한국외국어대학교

Final Report: Federated Session-Based Recommendations

Professor: Seok-Lyong Lee

Students:
Ericka Bermudez 202230093
Tori Bukit 202232062

Date: 17/06/2022

Index

Introduction

In an era of digitalized economy, it has become increasingly important to offer users innovative tools for decision-making. Recommender systems assist users by offering them a set of personalized items they might be interested in [1], meanwhile session-based recommendation systems aim to suggest these personalized items based on user-item interactions over a certain continuous period of time [2].

This paradigm of recommendation systems is very useful as it does not require all of a user's historical data, but rather focuses on their current interest during a session [2], but it still carries some privacy concerns as the user interactions are saved and sent to a centralized server to use for training. This concern can be addressed by Federated Learning (FL).

During this project, we first present a quick overview of our dataset by implementing data mining tools seen in class. Next, we use association analysis to give suggestions to an e-commerce website. Finally, we use our dataset to implement a FL scenario for session-based recommendation.

Contributions

We consider two scenarios: first, recommendations without FL, and then using this approach. In the first scenario, we assume that a company only collects data about anonymous users' sessions on their website. In the second, users train the model on their own devices, and the company does not store any kind of data. Our contributions can be summarized as follows:

- We adapt two implementations (GRU4REC and SR-SAN) and use a dataset without timestamp, metadata, or user data to produce recommendations.
- We use the GRU4REC implementation and use it with a FL approach. To the best of our knowledge, while there have been contributions in recommendation systems with a FL approach [3] the area of session-based recommendation systems under the same is still an unexplored area.

Background

Recommended systems offer users a set of personalized items or services that matches their preference [4]. The two more common tasks related to recommender systems are the prediction of user opinion (e.g., rating), called the prediction task, and the recommendation of a set of interesting or useful items to the user, the recommendation task [5].

One of the most popular classifications of recommendation systems was pointed out by Bobadilla et al., which groups them into a) demographic filtering, based on the principle that people with common personal attributes may have also common preferences, b) collaborative filtering recommender systems, that uses similar rating patterns to make recommendations, c) content-based recommendation, which uses item's descriptions, and d) hybrid recommender systems [6].

These recommendation systems utilize all historical user-item interactions to learn about each user's preferences. This takes into account two assumptions: first, that historical data is available and, second, that all of the historical interactions are equally important to their current preference [2].

In recent years, session-based recommender systems have emerged as a new paradigm of recommendation systems. Session-based recommendation systems aim to capture short-term and dynamic user preferences to provide more timely and accurate recommendations sensitive to the evolution of their session contexts [2]. They are based on a log of sessions in which users have interacted with a website [7] and each session is composed of multiple user-item interactions that happen together in a continuous period of time [2]

Due to the relevance of offering non-registered users suitable recommendations, session-based recommendation systems have been widely studied and there is continuous research on improving them, such as using dependency-aware and context-aware angles [7] [8] and approaches with graph neural networks [9].

Recommendation systems have been addressed by FL. FL allows decentralizing the process of machine learning by allowing devices to collaboratively learn a shared prediction model while keeping all the training data on the device [3]. This allows recommender systems to overcome privacy and network limitations.

Related work

Wang et al. [2] proposed a FL-based recommendation system that aimed to provide accurate recommendations on cold-start items. Their datasets included separated users, each with rated items and personal data, and used TensorFlow Federated platform to perform their own implementation.

Traditional FL has a considerable communication cost, and the performance of a federated recommendation system has also been considered by Cui et al. [10]. Their research proposed a model to optimize communication cost, recommendation precision, novelty, and diversity. They used the same datasets as Wang et al. [2] and achieved high-performance recommendations.

Ribero et al. [11] proposed a method of FL that avoided data collection and guaranteed user-level differential privacy, without communication and noise overhead. They used a synthetic dataset intended to simulate discrete processes such as ratings or counting event occurrences, a second healthcare-related dataset with patient data, and the Movielens 1m dataset that was also used in the two previous research.

Privacy in FL for recommendation systems has been considered [11] [12]. Datasets in both papers included rated items by users. Zhao et al. [13] proposed Fed4Rec, a privacy-preserving framework for page recommendation based on FL. They used the Globo Dataset [14] which contains user interaction logs. On their framework, each client retains user-specific parameters, and there are

public users, whose data is shared with the server. The model focus on different aspects of pages such as page topics.

While there has been attention to the benefits of a FL approach, to the best of our knowledge there has been no attempt to apply it for session-based recommendation, whose types of datasets do not possess the same characteristics of the previously addressed ones and cannot be subjected to the same treatment.

Dataset

The dataset used is JD Micro Behaviors for the area of computers [15]. The dataset contains users' micro-behaviors, where each line is a sequence of a user's micro behaviors in a session. For example, the line "2885143+5+683+9+4 3282626+5+679+3+52 3282626+7+679+49+52" represents one session of one user. Micro-behaviors are separated by blank space, "2885143+5+683+9+4" is one micro-behavior.

Each micro-behavior is in the format of "sku + behavior_type + category + time_interval + dwell_time", where "SKU" is id of the product, "behavior_type" is the type of micro-behavior, "category" is the leaf category of the product, time_interval is the time interval between two consecutive micro behaviors, dwell_time is the dwell time that user needs to move to the next product.

To use in our experiments, the data had a first pre-process. Two new columns where added: *user_id* will tell us that that micro behavior was conducted by certain user, meanwhile *seq* refers to the order in which the micro-behavior appeared in its session. An example of pre-processed data can be observed in Table 1.

Table 1 Pre-processed data

session_id	seq	sku	behavior_type	category	time_interval	dwell_time
0	0	2885143	5	683	9	4
0	1	3282626	5	679	3	52
0	2	3282626	7	679	49	52
0	3	3203662	5	679	191	10
0	4	1836048	5	681	122	9

There are 90 unique categories in the dataset. Dwell time and time interval are positively skewed, with very little micro-behaviors with more than 100 and 50 seconds, respectively. For sequence, most sessions had less than 20 micro-behaviors each. One more thing to notice is than most of products had very little micro-behaviors associated.

The description of behaviors can be seen in Table 2.

Table 2 Description of behavior types

behavior_type	Micro behaviors	Description
1	Home2Product	Browse the product from the homepage
2	ShopList2Product	Browse the product from the category page
3	Sale2Product	Browse the product from the sale page
4	Cart2Product	Browse the product from the carted page
5	SearchList2Product	Browse the product from the searched results
6	Detail_comments	Read the comments of the product
7	Detail_specification	Read the specification of the product
8	Detail_bottom	Read the bottom of page of the product
9	Cart	Add the product to the shopping cart
10	Order	Make an order

Association Analysis

Association analysis is the discovery of association rules showing attribute-value conditions that occur frequently together in a given set of data [16]. Many industries are interested in mining the association rules from their databases, as the discovery can help to do a redesign of a company's webpage, marketing, and other business decision making processes.

A typical example of association rule mining is market basket analysis. This process analyzes customer buying habits by finding associations between the different items that customers place in their "shopping baskets" [16].

We performed association analysis trying to use both products and category, but the results were not insightful (very low support) as the "shopping baskets" did not often share data, and were quite sparse. Sessions were often focused in few categories. Instead, association analysis was performed for behavior. For this, first the data was filtered to only sessions in which a product was bought. Then, we analyzed which other behaviors were related to this. Table 3 was obtained with a support equal or more than 0.7.

Table 3 Result of association analysis for bought products

support	itemsets
0.701164	(4)
0.97542	(5)
0.812419	(7)
0.935317	(9)
0.791721	(5, 7)
0.917206	(9, 5)
0.752911	(9, 7)
0.737387	(9, 5, 7)

Adding the product to the shopping cart (9) is to be expected as it is common to do before making an order. The most frequent behaviors are browsing the product from the searched results and read the specification of the product. A relevant combination is (5, 7, 9, which suggest that people browse the product from searched results, read the specification, and finally add to cart in order to buy. For this scenario, we offer some suggestions we could make to the company in the conclusions.

SR-SAN

Session-based Recommendation with Self Attention Networks (SR-SAN) is an approach for session-based recommendation proposed by Fang [17]. This approach uses self-attention networks as a remedy to the capturing of long-range dependencies between items in session-based recommendation. The self-attention networks allow SR-SAN capture global dependencies among all items of a session regardless of their distance, using a single latent vector to capture current interest and global interest. The architecture of SR-SAN can be observed in figure 1.

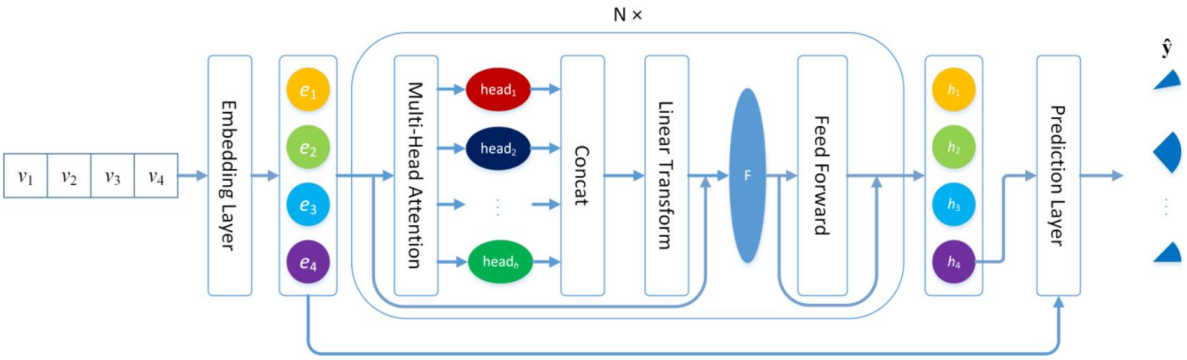

Figure 1 SR-SAN architecture [17]

SR-SAN is made of two parts: 1) Obtains item latent vectors wit self-attention networks and 2) making recommendation with prediction layer. The input for this algorithm is a dataset divided by sessions that uses a timestamp to divide the data.

SR-SAN Experiment

In this experiment, we performed an extra pre-process to use our data in SR-SAN. Our dataset does not contain a timestamp, and it was necessary to separate the dataset in SR-SAN original preprocess. For this, we created a fake timestamp based on the sequence of our dataset and random dates. This did not have a negative effect in our model, as dates are not used for prediction.

After doing the SR-SAN preprocessing, we could perform our experiment. For this, we realized, SR-SAN consumed too much resources, but had an attractive recall with little data. At 5 epochs and 1,000 sessions, the recall top 5 was 70.52% and MMR 60.15%, taking 90 seconds. On the other hand, it took around 57 minutes to train the model with 100,000 sessions, but we attained a very high recall top 5 of 95.21% and MMR of 78.47%.

In their paper, authors of SR-SAN do not mention the time of the training, but in their dataset, sessions have an average of ~6 micro-behaviors. Meanwhile, sessions in our have an average of ~27. The own nature of the SR-SAN uses the distance between items, and sessions this long might make it more costly.

From this experiment, we could see that SR-SAN has promising results. However, it consumes high resources and might not be the best model to perform FL.

GRU4Rec

GRU4Rec is one of the state-of-the-art algorithms that is used in session-based recommendation systems. This model used GRU-based RNN for predicting the sequence of the next items that the user will likely interact with within their session. The input for this algorithm is the actual session and the output is the item of the next session that the user is more likely to interact with. This algorithm will apply one hot encoding for the items in the input session, where the item will be presented with a vector with a length equal to the number of available items, and the value one is given to the corresponding column of the active item and the rest of the has zero value. All the vector input is then normalized to give better stability to the model. The core of this network architecture is the GRU layer and the GRU4Rec has an additional feedforward layer between the GRU layer and the output. We could modify GRU4Rec by adding more GRU layers in the middle of the network architecture or by changing the Feedforward layer's activation function to other types. The overall architecture of GRU4Rec could be seen in the Figure 2 [].

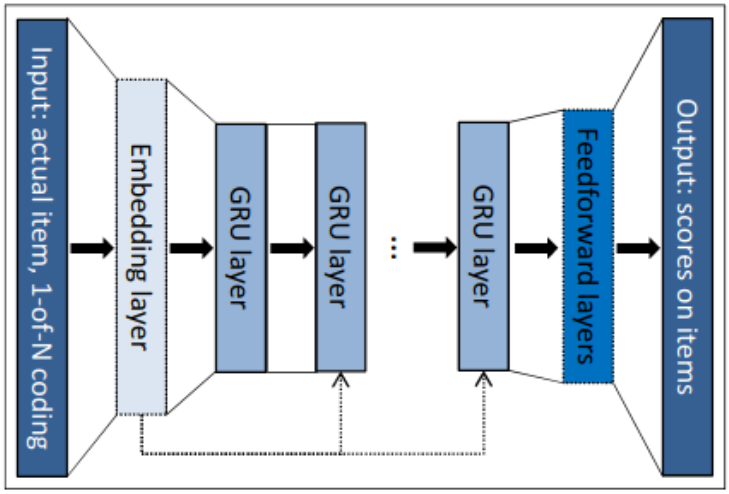

Figure 2 GRU4Rec architecture

Another thing worth mentioning about GRU4Rec is they use additional data preprocessing for the input data that could be processed by the GRU layer. RNN-based layers usually use in-sequence mini-batches for processing the task. For example, natural language processing uses a sliding window over the words of sentences and then creates mini-batches by putting these window fragments next to each other []. GRU4Rec uses session-parallel mini-batches to preprocess the

session input. The first step for creating these mini-batches is by creating the order for session input. After that, we use the first event of the first X session to form the input of the first mini-batch. The output of this mini-batch is the second event of the active session. We repeat this process to form the second mini-batch and so on. If any session ends, then we replace this session with the next one of our ordered sessions. Figure 3 illustrates how session-parallel mini-batches are created [].

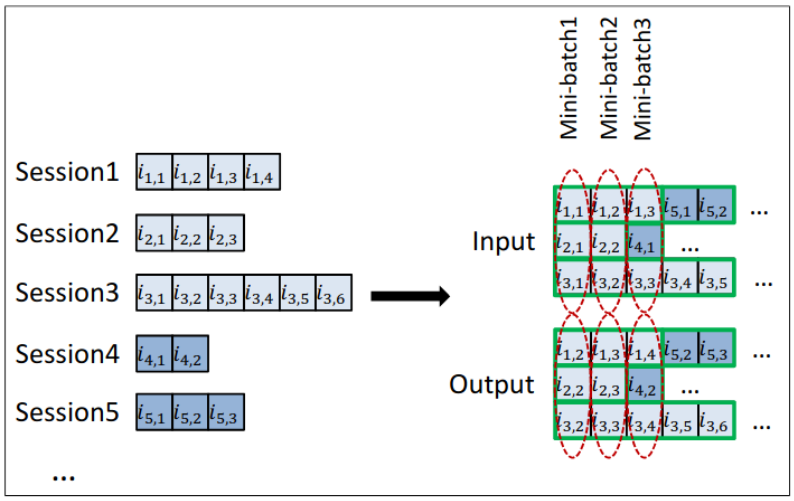

Figure 3 Session-parallel mini-batch

The last component of GRU4Rec is the lost function for optimizing the layer parameters. GRU4Rec supports pointwise and pairwise ranking loss. The difference between these two kinds of ranking loss is that pointwise ranking loss, will estimate the score or the rank of items independent of each other, and the loss is defined with the relevant items should be low. An example of this ranking loss is cross-entropy []. In the pairwise ranking loss, it will compare the score or rank the pairs of positive and negative items and the loss defined as the positive items should be lower than the negative items. An example of this ranking loss is BPR (Bayesian Personalized Ranking) [], Top1, and Top1-Max. Top1 and Top1-Max is the proposed ranking loss function in the GRU4Rec paper.

GRU4Rec Experiment

In this experiment, we will use to algorithm GRU4Rec to predict what is the next item or the next category that the user will interact with given the recent session data. In this GRU4REC experiment, the dataset already preprocesses so every event is represented as a row and we also add a sequence number to represent the order of events in a single session id. During the GRU4Rec experiment, we split the dataset into training and testing datasets in the proportion of 80:20 with the last 20% of session id being our testing dataset. At first, we will perform several experiments to obtain the best parameter setting in GRU4Rec that is most suitable for our dataset. Using one million session id for the next category prediction, we found the best parameter setting for JD.ID dataset is as follow:

From Table 4, with only 100 neurons we could achieve the 59% recall and 53% MRR. The best performing model for GRU4Rec (recall: 60% and MRR: 53%) is when the number of neurons is 1000 and we train with 10 epochs. From this result, we could conclude that the performance of using 1000 neurons and 100 neurons are not statistically different, meanwhile, the execution time is way shorter with the 100 neurons model only needing 20 seconds for predicting the next category. In the rest of the experiments, we will use the parameter set in number 1 as our configuration setting.

Table 5 show the result comparison for performing category recommendation. From the result, we could see that the performance of our model decreases as we use more session data for our training data. The worst performance is when we perform item recommendations using one million session data. GRU4Rec is only able to reach recall 3% and MRR 2%. We suspect the GRU4REC did not work well in our dataset because the item data is so sparse and only a small percentage of items appear more than once in our session data. GRU4Rec will perform best if the list of item candidates is smaller. The same phenomena could also be seen when performing category recommendations.

GRU4Rec Bidirectional

We also tried to change the GRU layer used in this GRU4Rec by using the GRU Bidirectional layer. From previous study, we found that bidirectional version in LSTM could outperform the uni-directional version of LSTM for sequence tagging case []. We try to change the layer of GRU4Rec in order to see if in this case the bi-directional version of GRU is also perform better. From the Table 6, we could see that the overall performance of Bidirectional-GRU4Rec is worse than the normal version of GRU4Rec. We could conclude that Bidirectional-GRU4Rec is not suitable for this recommendation task using JD.ID dataset.

Table 4 Benchmark of parameters

No	Loss Function	Neuron	Activation Function	Hidden Layer	Learning Rate	Epoch	Optimizer	Recall @5	MRR @5	Execution Time (s)
1	**TOP1-max**	**100**	**Tanh**	**1**	**0.05**	**10**	**Adagrad**	**59**	**53**	**20**
2	TOP1-max	1000	Tanh	1	0.01	5	Adagrad	60	52	56
3	TOP1-max	1000	Softmax	1	0.01	5	Adagrad	59	53	55
4	TOP1-max	1000	Tanh	1	0.01	10	Adagrad	60	53	50

Table 5 GRU4Rec experiment results

Session Count	Task	Recall @5	MRR @5
1000000	Category Recommendation	59	53
100000	Category Recommendation	62	56
1000000	Item Recommendation	3	2
100000	Item Recommendation	30	25

Table 6 GRU4Rec bidireccional experiment results

Session Count	Task	Recall @5	MRR @5
1000000	Category Recommendation	59	53
100000	Category Recommendation	56	52
1000000	Item Recommendation	1	1
100000	Item Recommendation	27	24

GRU4Rec in FL

From our experiment so far, GRU4Rec performs worst when the number of session IDs is so big. From this phenomenon, we will perform GRU4Rec in a FL setting to see how this algorithm work in the federated setting. In FL, the dataset is not sent to the centralized server so the learning must perform on each user device. In FL, the number of datasets used in each local training is way smaller than when performing training in a centralized way. When implementing GRU4Rec in the FL setting, we use the Flower framework to perform a simulation of FL. By using Flower, we could create several virtual clients from one computer and perform the simulation of FL. For implementing GRU4Rec in flower simulation, we will add additional methods to send and receive parameter updates from clients to the server and vice versa. For the FL algorithm itself, we use the FedAvg and this algorithm is already included in the Flower framework.

In this FL experiment, we will use 300 clients for simulation and we will divide the session id into these 300 clients. We will perform category recommendation task in this FL experiment. Because our dataset didn't have any user ID and only has a session ID, we cannot separate the dataset into each user that reflects the real-life scenario. We will separate these sessions by using binning techniques with the equal width technique. In our FL experiment, we also want to compare the result of GRU4Rec when we perform training by only using data from a single user (we called it the single setting) and in the FL setting. The main difference between the single setting and the FL setting is in the single setting the local weights are not aggregated with other local weights from other clients. Meanwhile, in FL setting the weights are aggregated for each round of training and we will use these global weights for testing in each of the local devices or clients. In this experiment, we will perform the experiment using 5 epochs for single setting and 5 round of training for FL setting. Table 7 shows the sample result of our experiments.

Table 7 GRU4Rec in FL

Client	Setting	Recall @5	MRR @5
0	Single	52	37
0	FL	54	37
105	Single	54	49
105	FL	53	51
36	Single	69	60
36	FL	74	68
126	Single	60	51
126	FL	77	67

From Table 7, we could see that the best performance in terms of recall is 77% which is the accuracy for client number 126. This recall value is greater than the recall value in the

centralized setting for GRU4Rec in Table 5 (average recall is 62% and MRR is 56% for 1000 session ID). The performance is also improved if we compare the in a single setting for only using data in client 126 (recall: 60% and MRR: 51%). For the worst performance, we could see in the client number 105, where in FL setting only achieve 53% recall with 51% MRR. If we compare to the single setting, the recall is 54% and MRR 49%, so the FL training is not improving the performance of the model for client number 105. We see a decrease in performance in recall and only improve the value for the MRR. Based on our experiment, we could see that GRU4Rec in FL setting could outperform the centralized training version of GRU4Rec for several clients. The other clients perform worse than the centralized setting with the worst recall we could see the value is 53% (There is no client with a recall value is less than 52%). We also could conclude that the FL setting doesn't always increase the performance of the model compared to single setting training. It is interesting to investigate this phenomenon in future study.

From Table 6, we could see that the best performance in term of recall is 77% which is the accuracy for client number 126. This recall value is greater than the recall value in the centralized setting for GRU4Rec in the Table 5 (average recall is 62% and MRR is 56% for 1000 session ID). The performance is also improved if we compare the in single setting for only using data in client 126 (recall: 60% and MRR: 51%). For the worst performance, we could see in the client number 105, where in FL setting only achieve 53% recall with 51% MRR. If we compare to the single setting, the recall is 54% and MRR 49%, so the FL training is not improving the performance of model for client number 105. We see the decrease performance in recall and only improve the value for the MRR. Based on our experiment, we could see that GRU4Rec in FL setting could outperform the centralized training version of GRU4Rec for several clients. The other clients perform worse than the centralized setting with the worst recall we could see the value is 53% (There is no client with the recall value is less than 52%). We also could conclude that the FL setting doesn't always increase the performance of model compare to single setting training. It is interesting to investigate this phenomenon in the future study.

Conclusion and Future Work

This project tries to perform exploration and data analytics for the real dataset to gain insight that could support the decision maker. This project also tries to design a recommendation system that performs well in this real dataset and still maintains greater privacy for the users. In this project, we use a real dataset from Chinese E-Commerce called JD. The data contain several session ID from a different user, where each session contain several event or behavior that the user perform during one session. In this project, we demonstrate that by performing data analytic techniques, we could get some of the insights that could help the decision makers in JD company. One of the examples of the insight is customers who bought the product, will usually visit the search page and read the product description item. Another insight that we found is the support for the customer who visits the sale product pages and then after that buying the product is not so high. From this finding, we could recommend the decision maker inspect their marketing strategy and their sales pages.

For the recommendation system, we show that using a session-based recommendation system algorithm could perform well for this real dataset. Here we implement two state-of-the-art algorithms (GRU4Rec and SR-SAN) for the experiment, and we found both of these algorithms have their strength and weakness. In terms of recall and MRR, SR-SAN outperforms GRU4Rec in every case of our experiment. But SR-SAN has a high computational cost and it takes more time for training the model compared to GRU4Rec. Another interesting thing for GRU4Rec that we found is the algorithms tend to produce lower accuracy as the training data grow bigger. We suspect that GRU4Rec is not performing quite well if the total number of item candidates is high. We found that GRU4Rec is performing better if the candidate items to be recommended are less than 1000 items (from category recommendation task experiments and item prediction experiments using only 10.000 session ID). We also conclude that GRU4Rec will perform better if the architecture used is simple (using only 100 neurons, 1 layer, unidirectional layer).

To increase the level of privacy for the user, we also try to implement GRU4Rec in the FL setting. By using the Flower framework, we can implement GRU4Rec in FL setting for 300 clients and we use the FedAvg algorithm for aggregating the weight. From our experiment, the performance of GRU4Rec is quite good compared to the result in a centralized setting. The highest recall score in FL clients is 77% which is better than the overall recall in centralized settings (59%) and the lowest recall score in FL clients is 53% which is not so a bit different from centralized settings. From our experiment in the FL settings, the aggregate weights are not always improving the performance of the model. In some of our clients in FL, the performance is still the same if we compare it using weight only from local training and using aggregate weight from the global model. We will inspect this result in future work to improve how the weight is aggregated for GRU4Rec in the FL settings.

In future works, we will try to implement the SR-SAN algorithm in the FL settings. From our experiment, we see that SR-SAN performs better than GRU4Rec even though it needs more computational power to run. We want to observe the performance of SR-SAN in the FL settings and try to reduce this computational cost in the FL settings. We are interested to improve this algorithm in the FL setting because the best of our knowledge, FL research is still growing and the work on a session-based recommendation system in the FL is still limited.

Another thing that we want to extend in the future is to create a multi-level-based recommendation session. Based on our experiment in GRU4Rec, this algorithm is suffered if the number of item candidates is too many. The next work will try to utilize the category recommendation to prune the item candidates for the item recommendation task. At the first level, we will predict what is the next category item that the user interacts with in the future. After that, the second level will predict the next item by using only candidate items from the same category output from level one. By performing this approach, we expect to see improvement in GRU4Rec because the number of candidates is reduced.

References

[1] O. A. Wahab, G. Rjoub, J. Bentahar and R. Cohen, "Federated against the cold: A trust-based federated learning approach to counter the cold start problem in recommendation systems," *Informatoin Sciences,* vol. 601, pp. 189-206, 2022.

[2] S. Wang, L. Cao, Y. Wang, Q. Z. Sheng, M. A. Orgun y D. Lian, «A Survey on Session-based Recommender Systems,» *ACM Comput. Surv,* May 2021.

[3] B. McMahan and D. Ramage, "Federated Learning: Collaborative Machine Learning without Centralized Training Data," Google AI Blog, 6 April 2017. [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. [Accessed 12 June 2022].

[4] R. Yera y L. Martinez, «Fuzzy Tools in Recommender Systems: A Survey,» *International Journal of Computational Intelligence Systems,* vol. 10, pp. 776-803, 2016.

[5] A. Gunawardana y G. Shani, «A Survey of Accuracy Evaluation Metrics of Recommendation Tasks,» *Journal of Machine Learning Research ,* vol. 10, pp. 2935-2962, 2009.

[6] J. Bobadilla, F. Ortega, A. Hernando y A. Gutierrez, «Recommender systems survey,» *Knowledge-Based Systems,* vol. 46, 2013.

[7] T. Wu, F. Sun, J. Dong, Z. Wang y Y. Li, «Context-aware session recommendation based on recurrent neural networks,» *Computers and Electrical Engineering,* 2022.

[8] D. Yan, T. Tang, W. Xie, Y. Zhang y Q. He, «Session-based social and dependency-aware software recommendation,» *Applied Soft Computing,* March 2022.

[9] C. Zhang, W. Zheng, Q. Liu, J. Nie and H. Zhang, "SEDGN: Sequence enhanced denoising graph neural network for session-based recommendation," *Expert Systems with Applications,* vol. 203, October 2022.

[10] Z. Cui, J. Wen, Y. Lan, Z. Zhang y J. Cai, «Communication-efficient federated recommendation model based on many-objective evolutionary algorithm,» *Expert Systems with Applications,* vol. 201, 2022.

[11] M. Ribero, J. Hendersson, S. Williamson and H. Vikalo, "Federating recommendations using differentially private prototypes," *Pattern Recognition,* vol. 129, September 22.

[12] D. Yan, Y. Zhao, Z. Yang, Y. Jin y Y. Zhang, «FedCDR: Privacy-preserving federated cross-domain recommendation,» *Digital Communications and Networks,* 5 May 2022.

[13] S. Zhao, R. Bharati, C. Borcea y Y. Chen, «Privacy-Aware Federated Learning for Page Recommendation,» de *IEEE International Conference on Big Data*, 2020.

[14] G. Moreira, "News Portal User Interactions by Globo.com," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/gspmoreira/news-portal-user-interactions-by-globocom. [Accessed 13 June 2022].

[15] Y. Gu, Z. Ding y S. Wang, «Hierarchical User Profiling for E-commerce Recommender Systems,» *WSDM '20: Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 223-231, 2020.

[16] J. Han y M. Kamber, Data Mining: Concepts and Techniques, 2000.

[17] J. Fang, «Session-based Recommendation with Self-Attention Networks,» *CoRR,* 2021.

[18] Hidasi, Balazs, A. Karatzoglou, L. Baltrunas y D. Tikk, «SESSION-BASED RECOMMENDATIONS WITH RECURRENT NEURAL NETWORKS,» de *International Conference on Learning Representations,* San Juan, 2016.

[19] H. Steck, «Gaussian ranking by matrix factorization,» de *ACM Conference on Recommender Systems*, Vienna, 2015.

[20] S. Rendle, C. Freudenthaler, Z. Gantner y L. Schmidt-Thieme, «BPR: Bayesian personalized ranking from implicit feedback,» de *Uncertainty in Artificial Intelligence*, Montreal, 2009.

[21] Z. Huang, W. Xu y K. Yu, «Bidirectional LSTM-CRF Models for Sequence Tagging,» *arXiv*, vol. 1508.01991, 2015.