A deep reinforcement learning based long-term recommender system

Jiacheng Liu [1] Gang Qiao [2] Hu Sun [2]

[1]Industrial & Operations Engineering

[2]Department of Statistics

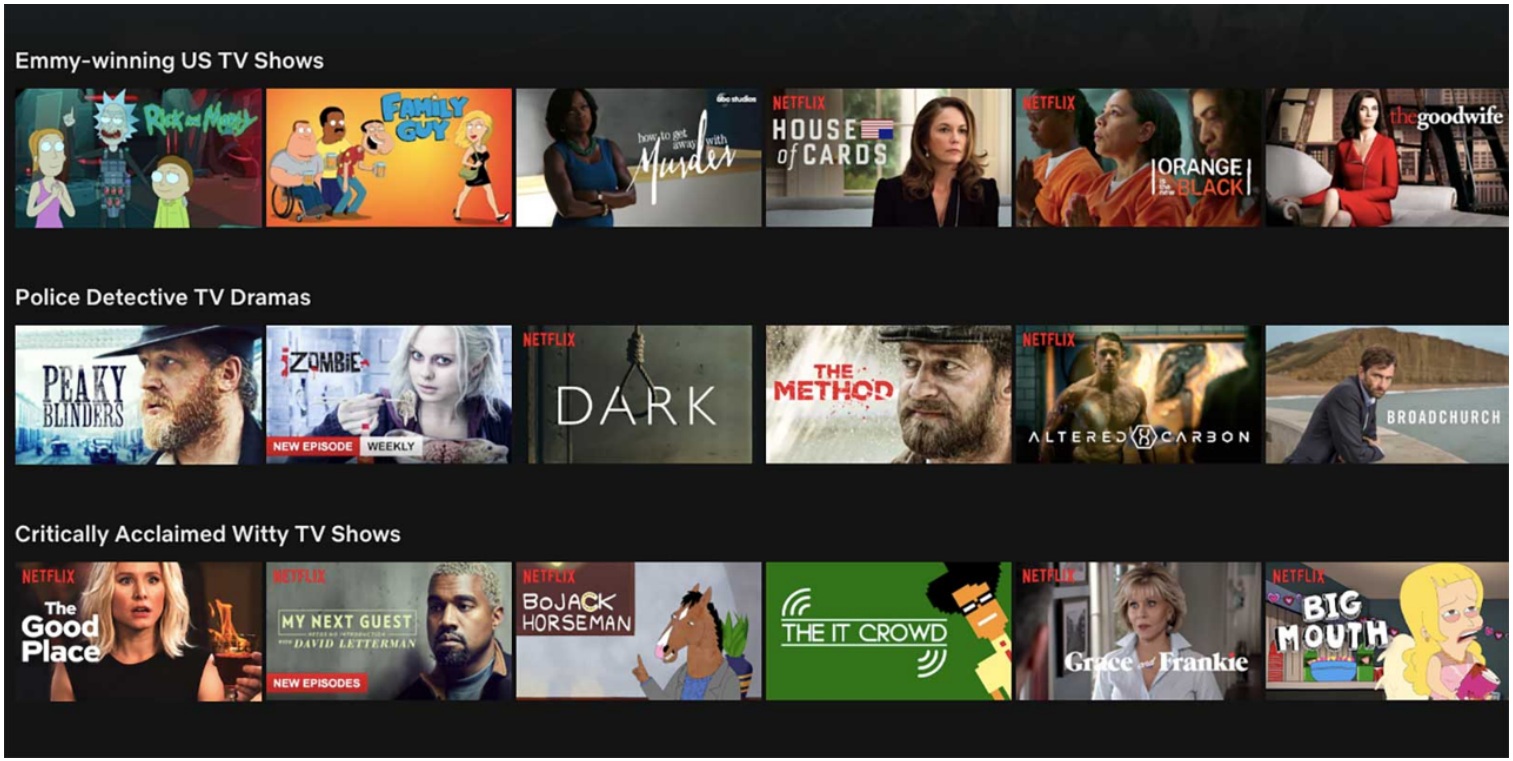
March 30, 2021

Overview

Recommender System

Everything is a Recommendation.

Personalized experiences are key to today's s internet businesses. Recommender Systems particularly provide users with personalized recommendations about content they may be interested in.



The Values of Recommender System

1. Netflix: 2/3 of the movies watched are recommended
2. Google News: recommendations generate 38 % more clickthrough
3. Amazon: 35 % sales from recommendations

Recommender System Methods

1. Content-based filtering
   - Recommendations based on item descriptions/features, and profile or past behavior of the "target" user only

Recommender System Methods

1. Content-based filtering
   - Recommendations based on item descriptions/features, and profile or past behavior of the "target" user only
2. Collaborative filtering (CF)
   - Look at the ratings of like-minded users to provide recommendations
   - Static view, ignore the dynamic and sequential nature of the recommendation problem
   - Cold-start issue since relies on the historical data

Recommender System Methods

1. Content-based filtering
   - Recommendations based on item descriptions/features, and profile or past behavior of the "target" user only
2. Collaborative filtering (CF)
   - Look at the ratings of like-minded users to provide recommendations
   - Static view, ignore the dynamic and sequential nature of the recommendation problem
   - Cold-start issue since relies on the historical data
3. Sequential Decision Process
   - Adaptability is crucial to RS since personal preferences of users always evolve over time
   - Short-term prediction, eg. RNN, SL, strict order with ground-truth labels
   - Long-term prediction,eg. RL, not actually previously selected, multiple optional correct targets without clear and unique labels.

RL for Recommender System

1. RL does not require an explicit target
2. RL aims to maximize the long-term returns
3. Current Problems
   - Dimensionality problem for complex systems
   - Ignored interaction between the recommendation agent and the user by RNN-based models

Problem Setting

1. Top-N model
2. Long-term prediction with high accuracy
   - Traditional evaluation eg. Leave-One-Out doesn't work
   - Popular metrics HR and NDCG
3. Deep reinforcement learning (DRL)
4. Recommendation situations
   - cold-start: no historical items provided to the test users
   - warm-start: whether in training or test, p% historical items are supplied to construct the users' history

Overall Framework

Environment different users
Agent RNN-based CF model
Action the top-N recommendation list for a particular user
Feedback whether the user accepts this recommendation list or not.

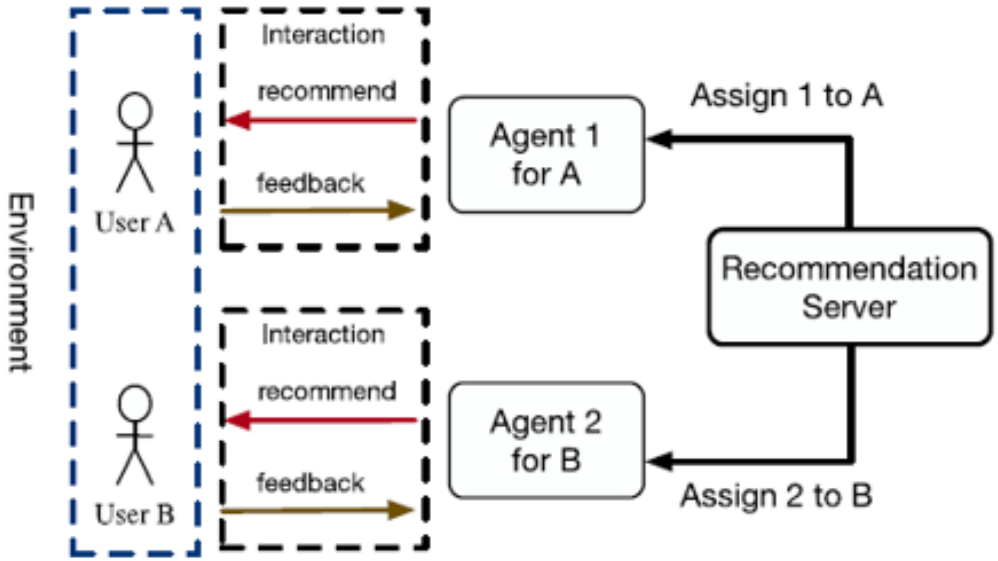


Figure: Overall recommendation processes

Overall Framework

1. State transition $\mathbb{T}$

$$s_{u,t} = \mathbb{T}(s_{u,t-1}, f_{u,t-1}, P^N_{u,t-1})$$

$s_{u,t}$ new state, $f_{u,t-1}$ feedback, $P^N_{u,t-1}$ recommendation list

2. Recommendation list generation $\mathbb{G}$

$$P^N_{u,t} = \mathbb{G}(s_{u,t}, I)$$

I all items(actions)

3. User feedback $\mathbb{F}$

$$f_{u,t} = \mathbb{F}(P^N_{u,t}, I_u)$$

Individual user

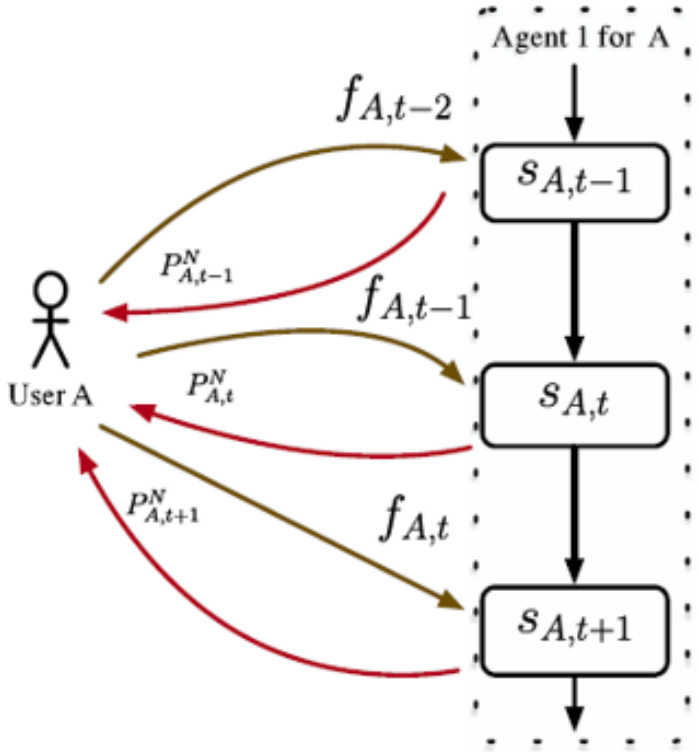


Figure: The recommendation processes for an individual user. At each time t, the red line is the generated top-N recommendation $P^N_{u,t}$ for user u, and the brown line is the feedback $f_{u,t}$ of u for recommendation $P^N_{u,t}$

Evaluation Metrics

Three metrics are used to compare model performance from the aspect of recommendation accuracy, recommendation timeliness and recommendation diversity:

1. Accuracy:

$$hit@N = \frac{\sum_u \frac{1}{|I_u|} \sum_{t=1}^{|I_u|} f_{u,t}}{\#user}$$

Evaluation Metrics

Three metrics are used to compare model performance from the aspect of recommendation accuracy, recommendation timeliness and recommendation diversity:

1. Accuracy:

$$hit@N = \frac{\sum_u \frac{1}{|I_u|} \sum_{t=1}^{|I_u|} f_{u,t}}{\#user}$$

2. Timeliness:

$$NDCG@N = \frac{\sum_u \frac{1}{|I_u|} \sum_{t=1}^{|I_u|} \frac{DCG@N(P_{u,t}^N)}{iDCG@N}}{\#user}$$

Evaluation Metrics

Three metrics are used to compare model performance from the aspect of recommendation accuracy, recommendation timeliness and recommendation diversity:

1. Accuracy:

$$hit@N = \frac{\sum_u \frac{1}{|I_u|} \sum_{t=1}^{|I_u|} f_{u,t}}{\#user}$$

2. Timeliness:

$$NDCG@N = \frac{\sum_u \frac{1}{|I_u|} \sum_{t=1}^{|I_u|} \frac{DCG@N(P_{u,t}^N)}{iDCG@N}}{\#user}$$

3. Diversity:

$$H_u = \frac{1}{m(m-1)} \sum_{u=1}^{m} \sum_{u \neq u'} [1 - \frac{Q_{uu'}}{N}]$$

Cold-start and warm-start

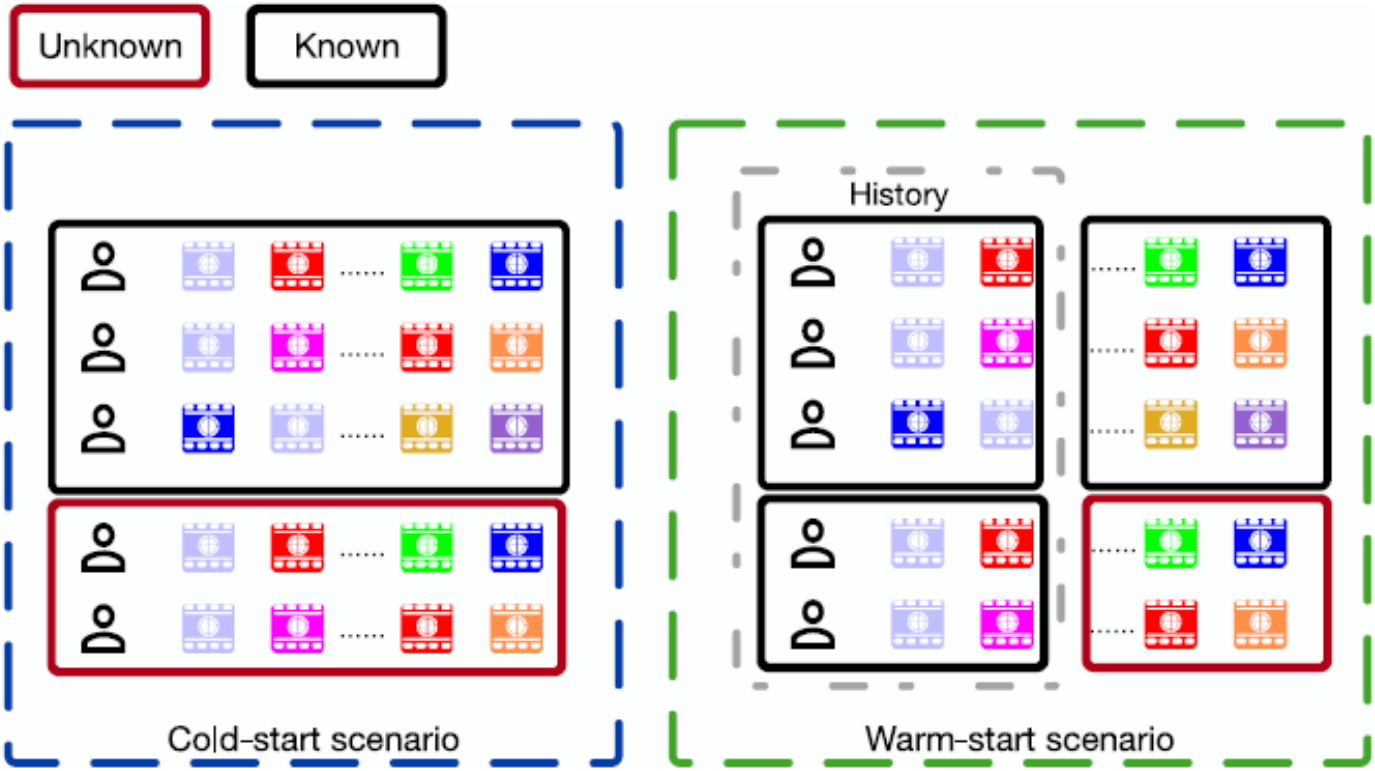


Figure: Cold-start and warm start scenarios. The objective of recommendation is to find as many items as possible in the unknown part on basis of the known items.

Cold-start model

1. We apply recurrent neural network (RNN) as the basic framework since the prediction made by RNN for each step correlates to the input of the current step and the RNN state of the last step, which fits well with the nature of MDP.
2. Major components of RNN: input layer, recurrent layer, output layer.
3. For input layer, learnable embedding is usually fed into the NN since the dense embedding vector can effectively learn the high-level abstracted information of items.
4. For recurrent layers, we use GRU.
5. For output layers, we adopt a dense layer with Softmax operation to predict the recommendation probabilities of items.

Output layer of RNN in the cold-start model

Let $\pi(i|s_{u,t})$ be the recommendation probabilities of item i given $s_{u,t}$, which is obtained from the RNN via output layer, the generation function is defined as

$$P_{u,t}^{N} = \mathbb{G}(s_{u,t}, I) = \text{TopN}_{i \in I} \left(\pi(i|s_{u,t}) \times m_{u,i}^{t} \right)$$

where $m_{u,i}^{t}$ is the element for item i's masking vector m_{u}^{t}, which is either 0 or 1 that indicates whether the item had been selected by the agent or user before.

Output layer of RNN in the cold-start model

The recommendation probability $\pi(i|s_{u,t})$ for item i at time t can be yielded as

$$\pi(i|s_{u,t}) = \text{Softmax}(o^i_{u,t}) = \text{Softmax}(\hat{\sigma}(W_s s_{u,t} + b_s))$$

To make the notation more clear, here $s_{u,t}$ is a l-dimensional vector obtained from the recurrent layer and $o^i_{u,t}$ is the ith element in the vector $o_{u,t}$. $\hat{\sigma}$ is the Relu activation and $W_s \in \mathcal{R}^{M \times l}$ and $b_s \in \mathcal{R}^M$ are parameter matrix and bias respectively. Specifically, the Softmax operator here is as follows:

$$\text{Softmax}(o^i_{u,t}) = \frac{\exp^{o^i_{u,t}}}{\sum_{k \in I} \exp^{o^k_{u,t}}}.$$

Recurrent layer of RNN in the cold-start model

The recurrent layer is to generate a new RNN state $s_{u,t}$ based on the previous state $s_{u,t-1}$ and the input of recurrent layer $\hat{a}_{u,t-1}$ (obtained from the input layer) to simulate the state transition process.

$$s_{u,t} = (1 - Z(\chi_{u,t}, s_{u_t-1})) \odot s_{u_t-1} + Z(\chi_{u,t}, s_{u_t-1}) \odot \tilde{S}(\chi_{u,t}, s_{u,t-1})$$

where $\odot$ denotes element-wise product, Z and $\tilde{S}$ are function for update gate and the function to generate candidate state shown as below:

$$Z(\chi_{u,t}, s_{u_t-1}) = \sigma(W_z \chi_{u,t} + U_z s_{u,t-1})$$

$$\tilde{S}(\chi_{u,t}, s_{u,t-1}) = \tanh(W \chi_{u,t} + U(j_{u,t} \odot s_{u,t-1}))$$

$$j_{u,t} = \sigma(W_j \chi_{u,t} + U_j s_{u,t-1}),$$

Here σ is the sigmoid activation, W_z, U_z, W, U, W_j, U_j are just parameter matrices.

Input layer of RNN in the cold-start model

In the input layer, we aim to transform the user feedback and the result of the previous recommendation step into a dense embedding.
The first step: select a representative item $\hat{a}_{u,t-1}$ from the recommendation list. For positive feedback, we directly use the recommendation item as the representative item, and for negative feedback, we use the item with the highest predicted recommendation probability to be the representative item.

$$\hat{a}_{u,t-1} = \text{argmax}_{a \in A_{u,t-1}} \left(\pi\left(a | s_{u,t-1} \times m_{u,i}^{t-1}\right) \right)$$

where $A_{u,t-1}$ is an auxiliary set for different situations of feedback.

Input layer of RNN in the cold-start model

The second step: incorporate the polar of feedback into the embedding of $\hat{a}_{u,t-1}$. We derive it by multiply $\hat{e}(\hat{a}_{u,t})$ (the $\hat{I}$-dimensional item embedding of $\hat{a}_{u,t}$) by the polar of feedback.

Last step: obtain the input of recurrent layer $\chi_{u,t}$ as follows:

$$\chi_{u,t} = E(\hat{a}_{u,t-1}) = \begin{cases} \hat{e}(\hat{a}_{u,t}) & \text{for } f_{u,t} > 0, \\ -\hat{e}(\hat{a}_{u,t}) & \text{otherwise.} \end{cases}$$

Cold-start model wrap-up

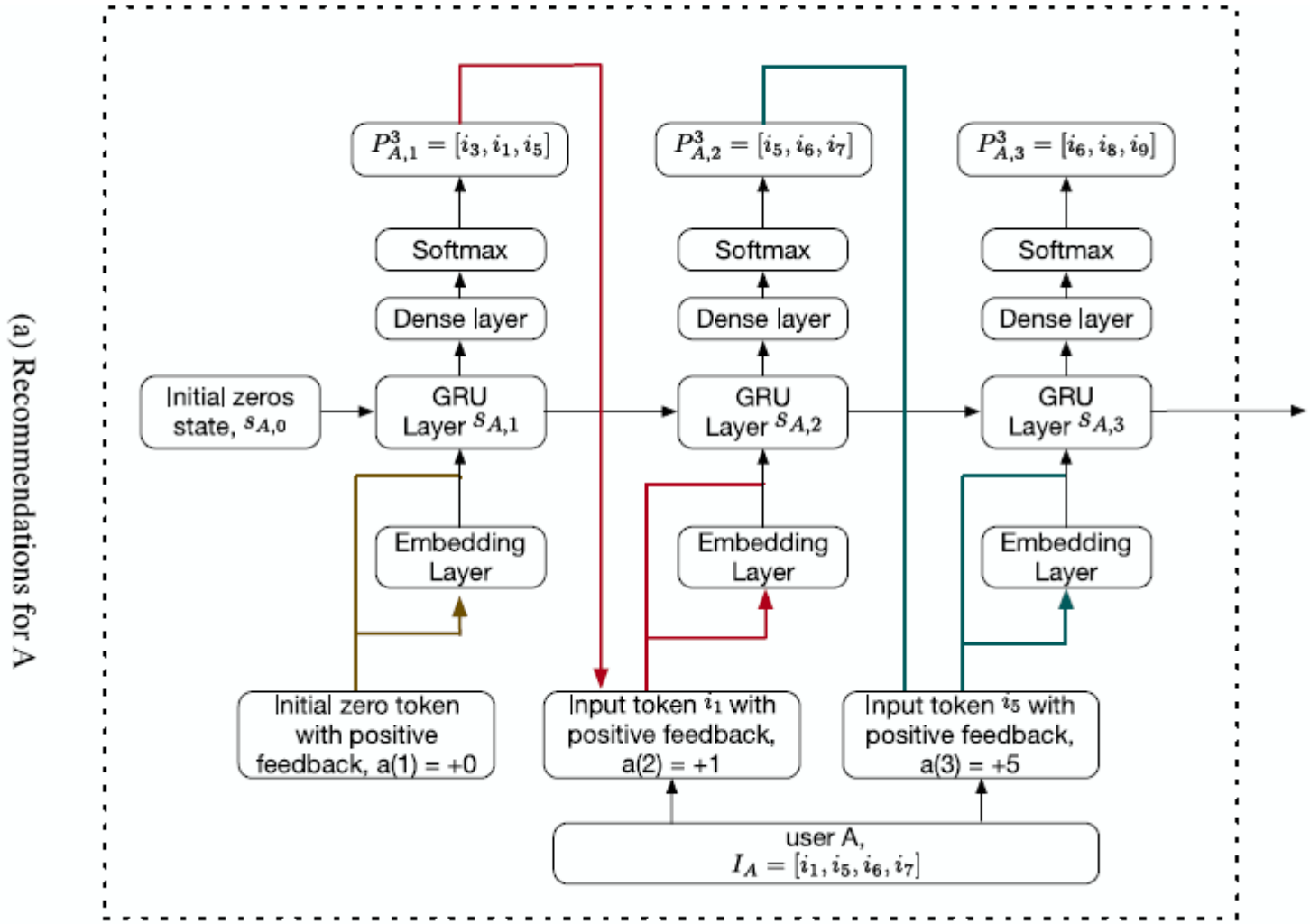


Figure: A cold-start model.

Warm-start model

We propose a more general recommendation model based on our cold-start model to incorporate historical information for the warm-start recommendation. It shares the same RNN framework, and the input/output layers are the same as the cold-start model. We only modify the recurrent layer. The key issues are:

1. how to represent the historical items,
2. how to balance the effects between the representation of historical items and the evolving state.

Warm-start model

First step: We represent the historical items by aggregate different numbers of items to obtain fixed-size features h_u by summing up the embeddings of all historical items to get the aggregation result:

$$h_u = \tanh(\sum_{i \in \tilde{I}_u} e(i)),$$

where $e(i)$ is the $\hat{I}$-dimensional item embedding to denote item i which needs to be learned and is different from $\hat{e}_i$ introduced before.

Warm-start model

Second step: Balance the effects between the representation of historical items h_u and the GRU state $s_{u,t}$. A new state involving both sides is denoted as follows:

$$\hat{s}_{u,t} = d_{u,t} \odot s_{u,t} + (1 - d_{u,t}) \odot h_u,$$

where the balance ratio $d_{u,t}$ is predicted by an additional gate operation:

$$d_{u,t} = \sigma(W_d h_u + U_d s_{u,t}),$$

where W_d and U_d are parameter matrices. The final item selection probability is obtained from $\hat{s}_{u,t}$:

$$\pi(i|\hat{s}_{u,t}) = \text{Softmax}(\hat{o}^i_{u,t}) = \text{Softmax}(\hat{\sigma}(W_s \hat{s}_{u,t} + b_s)).$$

Warm-start model wrap-up

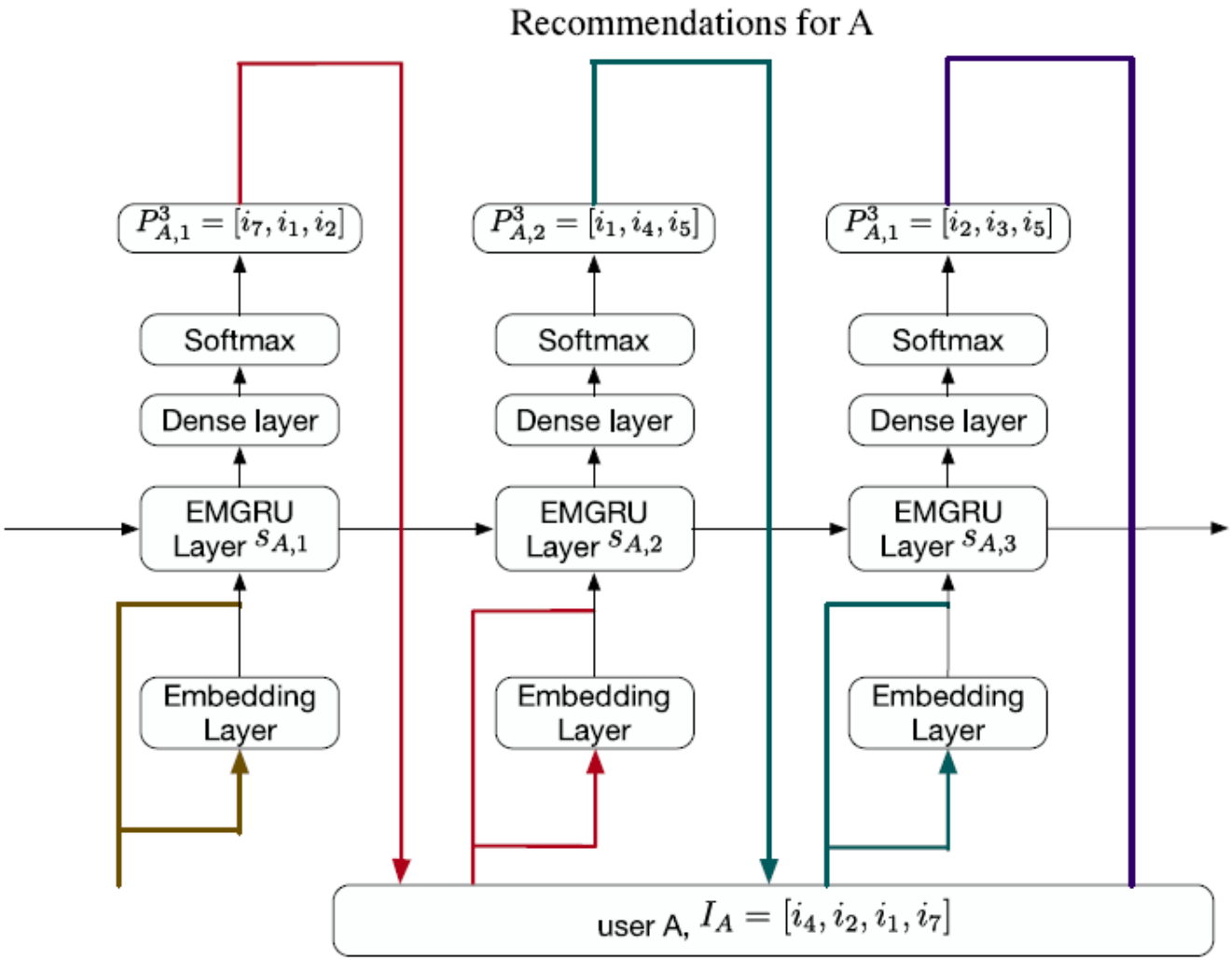


Figure: A warm-start model.

Reinforcement learning

In this paper, the parameter θ is learned via the episode E_u for each user u, and E_u means a complete interaction process for user u obtained from the agent with current parameter.
Formally, an episode E_u includes immediate reward $V_{u,t}$, state $s_{u,t}$ and action $\hat{a}_{u,t}$ for each time t, which can be denoted as follows:

$$E_u = [s_{u,1}, \hat{a}_{u,1}, f_{u,1}, V_{u,1}, \cdots, s_{u,M}, \hat{a}_{u,M}, f_{u,M}, V_{u,M}]$$

where $s_{u,t}, \hat{a}_{u,t}, f_{u,t}$ are obtained with the procedure introduced before, and $V_{u,t}$ can be calculated by

$$V_{u,t} = \begin{cases} 1.0 & f_{u,t} > 0, \\ -0.2 & otherwise. \end{cases}$$

The values 1.0 and -0.2 are determined empirically.

Reinforcement learning

To maximize the expected cost J, each action $\hat{a}_{u,t}$ has both an immediate reward $V_{u,t}$ and a long-term reward $R_{u,t}$ denoted as:

$$R_{u,t} = \sum_{k=0}^{M-k} \gamma^k V_{u,t+k},$$

and the objective function $J = \mathbb{E}_{s_{u,1}, a_{u,1}, \ldots}[R_{u,t}]$. According to REINFORCE, the agent parameter θ can be optimized via gradient ascent as follows:

$$\theta = \theta + \eta \nabla_\theta J$$

where the gradient $\nabla_\theta J(\theta)$ is

$$\nabla_\theta J(\theta) = \sum_{t=1}^{M} \gamma^{t-1} R_{u,t} \nabla_\theta \log \pi(\hat{a}_t | s_{u,t}).$$

Reinforcement learning

Algorithm 1 :Reinforcement Learning

Require: training set U, learning rate η, maximum size B, initialized θ.

while not stop **do**
  for u, I_u,$\hat{I}_u$ **in** U **do**
    $count \leftarrow 0$; $s_{u,0} \leftarrow$ zeros; $\hat{a}_{u,0} \leftarrow 0$;
    if $\hat{I}_u$ is $\emptyset$ **then**
      $m_u \leftarrow$ ones.
    else
      $m_u[i] \leftarrow 0$ for each item i in $\hat{I}_u$.
    end if
    while $count < |I_u + \hat{I}_u|$ **do**
      $E, m_u \leftarrow$ GenerateEpisode(I_u,$\hat{I}_u$,$s_{u,0}$,$\hat{a}_{u,0}$,$f_{u,0}$,m_u, B)
      $[s_{u,1}, \hat{a}_{u,1}, f_{u,1}, V_{u,1}, \ldots, s_{u,B}, \hat{a}_{u,B}, f_{u,B}, V_{u,B}] = E$
      Obtain $R_{u,t}$ according to Eq. (25)
      $\theta \leftarrow \theta + \eta \frac{1}{B} \sum_{t=1}^{B} \gamma^{t-1} R_{u,t} \nabla_\theta \log \pi(\hat{a}_t | s_{u,t})$
      $count += B$; $s_{u,0} \leftarrow s_{u,B}$, $\hat{a}_{u,0} \leftarrow \hat{a}_{u,B}$, $f_{u,0} \leftarrow f_{u,B}$
    end while
  end for
end while

Data

Three (offline) datasets are used in training the recommendation agent:

1. **MovieLens 100K**: A commonly used benchmark recommendation system dataset containing 100K movie ratings from $m = 943$ users and $I = 1682$ items.

Data

Three (offline) datasets are used in training the recommendation agent:

1. **MovieLens 100K**: A commonly used benchmark recommendation system dataset containing 100K movie ratings from $m = 943$ users and $I = 1682$ items.
2. **MovieLens 1M**: A expanded version of the **MovieLens 100K** dataset, containing 1M movie ratings from $m = 6040$ users and $I = 3900$ items.

Data

Three (offline) datasets are used in training the recommendation agent:

1. **MovieLens 100K**: A commonly used benchmark recommendation system dataset containing 100K movie ratings from $m = 943$ users and $I = 1682$ items.
2. **MovieLens 1M**: A expanded version of the **MovieLens 100K** dataset, containing 1M movie ratings from $m = 6040$ users and $I = 3900$ items.
3. **Steam**: A dataset of video game ratings, containing $m = 6818$ users and $I = 8023$ items. *This dataset is more sparse than the movie dataset. (Large action space)*

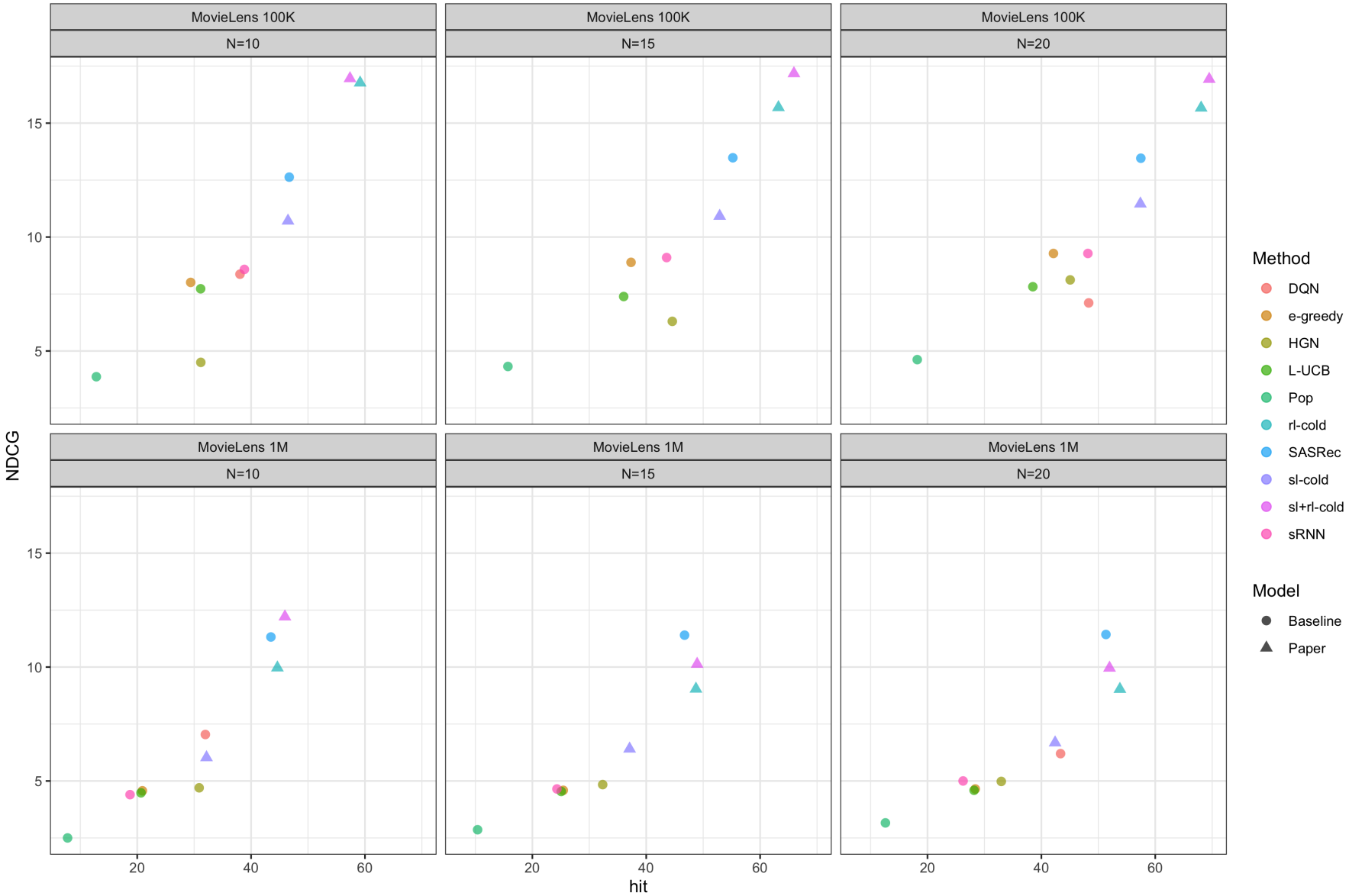
Baseline Methods

- **Pop**: Always recommend most popular items.
- **Linear-UCB** [7]: Linear upper confidence bound algorithm in a context-free multi-armed bandit setting.
- ϵ-**greedy** [7]: Choose optimal item with $p = 1 - \epsilon$ and explore a random item with $p = \epsilon$, other settings are similar to Linear UCB.
- **session-based RNN** [2] (**sRNN**): RNN-based model for next-item prediction (short-term).
- **BPR-MF** [5]: A matrix-factorization based Bayesian personalized-ranking algorithm (popular for Top-N recommendation).

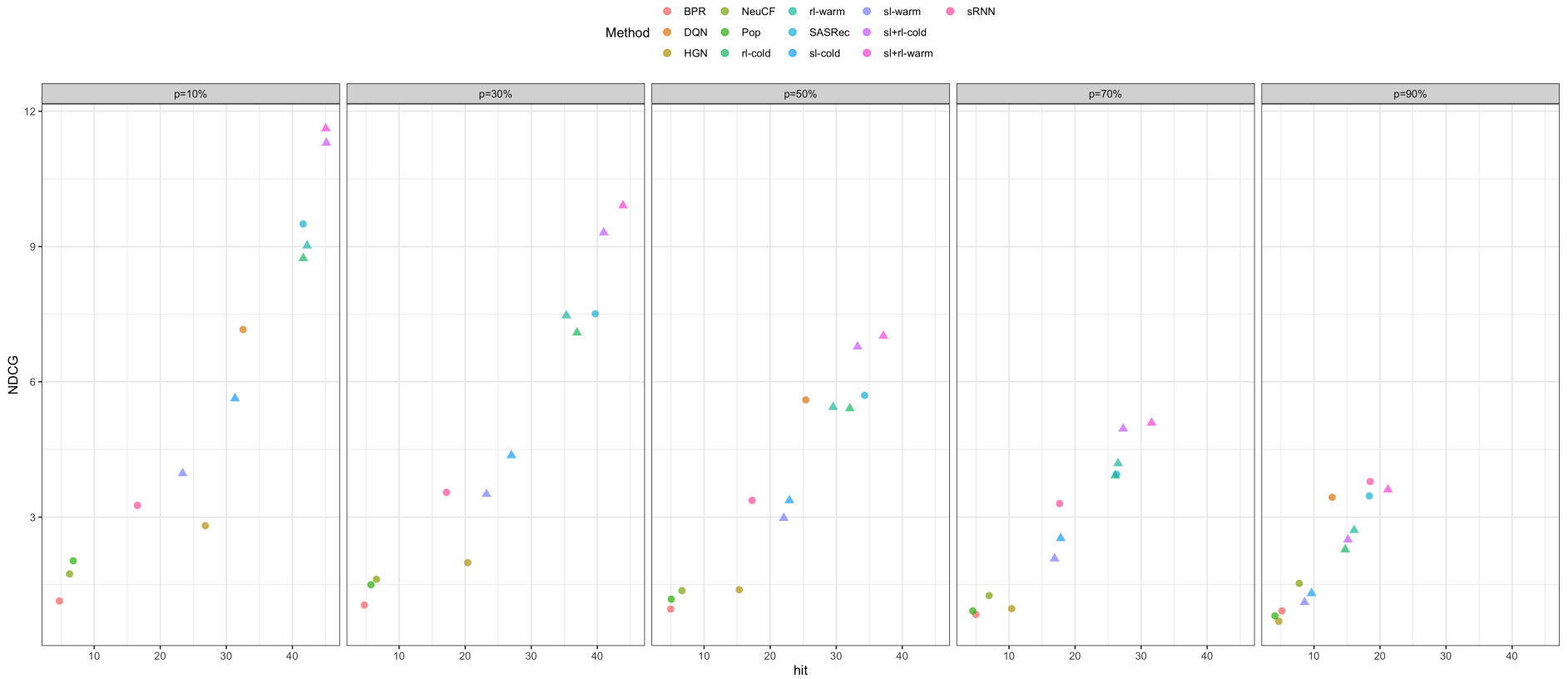
Baseline Methods (Continued)

- **NeuCF** [1]: Neural network-based Collaborative Filtering. (State-of-the-art for Top-N recommendation)
- **DQN** [6]: Deep-RL based Deep Q-learning Network (value-based learning).
- **SASRec** [3]: Self-attention (on user's history) based RNN-model.
- **HGN** [4]: hierarchical gating network learning group-level representations (learn a sequence of items to predict future items).

Cold-start Result



Warm-start Result



Sparse Dataset Result

Table 5
Performance comparisons on Steam.

Method	Steam					
	$p = 0\%$			$p = 50\%$		
	hit@10	N@10	D@10	hit@10	N@10	D@10
sRNN	18.71%	2.20%	98.60%	9.83%	1.02%	98.44%
SASRec	20.06%	2.95%	85.82%	17.94%	2.08%	90.39%
HGN	19.86%	2.67%	73.04%	11.59%	1.14%	88.30%
DQN	12.28%	2.29%	72.03%	9.79%	1.35%	71.47%
sl+rl-cold	25.57%	5.59%	95.12%	16.52%	2.66%	95.51%
sl+rl-warm	–	–	–	20.99%	3.21%	98.70%

Figure: Performance on the sparse dataset **Steam**. Diversity measure H_u is denoted as D. Only $N = 10$ scenario is considered here.

Ablation Study

Table 7
Ablation studies on MovieLens 100K, 1M and Steam under warm-start recommendation scenario ($p = 50\%$).

Model	Components				100K		1M		Steam	
	SL	RL	FI	EMGRU	Hit@10	N@10	Hit@10	N@10	Hit@10	N@10
Full	√	√	√	√	46.99%	9.64%	37.14%	7.02%	20.99%	3.21%
Model-1	√	×	√	√	33.82%	5.37%	22.07%	2.98%	12.52%	1.16%
Model-2	×	√	√	√	42.03%	9.64%	29.53%	5.44%	7.59%	1.35%
Model-3	√	√	×	√	43.46%	9.15%	35.20%	6.39%	20.42%	3.14%
Model-4	√	√	√	×	45.93%	9.38%	33.24%	6.78%	16.52%	2.66%
Model-5	√	√	×	×	35.79%	6.94%	33.84%	5.98%	17.19%	2.65%
Model-6	√	×	×	×	24.23%	3.43%	14.93%	1.81%	10.36%	0.98%

Figure: Ablation study with warm-start scenario. 4 model components are considered: 1) (SL) supervised-learning as pre-training; 2) (RL) reinforcement learning; 3) (FI) combining both user's feedback and representative item from previous Top-N list as input; 4) (EMGRU) using the EMGRU cell for embedding the warm-start history.

Long-term Performance

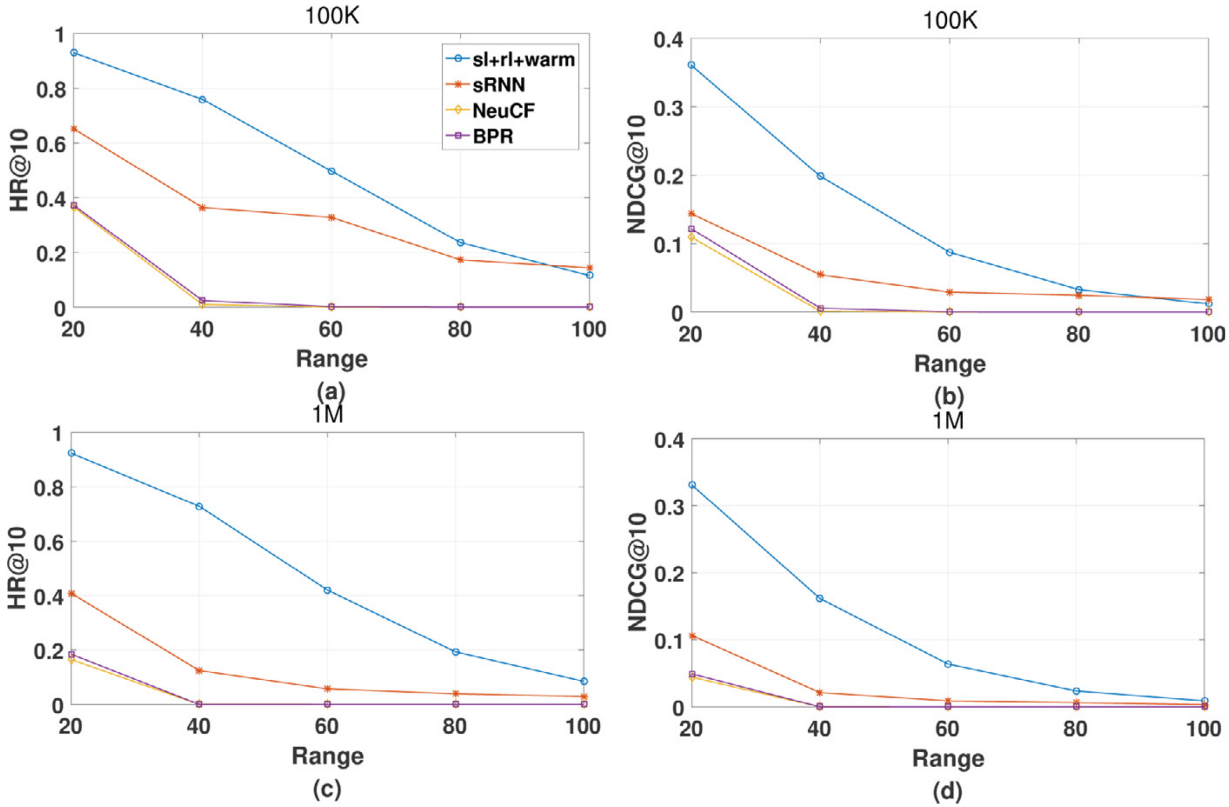


Fig. 10. Performance on different stages with $p = 50\%$.

Figure: Long-term performance evaluation. All users in the test set with above-average l_u are considered. All interactions with the users are broken down into 5 stages based on percentiles: $0 \sim 20\%, 20 \sim 40\%$, $40 \sim 60\%$ $60 \sim 80\%$, $80 \sim 100\%$.

Recommendation Behavior

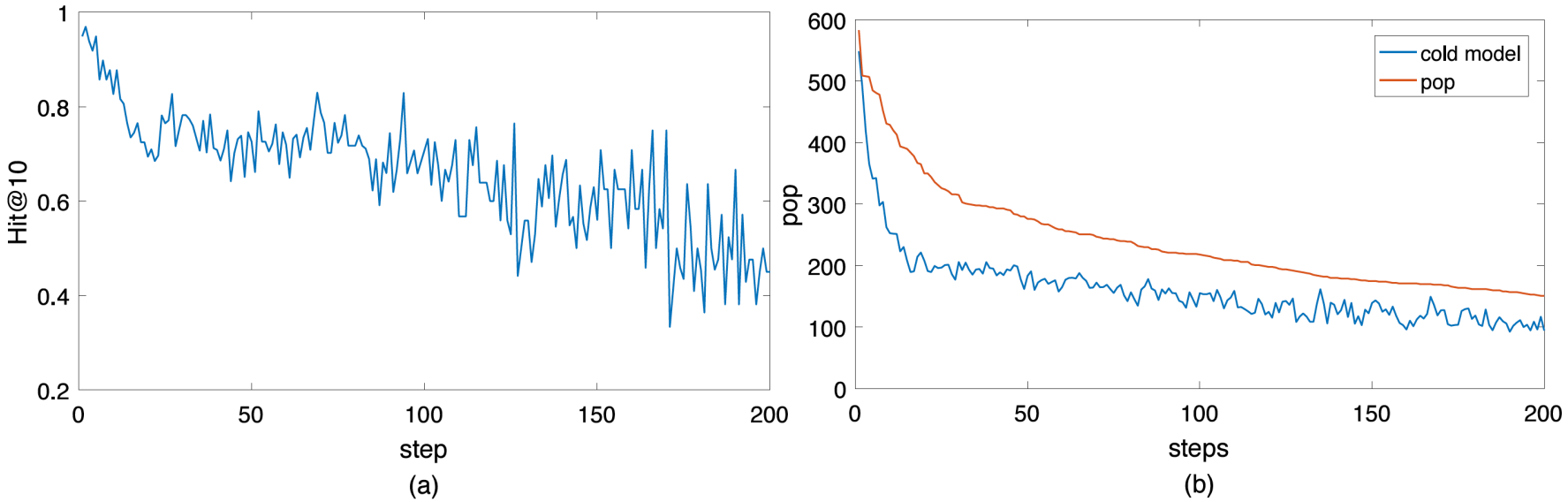


Figure: Recommendation results for cold-start on **MovieLens 100K**. (a):the average of *Hit@10* for each step t of test users; (b): the average of popularity of selected items for each step t of test users. Blue lines: the results on cold-start model. Orange line denotes the distribution of popularity of items, where the point at step t represents the popularity of the t-th popular item.

Project Goal

Experiment on different value-based/policy-based RL (vs. non-RL) methods to compare their efficacy under different recommendation scenarios:

- cold-start vs. warm-start
- small action space vs. large action space

Question?

References I

[1] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[2] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[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] Chen Ma, Peng Kang, and Xue Liu. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 825–833, 2019.

References II

[5] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[6] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1040–1048, 2018.

[7] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1411–1420, 2013.