School of Informatics

MSc Project Progress Report
Recommender Systems: looking further into the future

Kyriakos Kyriakou
July 2023

Date: Thursday 13th July, 2023

Supervisor: David Wardrope, Timos Korres

1 Project Goals

By pursuing the following project goals, the research aims to enhance the accuracy and effectiveness of recommender systems, leading to improved customer satisfaction, increased subscriptions, and overall advancements in the field of recommendations. General project goals:

1. Develop a baseline sequential recommender system for next-item predictions.
2. Extend the system to predict the sequence of a user's future interactions.
3. Experiment with different techniques to improve the future item window predictor.
4. Evaluate the effectiveness of using a future interaction window predictor for making recommendations.
5. Determine the optimal interaction window for predicting future behaviors.
6. Compare the proposed approach with state-of-the-art methods for recommendation methods.

2 Progress & Methods

As mentioned previously, this study will focus on developing a sequential recommender system with an interaction window predictor to enhance the accuracy and efficiency of the system.

Data Collection & Preparation: We collected potential datasets, such as MovieLens [1], and Retail Rocket [2] from online available sources[1] and use them to build the system. Exploratory Data Analysis has been already carried out for both datasets (will probably include some graphs in the actual thesis) but only MovieLens 1M has used so far for training. The first initial goal of the project was to develop a time-window future predictor but our datasets does not allow temporal splitting that will enable the models learn. Thereafter, we switched to a future interaction window-based predictor. Our system requires relatively long sequences of user-items interactions in order to ensure robustness in predicting multiple items in the future. We keep, as Kang et al. in the SASRec study, only sequences that have more than 5 interactions [3].
For preparing the data we have created different data preparation methods for different techniques. Next, we will state all the techniques / models that have been implemented so far, explain them briefly, and state how data is prepared for each. The architecture used so far in the project is SASRec [3].

Next item training (Baseline): We use the standard approach of predicting the next item for the baseline system as in [3]. SASRec was selected as a baseline as it's popular across the literature. We use next item training of SASRec (no future window training involved), and we evaluate (both valid and test) on a window size items. We partitioning the data into three categories: training, validation, test. We split the data based on a percentage. Currently we use 80% of the data sequence of each user as training (oldest interactions), 10% for validation (a bit most recent), and 10% for testing (most recent data). For example: $mostrecent \rightarrow$
$\| - - - - - - - - - train 80\% - - - - - - - - - - \| - - valid 10\% - - \| - - test 10\% - - \|$

Next item training + Independent Feeding Split: Independent feeding split is a new data partitioning approach implemented to try to improve our future predictor. We split like

[1] MovieLens Dataset, Retail Rocket Dataset

before in train - valid - test (80%,10%,10%) but this time instead of feeding to the model the training sequence, we split the training sequence further into input and target sets. In our implementation so far we use 90% of the train data for input and 10% for target. After the splitting, we create multiple training sequences for each user where we append to each an item from the target. For example, if the input sequence for a user is $[1, 2, 3, 4, 5, 6]$ and the target $[7, 8, 9]$, we create new sequences as follows:

$$[1, 2, 3, 4, 5, 6, 7]$$
$$[1, 2, 3, 4, 5, 6, 8]$$
$$[1, 2, 3, 4, 5, 6, 9]$$

This allows new item embeddings to be created while training as next item prediction, which potentially will allow capturing patterns to predict better future items.

Next item training + Sequential Feeding Split: Sequential feeding split is another data partitioning approach implemented. We split exactly like the independent feeding method into train (input 90%, target 10%), valid, test as 80%-10%-10%. We then create multiple training sequences for each user where we append to each sequentially the target items like teacher forcing. For example, if the input sequence for a user is $[1, 2, 3, 4, 5, 6]$ and the target $[7, 8, 9]$, we create new sequences as follows:

$$[1, 2, 3, 4, 5, 6, 7]$$
$$[1, 2, 3, 4, 5, 6, 7, 8]$$
$$[1, 2, 3, 4, 5, 6, 7, 8, 9]$$

This again, will create different item embeddings for next item training, which could lead to more accurate predictions.

All action prediction: We implemented all action prediction from Pinnerformer, where we use the last embedding in the sequence to predict the next k future items [4]. We split like before into train-valid-test, where the the train has input and target sequences. We use the target sequences as the positive samples of the model to be trained using the last embedding instead of 32 used in the paper [4]. If the target positive samples are less than our window predictor, we sample items randomly from the training target window to populate it. We use 1:1 ration positives:random negatives samples.

Dense all action prediction: We implemented dense all action prediction from Pinnerformer, where we use the different random item embeddings in the sequence to predict, for each, a future item (one positive sample) from the target sequence [4]. Split is the same as All action prediction.

Super Dense all action prediction: We extended dense all action prediction from Pinnerformer. Instead of using each different random item embeddings in the sequence to predict single future item from the target sequence, we use each random item embedding to predict all the target items (all positive samples). Split is the same as All action prediction.

Future rolling window prediction: We extended all action prediction from Pinnerformer. Instead of using the last embedding to predict the positive target items, for each item position in the training sequence we create multiple target windows of length k to predict the next items. For example: for seq. $[1, 2, 3, 4]$ and target $[5, 6, 7]$, we predict the window size items as follows:

Window size $k = 3$

For item embedding 1, the model will try to predict $[2, 3, 4]$,
For item embedding 2, the model will try to predict $[3, 4, 5]$,
For item embedding 3, the model will try to predict $[4, 5, 6]$,
For item embedding 4, the model will try to predict $[5, 6, 7]$

For the aforementioned methods to train the window-based predictor we also incorporated some training strategies to help the model train better:

Masking: Inspired by Sun et al. in BERT4Rec, we implemented a similar masking technique to help the model train better and possibly understand more patterns to predict better in the future. In All action prediction, we create a masked sequence as well based on a percentage. For example in the input training sequence of $[1, 2, 3, 4, 5]$ we mask randomly items to get a new masked sequence $[1, 2, 0, 4, 5]$. We use 2 losses. The first one we call it target loss and it's used for predicting the targets (pos. samples) normally like before, and the second loss (masked loss) is calculated based on the predictions of the masked items (we pass the original training sequence and the masked and we try to predict the masked items). Then we combine the losses and optimize the model.

Teacher Forcing: In this training strategy, our recommender system is trained by providing both the input sequence and the corresponding target sequence. For example, when training on the sequence [1,2,3,4,5] with the target sequence [6,7,8], the model predicts the next item based on the input sequence while appending the correct target item to the input sequence and generating new item embedding. This process iteratively continues, creating updated sequences and predicting subsequent target items. However because the embedding length is fixed, we remove the oldest interaction from the sequence for each appended item.

Auto-regressive: Similarly to teacher forcing, our recommender system predicts the next item in the sequence based on the previous items. However, here the predicted item is then appended to the sequence, and the process repeats for subsequent predictions.

The experiments conducted so far use three losses: **Binary Cross Entropy**, **Uniform Sampled Softmax**, and **Sampled Softmax with LogQ correction** [3, 4].

System Evaluation: We utilized four different metrics for evaluation.

$$NDCG@N_i = \frac{1}{|U|} \sum_{u \in U} \frac{1}{log_2(rank_i + 2)} \quad (1) \qquad NDCG_{avg} = \frac{1}{K} \sum_{i=1}^{K} NDCG@N_i \quad (2)$$

In the first equation, $NDCG@N_i$ denotes the NDCG for i-th position in the top-N recommendation, averaged across all users. If users have less than the positions in the their valid/test sets we don't evaluate them. Then the second equation, the average NDCG over the future window size K is calculated.

Then, Hit Rate (HT@N) is calculated with a similar way. If item is in top-N increases by 1. Then the average over users is taken, and then the average over the window.

We implemented Order_Score@N metric, a novel metric for evaluating the position of a correct recommendation within a window of future interactions. The order of recommendations is important in our system. For every correct prediction, we calculate the Order Score by taking the difference between the total number of future positions K and the absolute difference of the true position and predicted position, divided by the number of future positions. Essentially, this score is higher for recommendations whose predicted and actual positions are closer:

$$Order\_Score@N_i = \frac{1}{|U|} \sum_{u \in U} \frac{K - |i - j|}{K} \quad (3) \qquad Order\_Score_{avg} = \frac{1}{K} \sum_{i=1}^{K} Order\_Score@N_i \quad (4)$$

where, i is the true position in the future, and j is the predicted position (rank). This score is calculated for each top-N prediction independently. If the item is not in top-N, the metric is 0.

Finally, we employ Kendall's Tau to measure the correlation between the true ranking of items and their ranking as predicted by the model in the K future window. The higher the Kendall's Tau, the more the predicted ranking resembles the true ranking.

$$Kendall's\tau = \frac{1}{|U|} \sum_{u \in U} \tau_u(true\_positions, predicted\_rankings) \tag{5}$$

Kendall's tau is using the top-1 predictions of the model to be calculated, thus harder to get good performance.

Models	NDCG@10		HR@10		Order_Score@10		Kendall's Tau	
	BCE	Unif. SS	BCE	Unif. SS	BCE	Unif. SS	BCE	Unif. SS
Baseline	0.5556	0.5557	0.8164	0.8055	0.5278	0.5211	0.1450	0.1501
Independent Feeding Split	0.5226	0.5609	0.7929	0.8115	0.5087	0.5234	0.1061	0.1356
Sequential Feeding Split	**0.5582**	**0.5660**	**0.8193**	**0.8161**	**0.5313**	**0.5272**	**0.1467**	**0.1509**
All Action	0.4233	0.4329	0.6843	0.6908	0.4298	0.4349	0.0873	0.0895
Dense All Action	0.2853	0.3077	0.5184	0.5478	0.3166	0.3337	0.0492	0.0466
Super Dense All Action	0.2961	0.3055	0.5397	0.5468	0.3254	0.3332	0.0434	0.0486
Future Rolling Window	0.2711	0.2827	0.5077	0.5274	0.3025	0.3161	0.0345	0.0360

Table 1: Results so far for the experiments

Models	NDCG@10		HR@10		Order_Score@10		Kendall's Tau	
	BCE	Unif. SS	BCE	Unif. SS	BCE	Unif. SS	BCE	Unif. SS
Baseline	**0.5556**	**0.5557**	**0.8164**	**0.8055**	**0.5278**	**0.5211**	**0.1450**	**0.1501**
All Action	0.4233	0.4329	0.6843	0.6908	0.4298	0.4349	0.0873	0.0895
All Action + Masking (15%)	0.4234	0.4161	0.6845	0.6669	0.4310	0.4188	0.0893	0.0854
All Action + Autoregressive	0.4360	0.4214	0.7028	0.6763	0.4428	0.4225	0.0811	0.0679
All Action + Teacher Forcing	0.4733	0.4694	0.7349	0.7297	0.4693	0.7235	0.1072	0.1068

Table 2: Results of all action prediction with different training strategies

3 Future Plans

We are planning to implement a combined model of SASRec next item training and all action / dense / future rolling. We will evaluate this model and experiment how the implemented training strategies work with it. Also we are planning to incorporate time embeddings (time2vec) in the SASRec to improve the model [5]. Furthermore, we will experiment with different window sizes. If we outperform baseline, we then start writing of the thesis, otherwise it could be worth looking into transfer learning for the item embeddings and/or temporal window split with the bigger Movielens 20M dataset. Statistical testing might be conducted if results are not clear. Lastly, if time allows we might experiment with Retail Rocket dataset.

References

[1] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[2] Roman Zykov, Noskov Artem, and Anokhin Alexander. Retailrocket recommender system dataset, 2022.

[3] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[4] Nikil Pancha, Andrew Zhai, Jure Leskovec, and Charles Rosenberg. Pinnerformer: Sequence modeling for user representation at pinterest. *arXiv preprint arXiv:2205.04507*, 2022.

[5] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019.