School of Informatics

Progress Report
Recommender Systems: looking further into the future

S2406121
July 2023

Date: Thursday 13th July, 2023

1 Project Goal

The next-item prediction approach is widely used for the sequential recommender system, which takes a time-ordered user sequence to predict the next most likely user action. This project aims to develop a new window-based approach that could potentially outperform the next item prediction. Different from focusing on the next item, this novel approach asks the model to predict all items that the user could interact with within a given time window. Illustrated in Fig 1, the training objective of these two approaches is different, where next-item prediction is only interested in the next item in each time step, but the window-based prediction is going to use the input sequence to predict all items in the target window.

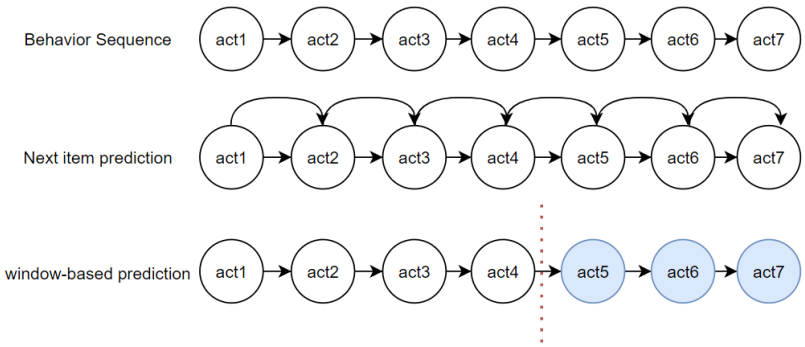

Figure 1: Difference between next-item prediction and window-based prediction

Therefore, the goal of the project is to develop a window-based sequential recommender system with a better recommendation performance based on recall and better recommendation diversity based on P90 coverage than the current state-of-the-art recommender systems. The objective responds to users' requirements for a more personalised user experience, increasing their satisfaction with the recommendation results.

2 Methodology

This project can be divided into 4 work packages (WPs). Firstly, a sufficient background reading should be conducted to understand the work of the current SOTA recommender systems. Following this, dataset handling is also significant. In the third WP, we need to develop some novel models that can be used as the window predictor. Finally, the analysis of experiment results will be conducted based on the evaluation of model performance. In addition, all datasets we are using to train and test the model are large-scale publically available datasets such as MovieLength-20M dataset.

Background Reading: To enhance the understanding of current SOTA recommender systems majorly focusing on transformer-based recommender systems, it is essential to do an intensive reading task at first.

Data handling: Different from next item prediction, window predictor requires a bag of target items, so we need to handle the dataset more carefully before applied to the recommender. Because the time window has relatively high demand requirements on the dataset, we first split the dataset by percentage to fit the window structure, which could be simply applied to all datasets as shown in Fig 2. When the model works fine with the percentage-split data, we split

the dataset by time window and feed the dataset to the model.

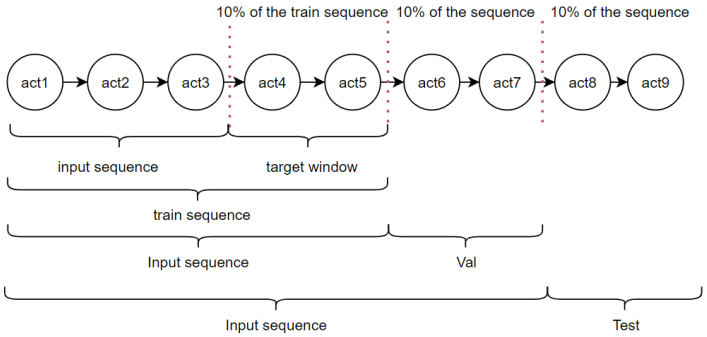

Figure 2: splitting data by percentage

Model Development: To test which structure is more suitable for a window-based sequential recommender system, we try to implement five training objectives for the model. The simplest one called all action prediction is just using the last context-related item embedding of the transformer structure to represent the user embedding and then doing a softmax with the item embedding matrix to make the recommendation.

Analysis: After implementing all of these models successfully and utilizing the suitable features to enhance the recommendation, we finally analyse the model performance with the baseline model that uses the next item prediction method.

3 Accomplished Progress

Background Reading: Currently, an extensive background reading has been completed, leading to a comprehensive understanding of the SOTA approaches.

Data Handling: As mentioned above, for normal next-item prediction, we just need the time-ordered sequence, and we also need different approaches to split data by percentage and time. Therefore, including the time-ordered sequence, we have implemented three kinds of data partition strategies to fit our requirements.

Model Development:

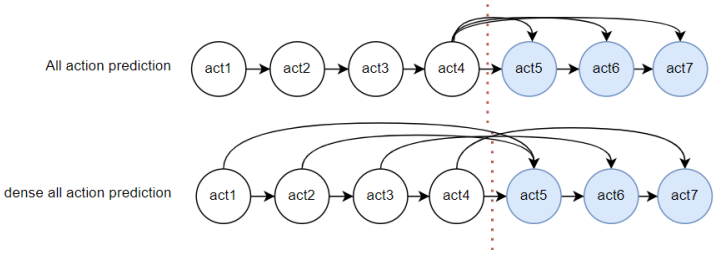

Figure 3: Two window predictors

As previously mentioned, we have implemented five window-based recommender systems, two of which are shown in Fig 3. For all action prediction, we ask the model to predict all items in

the target window by the final item embedding of the transformer output. For dense all action prediction, we assume each action has some influence on the prediction, so we ask each action in the input sequence to predict one randomly selected item in the target window.

Transfer Learning: Due to some experiment results, the baseline (next item prediction) outperforms all window-based models. Inspired by the supervisor and a paper on a window-based predictor, we tried fixed item embedding by transfer learning instead of training from scratch. Transfer learning can help a lot to improve the performance of all window-based recommender systems according to our experiment results. Based on the experiments, the next item prediction will generate a more stable item embedding training from scratch than window-based predictors, we then transfer the item embedding to our proposed window-based models. Two approaches are carried out to apply transfer learning to our models, the first one is we just freeze the pre-trained item embedding, and the other is we choose a much smaller learning rate for the item embedding than other parameters. Both of these two approaches improve window predictors than training the model from scratch.

4 Future Steps

More reading: Reading will not stop during the whole progress, we always need to do more readings when we want to incorporate the model with some new features. we currently trying to implement time features to the model by Time2Vec, so we need to do more reading on this part.

Evaluation metric: The evaluation we currently used is recall, which only considers the recommendation quality. Therefore, to achieve the goal of our project, we also need to correctly implement P90 coverage, which evaluates the recommendation diversity.

Incorporate time features: The final goal of our project is going to predict all possible actions in next k days, which is a time-sensitive task, so we assume incorporating time features should also improve the model. There are two ways of incorporating time features from reading. We can just use Time2Vec to get the time feature or we can use a personalized time interval process. The latter one is more complicated and has more memory cost, which can easily result in a memory error. The major objective at the moment is to correctly implement this task.

Results Analysis: The final step of our project is to compare the model performance on recommendation quality and diversity. Once we have conducted formal experiments, it will be crucial to document the outcomes and analyse the performance of the model on different metrics.