Exact-K Recommendation via Maximal Clique Optimization

Yu Gong[1*], Yu Zhu[1*], Lu Duan[2], Qingwen Liu[1], Ziyu Guan[3*], Fei Sun[1], Wenwu Ou[1], Kenny Q. Zhu[4]
[1] Alibaba Group, China [2] Zhejiang Cainiao Supply Chain Management Co., Ltd, China
[3] Xidian University, China [4] Shanghai Jiao Tong University, China
{gongyu.gy,zy143829,xiangsheng.lqw,ofey.sf}@alibaba-inc.com,duanlu.dl@cainiao.com
zyguan@xidian.edu.cn,santong.oww@taobao.com,kzhu@cs.sjtu.edu.cn

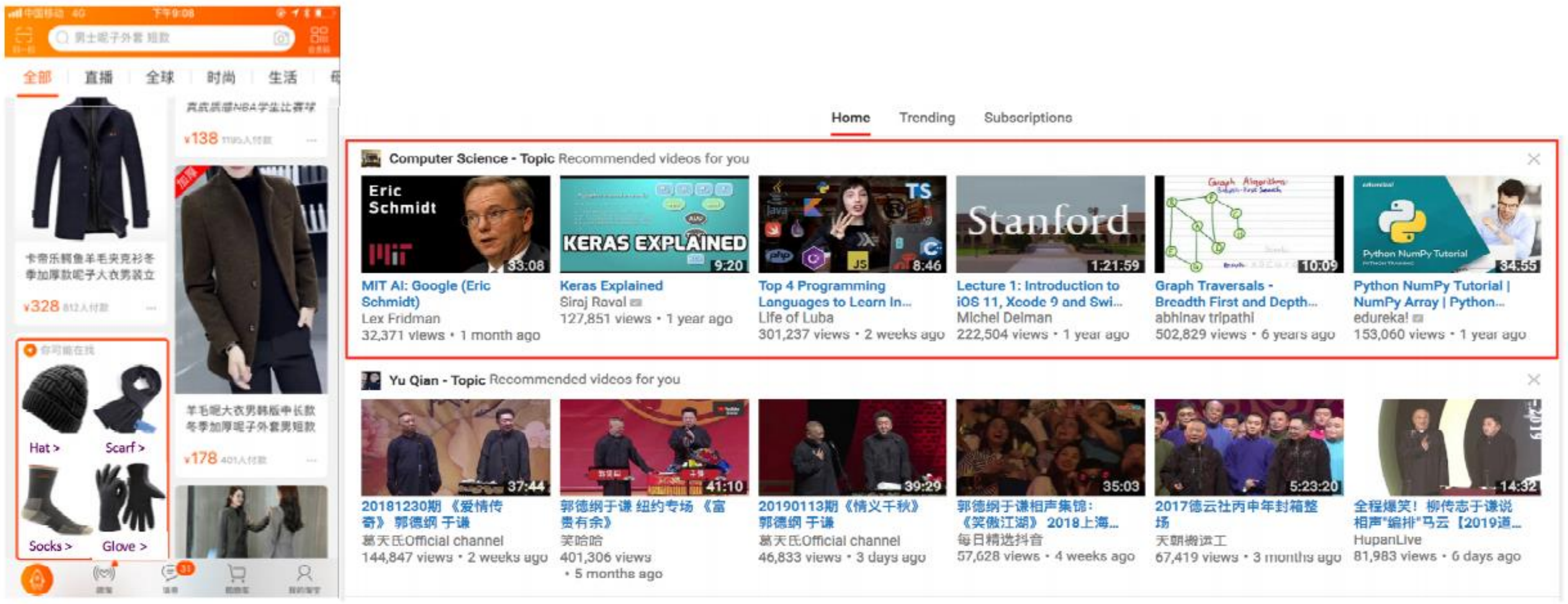
KDD 2019

张峻伟
2019/8/23

Introduction

- **top-K recommendation**: a **ranking optimization problem** which assumes that "better" items should be put into top positions
- **exact-*K* recommendation**: a (constrained) **combinatorial optimization problem** which tries to maximize the joint probability of the set of items in a card.

interact with each other



Hat >
Scarf >
Socks >
Glove >
Home
Trending
Subscriptions
Computer Science - Topic Recommended videos for you
Yu Qian - Topic Recommended videos for you

(a) Taobao RS (b) YouTube RS

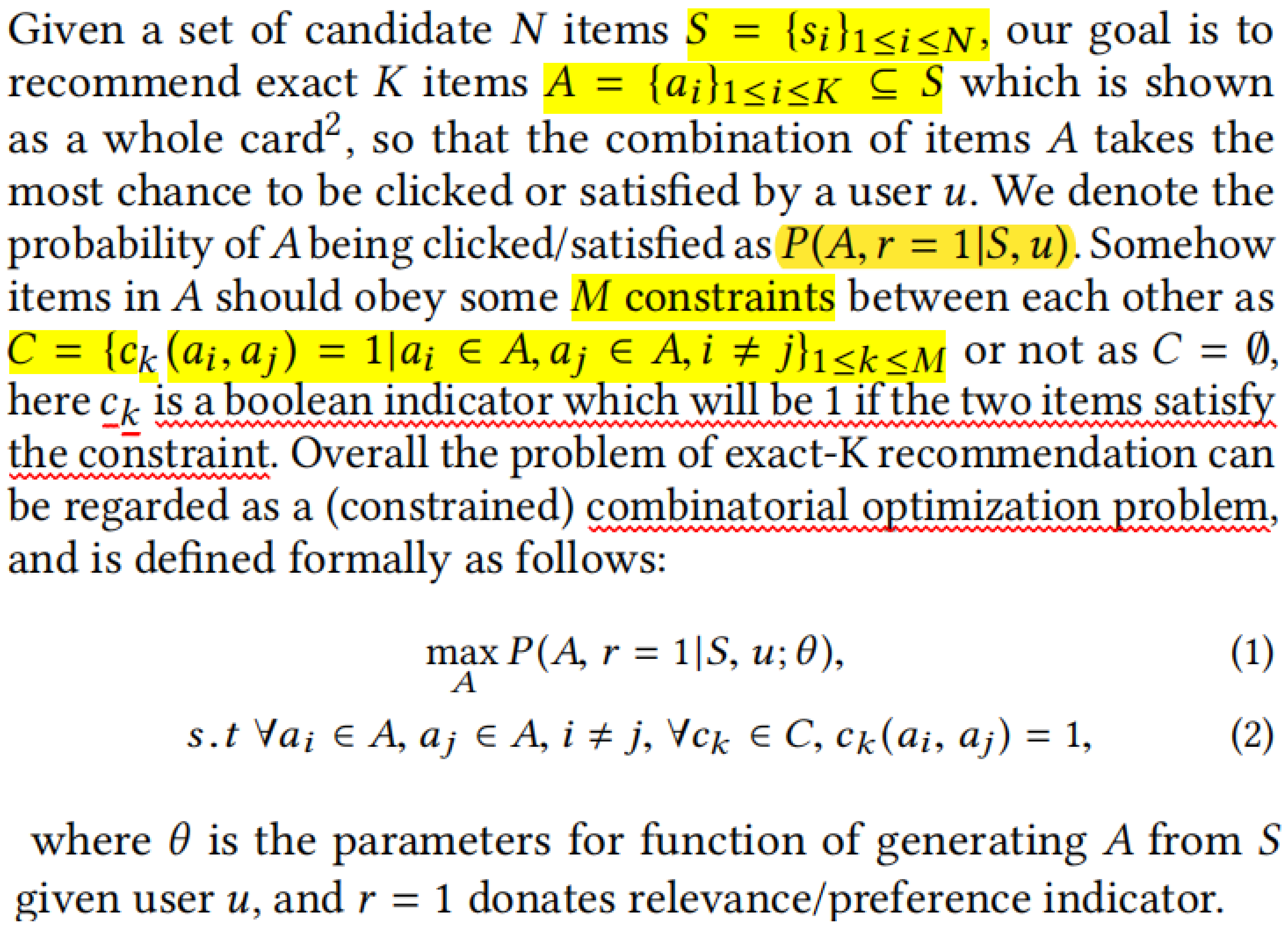
Figure 1: Show cases for exact-K recommendation in Taobao and YouTube.

Contributions

- We formally **define the exact-K recommendation problem** and innovatively reduce it to a Maximal Clique Optimization problem based on graph.（基于图的最大团优化问题）
- We propose ***Graph Attention Networks* (GAttN)** with an **Encoder-Decoder framework** which can end-to-end learn the joint distribution of K items and generate an optimal card containing K items. We adopt well-designed ***Reinforcement Learning from Demonstrations* (RLfD)** which combines the advantages in **behavior cloning and reinforcement learning**, making it sufficient-and-efficient to train GAttN.
- We conduct extensive experiments on three datasets.

Problem Definition

- **Exact-K recommendation**



Given a set of candidate N items $S = \{s_i\}_{1 \le i \le N}$, our goal is to recommend exact K items $A = \{a_i\}_{1 \le i \le K} \subseteq S$ which is shown as a whole card[2], so that the combination of items A takes the most chance to be clicked or satisfied by a user u. We denote the probability of A being clicked/satisfied as $P(A, r = 1|S, u)$. Somehow items in A should obey some M constraints between each other as $C = \{c_k(a_i, a_j) = 1 | a_i \in A, a_j \in A, i \neq j\}_{1 \le k \le M}$ or not as $C = \emptyset$, here c_k is a boolean indicator which will be 1 if the two items satisfy the constraint. Overall the problem of exact-K recommendation can be regarded as a (constrained) combinatorial optimization problem, and is defined formally as follows:

$$\max_{A} P(A, r = 1|S, u; \theta), \tag{1}$$

$$s.t\ \forall a_i \in A, a_j \in A, i \neq j, \forall c_k \in C, c_k(a_i, a_j) = 1, \tag{2}$$

where θ is the parameters for function of generating A from S given user u, and $r = 1$ donates relevance/preference indicator.

In another perspective, we construct a graph $\mathbb{G}(\mathcal{N}, \mathcal{E})$ containing N nodes, in which each node n_i in $\mathcal{N}$ represents an item s_i in candidate item set S, each edge e_{ij} in $\mathcal{E}$ connecting nodes (n_i, n_j) represents that items s_i and s_j should satisfy the constraints or there is no constraint (a.k.a $\mathbb{G}$ is now a complete graph), i.e $\forall c_k \in C, c_k(s_i, s_j) = 1$ or $C = \emptyset$, so it is an undirected graph here.

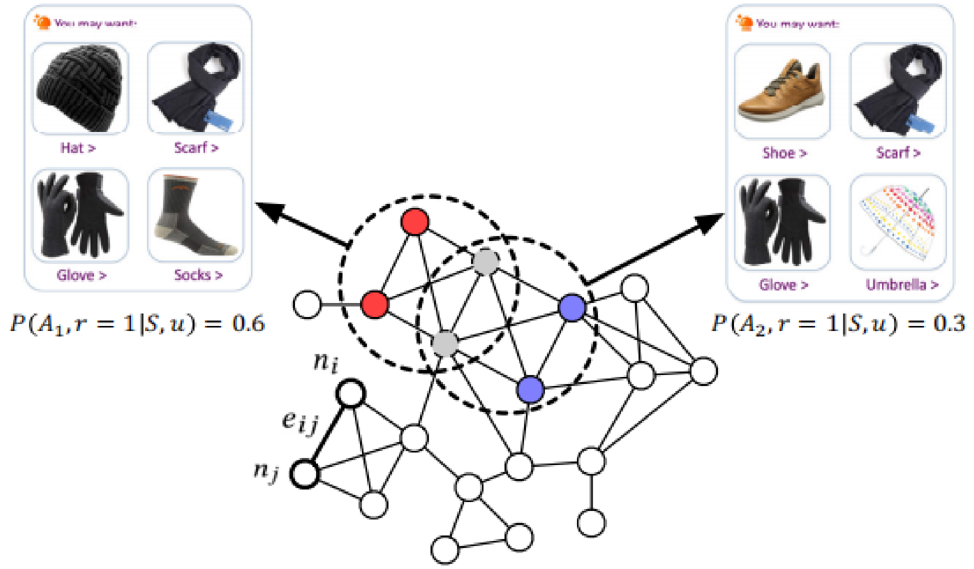
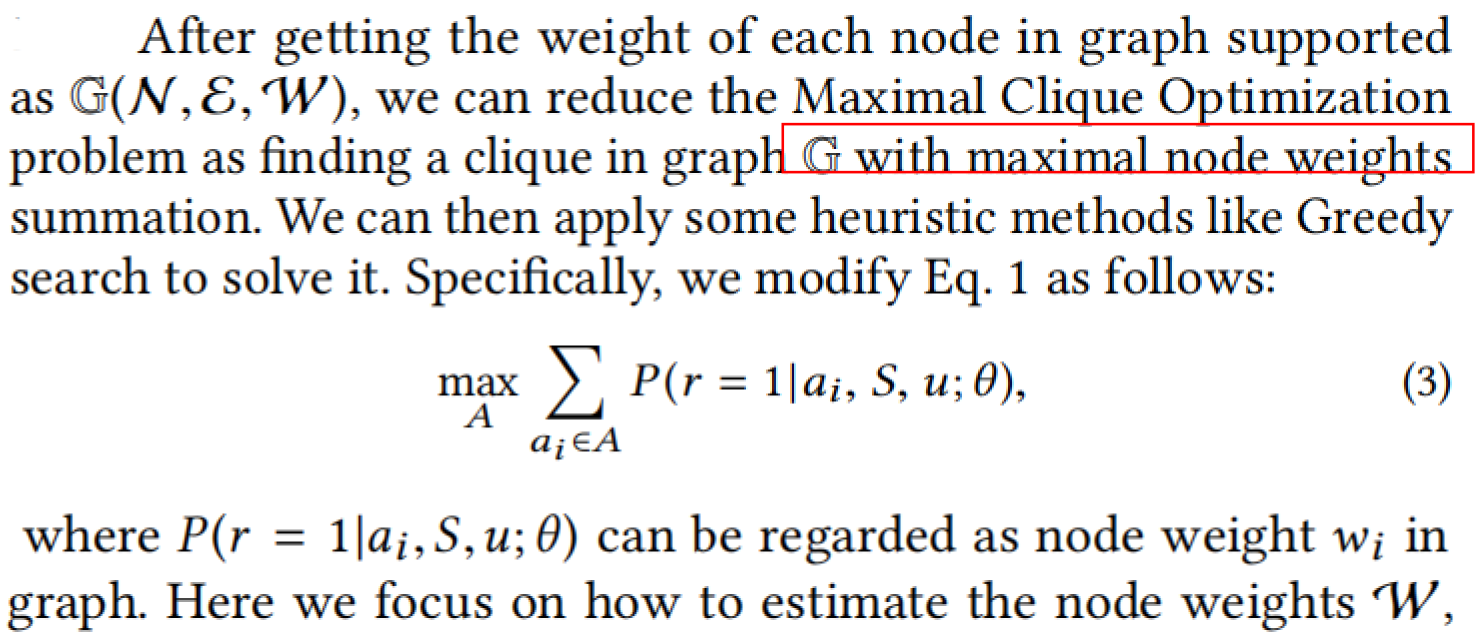


Figure 2: Illustration for a specific graph $\mathbb{G}$ with $N = 20$ and $K = 4$. We show two different cliques (red and blue) in graph and the corresponding cards (A_1 and A_2) each with 4 items. We suppose that $P(A_1, r = 1|S, u) > P(A_2, r = 1|S, u)$ means that card A_1 takes more chance to be satisfied than card A_2 given user u and candidate item set S.

● Naive Node-Weight Estimation Method

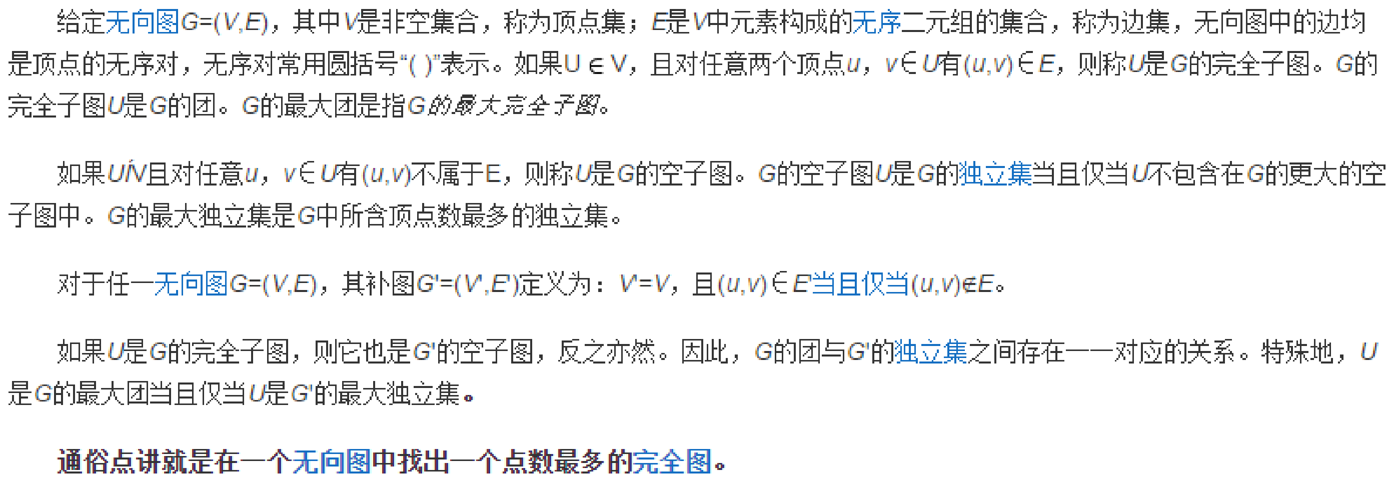


After getting the weight of each node in graph supported as $\mathbb{G}(\mathcal{N}, \mathcal{E}, \mathcal{W})$, we can reduce the Maximal Clique Optimization problem as finding a clique in graph $\boxed{\mathbb{G}\text{ with maximal node weights}}$ summation. We can then apply some heuristic methods like Greedy search to solve it. Specifically, we modify Eq. 1 as follows:

$$\max_{A} \sum_{a_i \in A} P(r = 1 | a_i, S, u; \theta), \tag{3}$$

where $P(r = 1|a_i, S, u; \theta)$ can be regarded as node weight w_i in graph. Here we focus on how to estimate the node weights $\mathcal{W}$,

【最大团问题】



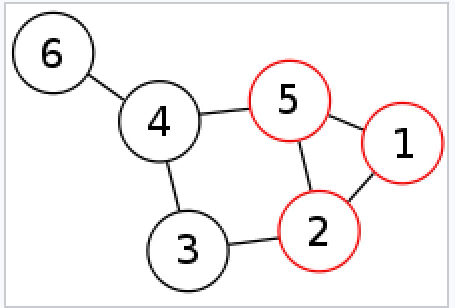
给定无向图G=(V,E)，其中V是非空集合，称为顶点集；E是V中元素构成的无序二元组的集合，称为边集，无向图中的边均是顶点的无序对，无序对常用圆括号“( )”表示。如果U ∈ V，且对任意两个顶点u，v∈U有(u,v)∈E，则称U是G的完全子图。G的完全子图U是G的团。G的最大团是指G的最大完全子图。

如果UÍV且对任意u，v∈U有(u,v)不属于E，则称U是G的空子图。G的空子图U是G的独立集当且仅当U不包含在G的更大的空子图中。G的最大独立集是G中所含顶点数最多的独立集。

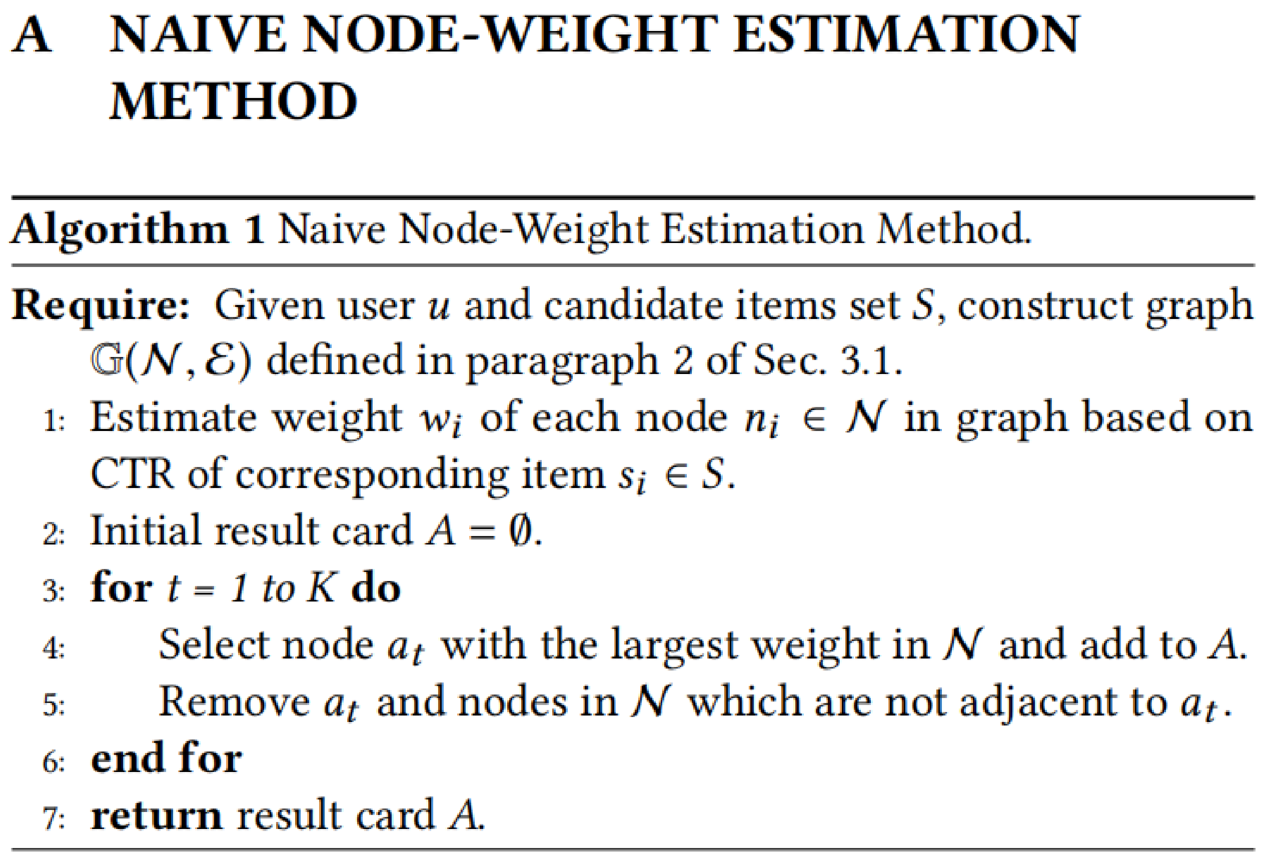
对于任一无向图G=(V,E)，其补图G'=(V',E')定义为：V'=V，且(u,v)∈E'当且仅当(u,v)∉E。

如果U是G的完全子图，则它也是G'的空子图，反之亦然。因此，G的团与G'的独立集之间存在一一对应的关系。特殊地，U是G的最大团当且仅当U是G'的最大独立集。

通俗点讲就是在一个无向图中找出一个点数最多的完全图。



The graph shown has one maximum clique, the triangle {1,2,5}, and four more maximal cliques, the pairs {2,3}, {3,4}, {4,5}, and {4,6}.



A NAIVE NODE-WEIGHT ESTIMATION METHOD

Algorithm 1 Naive Node-Weight Estimation Method.

```
Require: Given user $u$ and candidate items set $S$, construct graph
    $\mathbb{G}(\mathcal{N}, \mathcal{E})$ defined in paragraph 2 of Sec. 3.1.
1: Estimate weight $w_i$ of each node $n_i \in \mathcal{N}$ in graph based on
   CTR of corresponding item $s_i \in S$.
2: Initial result card $A = \emptyset$.
3: for $t$ = 1 to $K$ do
4:     Select node $a_t$ with the largest weight in $\mathcal{N}$ and add to $A$.
5:     Remove $a_t$ and nodes in $\mathcal{N}$ which are not adjacent to $a_t$.
6: end for
7: return result card $A$.
```

weaknesses of such method are obvious:

- CTR estimation for each item is independent;
- Combinational characteristic of the *K* items in a card is not considered;
- Problem objective is not optimized directly but substituted with a reduced heuristic objective which will unfortunately fall into sub-optimal.

Approach

we first propose *Graph Attention Networks* (GAttN) which follows the Encoder-Decoder Pointer framework with Attention mechanism. Then we adopt Reinforcement Learning from Demonstrations (RLfD) which combines the advantages in behavior cloning and reinforcement learning, making it sufficient-and-efficient to train the networks.

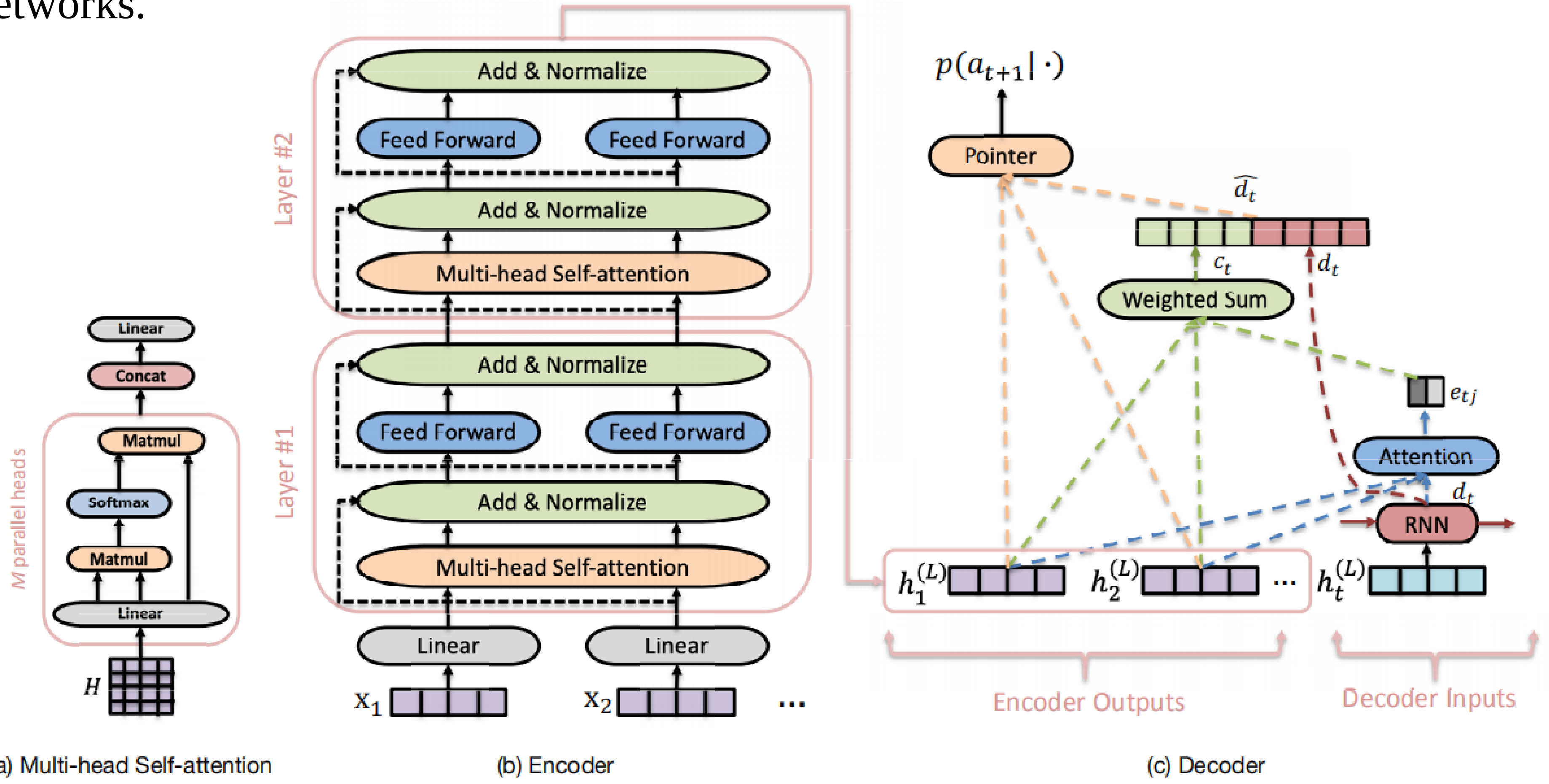
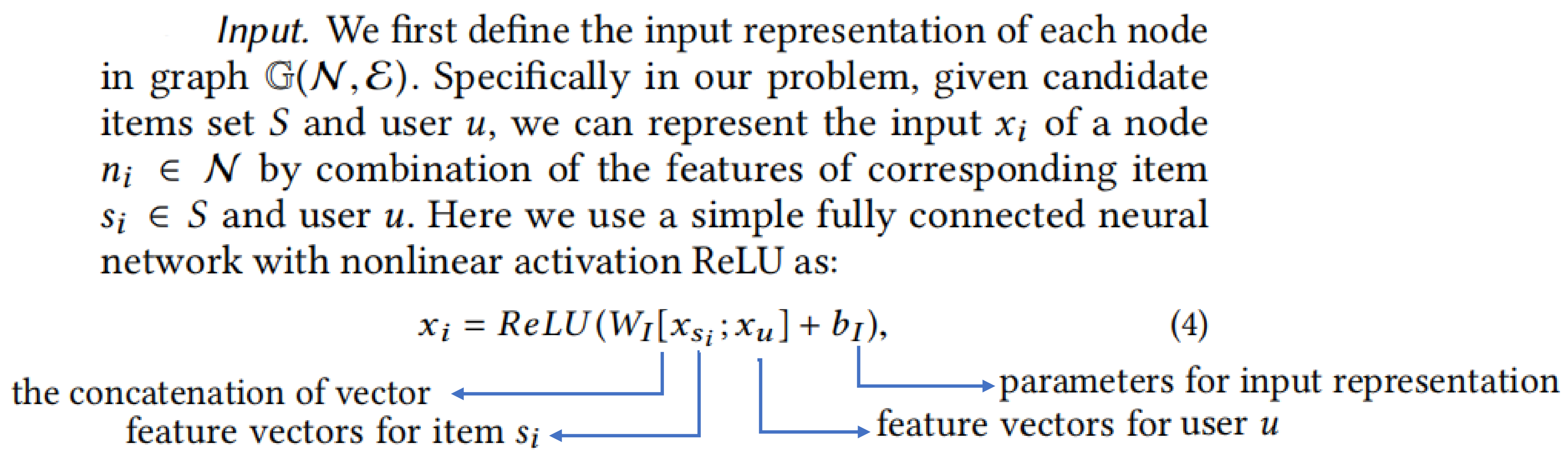


Figure 3: The key modules of Graph Attention Networks (GAttN).

● Graph Attention Networks

☐ Input



Input. We first define the input representation of each node in graph $\mathbb{G}(\mathcal{N}, \mathcal{E})$. Specifically in our problem, given candidate items set S and user u, we can represent the input x_i of a node $n_i \in \mathcal{N}$ by combination of the features of corresponding item $s_i \in S$ and user u. Here we use a simple fully connected neural network with nonlinear activation ReLU as:

$$x_i = ReLU(W_I[x_{s_i}; x_u] + b_I), \tag{4}$$

the concatenation of vector

feature vectors for item s_i

parameters for input representation

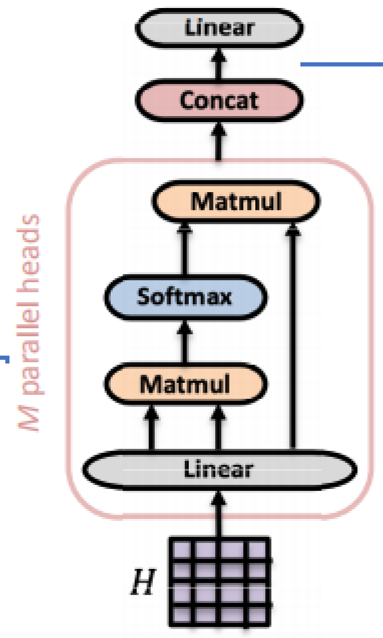
feature vectors for user u

☐ Encoder

Since the order of nodes in a undirected graph is meaningless, any permutation of the inputs results in the same output representations. (RNN-traditional encoder).
The representation for a node should consider the other nodes in graph, as there can exist some underlying structures in graph that nodes may influence between each other. (use self-attention)

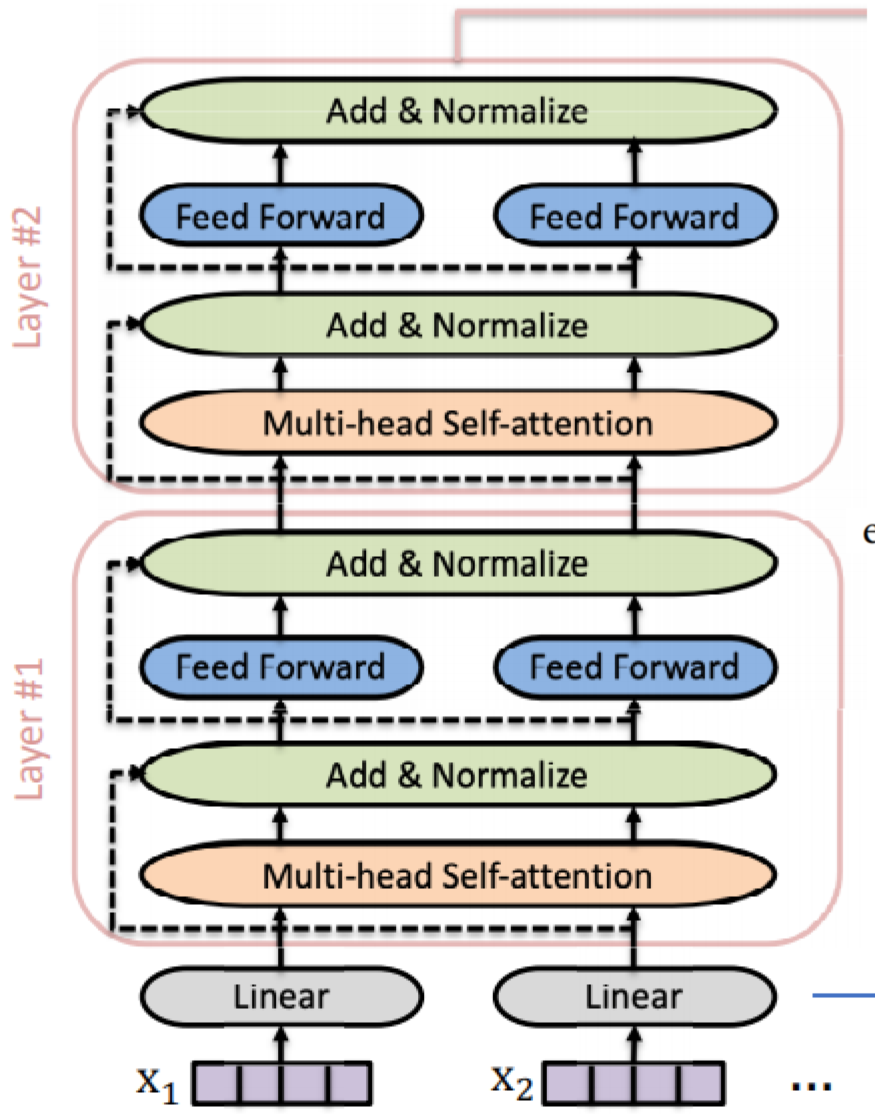
$$\text{SelfAttention}(E) = softmax\left(\frac{EE^T}{\sqrt{d}}\right)E,$$

$$\widehat{H^{(l)}} = [head_1; \ldots ; head_M]W_O,$$
$$\text{where } head_i = \text{SelfAttention}(H^{(l-1)}W_{hi}),$$



(a) Multi-head Self-attention

MHSA



(b) Encoder

embedding outputs of node n_i in FF sub-layers

$$h_i^{(l)} = W_{F2}ReLU(W_{F1}\widehat{h_i^{(l)}} + b_{F1}) + b_{F2},$$

embedding outputs of node n_i in MHSA

$$h_i^{(0)} = W_E x_i + b_E.$$

☐ Decoder

RNN has been widely used to map the encoder embeddings to a correlated output sequence, so does in our proposed framework. our goal is to **optimize P (A|S,u; θ)** (here we omit relevance score of r = 1), it is a **joint probability** and can be decomposed by the chain rule as follows

$$P(A|f(S, u; \theta) = \prod_{i=1}^{K} p(a_i|a_1, \ldots, a_{i-1}, S, u; \theta)$$

represent encoder as $f(S, u; \theta_e)$

$$= \prod_{i=1}^{K} p(a_i|a_1, \ldots, a_{i-1}, f(S, u; \theta_e); \theta_d),$$

$$p(a_i|a_1, \cdots, a_{i-1}, f(S, u; \theta_e); \theta_d) = p(a_i|d_i, f(S, u; \theta_e); \theta_d),$$

state vector

$$d_i = \begin{cases} g(0, 0) & \text{if } i = 1, \\ g(d_{i-1}, a_{i-1}) & \text{otherwise,} \end{cases}$$

where $g(d, a)$ is usually a non-linear function cell in LSTM or GRU that combines the previous state and previous output (embedding of the corresponding node *a* from encoder) in order to produce the current state.

$$p(a_t|d_t, f(S, u; \theta_e); \theta_d) = softmax(u_{tj}),$$ $$u_{tj} = \begin{cases} v_{D2}^T \tanh(W_{D3}\widehat{d_t} + W_{D4}h_j^{(L)}), & \text{otherwise} \\ -\infty, & \text{if node } n_j \text{ should be masked,} \end{cases}$$

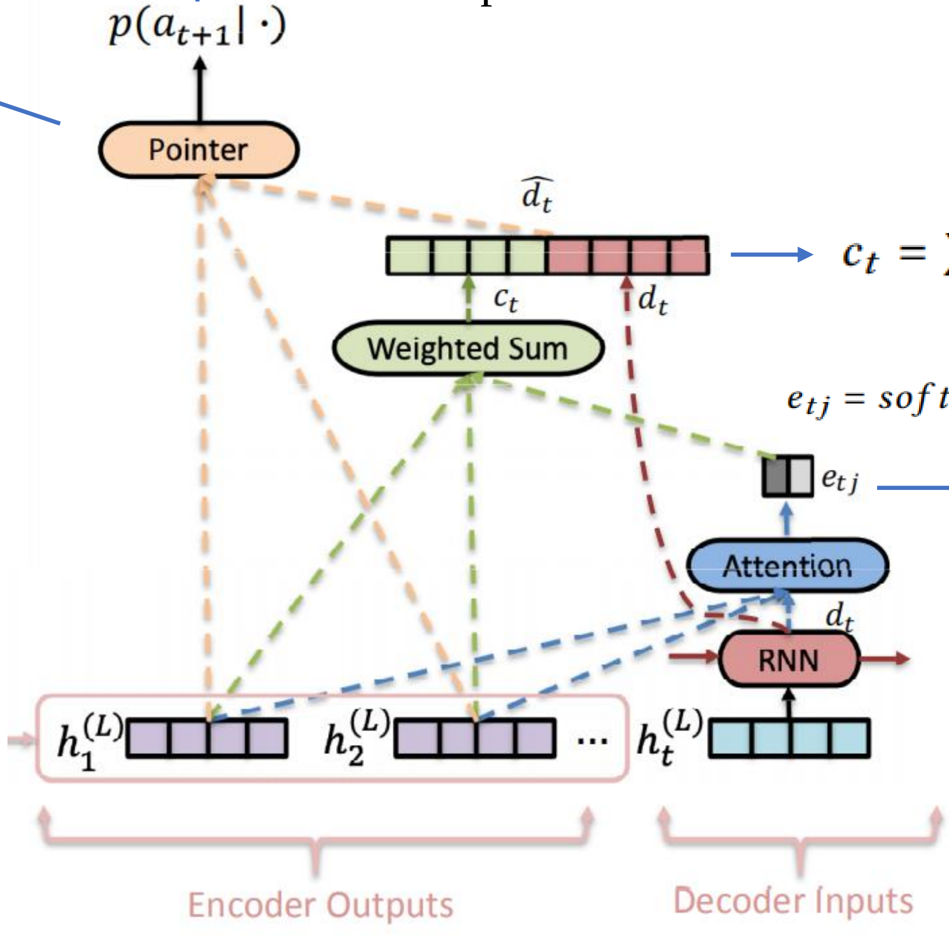
At the period of decoder inference, apply technique of **beam search**

an specific attention mechanism named **pointer**

It allows the decoder to look at the whole input graph G(N, E) at any time and select a member of the input nodes N as the final outputs *A*

a specific attentive pointer with **masking scheme** to generate feasible clique from graph.

(1) nodes already decoded are not allowed to be pointed, (2) nodes will be masked if disobey the clique constraint rule among the decoded subgraph. And we compute the attention values as follows



(c) Decoder

$$c_t = \sum_{j=1}^{N} e_{tj} h_j^{(L)}, \quad \widehat{d_t} = [d_t; c_t],$$

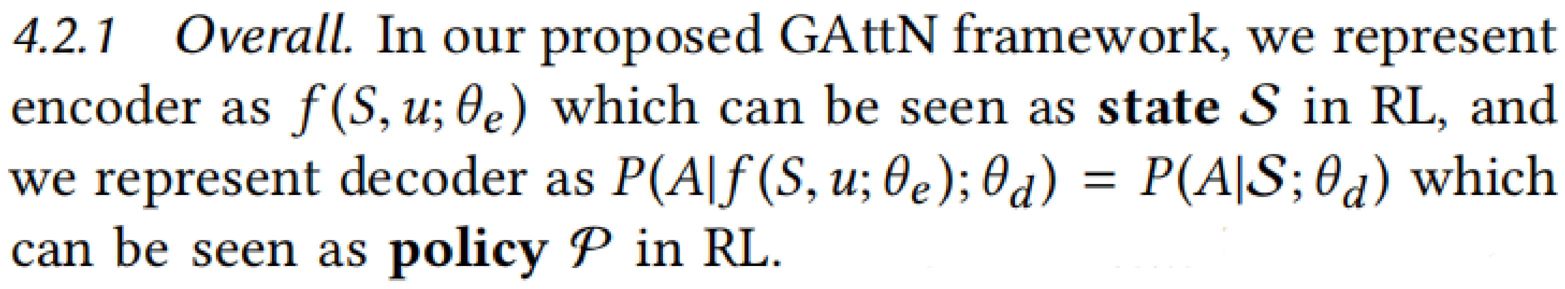
$$e_{tj} = softmax(v_{D1}^T \tanh(W_{D1}d_t + W_{D2}h_j^{(L)})), j \in \{1, \ldots, N\},$$

decoder output hidden states

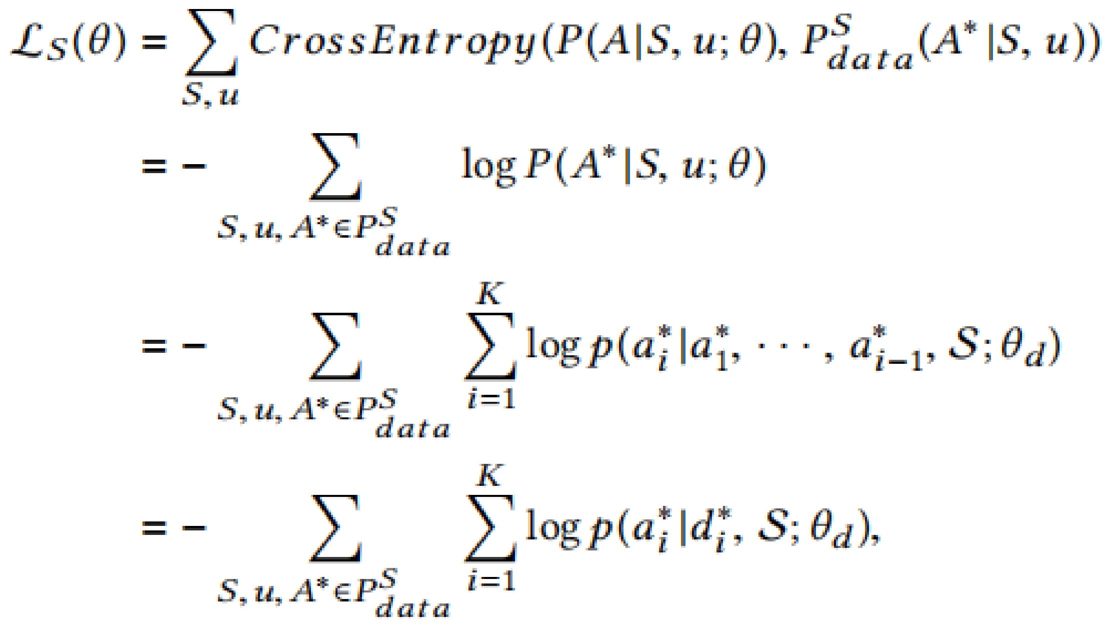
http://www.360doc.com/content/18/0618/09/17563728_763230413.shtml

To summarize, *decoder* receives embedding representations of nodes in graph G from encoder, and selects clique *A* of *K* nodes with attention mechanism. With the help of RNN and beam search, decoder in our proposed GAttN framework is able to capture the combinational characteristics of the *K* items in a card.

● Reinforcement Learning from Demonstrations



4.2.1 Overall. In our proposed GAttN framework, we represent encoder as $f(S, u; \theta_e)$ which can be seen as **state** $\mathcal{S}$ in RL, and we represent decoder as $P(A|f(S, u; \theta_e); \theta_d) = P(A|\mathcal{S}; \theta_d)$ which can be seen as **policy** $\mathcal{P}$ in RL.



$$\begin{aligned}
\mathcal{L}_S(\theta) &= \sum_{S,u} CrossEntropy(P(A|S, u; \theta), P^S_{data}(A^*|S, u)) \\
&= - \sum_{S,u,A^* \in P^S_{data}} \log P(A^*|S, u; \theta) \\
&= - \sum_{S,u,A^* \in P^S_{data}} \sum_{i=1}^{K} \log p(a_i^*|a_1^*, \cdots, a_{i-1}^*, \mathcal{S}; \theta_d) \\
&= - \sum_{S,u,A^* \in P^S_{data}} \sum_{i=1}^{K} \log p(a_i^*|d_i^*, \mathcal{S}; \theta_d),
\end{aligned}$$

Learning from demonstrations is much sample efficient and can speed up learning process, leveraging demonstrations to direct what would otherwise be uniform random exploration and thus speed up learning. While the demonstration trajectories may be noisy or sub-optimal, so policy supervised from such demonstrations will be worse too. And learning from demonstrations is not directly targeting the objective which makes the policy fall into local-minimal.

https://arxiv.org/pdf/1709.10089.pdf

☐Learning from Demonstrations

Learning from demonstrations can be seen as behavior cloning imitation learning

It applies supervised learning for policy (mapping states to actions) using the demonstration trajectories as ground-truth. We collect the ground truth clicked/satisfied cards $A^* = \{a_i^*\}_{1 \le i \le K}$ given user u and candidate items set S as demonstration trajectories

cross entropy of the generated cards

the generated cards

demonstrated cards

$$\begin{aligned}
\mathcal{L}_S(\theta) &= \sum_{S,u} CrossEntropy(P(A|S,u;\theta), P_{data}^S(A^*|S,u)) \\
&= -\sum_{S,u,A^* \in P_{data}^S} \log P(A^*|S,u;\theta) \\
&= -\sum_{S,u,A^* \in P_{data}^S} \sum_{i=1}^{K} \log p(a_i^*|a_1^*, \cdots, a_{i-1}^*, \mathcal{S};\theta_d) \qquad (15) \\
&= -\sum_{S,u,A^* \in P_{data}^S} \sum_{i=1}^{K} \log p(a_i^*|d_i^*, \mathcal{S};\theta_d),
\end{aligned}$$

state vector estimated by a RNN

This means that the decoder model focuses on learning to output the next item of the card given the current state of the model and previous ground-truth items

SUPERVISE with Policy-sampling.

We change the training process from fully guided using the true previous item, towards using the generated item from trained policy instead. The loss function for *Learning from Demonstrations* is now as follows

$$\begin{aligned}\mathcal{L}_S(\theta) &= \sum_{S,u} CrossEntropy(P(A|S,u;\theta), P^S_{data}(A^*|S,u)) \\ &= -\sum_{S,u,A^*\in P^S_{data}} \log P(A^*|S,u;\theta) \\ &= -\sum_{S,u,A^*\in P^S_{data}} \sum_{i=1}^{K} \log p(a_i^*|\boxed{a_1^*,\cdots,a_{i-1}^*,}\,\mathcal{S};\theta_d) \\ &= -\sum_{S,u,A^*\in P^S_{data}} \sum_{i=1}^{K} \log p(a_i^*|d_i^*,\mathcal{S};\theta_d),\end{aligned} \tag{15}$$

$$\Longrightarrow$$

$$\begin{aligned}\mathcal{L}_S(\theta) &= -\sum_{S,u,A^*\in P^S_{data}} \sum_{i=1} \log p(a_i^*\,\boxed{a_1,\cdots,a_{i-1},}\,\mathcal{S};\theta_d) \\ &= -\sum_{S,u,A^*\in P^S_{data}} \sum_{i=1}^{K} \log p(a_i^*|d_i,\mathcal{S};\theta_d),\end{aligned} \tag{16}$$

◻Learning from Rewards

The objective of exact-K recommendation is to maximize the chance of being clicked or satisfied for the selected card *A* given candidate items set *S* and user *u*.

we transfer estimation of reward function to the problem of CTR estimation for a card *A* given user *u* as $P(r = 1|A, u; \phi)$, and the loss function for training it is as follows:

explicit feedback data in which users click cards

$$\mathcal{L}_D(\phi) = -\sum_{A,u,r^* \in P^D_{data}} \Big(r^* \log\big(P(r=1|A,u;\phi)\big) + (1-r^*)\log\big(1-P(r=1|A,u;\phi)\big)\Big). \quad (17)$$

$$P(r=1|A,u;\phi) = \sigma\Big(W_{R2}ReLU\big(W_{R1}\big[[x_{a_i} \odot x_u]_{i=1}^{K};[x_{a_i}]_{i=1}^{K};x_u\big] + b_{R1}\big) + b_{R2}\Big), \quad (18)$$

REINFORCE with Hill-climbing.

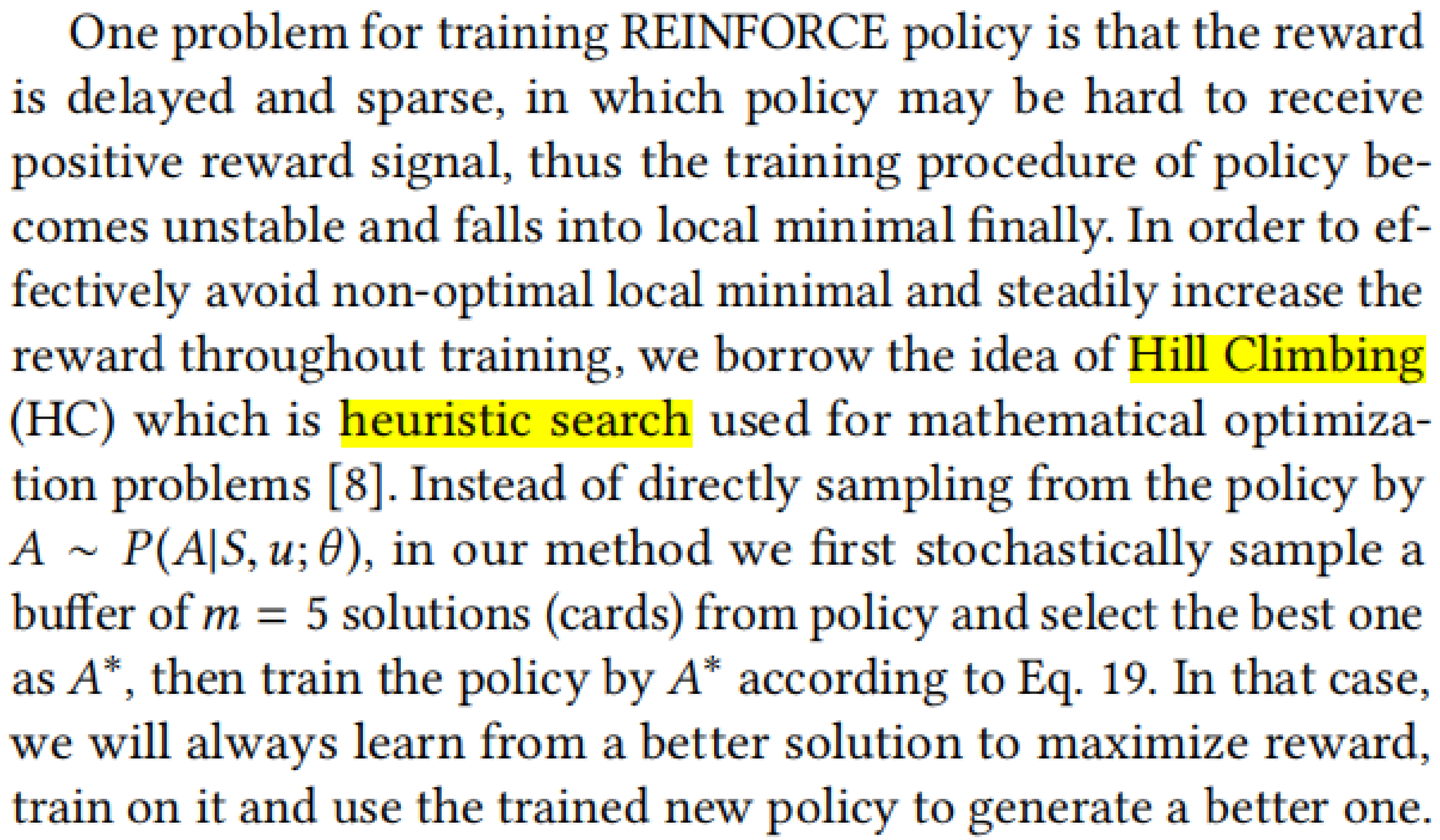
After we get the optimized reward function represented as $P(r = 1|A, u; \phi^*)$, we use policy gradient based reinforcement learning (REINFORCE) [29] to train the policy. And its loss function given previously defined dataset $P^S_{data}(\cdot|S, u)$ is derived as follows:

$$\begin{aligned} \mathcal{L}_R(\theta) &= -\sum_{S,u \in P^S_{data}} \mathbb{E}_{A \sim P(A|S,u;\theta)}[R(A, u)] \qquad (19) \\ &= -\sum_{S,u \in P^S_{data}} R(A, u) \sum_{i=1}^{K} \log p(a_i | a_1, \cdots, a_{i-1}, \mathcal{S}; \theta_d), \end{aligned}$$

where $\mathcal{S}$ is previously defined encoder state, $R(A, u)$ is the delayed reward [31] obtained after the whole card A is generated and is estimated by the following equation:

$$R(A, u) = 2 \times \left(P(r = 1|A, u; \phi^*) - 0.5\right), \qquad (20)$$

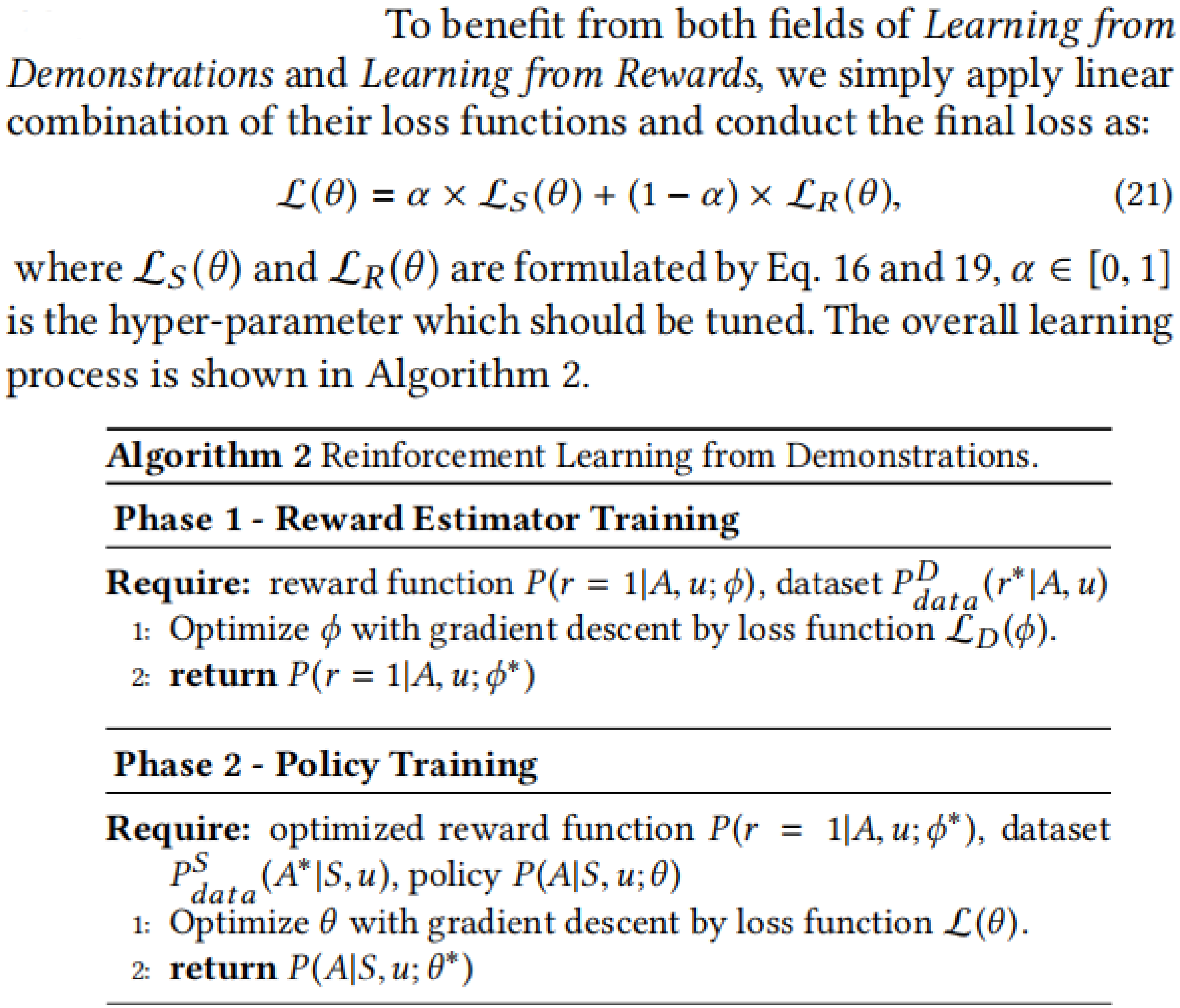
here we rescale the value of reward between -1.0 to 1.0.



One problem for training REINFORCE policy is that the reward is delayed and sparse, in which policy may be hard to receive positive reward signal, thus the training procedure of policy becomes unstable and falls into local minimal finally. In order to effectively avoid non-optimal local minimal and steadily increase the reward throughout training, we borrow the idea of Hill Climbing (HC) which is heuristic search used for mathematical optimization problems [8]. Instead of directly sampling from the policy by $A \sim P(A|S, u; \theta)$, in our method we first stochastically sample a buffer of $m = 5$ solutions (cards) from policy and select the best one as A^*, then train the policy by A^* according to Eq. 19. In that case, we will always learn from a better solution to maximize reward, train on it and use the trained new policy to generate a better one.

https://baike.baidu.com/item/%E7%88%AC%E5%B1%B1%E7%AE%97%E6%B3%95/3252408?fr=aladdin

☐Combination



To benefit from both fields of *Learning from Demonstrations* and *Learning from Rewards*, we simply apply linear combination of their loss functions and conduct the final loss as:

$$\mathcal{L}(\theta) = \alpha \times \mathcal{L}_S(\theta) + (1 - \alpha) \times \mathcal{L}_R(\theta), \tag{21}$$

where $\mathcal{L}_S(\theta)$ and $\mathcal{L}_R(\theta)$ are formulated by Eq. 16 and 19, $\alpha \in [0, 1]$ is the hyper-parameter which should be tuned. The overall learning process is shown in Algorithm 2.

Algorithm 2 Reinforcement Learning from Demonstrations.

Phase 1 - Reward Estimator Training

```
Require: reward function $P(r = 1|A, u; \phi)$, dataset $P^D_{data}(r^*|A, u)$
1: Optimize $\phi$ with gradient descent by loss function $\mathcal{L}_D(\phi)$.
2: return $P(r = 1|A, u; \phi^*)$
```

Phase 2 - Policy Training

```
Require: optimized reward function $P(r = 1|A, u; \phi^*)$, dataset
    $P^S_{data}(A^*|S, u)$, policy $P(A|S, u; \theta)$
1: Optimize $\theta$ with gradient descent by loss function $\mathcal{L}(\theta)$.
2: return $P(A|S, u; \theta^*)$
```

Experiments

- ☐ Does *GAttN with RLfD* method outperform the baseline methods in exact-K recommendation problem?
- ☐ How *Graph Attention Networks* (GAttN) framework work for modeling the problem?
- ☐ How does the optimization framework *Reinforcement Learning from Demonstrations* (RLfD) work for training the model?

● Experimental Settings

- ☐ *Datasets*

Table 3: Statistics of the experimented datasets.

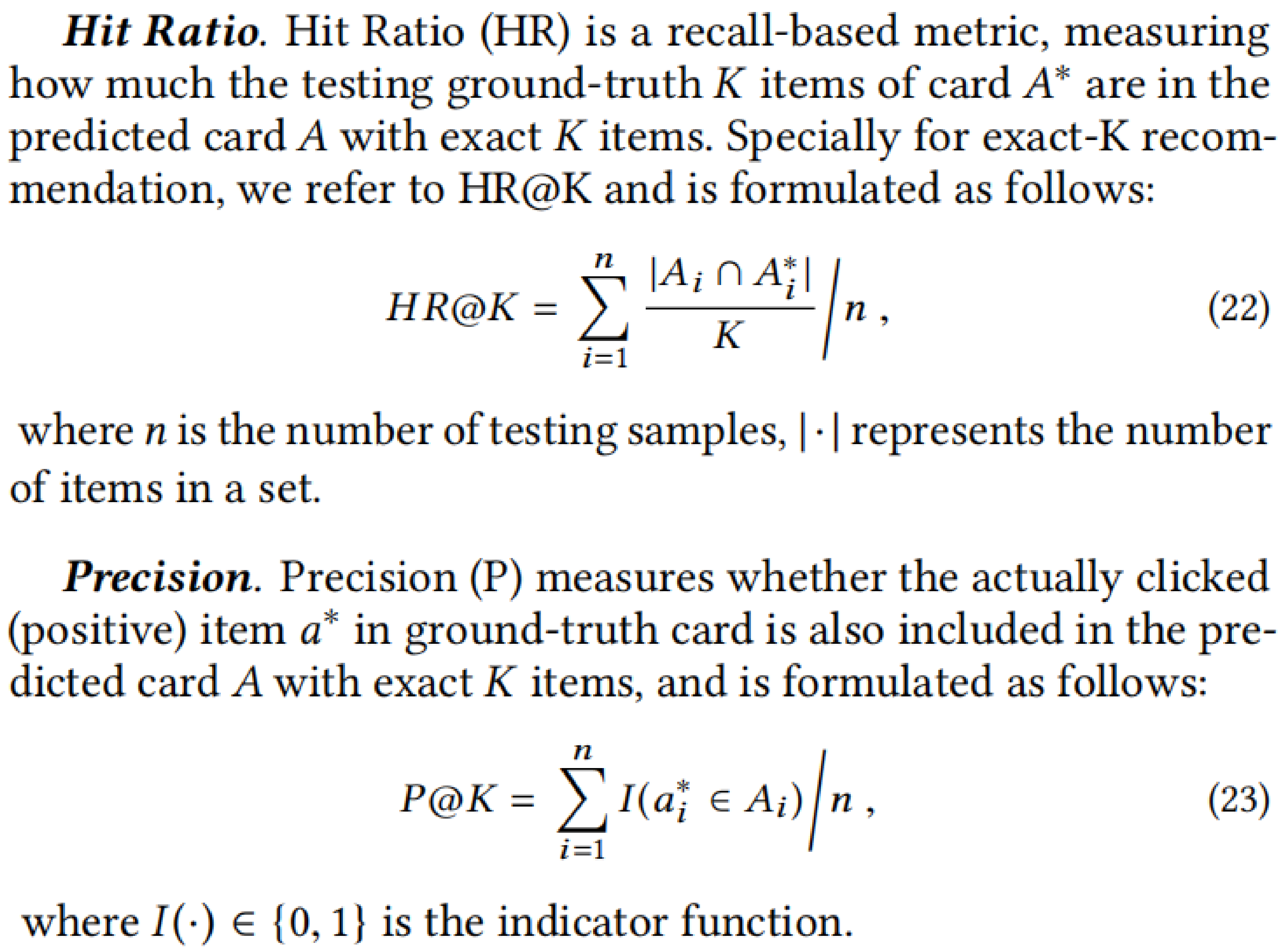
Dataset	User#	Card#	Item#	Sample#
MovieLens(K=4,N=20)	817	40036	1630	40036
MovieLens(K=10,N=50)	485	33196	1649	33198
Taobao(K=4,N=50)	581055	310509	3148550	1116582

Table 4: Show case of the dataset.

	user	card	candidate items	card label	positive item
sample#1	1	1,2,3,4	1,2,3,4,...,20	1	2
sample#2	1	1,4,5,6	1,2,3,4,...,20	0	/
			...		

(We take $K = 4$ and $N = 20$ for example. Items and users are represented as IDs here. Card label represents whether the card is clicked or satisfied by user (labeled as 1) or not (labeled as 0). Positive item is the actually clicked item in card by user.)

☐ *Evaluation Protocol*



Hit Ratio. Hit Ratio (HR) is a recall-based metric, measuring how much the testing ground-truth K items of card A^* are in the predicted card A with exact K items. Specially for exact-K recommendation, we refer to HR@K and is formulated as follows:

$$HR@K = \sum_{i=1}^{n} \frac{|A_i \cap A_i^*|}{K} \Big/ n \,, \tag{22}$$

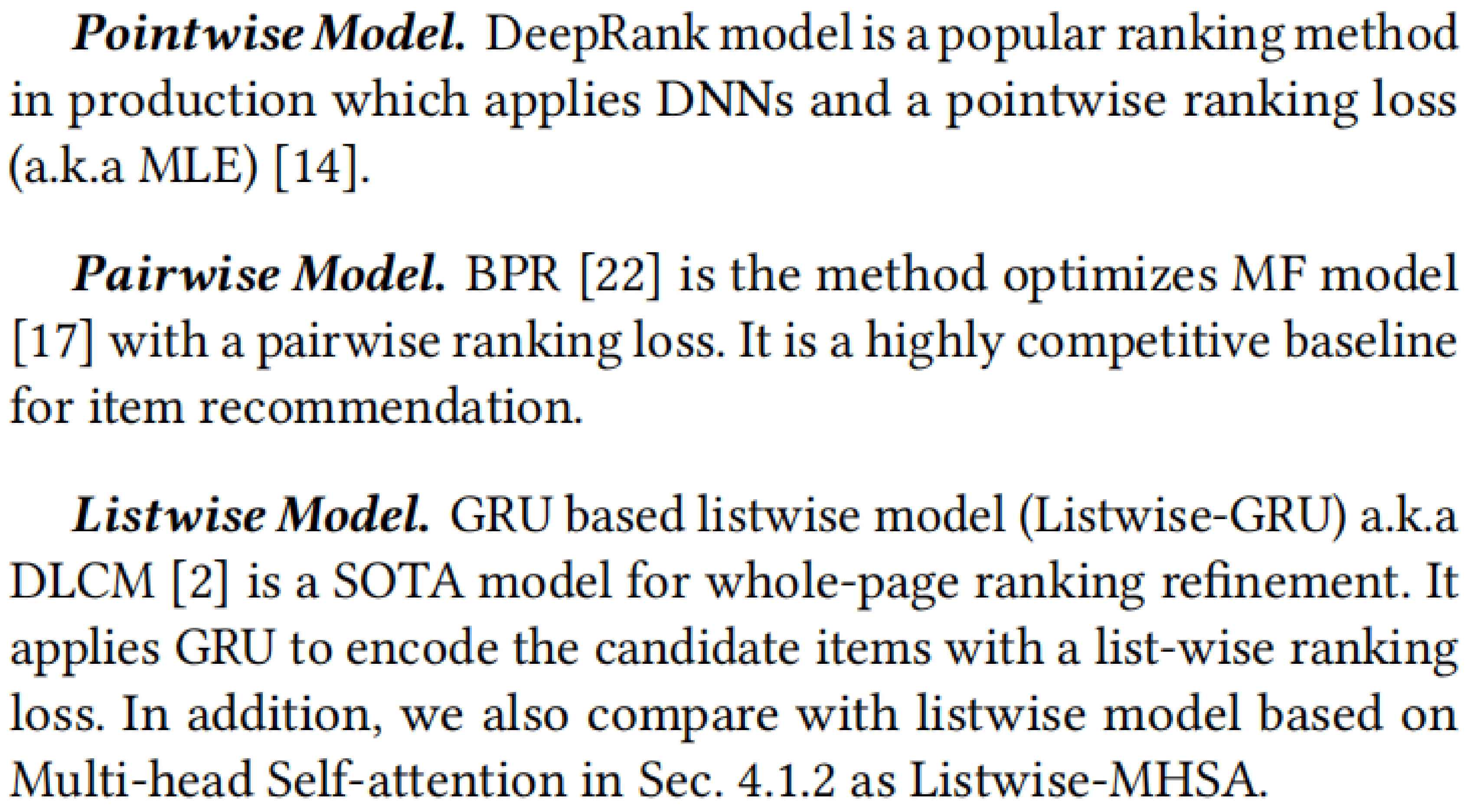
where n is the number of testing samples, $|\cdot|$ represents the number of items in a set.

Precision. Precision (P) measures whether the actually clicked (positive) item a^* in ground-truth card is also included in the predicted card A with exact K items, and is formulated as follows:

$$P@K = \sum_{i=1}^{n} I(a_i^* \in A_i) \Big/ n \,, \tag{23}$$

where $I(\cdot) \in \{0, 1\}$ is the indicator function.

Baselines



Pointwise Model. DeepRank model is a popular ranking method in production which applies DNNs and a pointwise ranking loss (a.k.a MLE) [14].

Pairwise Model. BPR [22] is the method optimizes MF model [17] with a pairwise ranking loss. It is a highly competitive baseline for item recommendation.

Listwise Model. GRU based listwise model (Listwise-GRU) a.k.a DLCM [2] is a SOTA model for whole-page ranking refinement. It applies GRU to encode the candidate items with a list-wise ranking loss. In addition, we also compare with listwise model based on Multi-head Self-attention in Sec. 4.1.2 as Listwise-MHSA.

● Performance Comparison

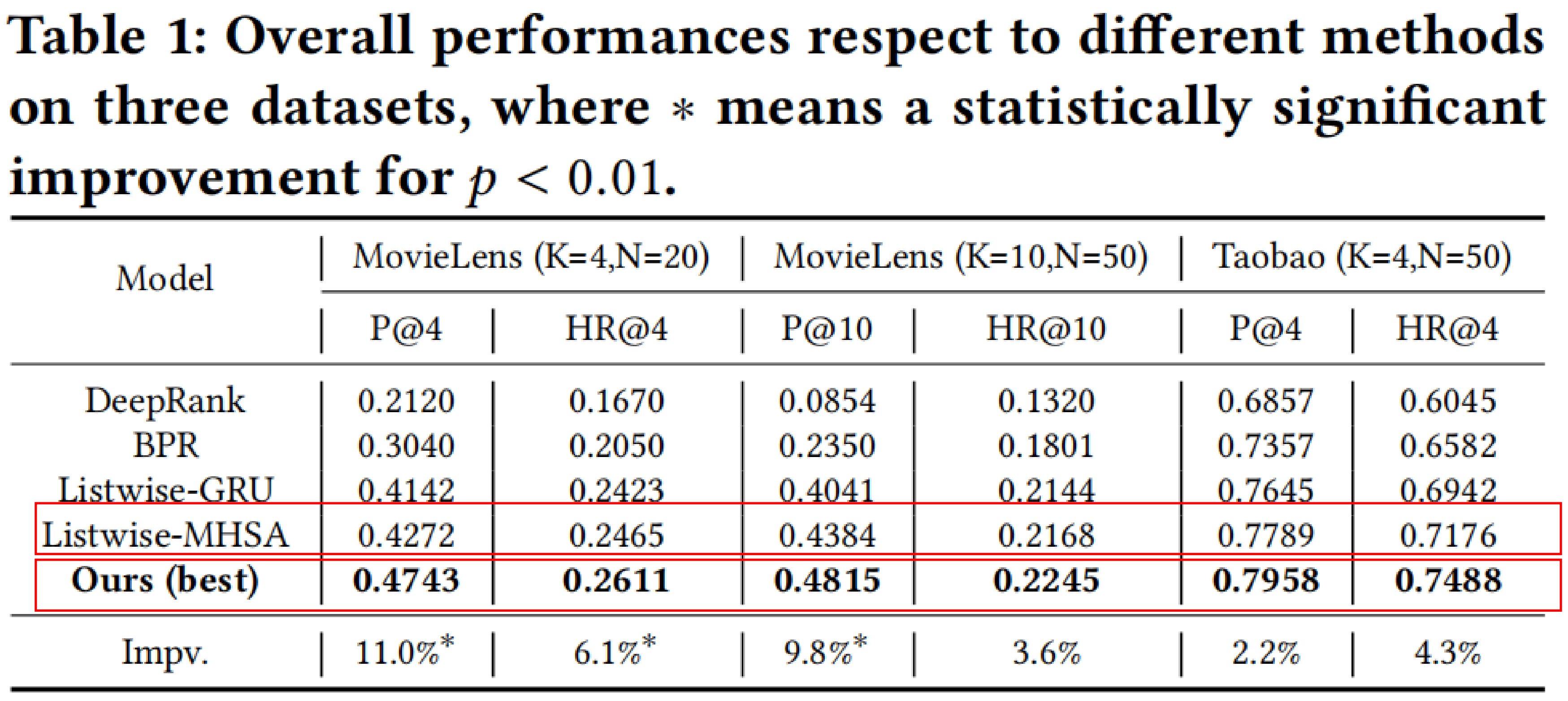


Table 1: Overall performances respect to different methods on three datasets, where $*$ means a statistically significant improvement for $p < 0.01$.

Model	MovieLens (K=4,N=20)		MovieLens (K=10,N=50)		Taobao (K=4,N=50)	
	P@4	HR@4	P@10	HR@10	P@4	HR@4
DeepRank	0.2120	0.1670	0.0854	0.1320	0.6857	0.6045
BPR	0.3040	0.2050	0.2350	0.1801	0.7357	0.6582
Listwise-GRU	0.4142	0.2423	0.4041	0.2144	0.7645	0.6942
Listwise-MHSA	0.4272	0.2465	0.4384	0.2168	0.7789	0.7176
Ours (best)	**0.4743**	**0.2611**	**0.4815**	**0.2245**	**0.7958**	**0.7488**
Impv.	11.0%*	6.1%*	9.8%*	3.6%	2.2%	4.3%

● Analysis for GAttN

how the self-attention works in encoder (see Fig. 4) based on Taobao dataset.

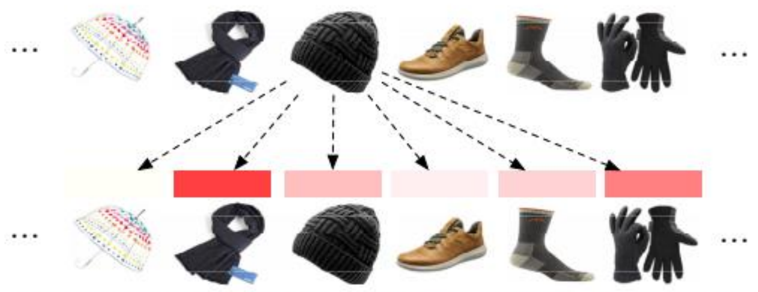


Figure 4: An example of the attention mechanism in the encoder self-attention in layer 2. The higher attention weights of the item, the darker color of the grid. We take item "hat" for example and only show attention weights in one head.

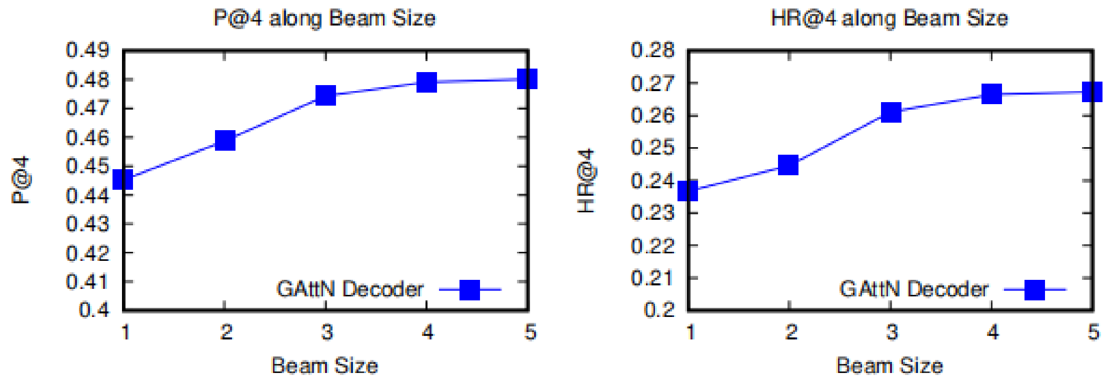


Figure 5: Performance of P@4 and HR@4 with different beam size in GAttN decoder.

● Analysis for RLfD

(1) Set $\alpha = 0$ in Eq. 21 (only reinforcement loss as Eq. 19), and compare *RL(w/ hill-climbing)* with *RL(w/o hill-climbing)*.
(2) Set $\alpha = 1$ in Eq. 21 (only supervised loss as Eq. 16), and compare *SL(w/ policy-sampling)* with *SL(w/o policy-sampling)*.
(3) Finetune α in Eq. 21 and figure out the influence to the combination of SL with RL.

Table 2: Performance for different settings in RLfD.

	Settings in RLfD	MovieLens (K=4,N=20)	
		P@4	HR@4
1	RL(w/o hill-climbing)	0.3340	0.2314
2	RL(w/ hill-climbing)	0.3573	0.2330
3	SL(w/o policy-sampling)	0.4095	0.2401
4	SL(w/ policy-sampling)	0.4272	0.2465
5	RL(w/o hill-climbing) + SL(w/ policy-sampling)	0.4495	0.2514
6	RL(w/ hill-climbing) + SL(w/o policy-sampling)	0.4472	0.2534
7	**RL(w/ hill-climbing) + SL(w/ policy-sampling)**	**0.4743**	**0.2611**

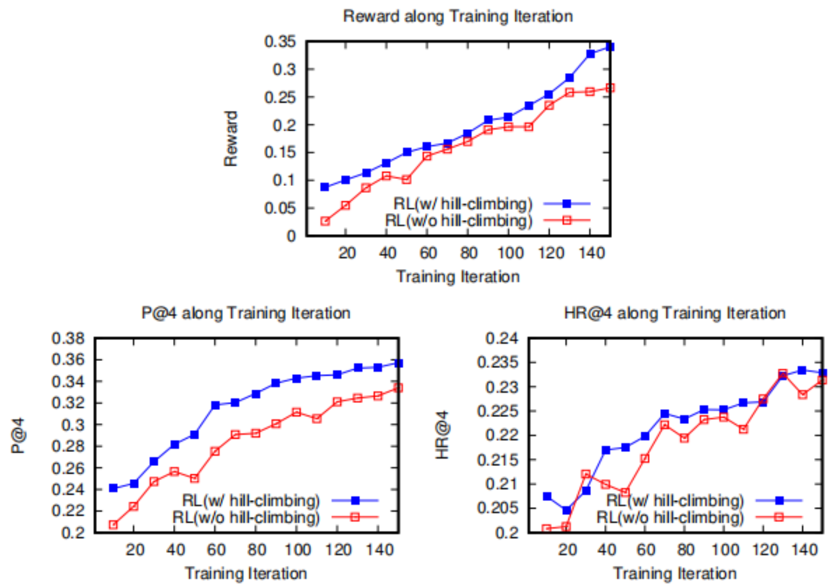


Figure 6: Learning curves respect to Reward, P@4 and HR@4 for RL with (w/) or without (w/o) hill-climbing.

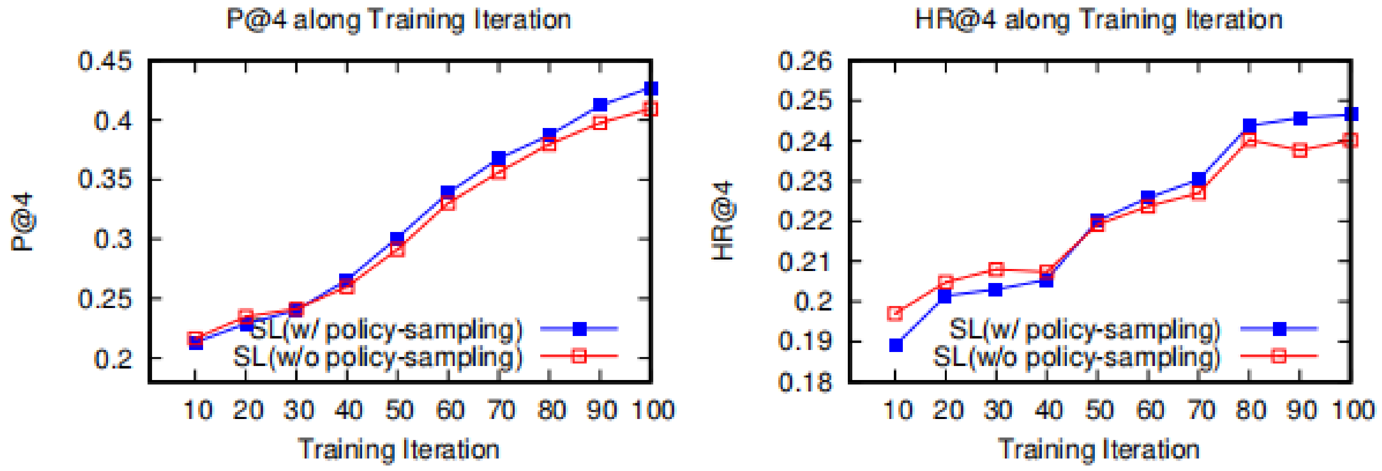


Figure 7: Learning curves respect to P@4 and HR@4 for SL with (w/) or without (w/o) policy-sampling.

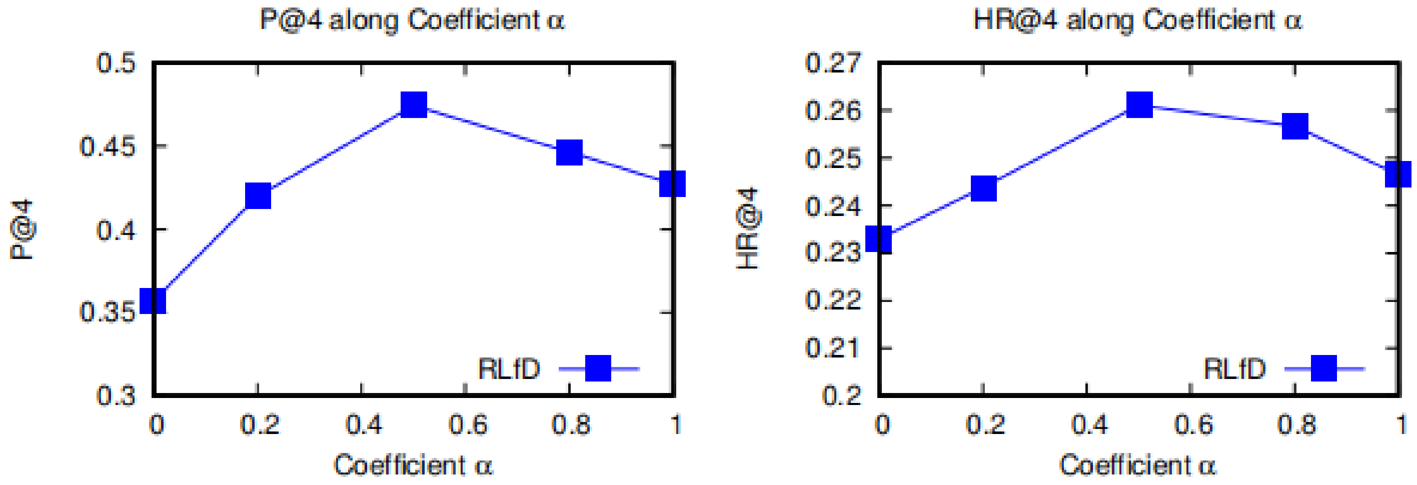


Figure 8: Performance of P@4 and HR@4 with different coefficients α in loss defined in Eq. 21.

Conclusion & Future Work

- In the first step, we give a formal problem definition, then reduce it to a Maximal Clique Optimization problem which is a combinatorial optimization problem and NP-hard.
- We propose a novel approach of GAttN with RLfD. In our evaluation, we perform extensive analysis to demonstrate the highly positive effect of our proposed method targeting exact-K recommendation problem.
- In our future work, we plan to adopt adversarial training for the components of Reward Estimator and Reinforce learning, regarding as discriminator and generator in GAN’s perspective. Moreover further online A/B testing in production will be conducted

总结 & 启发

- Exact-K Recommendation （数据集、评价指标、实验设置）
- 多个细节方法的融合

thank you