Report for “Dynamic Graph Neural Networks for Sequential Recommendation”

John (Ziyuan) Zhou

University of Washington Tacoma, Tacoma, Washington, United States, zzhou98@uw.edu

**1. Introduction**

In the article “Dynamic Graph Neural Networks for Sequential Recommendation”, the authors point out one of the core problems of sequential recommendation, which is modeling user preference from his historical sequences. More precisely, most of existing methods only model users’ interests within their own sequences and ignore the dynamic collaborative signals among different user sequences, such as Markov-chain, RNN-based models, CNN-based models. Thus, they cannot efficiently explore users’ preferences. To deal this problem, there are two importance need to be solved:

* How to dynamically represent user-item interactions with a graph: Most existing methods represent user-item interactions as a static bipartite graph, and this results in the failure of recording the interaction order of the user-item pair.
* How to explicitly encode the dynamic collaborative signal for each user sequence: For each user sequence, its dynamic associated items and users form a graph structure, which contains more time/order information than conventional static graph.

Therefore, the authors propose a novel method named Dynamic Graph Neural Network for Sequential Recommendation (DGSR) as the solution. This method will explore interactive behaviors between users and items through dynamic graph.

**2. Related Work**

**2.1 Sequential Recommendation**

In the setting of sequential recommendation, we have the set of users and items, which represented by and. (Figure.1) For each user , its action sequence is denoted as where . The corresponding timestamp sequence of is .

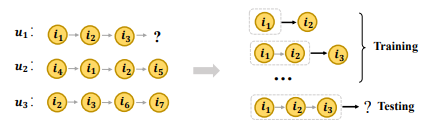
****

Figure.1 Illustration of user-item sequential interaction.

**2.2 Dynamic Graph**

Generally, there are two types of dynamic graphs, which are discrete-time dynamic graphs and continuous-time graphs. In this paper, the authors adopt continuous-time dynamic graph in their work. The dynamic network can be defined as . (Figure.2) is node set representing as . is the edge set, and represents the interacti9on between and at time . Therefore, the edge between and is usually represented by .

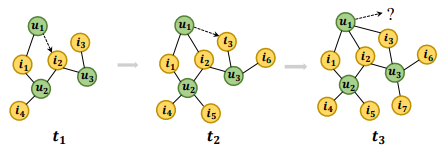
****

Figure.2 Illustration of user-item graph

**3. Methodology**

For the methodology of DGSR, there are four components as follows (Figure.3): dynamic graph construction, sub-graph sampling, dynamic graph recommendation networks, and prediction layer.

Graphical user interface, application

Description automatically generated

Figure.3: Overview of DGSR framework

**3.1 Dynamic Graph Construction**

In this section, the authors describe how to convert all user sequences into a dynamic graph annotated with time and order information on edges. Unlike the general edge representation, when user  acts on the item  at time , the edge  is established between  and  is represented by the tuple . is the timestamp when interaction occurred.  is the order of - interaction – the position of item  in all items that  has interacted with.  vice versa.

**3.2 Sub-graph Sampling**

The goal of this section is to devise a sub-graph sampling strategy to dynamically extract sub-graphs containing user’s sequence and associated sequences. Therefore, reduce the computational cost. The entire algorithm is shown in Figure.4, and it can be devised in three steps:

* Step 1 (Line 5,6,8): Take user node as the anchor node and select its most recent first-order neighbors from graph. is the historical item that has interacted with and its max length is .
* Step 2 (Line 11,12,14): For each item in , each (written as ) will be used as an anchor node to sample the set of users who have interacted with them.
* Step 3 (Line 7,13): Record user and item nodes that have been used to be anchor node to avoid repeated sampling.

Text, letter

Description automatically generated

Figure.4 Sub-graph Sampling

The output of this section is , which contains the nodes of the sequence and its associated sequences. User and items nodes in these sequences are linked to each other, through stacking and relationships in

**3.3 Dynamic Graph Recommendation Networks**

The purpose of this section is to encode user’s preference from the sub-graph. It consists of message propagation and node updating components and takes the  and  from previous step as inputs. Each input contains two types of information which are long-term and short-term. For item to user, long-term reflects user’s inherent characteristics and general preference induced from user’s all historical items and short-term reflects user’s latest interest. For user to item, long-term reflects general characters of the item and short-term reflects newest property of item. The formulas to calculate all the information are the same as follow:

* Long-term:

1. Graph neural networks: intuitive approach that aggregate all neighbor node embedding directly. and ∈ are encoding matrix parameters of item and user in -th layer. and are the numbers of ’s and’s neighbor nodes.

Text, whiteboard

Description automatically generated

1. Recurrent neural networks: effective network to model the sequence dependencies. Therefore, GRU net is utilized. and are into GRU in chronological order.

A picture containing text

Description automatically generated

1. Dynamic Graph Attention Mechanism: defined by combining graph attention mechanism and the encoding of sequence information. First, we have a relative order-aware attention mechanism to differentiate the importance weight of items to the user. is the relative order of item to the last item in the neighbors of the user node, and its calculation is . For each value , there is a unique parameter vector as the relative order embedding to encode the order information. is the dimension of the embeddings. Second, we calculate the weighting score () between user and its neighbors by SoftMax function.

Diagram, schematic

Description automatically generated



1. Long-term Preference of User:



1. Long-term Preference of Item:



Text, whiteboard

Description automatically generated

* Short-term: Attention mechanism is used to model the explicit effectiveness between last interaction with historical interactions.

Text

Description automatically generated

Text, letter

Description automatically generated

Message propagation will encode long-term and short-term information, and node updating will aggregate the long-term embedding, short-term embedding, and the previous layer embedding to update the node’s representation.

* User node updating:



* Item node updating:



* 1. **Prediction Layer**

In the last section, it will determine the items that user may interact with next, by using the outputs from previous step which is the multiple embedding of node. The steps are the same as follow:

* Concatenate user multiple embeddings to get the final embedding for node .



* Define the link function for each candidate item ∈ . is the trainable transformation matrix.



* Generate the score vector of for each candidate item, where .
* Calculate the normalized vector of user ’s score for candidate item.



The objective function is as follow, where denotes the one-hot encoding vector of the ground truth items for next interaction of , denotes all model parameters, is norm, and λ is to control regularization strength.

Logo, company name

Description automatically generated

**4.Experiment**

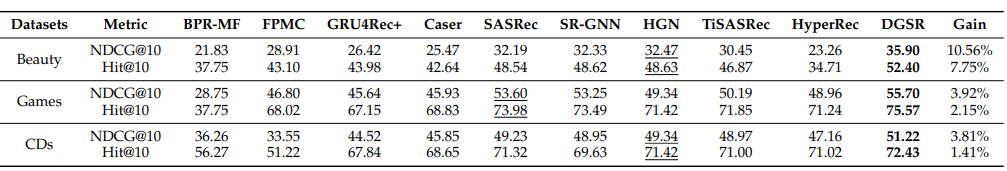
**4.1 Experiment Setting**

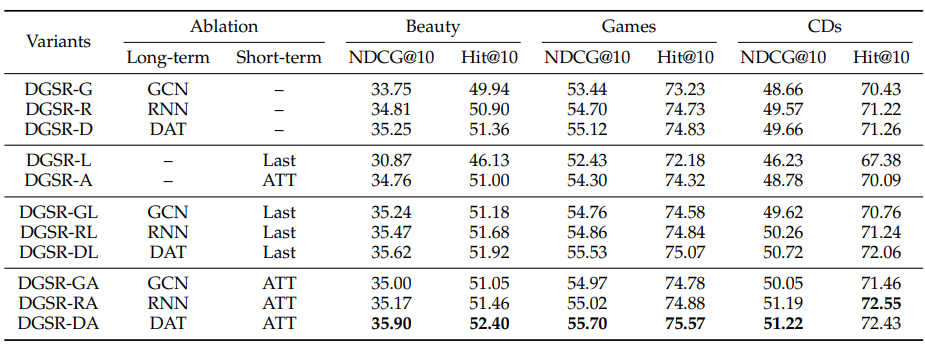
* Datasets: Amazon-CDs, Amazon-Games, Amazon-Beauty
* Baselines: BPR-MF, FPMC, GRU4Rec+, Caser, SASRec, SR-GNN, HGN, TiSARec, HyperRec.
* Evaluation Metrics: Hit@10, NDCG@K

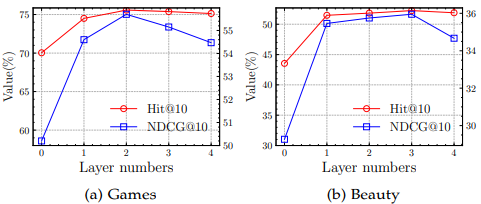
**4.2 Results**

The experiments explore three questions: 1). overall performance of DGSR compared to the other state-of-the-art sequential recommendation methods, 2). efficiency of the dynamic graph recommendation networks component in DGSR, and 3). hyper-parameter settings on DGSR.

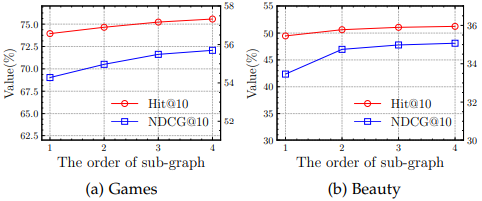
* Question 1: The results shows that DGSR on those three different datasets achieve best performance comparing with the other baseline methods.



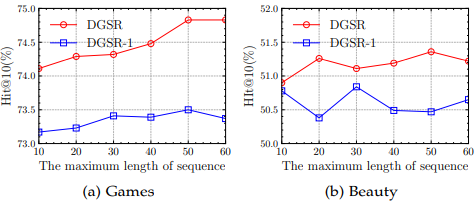
* Question 2: The experiment conducts 11 variants of DGSR with combination of components in long-term and short-term. The result shows that DGSR-DA has the best performance.
* Question 3: The experiment on DGRN Layer numbers, subgraph-sampling size, maximum sequence length and embedding size shows their relation to the model’s performance. The authors use NDCG@10 and HIT@10 as the measurement, and the results are same as follow:
  + While the numbers of DGRN layer increase, the model performance is more stable.



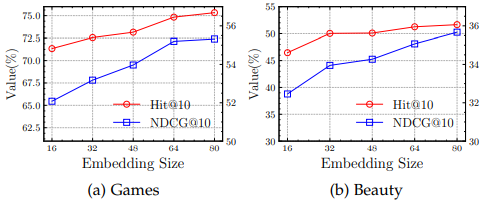
* + While the sub-graph sampling size increase, the model performance tends to be stable.



* + Increasing the maximum sequence length can improve the performance of the model. But blindly increasing will bring noise and cause the performance to degrade.

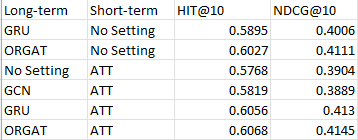


* + The performance will increase with the increased embedding size.



**4.3 Result from Code Demo**

The authors offer the code for the second question. In the code demo, I tried to run the 11 variants of DGSR with combination of components in long-term and short-term. To eliminate any other elements influent performance testing, all the variants use RNN as the user/item node updating method. Due to the limitation of my computation recourse, I only successfully generated the training, testing, and validating data from the Games dataset by using 25GB RAM. Besides this, there are several errors in the authors' code.  All the variants setting with using Last as short-term didn't work, and the code also miss the part of using GNC as long-term and no setting on short-term. Therefore, I generated 6 results out of 11. The results are in the following table.



According to the results from HIT@10 and NDCG@10 metrics, the variants setting with ORGAT (which also named DAT in article) as long-term and ATT as short-term has the best performance. Therefore, my code demo demonstrates the DGSR-DA has the best performance.

**5.Conclusion**

Overall, the authors offer the state-of-the-art method that improving the sequential recommendation by efficiently using the dynamic collaborative signals among different user sequences. The content of the article is clear and understandable to me as an audience. However, besides the paper, I wish they can also provide the testing code for the other two problems, and also fix the bug I mentioned in the previous section.