation. In Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'20). ACM, 1172–1180.

[8] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated graph sequence neural networks. In Proceedings of the International Conference on Learning Representations (ICLR'16).

[9] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-term attention/memory priority model for session-based recommendation. In Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'18). ACM, 1831–1839.

[10] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In Proceedings of the International Conference on Information and Knowledge Management (CIKM'17). ACM, 1419–1428.

[11] 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 Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'19). 3940–3946.

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

[13] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. 2010. The YouTube video recommendation system. In Proceedings of the ACM Conference on Recommender Systems (RecSys'10). ACM, 293–296.

[14] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In Proceedings of the International World Wide Web Conference (WWW'10). ACM, 811–820.

[15] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xianling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20). ACM, 169–178.