



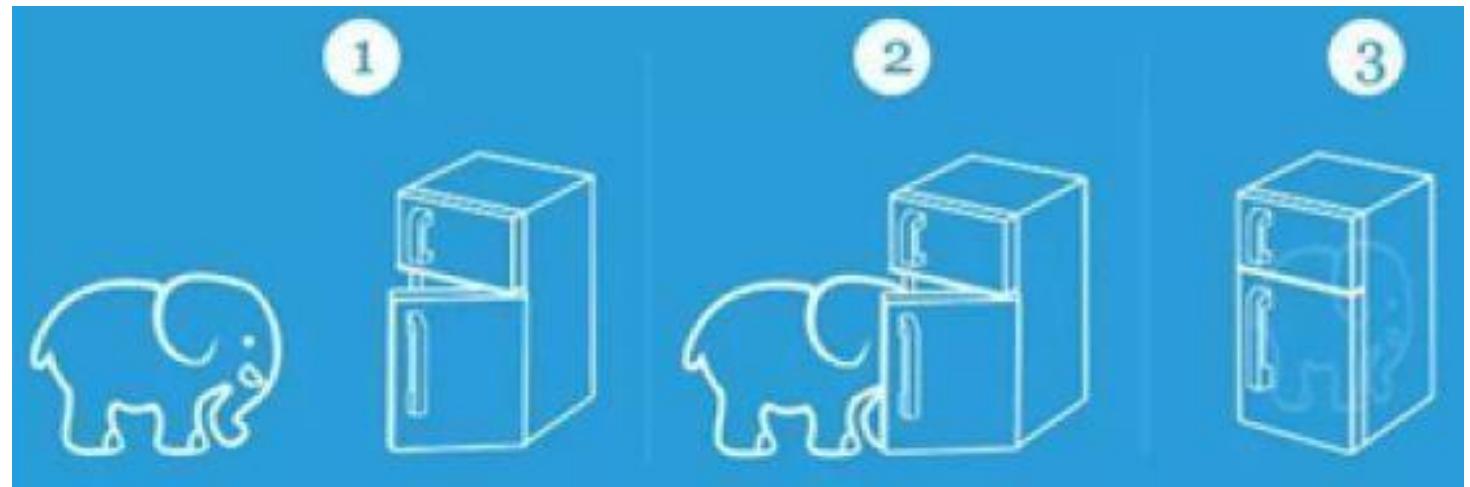
# IR APPLICATIONS USING DQN

Dell Zhang

# How to Put an Elephant into a Fridge?

You only need three steps:

- 1) Open the fridge;
- 2) Put the elephant in;
- 3) Close the door.



# How to Solve a Real-World Problem using RL?

You only need three steps:

1. Formulate the problem as an MDP;
2. Choose an RL algorithm for the MDP;
3. Run the RL algorithm.

# Recommender Systems

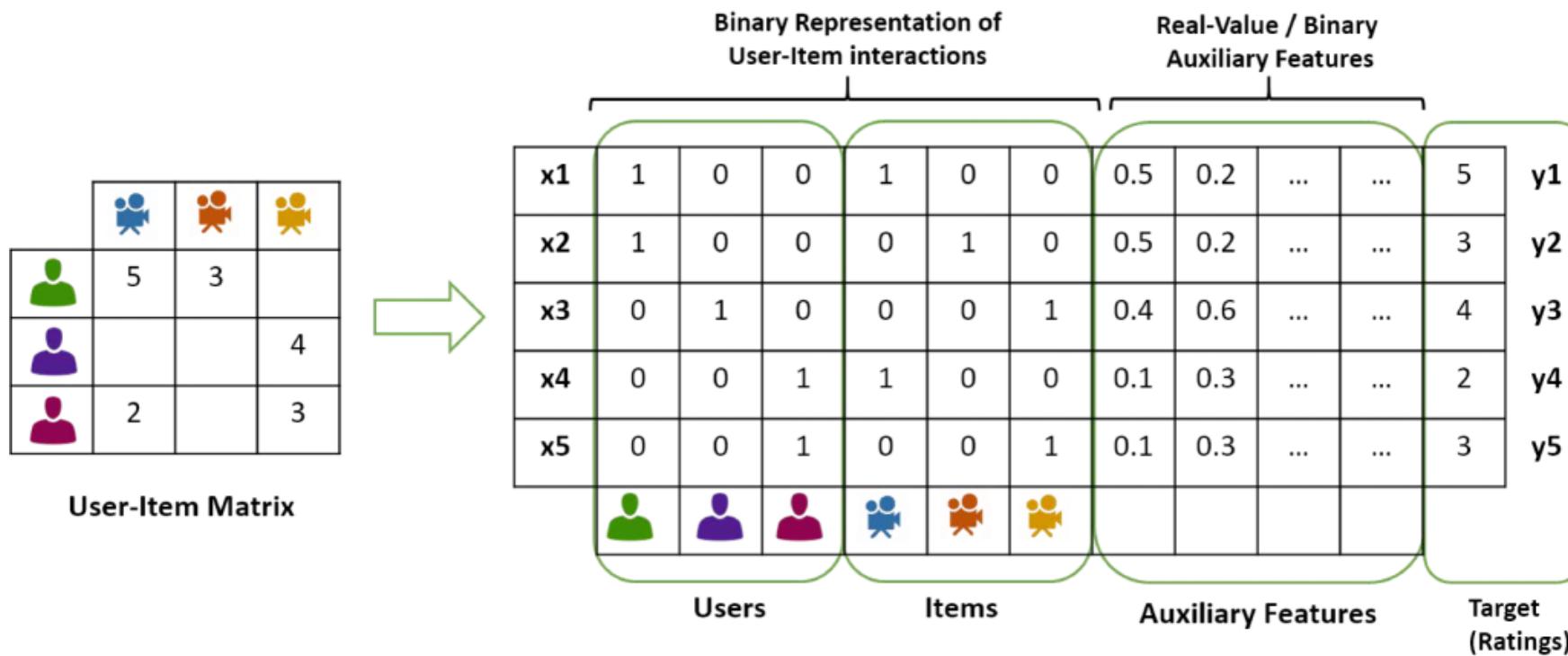
Mitigate the information overload problem by making personalized suggestions to users



# Recommender Systems

## Challenges of Existing Recommender Systems (e.g., Factorization Machines)

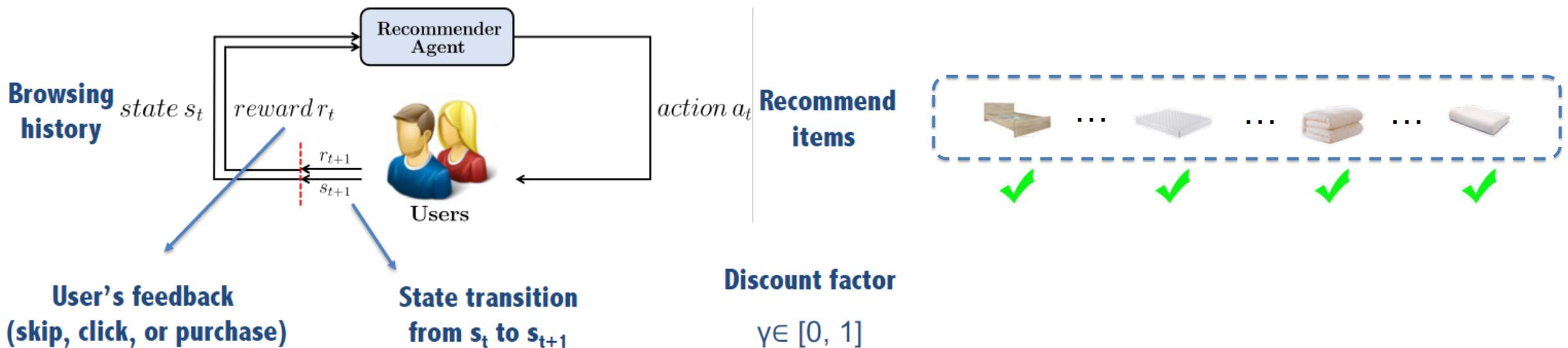
- Recommendation procedure as static process, following a fixed greedy strategy.
- Maximizing the immediate (short-term) reward from users



# Recommender System

## Advantages of Using RL

- Continuously updating the recommendation strategies
- Maximizing the long-term reward from users



# DRN: A Deep Reinforcement Learning Framework for News Recommendation

Guanjie Zheng<sup>†</sup>, Fuzheng Zhang<sup>§</sup>, Zihan Zheng<sup>§</sup>, Yang Xiang<sup>§</sup>

Nicholas Jing Yuan<sup>§</sup>, Xing Xie<sup>§</sup>, Zhenhui Li<sup>†</sup>

Pennsylvania State University<sup>†</sup>, Microsoft Research Asia<sup>§</sup>  
University Park, USA<sup>†</sup>, Beijing, China<sup>§</sup>

gjz5038@ist.psu.edu,{fuzzhang,v-zihanzhe,yaxian,nicholas.yuan,xingx}@microsoft.com,jessieli@ist.psu.edu

# News Recommendation

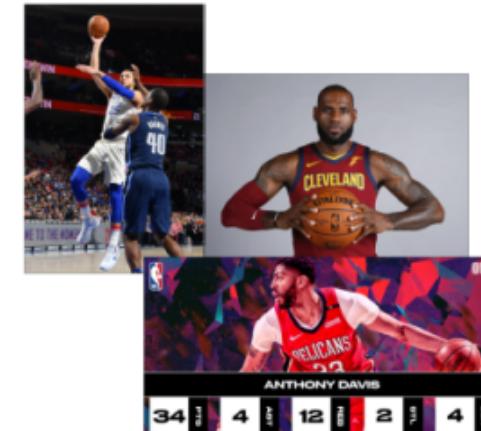
## Long-Term Effect in Recommendation



First round



Second round



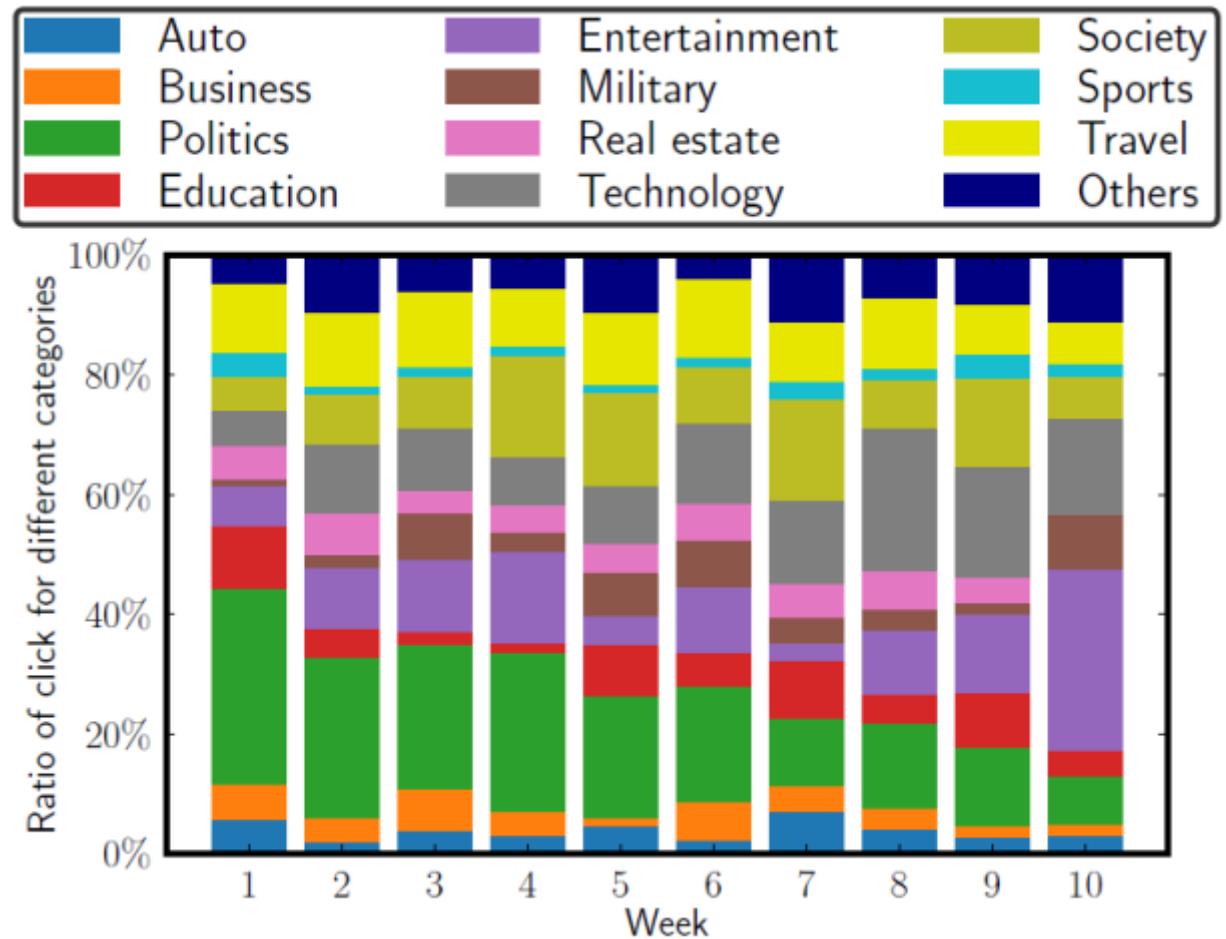
Images from:

1. <https://www.ohio.com/akron/sports/cavs/cavaliers-j-r-smith-expects-lebron-james-to-keep-opening-night-streak-alive-despite-sprained-ankle>
2. <https://www.cnbc.com/2018/03/05/kobe-bryant-has-won-an-oscar-heres-what-he-says-it-takes-to-succeed.html>
3. <https://twitter.com/NBA>
4. <https://www.stgeorgeutah.com/news/archive/2016/06/11/dig-alert-significant-weather-strong-thunderstorm-rolls-through-washington-county/#.WsqHUhPwb6>
5. [http://www.clipartpanda.com/clipart\\_images/question-mark-36565633](http://www.clipartpanda.com/clipart_images/question-mark-36565633)

# News Recommendation

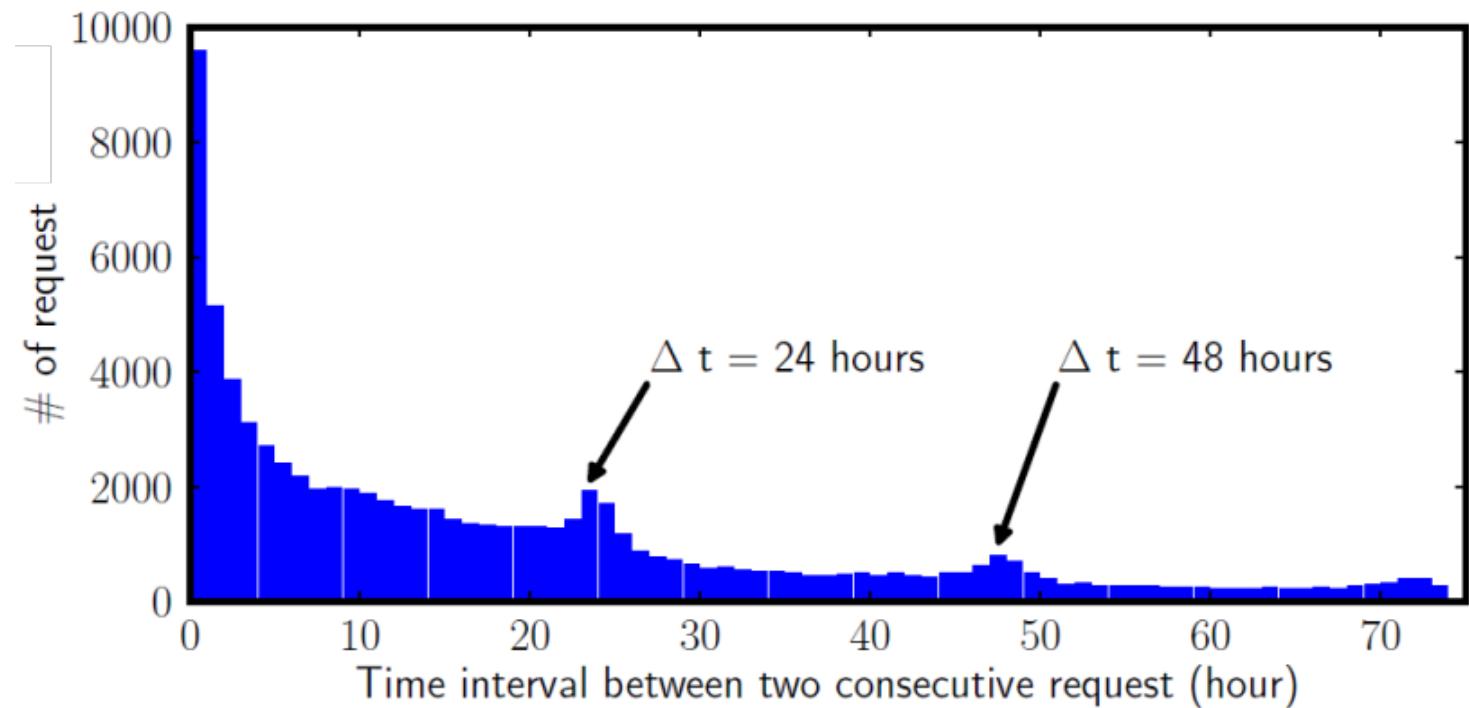
- News recommendation is dynamic!

- The life period for news is usually very short.  
(last-click-time - publish-time  $\approx$  4.1 hours)
- User's interest may change during time.



# News Recommendation

- Can we utilize user feedback other than click/skip?  
(e.g., how frequent user returns)
  - User's clicks on news are usually very dense in a short period.  
Then, user usually leave the app!
  - User may return everyday!



# News Recommendation

- Should we keep recommending similar items?
  - Will you get bored if all the recommended news are from NBA when you are browsing sports news?



Lebron James will be the MVP! |

Tony Parker has come back from injury! |

Paul Gasol promises to help the Spurs in the playoff.

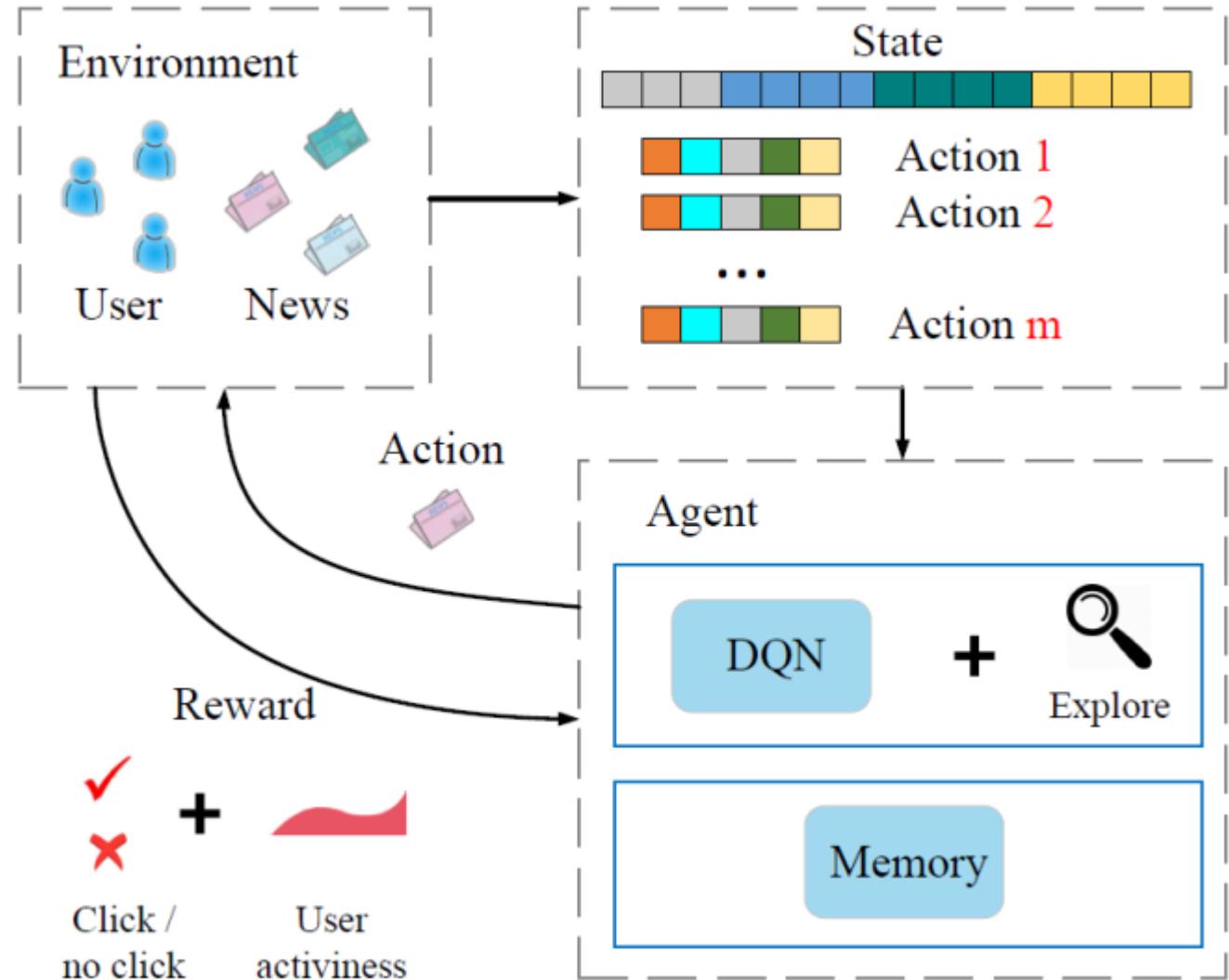
# News Recommendation

Motivation	Solution
<b>Long term effect in recommendation</b> <ul style="list-style-type: none"><li>• Dynamic nature of news recommendation</li><li>• Consider more measures for long term effect</li><li>• Recommendation diversity</li></ul>	<b>Deep reinforcement learning (DRL)</b> <ul style="list-style-type: none"><li>• Online learning feature of DRL</li><li>• Reward function design of DRL</li><li>• Explore in DRL</li></ul>

# News Recommendation

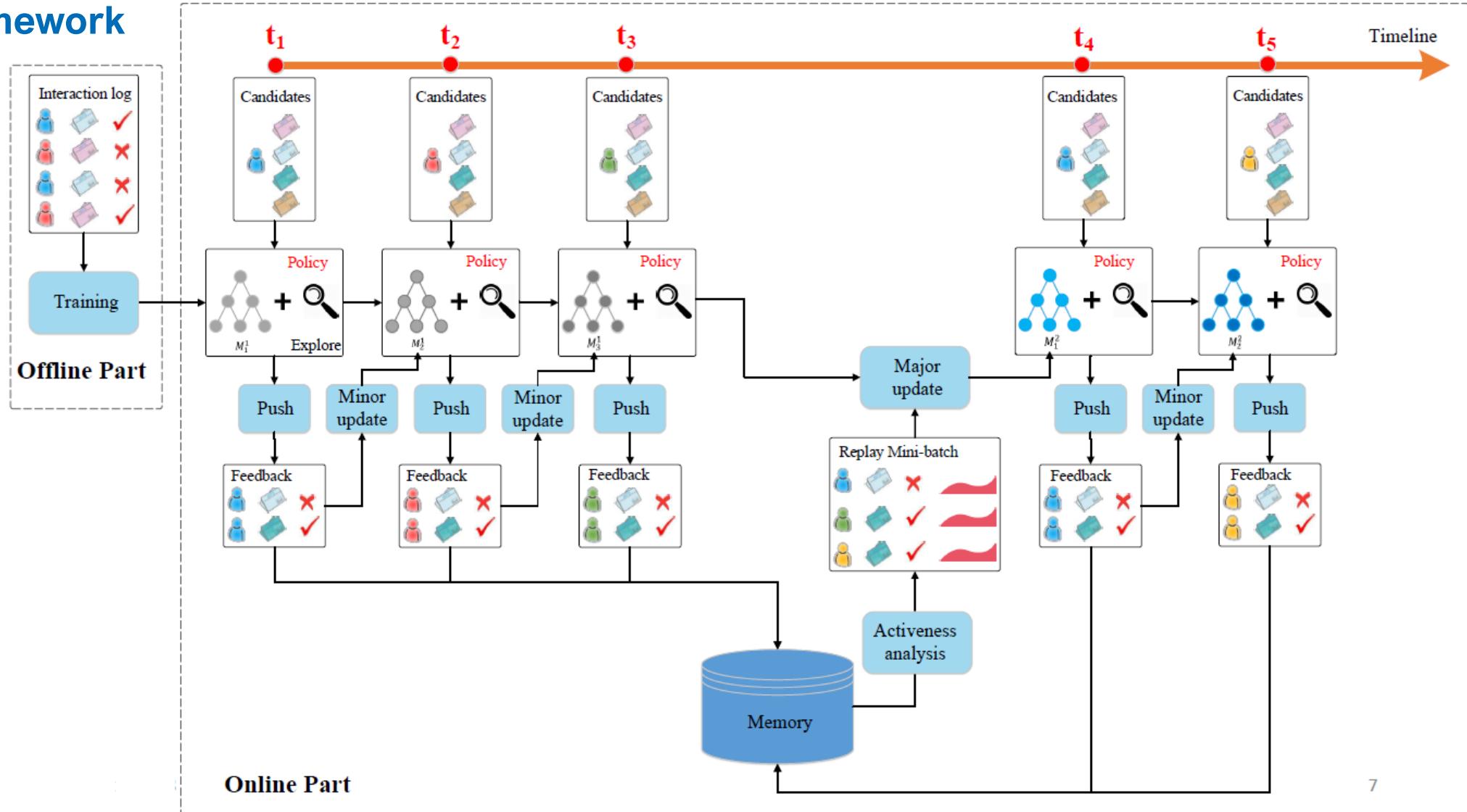
## Recommendation as MDP

- Environment: User Pool + News Pool
- Agent: Recommendation Algorithm
- State: Feature Representation for Users
- Action: Feature Representation for News
- Reward: User Feedback
  - click/skip labels
  - estimation of user activeness



# News Recommendation

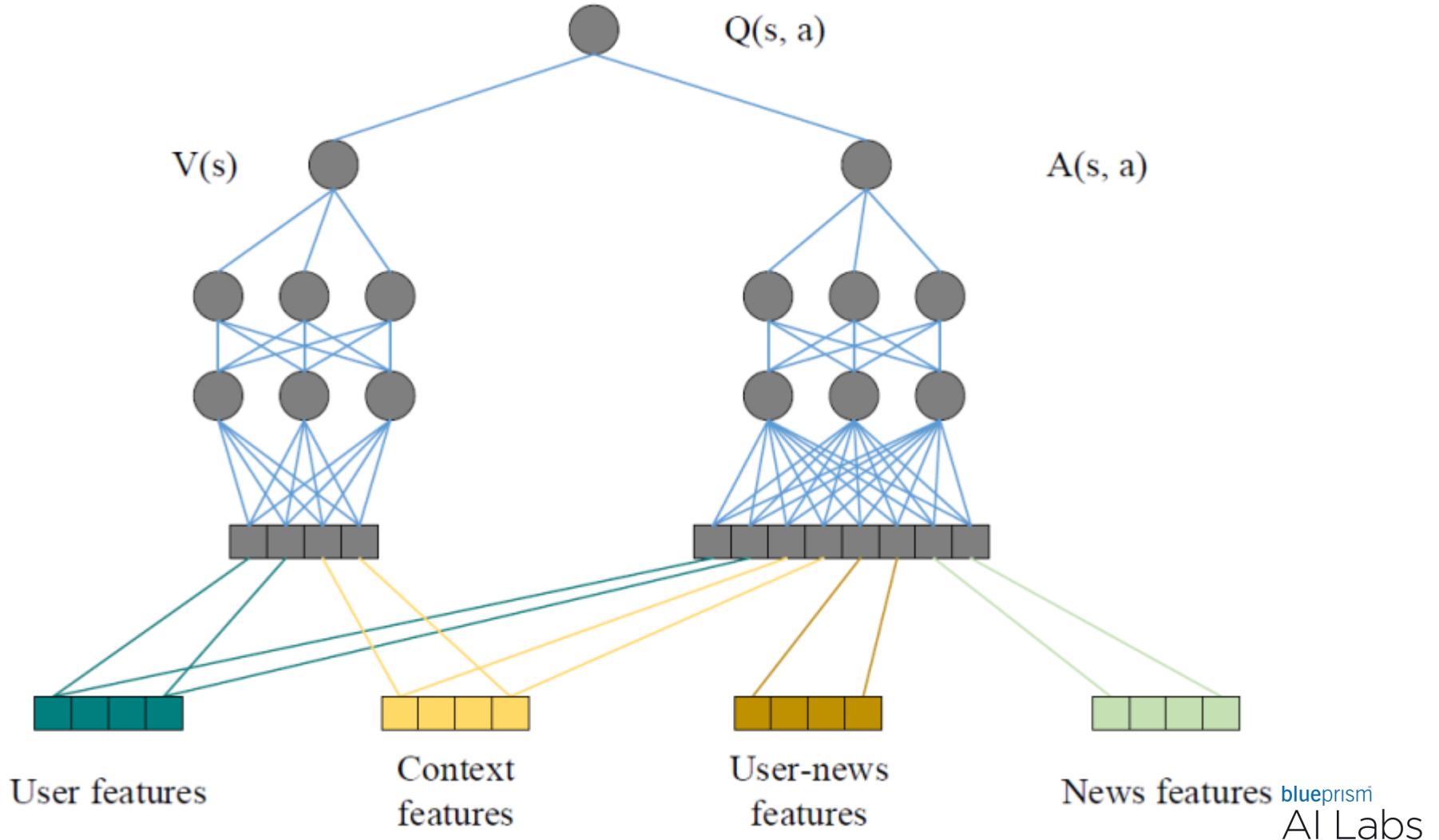
## Model Framework



# News Recommendation

## Dueling Network Architecture

- Value function
- Advantage function

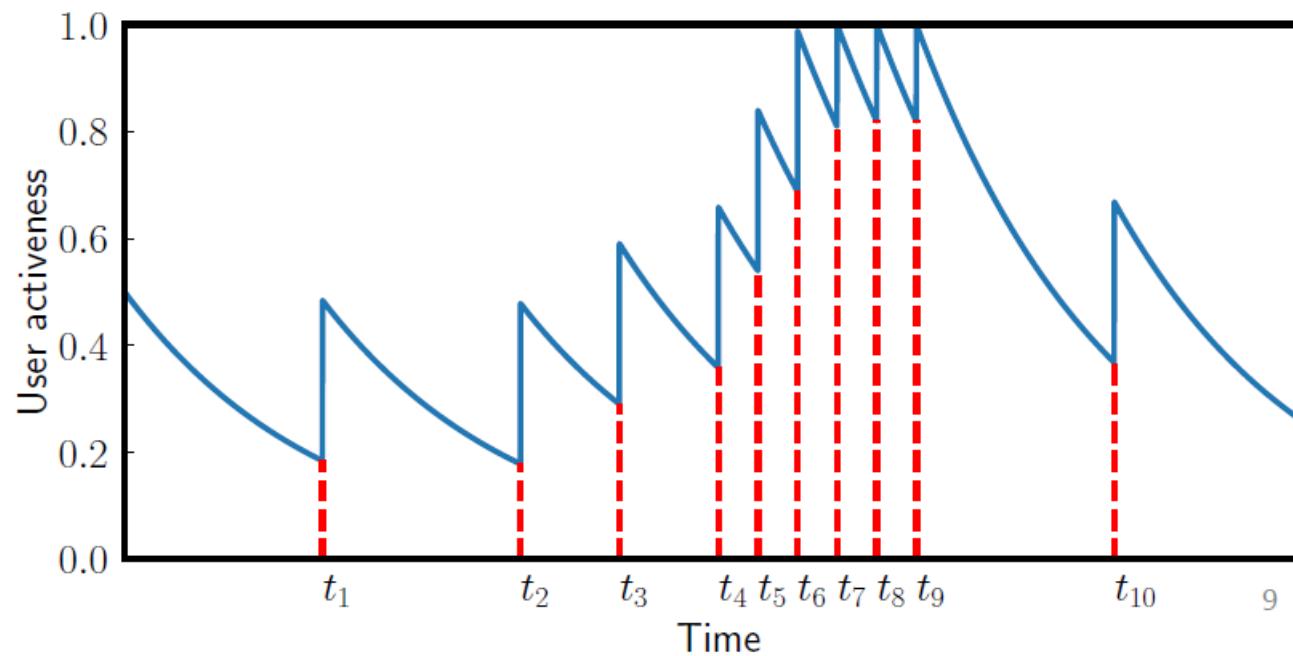


# News Recommendation

## User Activeness Modelling

(Survival Analysis)

- Hazard function  $\lambda(t) = \lim_{dt \rightarrow 0} \frac{Pr\{t \leq T < t + dt | T \geq t\}}{dt}$
- User activeness  $S(t) = e^{-\int_0^t \lambda(x)dx}$

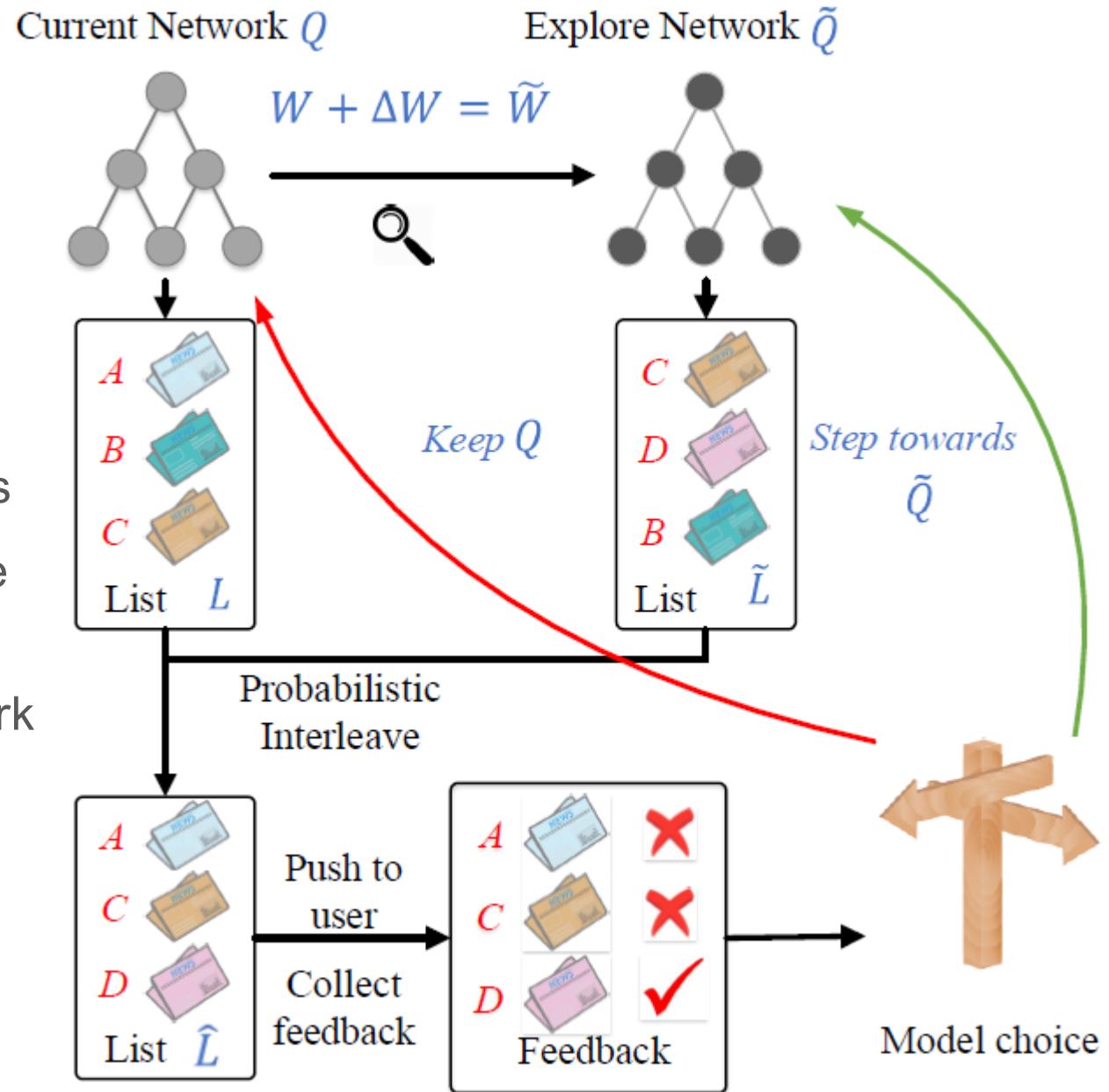


# News Recommendation

## Effective Exploration

(Dueling Bandit Gradient Descent)

- Step1: get recommendation from  $Q$  and  $\tilde{Q}$   
 $\Delta W = \alpha \cdot \text{rand}(-1, 1) \cdot W$
- Step2: probabilistic interleave these two lists
- Step3: get feedback from user and compare the performance of two networks
- Step4: if  $\tilde{Q}$  performs better, update  $Q$  network towards it  
 $W' = W + \eta \tilde{W}$



## **Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning**

Xiangyu Zhao  
Data Science and Engineering Lab  
Michigan State University  
zhaoxi35@msu.edu

Liang Zhang  
Data Science Lab  
Intelligent Advertising Lab  
JD.com  
zhangliang16@jd.com

Zhuoye Ding  
Data Science Lab  
JD.com  
dingzhuoye@jd.com

Long Xia  
Data Science Lab  
JD.com  
xialong@jd.com

Jiliang Tang  
Data Science and Engineering Lab  
Michigan State University  
tangjili@msu.edu

Dawei Yin  
Data Science Lab  
JD.com  
yindawei@acm.org

# Recommend with Positive and Negative Feedback

## Why Negative Feedback?

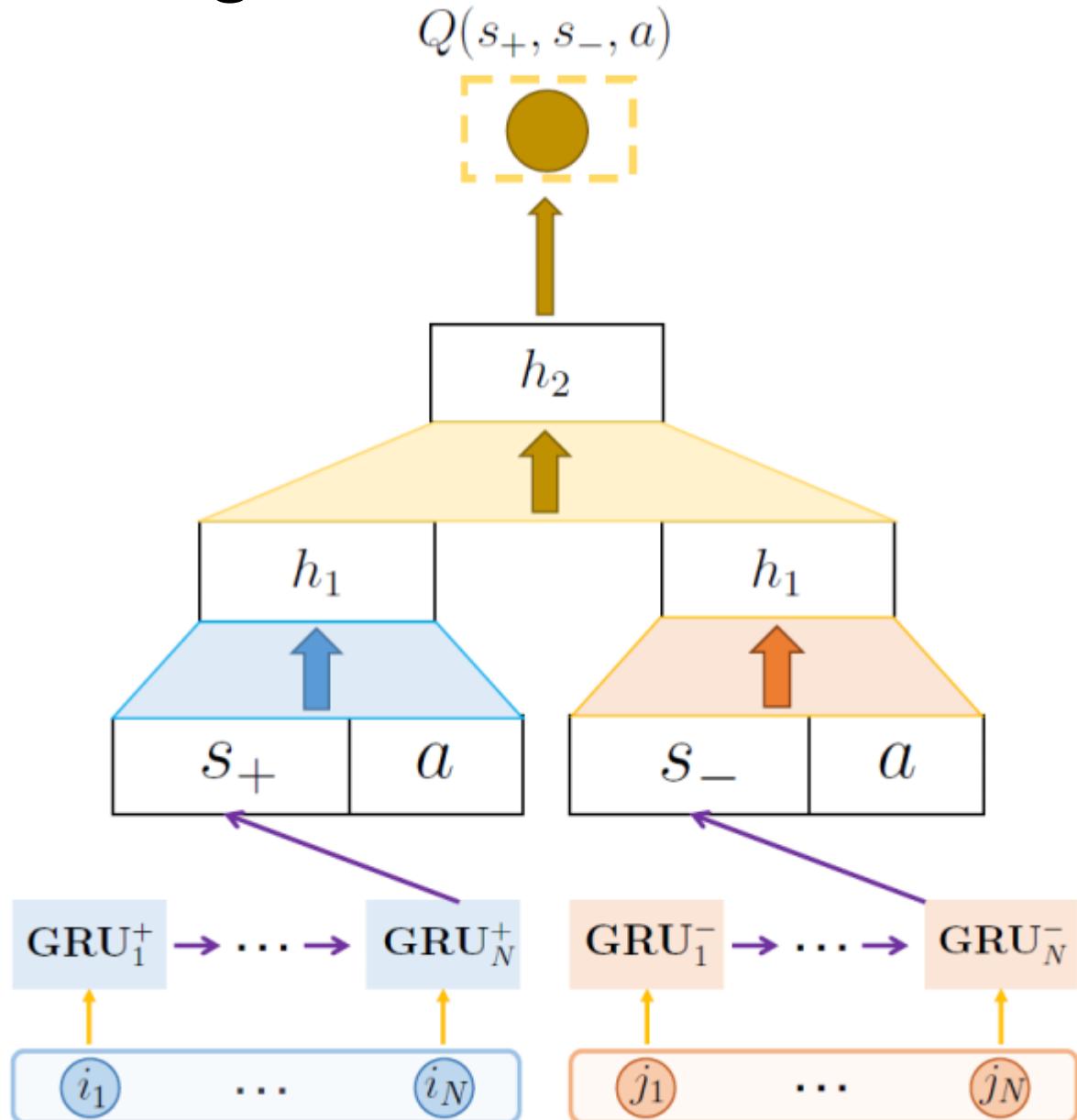
- Positive: click or purchase
- Negative: skip  
(what users may not like)



# Recommend with Positive and Negative Feedback

## Network Architecture

- RNN with GRU to capture users' *sequential preference*.
- Recommend an item that is *similar* to the clicked/purchased items (left part), while *dissimilar* to the skipped items (right part).
- This architecture can assist DQN to capture distinct contributions of the positive and negative feedback to recommendations.



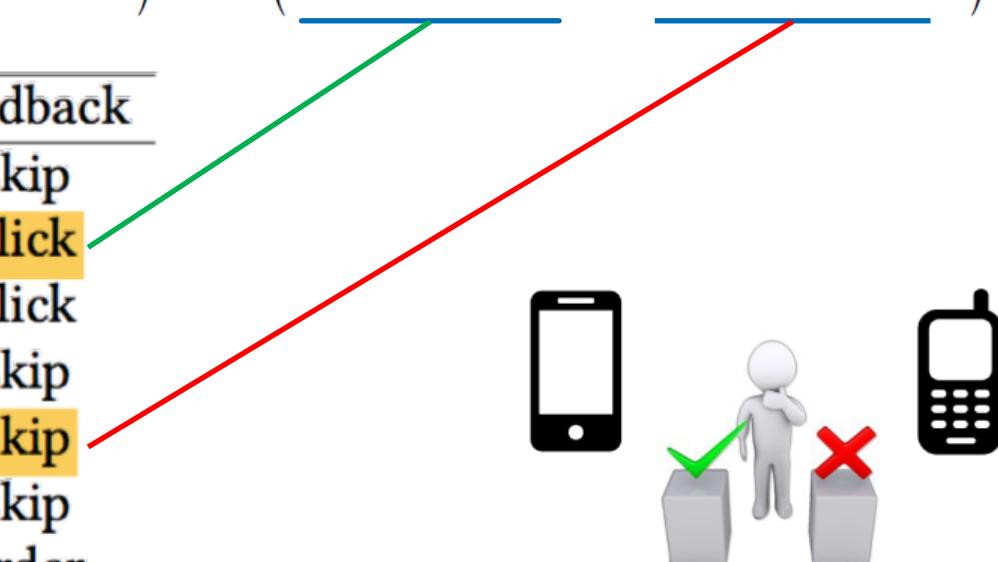
# Recommend with Positive and Negative Feedback

## The Pairwise Regularization Term

- RA often recommends items belong to the same category, while users click/purchase a part of them and skip others.
- We design a pairwise regularization term to maximize the difference of Q-values between competing items.

$$L(\theta) = \mathbb{E}_{s, a, r, s'} \left[ \left( y - Q(s_+, s_-, a; \theta) \right)^2 - \alpha \left( Q(s_+, s_-, a; \theta) - Q(s_+, s_-, a^C; \theta) \right)^2 \right]$$

Time	State	Item	Category	Feedback
1	$s_1$	$a_1$	A	skip
2	$s_2$	$a_2$	B	click
3	$s_3$	$a_3$	A	click
4	$s_4$	$a_4$	C	skip
5	$s_5$	$a_5$	B	skip
6	$s_6$	$a_6$	A	skip
7	$s_7$	$a_7$	C	order



# Jointly Learning to Recommend and Advertise

Xiangyu Zhao  
Michigan State University  
[zhaoxi35@msu.edu](mailto:zhaoxi35@msu.edu)

Xudong Zheng  
Bytedance  
[zhengxudong.alpha@bytedance.com](mailto:zhengxudong.alpha@bytedance.com)

Xiwang Yang  
Bytedance  
[yangxiwang@bytedance.com](mailto:yangxiwang@bytedance.com)

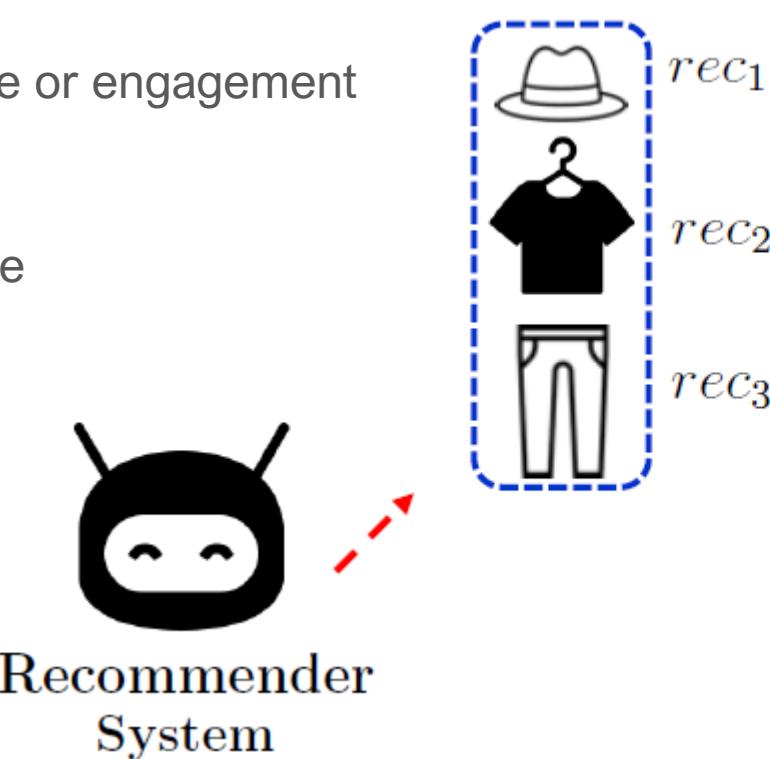
Xiaobing Liu  
Bytedance  
[will.liu@bytedance.com](mailto:will.liu@bytedance.com)

Jiliang Tang  
Michigan State University  
[tangjili@msu.edu](mailto:tangjili@msu.edu)

# Jointly Learning to Recommend and Advertise

## Recommender System

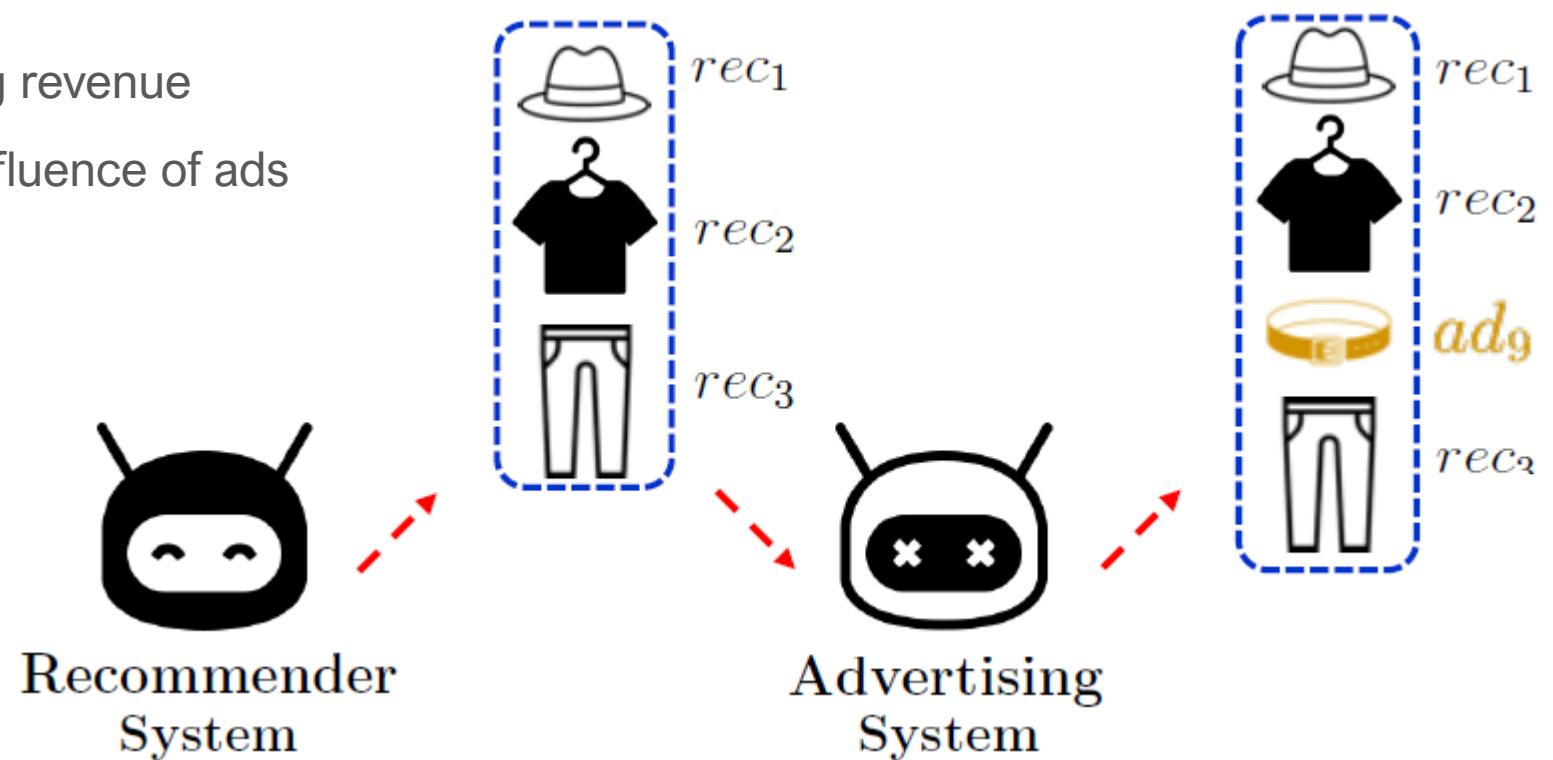
- Goal
  - long-term user experience or engagement
- Challenge
  - combinatorial action space



# Jointly Learning to Recommend and Advertise

## Advertising System

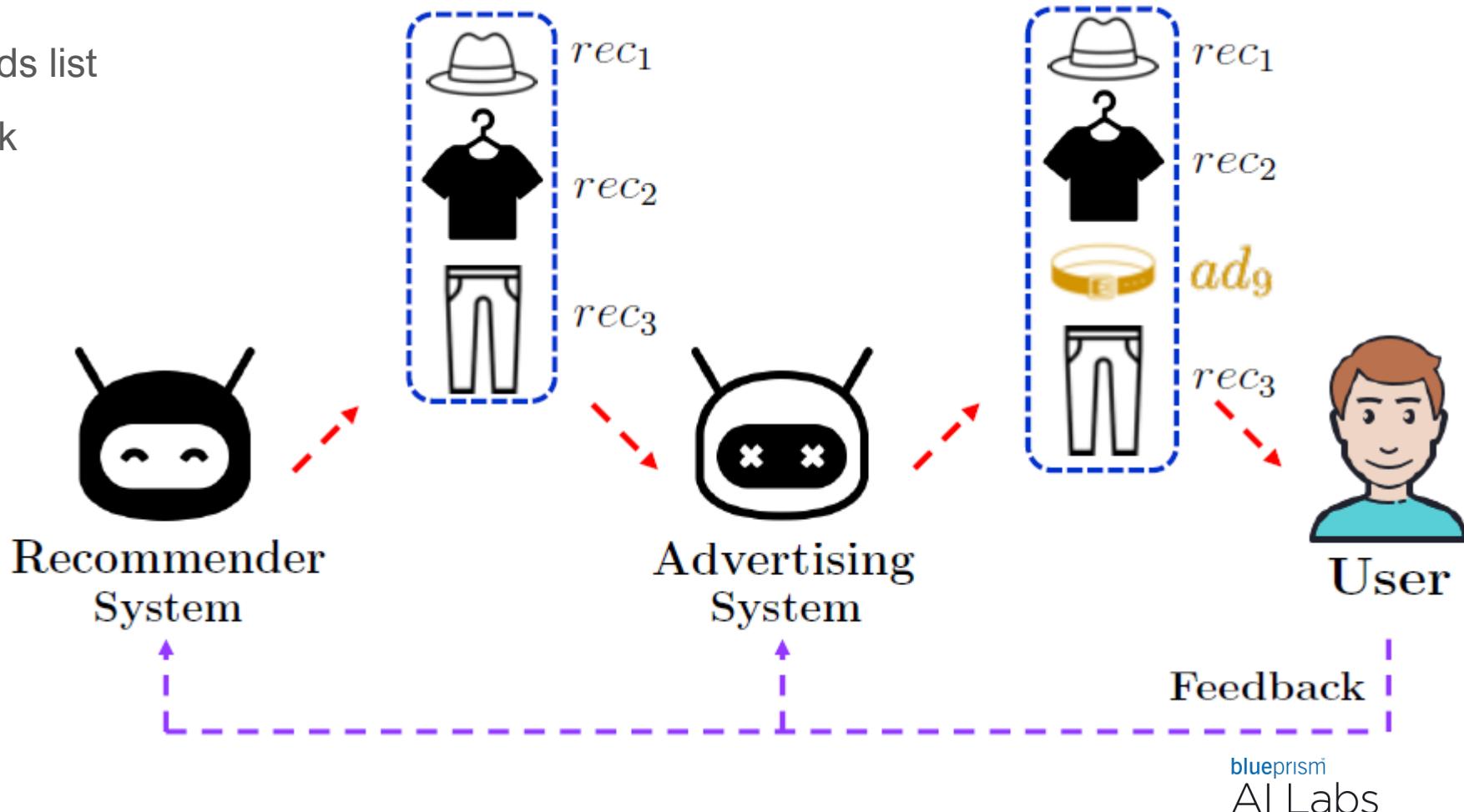
- Goal:
  - maximize the advertising revenue
  - minimize the negative influence of ads on user experience
- Challenge:
  - interpolate an ad?
  - the optimal location
  - the optimal ad



# Jointly Learning to Recommend and Advertise

## Systems Update

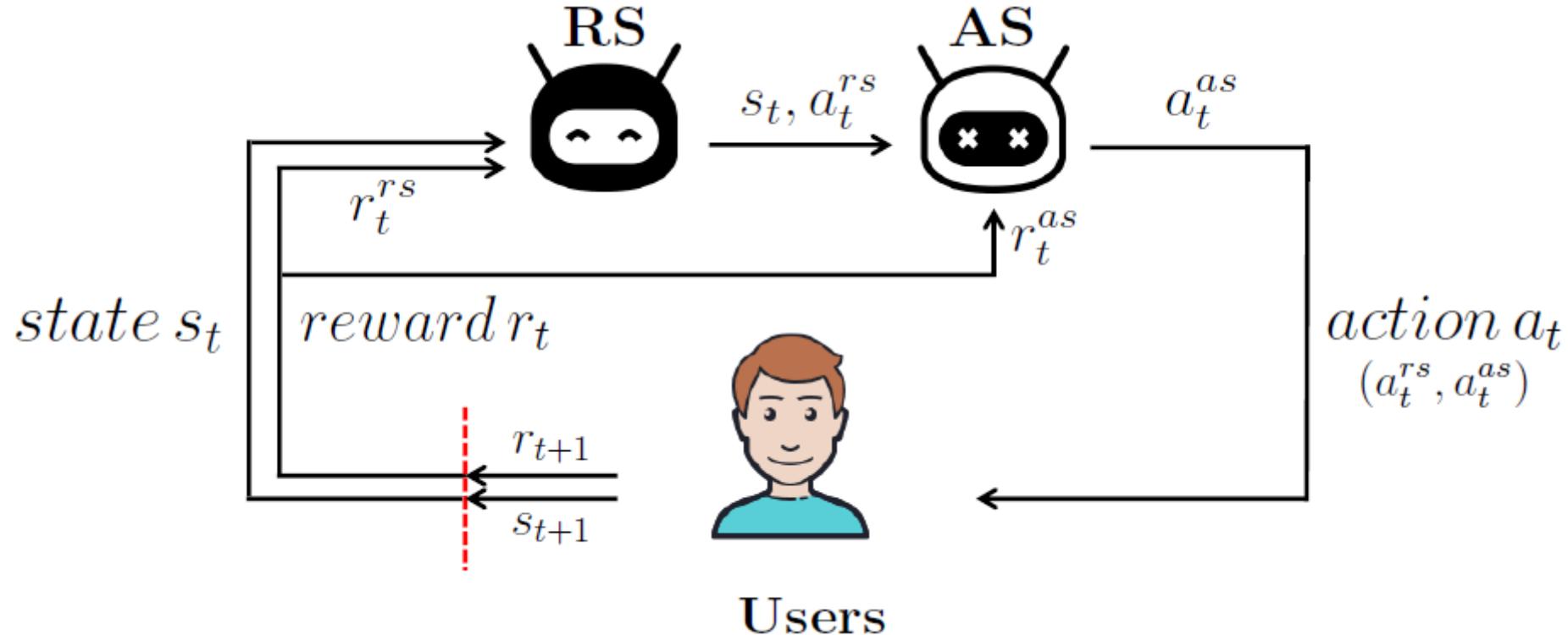
- Target User:
  - browses the mixed rec-ads list
  - provides her/his feedback



# Jointly Learning to Recommend and Advertise

## Two-level DQN

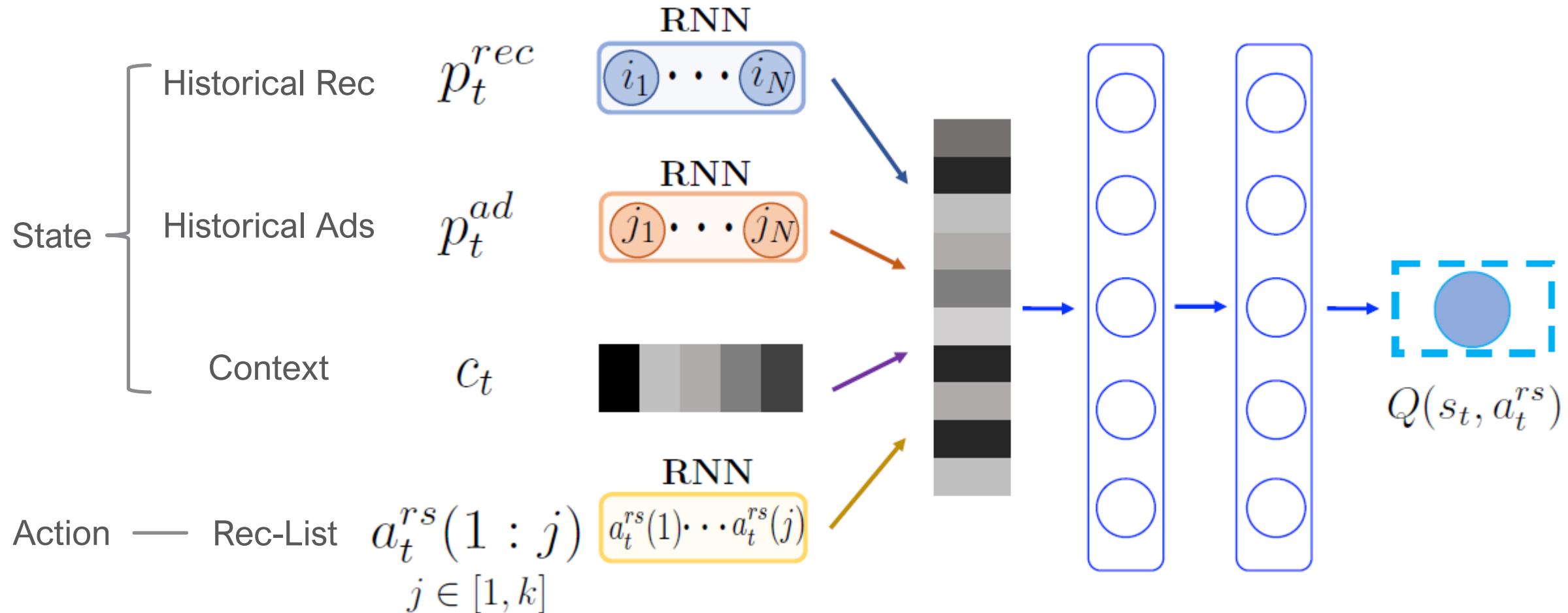
- 1st-level: recommender system (RS)
- 2nd-level: advertising system (AS)



# Jointly Learning to Recommend and Advertise

## Cascading DQN for RS

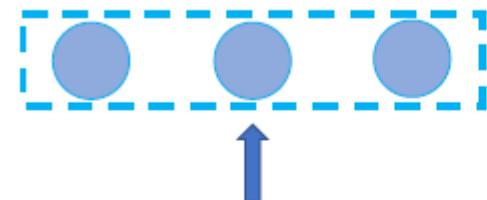
(Generates a list by sequentially selecting items in a cascading manner)



# Jointly Learning to Recommend and Advertise

## Novel DQN for AS

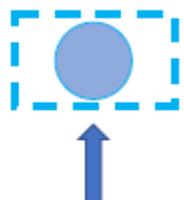
$$Q(s_t, a)^1 \dots Q(s_t, a)^{k+1}$$



$state s_t$   
(a)

Conventional DQN  
Architecture

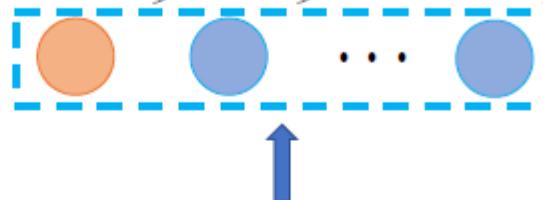
$$Q(s_t, a_t)$$



$state s_t$   $action a_t$   
(b)

Conventional DQN  
Architecture

$$Q(s_t, a_t^{ad})_0 \dots Q(s_t, a_t^{ad})_{k+1}$$

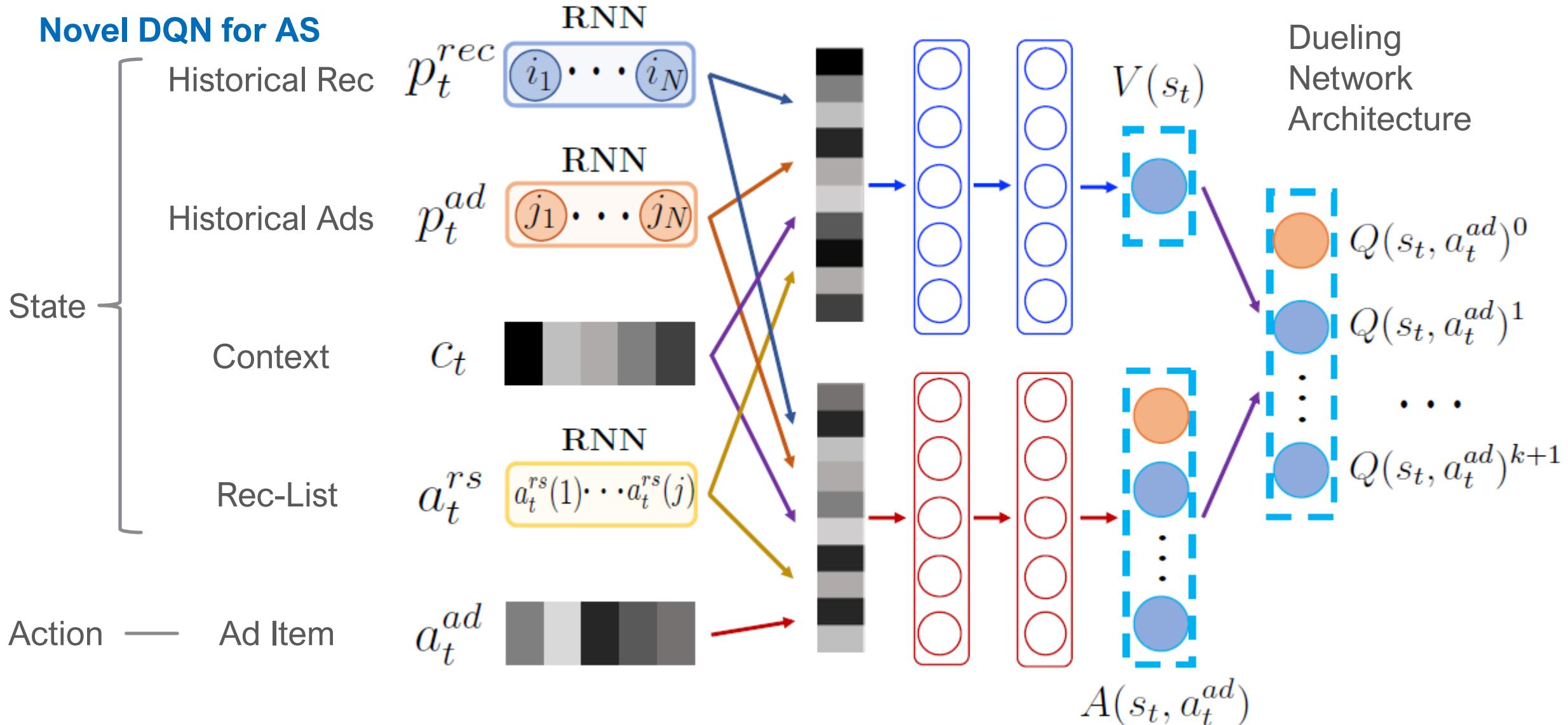


$state s_t$   $action a_t^{ad}$   
(c)

Proposed DQN  
Architecture

- interpolate an ad?
- the optimal location
- the optimal ad

# Jointly Learning to Recommend and Advertise





# IR APPLICATIONS USING REINFORCE

Dell Zhang

## Reinforcement Learning to Rank with Markov Decision Process

Zeng Wei, Jun Xu\*, Yanyan Lan, Jiafeng Guo, Xueqi Cheng  
CAS Key Lab of Network Data Science and Technology,  
Institute of Computing Technology, Chinese Academy of Sciences  
zengwei@software.ict.ac.cn, {junxu, lanyanyan, guojiafeng, cxq}@ict.ac.cn

Session 5A: Retrieval Models and Ranking 3

SIGIR'17, August 7-11, 2017, Shinjuku, Tokyo, Japan

<https://dl.acm.org/doi/10.1145/3077136.3080775>

## Adapting Markov Decision Process for Search Result Diversification

Long Xia, Jun Xu\*, Yanyan Lan, Jiafeng Guo, Wei Zeng, Xueqi Cheng  
CAS Key Lab of Network Data Science and Technology,  
Institute of Computing Technology, Chinese Academy of Sciences  
{xialong, zengwei}@software.ict.ac.cn, {junxu, lanyanyan, guojiafeng, cxq}@ict.ac.cn

Session 3B: Learning to Rank

SIGIR '20, July 25–30, 2020, Virtual Event, China

<https://dl.acm.org/doi/10.1145/3397271.3401148>

## Reinforcement Learning to Rank with Pairwise Policy Gradient

Jun Xu<sup>1,2</sup>, Zeng Wei<sup>3</sup>, Long Xia<sup>4</sup>, Yanyan Lan<sup>5</sup>, Dawei Yin<sup>3</sup>, Xueqi Cheng<sup>5</sup>, Ji-Rong Wen<sup>1,2</sup>

<sup>1</sup>Gaoling School of Artificial Intelligence, Renmin University of China

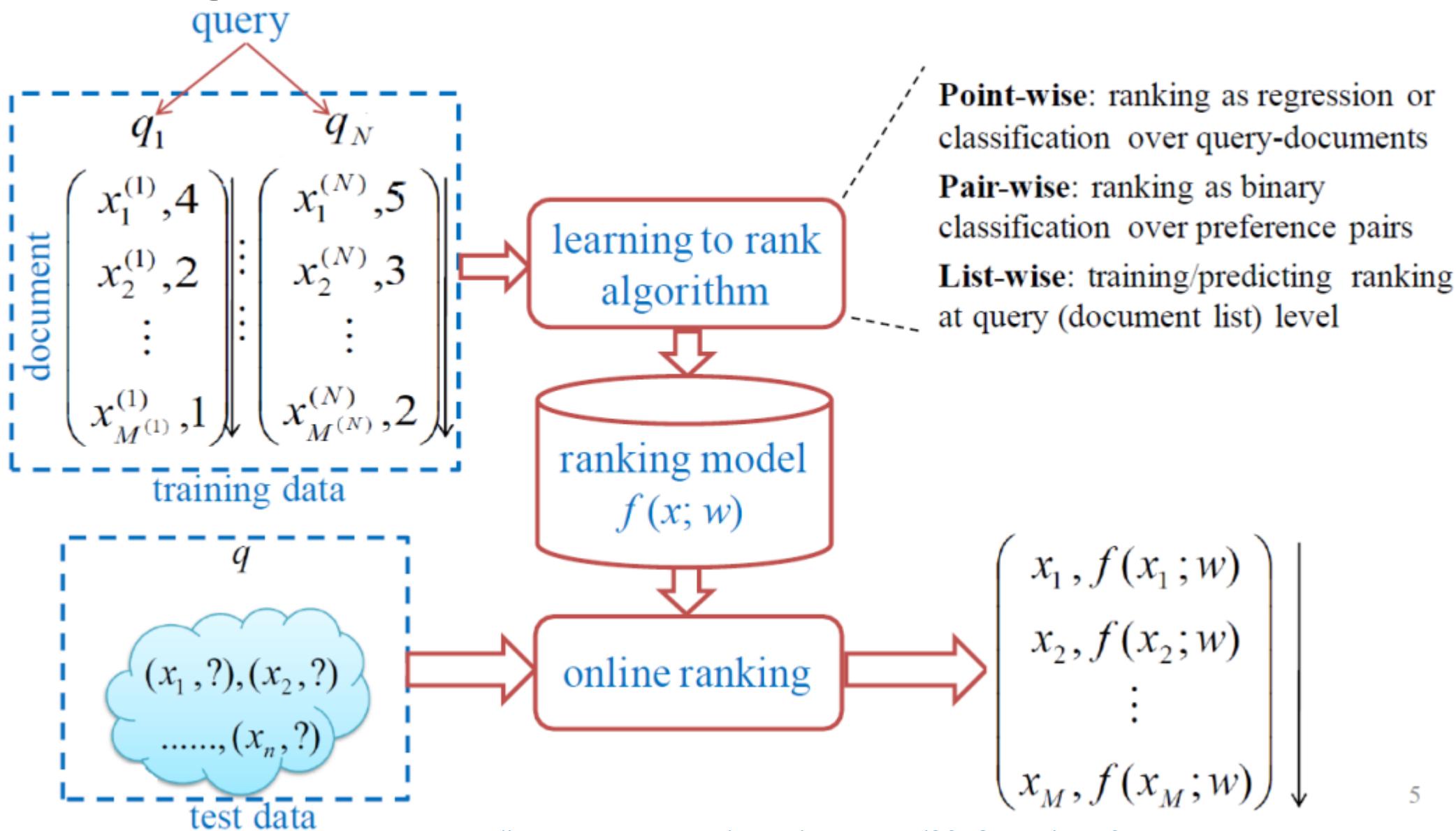
<sup>2</sup>Beijing Key Laboratory of Big Data Management and Analysis Methods

<sup>3</sup>Baidu Inc.; <sup>4</sup>School of Information Technology, York University;

<sup>5</sup>CAS Key Lab of Network Data Science & Technology, Institute of Computing Technology

{junxu, jrwen}@ruc.edu.cn; weizeng@baidu.com; longxia@yorku.ca; yindawei@acm.org; {yanyanlan, cxq}@ict.ac.cn

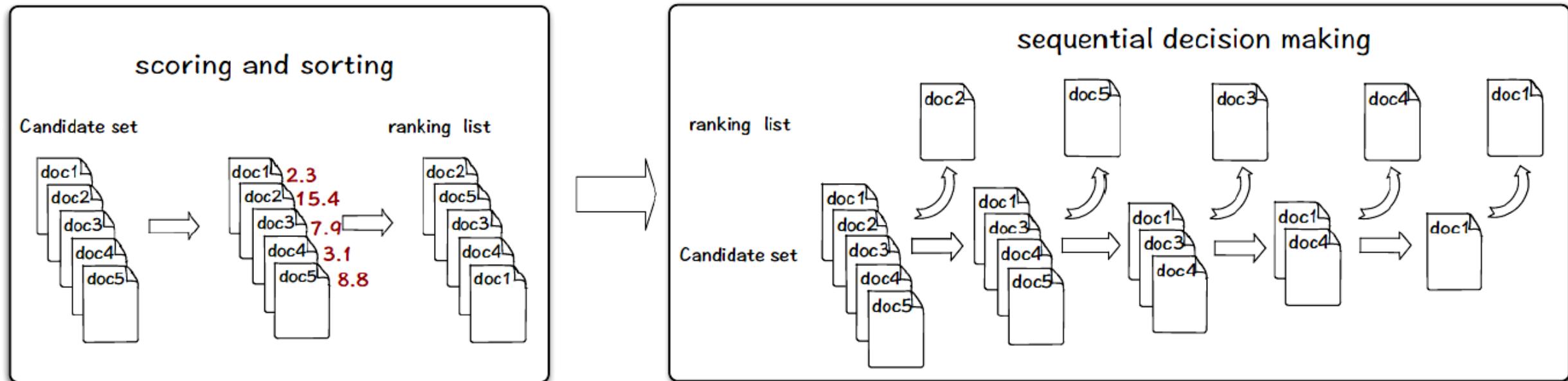
# Learning to Rank



# Learning to Rank

## Sequential Decision Making: *Beyond Independence Relevance*

- Document Ranking Positions
- Document Interdependencies

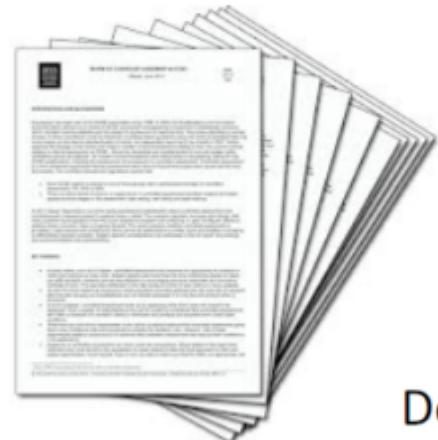


# Learning to Rank

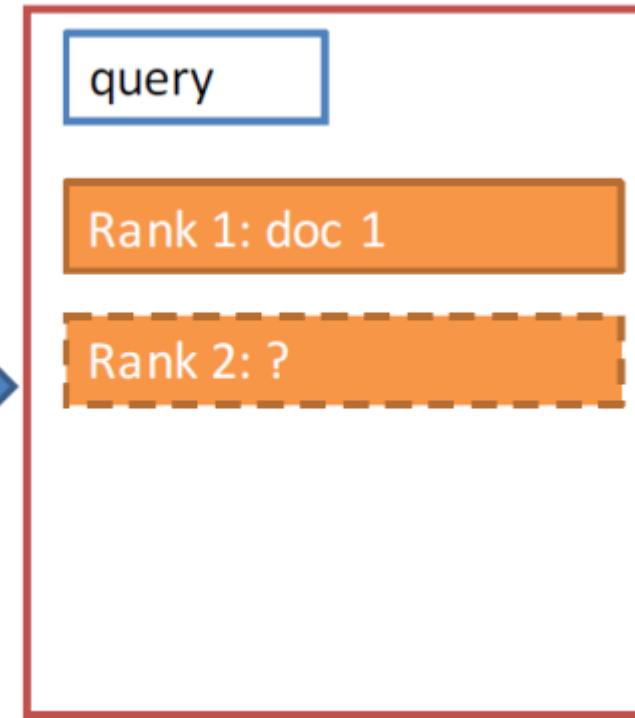
## Ranking as MDP

- Mimic user top-down browsing behaviours
- Model dynamic information needs with MDP state

Candidate document set

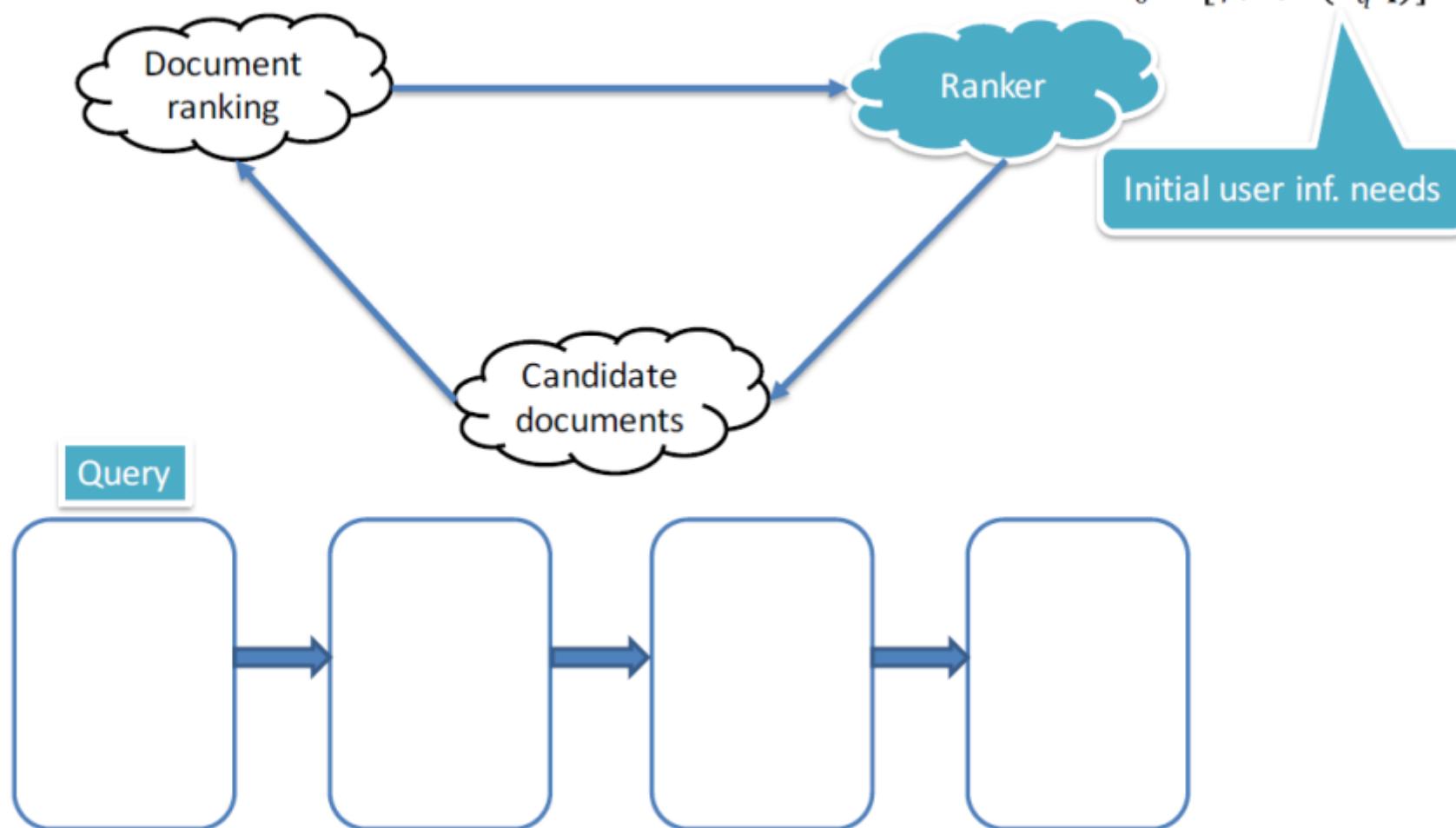


Decide which doc  
should be selected for  
the 2<sup>nd</sup> position



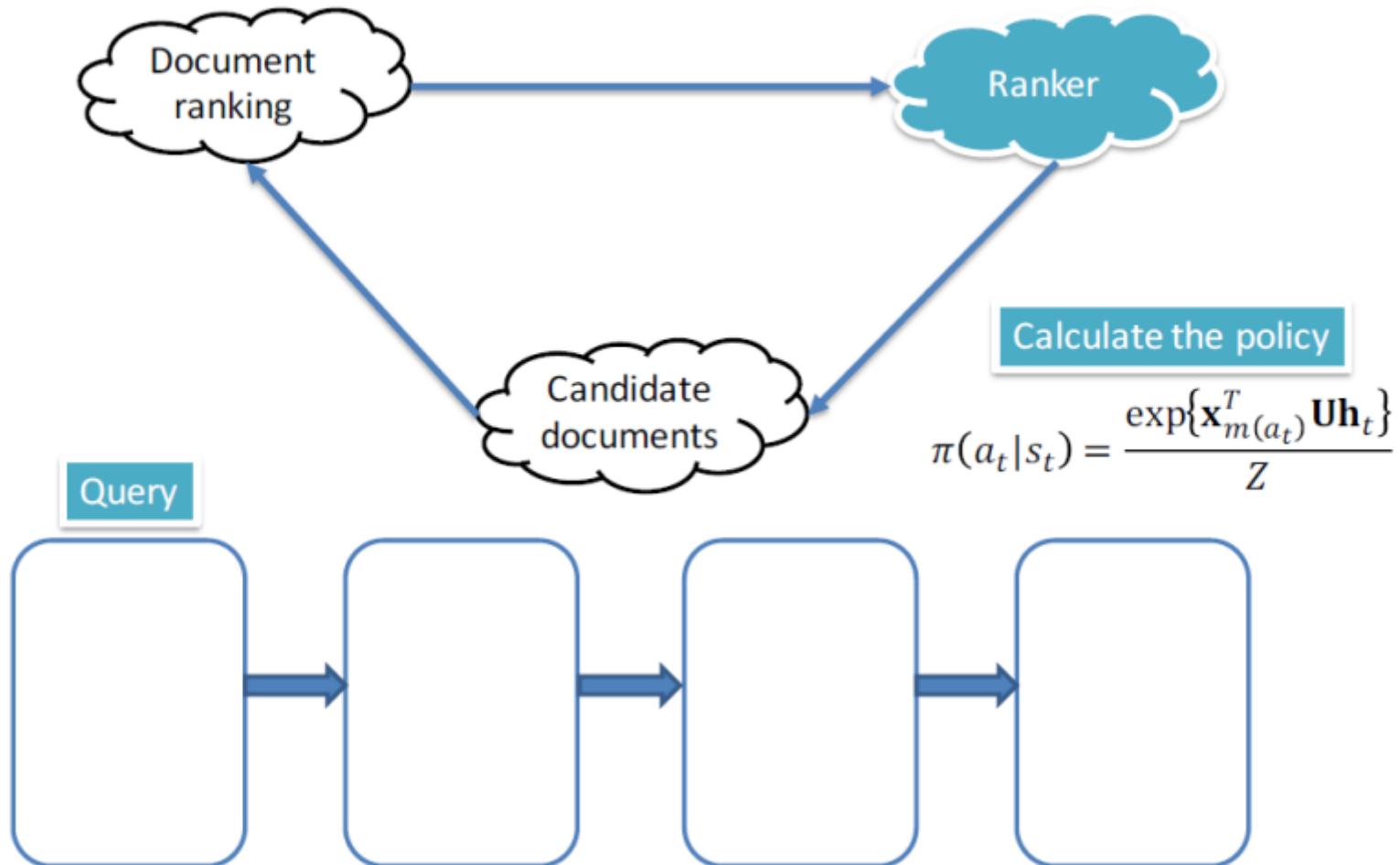
# Learning to Rank

Ranking Process: Initialize State



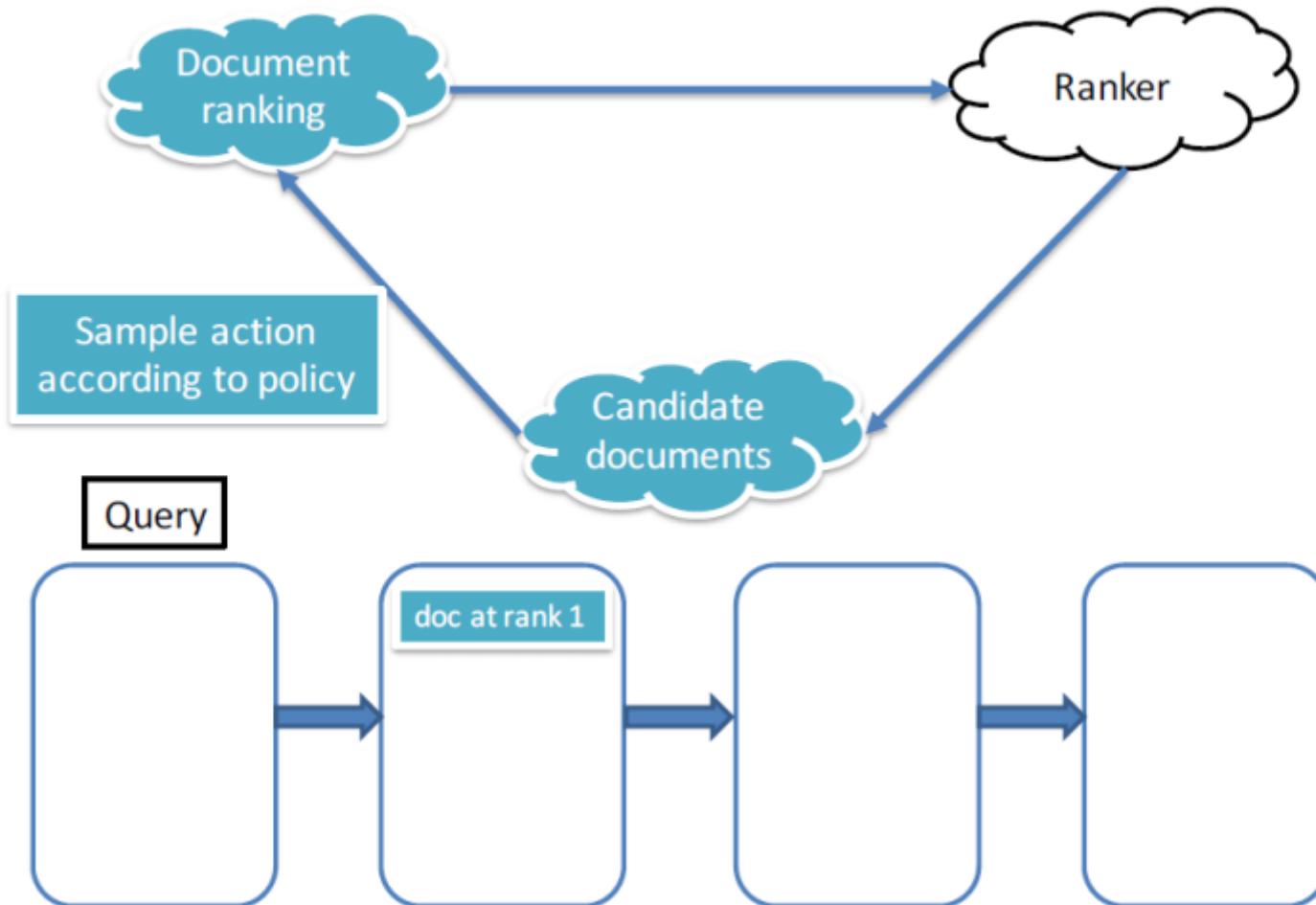
# Learning to Rank

Ranking Process: Policy



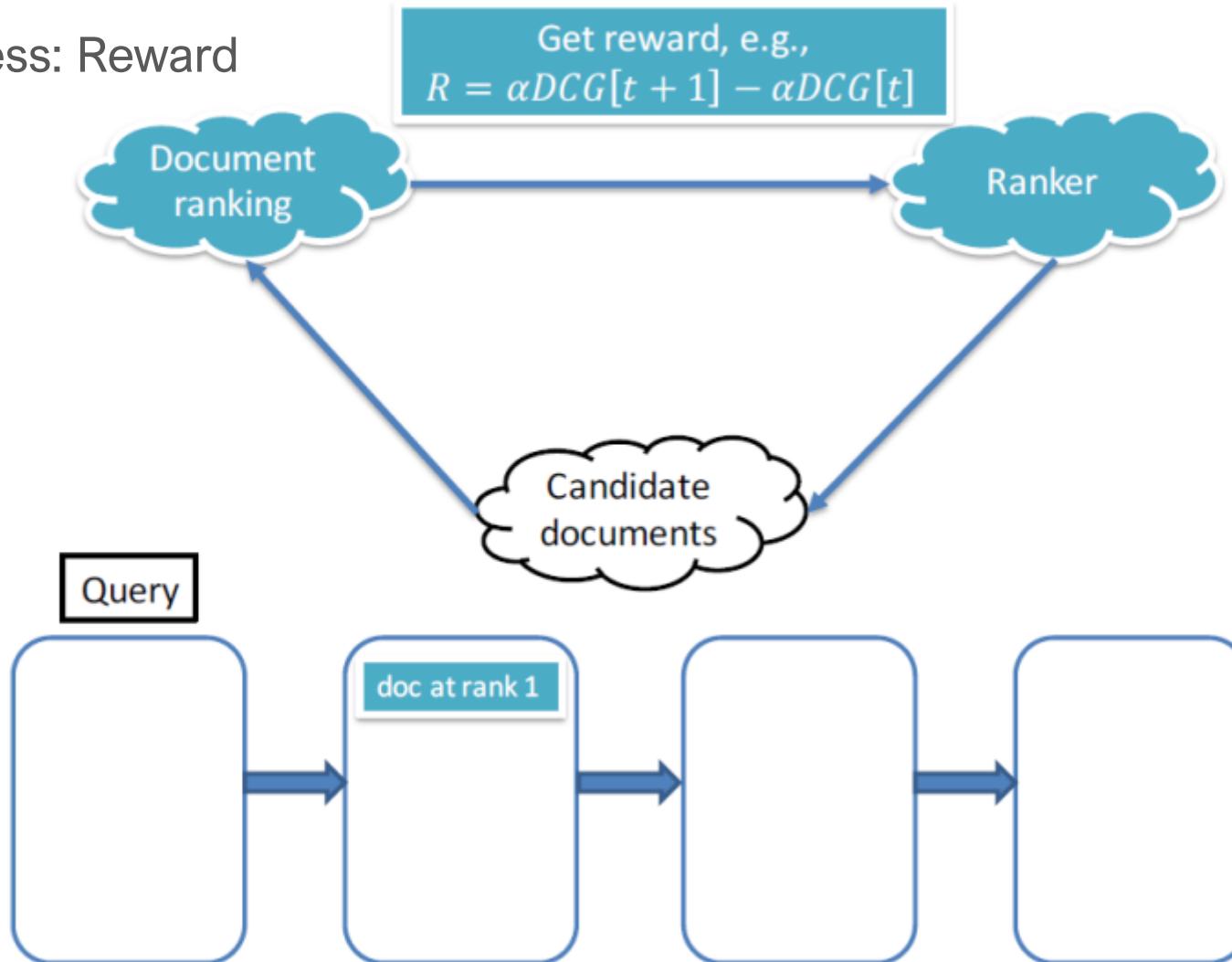
# Learning to Rank

Ranking Process: Action



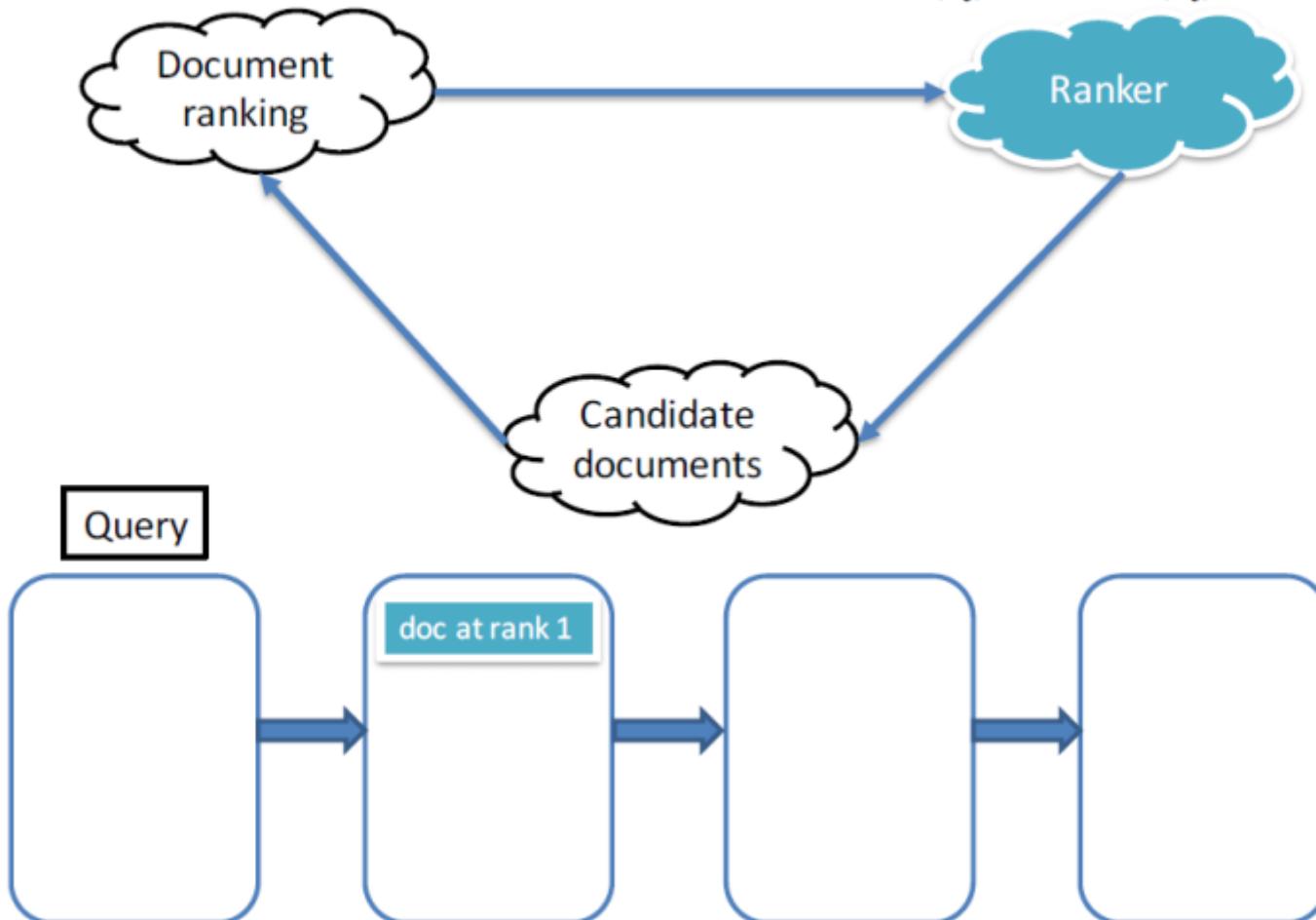
# Learning to Rank

Ranking Process: Reward



# Learning to Rank

Ranking Process: State Transition

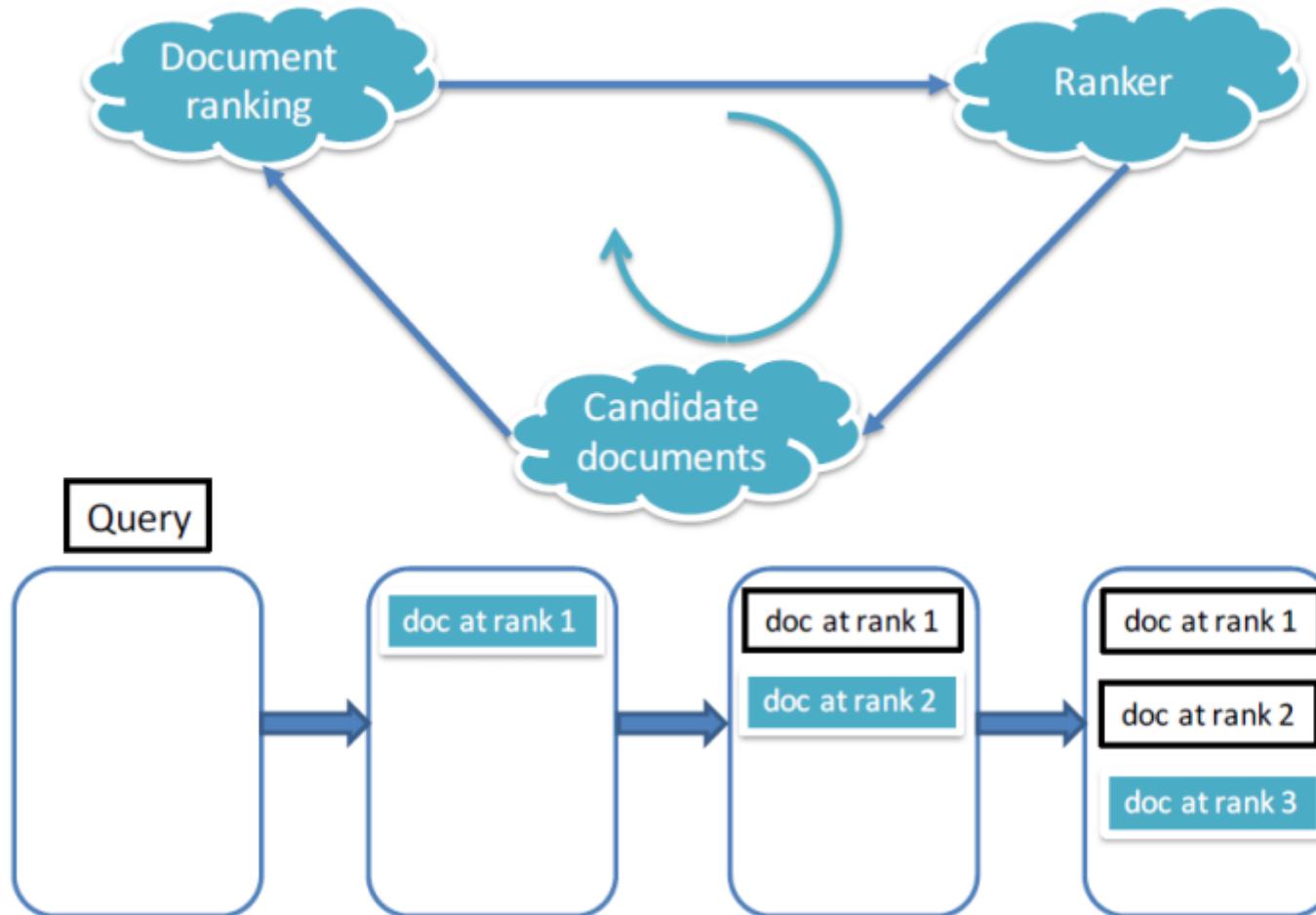


Update ranked list, candidate set, and latent vector

$$s_{t+1} = [Z_t \oplus \{\mathbf{x}_{m(a_t)}\}, X_t \setminus \{\mathbf{x}_{m(a_t)}\}, \sigma(\mathbf{Vx}_{m(a_t)} + \mathbf{Wh}_t)]$$

# Learning to Rank

Ranking Process: Iterate



# Learning to Rank

## Text Retrieval

MDP configuration

text retrieval

state at time  $t$ :  $s_t$

$$s_t = [t, X_t]$$

initial state  $s_0$

$s_0 = \mathcal{S}(Q, X) = [0, X]$ , where  $X$  contains all docs

state transition  $\mathcal{T}(s_t, a_t)$

$$\mathcal{T}(s_t, a_t) = [t + 1, X_t \setminus \{d_{m(a_t)}\}]$$

reward  $\mathcal{R}(s_t, a_t)$

$$\mathcal{R}(s_t, a_t) = \begin{cases} 2^{y_{m(a_t)}} - 1 & t = 0 \\ (2^{y_{m(a_t)}} - 1)/\log_2(t + 1) & t > 0 \end{cases}$$

query-document  
relevance features

policy  $\pi(a_t | s_t; \theta)$

$$\pi(a_t | s_t; \theta) = \frac{\exp\left\{\theta^T \phi(Q, d_{m(a_t)})\right\}}{\sum_{a \in A(s_t)} \exp\left\{\theta^T \phi(Q, d_{m(a)})\right\}},$$

where  $\phi(Q, d_{m(a_t)})$  is ranking feature vector

$$\phi(Q, d_{m(a_t)}) = \sum_{a \in A(s_t)} \pi(a | s_t; \theta) \phi(Q, d_{m(a)})$$

Gradient  $\nabla \log \pi(a | s)$

# Learning to Rank

## Search Result Diversification

user's perceived utility from preceding documents

MDP configuration

search result diversification

state at time  $t$ :  $s_t$

$s_t = [Z_t, X_t, h_t]$ , where  $Z_t$  is ranked doc list,  $h_t$  is latent vector

initial state  $s_0$

$s_0 = \mathcal{S}(Q, X) = [\emptyset, X, \sigma(V_q q)]$ , where  $q$  is query embedding

state transition  $\mathcal{T}(s_t, a_t)$

$\mathcal{T}(s_t, a_t) = [Z_t \cup \{d_{m(a_t)}\}, X_t \setminus \{d_{m(a_t)}\}, \sigma(Vx_{m(a_t)} + Wh_t)]$ , where  $x_{m(a_t)}$  is doc embedding, and  $V, W$  are parameters

reward  $\mathcal{R}(s_t, a_t)$

$\mathcal{R}(s_t, a_t) = \alpha\text{-DCG}[t+1] - \alpha\text{-DCG}[t]$ , where  $\alpha\text{-DCG}[0] = 0$

policy  $\pi(a_t | s_t; \theta)$

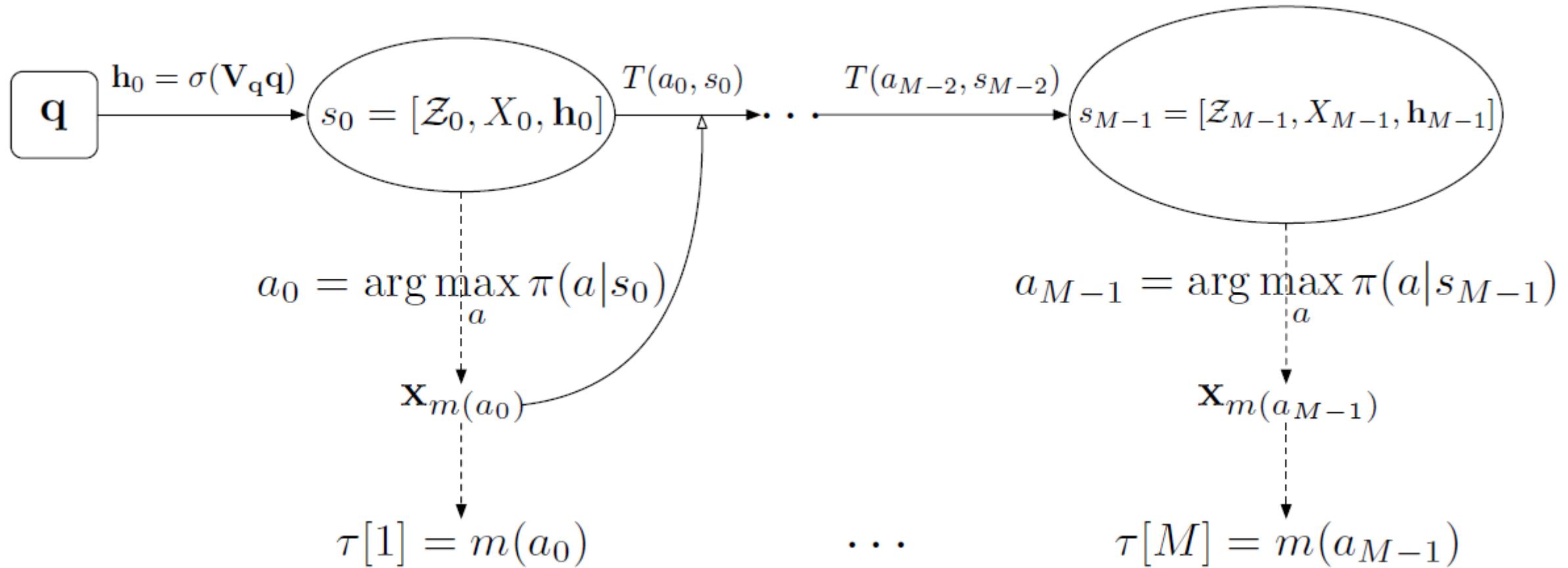
$$\pi(a_t | s_t) = \frac{\exp\left\{x_{m(a_t)}^T U h_t\right\}}{\sum_{a \in A(s_t)} \exp\left\{x_{m(a)}^T U h_t\right\}},$$
 where  $U$  is the parameter matrix

Gradient  $\nabla \log \pi(a | s)$

refer to [35] for details

# Learning to Rank

## Search Result Diversification



# Learning to Rank

## Search Result Diversification

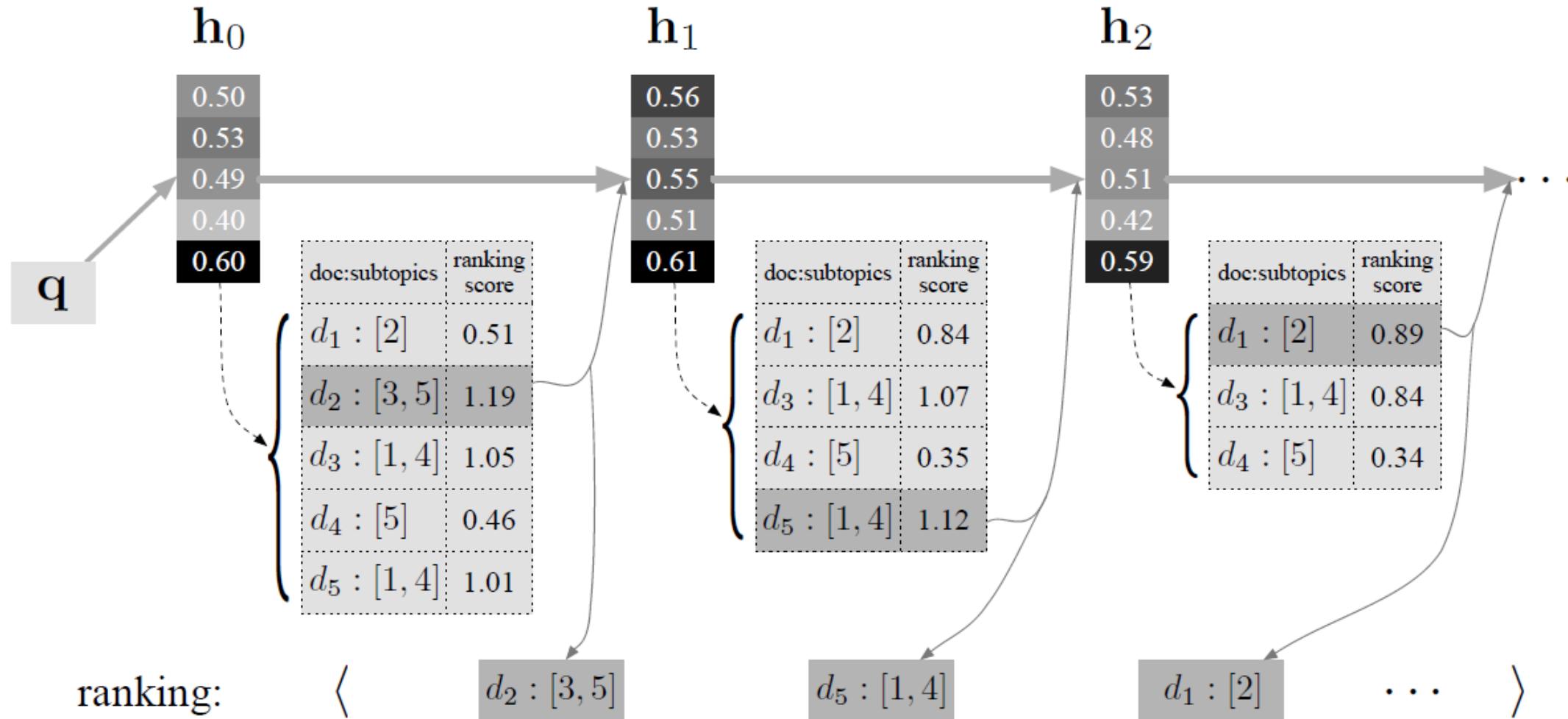
- Consider query #93 “raffles” which is ambiguous and multi-faceted (containing five subtopics).
  - The five top ranked documents in the Clueweb09 collection are as follows. The subtopics covered by each of the documents are shown in the square brackets.

[1] : “Raffles Hotel in Singapore”	$d_1$ : “Stamford Raffles – Wikipedia, the free encyclopedia” [2]
[2] : “Sir Stamford Raffles”	$d_2$ : “Fundraiser Raffle Ideas” [3, 5]
[3] : “organizing a raffle”	$d_3$ : “Luxury Hotel Guide   Raffles Hotels” [1, 4]
[4] : “the Raffles hotel in Dubai”	$d_4$ : “National Corvette Museum – Corvette Raffles” [5]
[5] : “car raffles”	$d_5$ : “Raffles Hotels and Resorts” [1, 4]

- The search results may contain redundant information. It is necessary to consider the novelty of a document given preceding documents.
- Goal: to cover as much subtopics as possible with a few documents.

# Learning to Rank

## Search Result Diversification



# Learning to Rank

## Advantages of Using RL

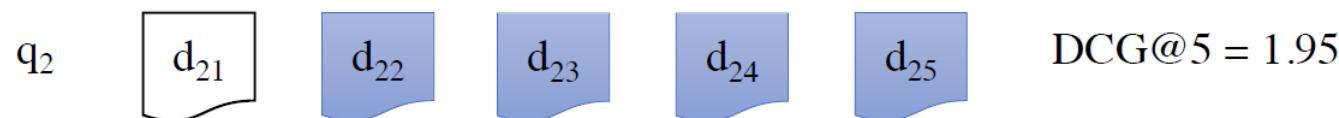
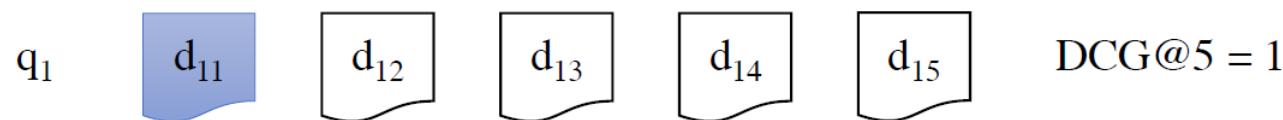
- **Text Retrieval**
  - Utilizes the IR measures calculated at all the ranking positions as supervision information for training.
  - Optimizes the IR measure directly on the training data without any approximation or upper bounding.
- **Search Result Diversification**
  - Unified criterion (additional utility user can perceive) for selecting documents at each iteration
  - End-to-end learning of the diverse ranking model: no need of handcrafted features.
  - Utilizes both the immediate rewards and the long-term returns as the supervision information during training.

# Learning to Rank

## Standard REINFORCE Algorithm

- Problem

- The gradient is estimated according to the sampled document list weighted by its *absolute* performance score.
- However, the performance of the optimal ranking for a difficult query may be lower than the performance of a suboptimal ranking for an easy query. Since the easy queries always generate document lists with high-performance scores while policy gradient treats them equally, the trained ranking model may be biased to easy queries.



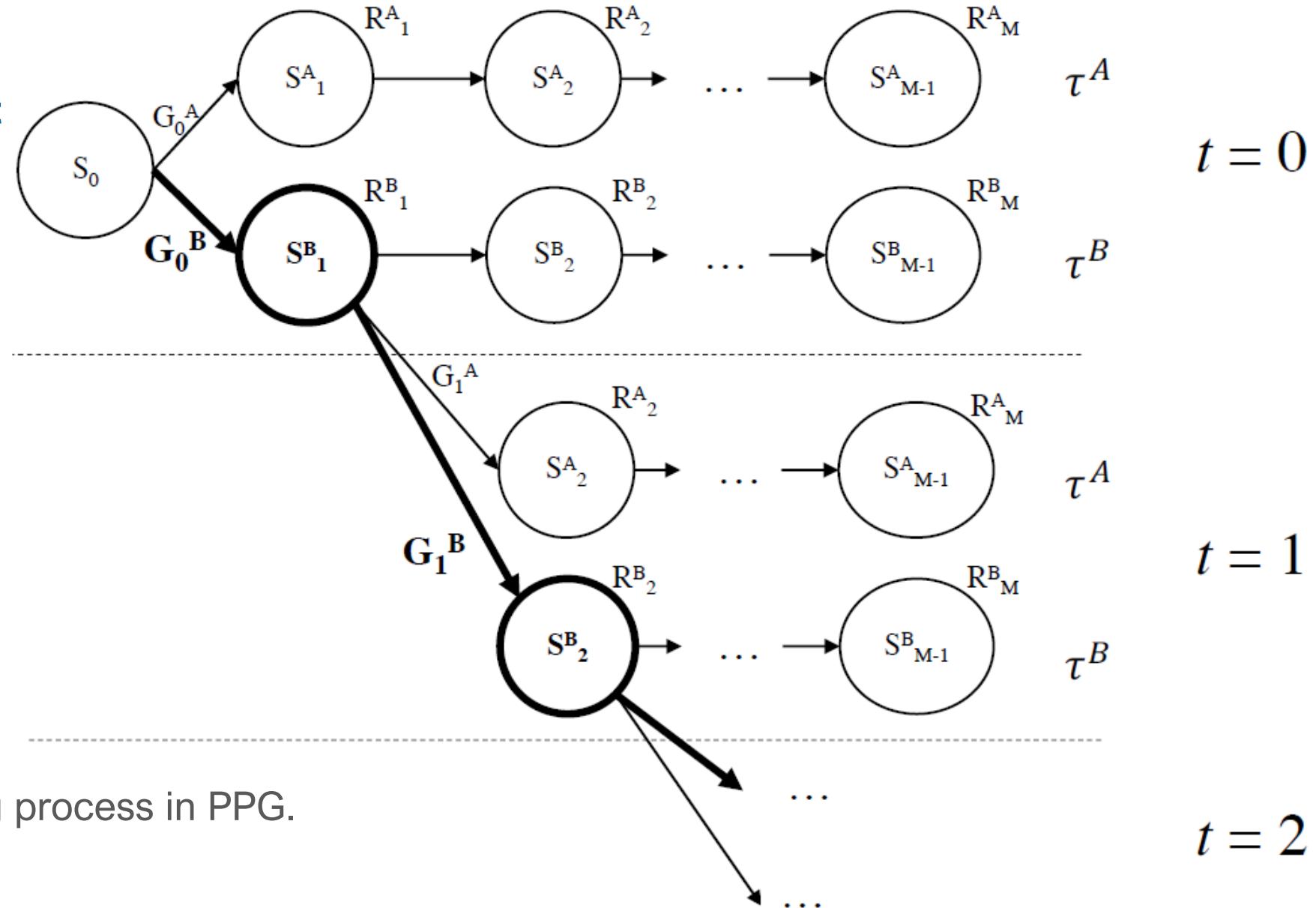
# Learning to Rank

## Pairwise Policy Gradient

- Motivation
  - The *relative ordering nature of IR ranking*: (1) we care more about the relative goodness of a document compared to others, rather than its absolute relevance score; (2) the comparisons should only be conducted between the documents within the same query.
  - Estimating a reasonable baseline function for IR ranking is very difficult, due to the extremely huge and separated state space.
- Solution
  - Pairwise Policy Gradient (PPG) is a policy gradient algorithm using intra-query pairwise comparisons.
  - In its learning procedure, PPG repeats the process of sampling a pair of document lists starting from the same state (and thus within the same query), comparing these two lists in terms of their performance scores and gradient directions, and finally update the MDP parameters with the comparison results.

# Learning to Rank

## Pairwise Policy Gradient



# Learning to Rank

## Pairwise Policy Gradient

- A variation of REINFORCE with baseline.
- Good theoretical convergence properties:
  - unbiased
  - low variance

---

### Algorithm 1 Pairwise Policy Gradient (PPG)

---

**INPUT:** Training set  $D = \{(Q^{(n)}, X^{(n)}, Y^{(n)})\}_{n=1}^N$ , learning rate  $\eta$ , discount factor  $\gamma$ , and reward function  $R$

- 1: Initialize  $\theta \leftarrow$  random values
- 2: **repeat**
- 3:    $\Delta\theta \leftarrow 0$
- 4:   **for all**  $(Q, X, Y) \in D$  **do**
- 5:     Initial state  $S = S(Q, X)$  {Init state with  $Q$ }
- 6:     **for**  $t = 0$  **to**  $M - 1$  **do**
- 7:        $\tau^A = \{S, A_t, R_{t+1}^A, S_{t+1}^A, \dots, S_{M-1}^A, A_{M-1}, R_M^A\} \sim \pi(\theta)$
- 8:        $G^A \leftarrow \sum_{k=1}^{M-t} \gamma^{k-1} R_{t+k}^A$  {long term return of  $\tau^A$ }
- 9:        $\tau^B = \{S, B_t, R_{t+1}^B, S_{t+1}^B, \dots, S_{M-1}^B, B_{M-1}, R_M^B\} \sim \pi(\theta)$
- 10:        $G^B \leftarrow \sum_{k=1}^{M-t} \gamma^{k-1} R_{t+k}^B$  {long term return of  $\tau^B$ }
- 11:        $\Delta\theta \leftarrow \Delta\theta + (G^A - G^B) \cdot (\nabla \log \pi(A_t | S_t; \theta) - \nabla \log \pi(B_t | S_t; \theta))$  {according to Equation (5)}
- 12:      $S \leftarrow \begin{cases} S_{t+1}^A & G^A \geq G^B \\ S_{t+1}^B & \text{otherwise} \end{cases}$
- 13:   **end for**
- 14: **end for**
- 15:    $\theta \leftarrow \theta + \eta \Delta\theta$
- 16: **until** converge
- 17: **return**  $\theta$

---

# A Reinforcement Learning Framework for Relevance Feedback

Ali Montazer alghaem  
Center for Intelligent Information  
Retrieval  
University of Massachusetts Amherst  
[montazer@cs.umass.edu](mailto:montazer@cs.umass.edu)

Hamed Zamani  
Microsoft  
[hazamani@microsoft.com](mailto:hazamani@microsoft.com)

James Allan  
Center for Intelligent Information  
Retrieval  
University of Massachusetts Amherst  
[allan@cs.umass.edu](mailto:allan@cs.umass.edu)

# Relevance Feedback

## Problem Statement and Motivation

- Let  $Q = \{q_1, q_2, \dots, q_m\}$  be a query with  $m$  terms issued by a user  $u$ , and  $F = \{D_1, D_2, \dots, D_k\}$  be a set of  $k$  (real or pseudo-) relevance feedback documents.
- The objective is to optimize the evaluation function  $\text{eval}(u, Q, \mathcal{M}(\theta_Q^*; \mathcal{C}))$ , where  $\mathcal{M}$  and  $\mathcal{C}$  denote the retrieval model and document collection, respectively.  
The function  $\text{eval}$  is usually *non-differentiable*, e.g.,  $AP$ ,  $nDCG$ , and  $\alpha$ - $nDCG$ .
- We assume that the retrieval model  $\mathcal{M}$  is given and focus on estimating an accurate query model for the query  $Q$ :  $\theta_Q^* = \mathcal{R}(Q, F; \mathcal{C}, \Omega)$  where  $\mathcal{R}$  denotes the relevance feedback model with the parameter set  $\Omega$ .
- Existing (pseudo-) relevance feedback methods do not directly optimize the ultimate objective, i.e., retrieval performance.

# Relevance Feedback

## RF as MDP

- Environment:  $\mathcal{C}, \mathcal{M}$
- Agent: the query model that expands the query and reweights the query terms
- State:  $Q, F; \mathcal{C}, \Omega^t$
- Action: To sample  $K$  terms  $S^t = \{w_1, w_2, \dots, w_K\}$  from the query model distribution learned by  $\mathcal{R}$  without replacement (i.e.,  $w \sim \theta_Q^t = \mathcal{R}(Q, F; \mathcal{C}, \Omega^t)$ ), and then *linearly interpolate* the normalised  $K$ -term query model with the original query model (estimated using MLE).

$$p(w|\hat{\theta}_Q^t) = \alpha p(w|\theta_Q^{MLE}) + (1 - \alpha) \underbrace{p'(w|\theta_Q^t)}_{\vdots} \\ p'(w|\theta_Q^t) = \begin{cases} \frac{p(w|\theta_Q^t)}{Z} & \text{if } w \in S^t \\ 0 & \text{otherwise} \end{cases}$$

- Reward:

$$\text{Reward}(Q, t) = \text{eval}(u, Q, \mathcal{M}(\hat{\theta}_Q^t; \mathcal{C})) - \text{eval}(u, Q, \mathcal{M}(\hat{\theta}_Q^{t-1}; \mathcal{C}))$$

# Relevance Feedback

## Policy Network Architecture

- We cast the problem of estimating a relevance feedback distribution to estimating a relevance feedback weight for each term using a softmax operator.

$$p(w|\mathcal{R}) = \frac{\exp(\mathcal{R}'(w, Q, F; C, \Omega))}{\sum_{w' \in V} \exp(\mathcal{R}'(w', Q, F; C, \Omega))}$$

- A single model  $\mathcal{R}'$  is trained to take information related to a candidate feedback term  $w$  as input and produce a feedback weight.

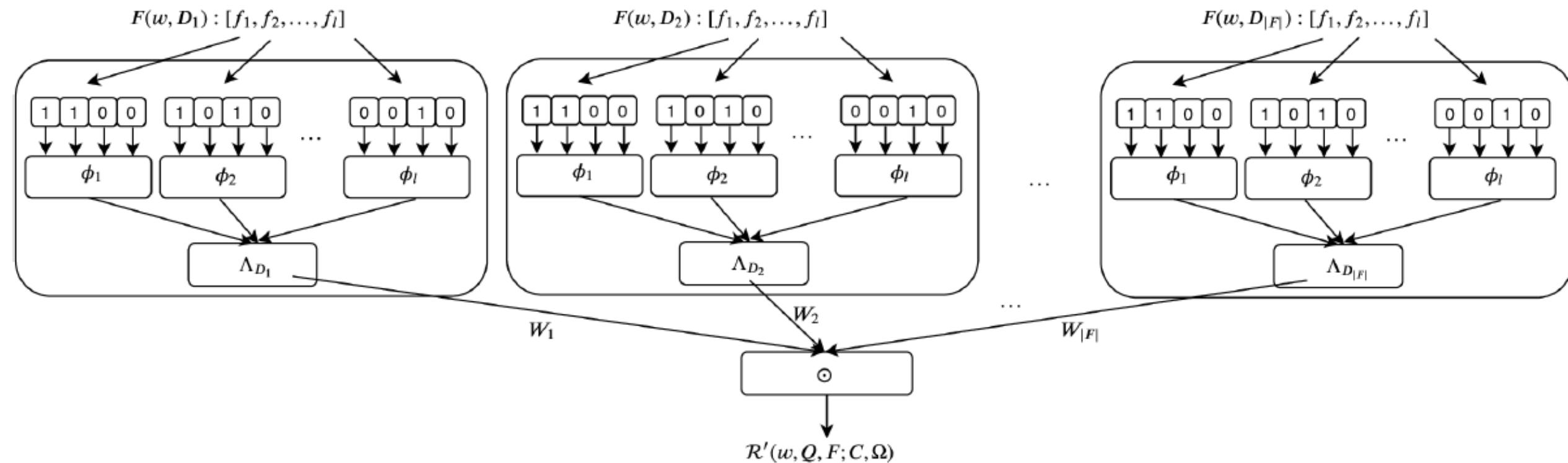
$$\mathcal{R}'(w, Q, F; C, \Omega) = \odot_{i=1}^{|F|} (\Lambda(\phi_1(w, Q, D_i; C), \dots, \phi_l(w, Q, D_i; C)), \mathcal{W}_i)$$

- We use a component-based architecture that consists of a set of sub-networks  $\phi_1, \phi_2, \dots, \phi_l$ , each modelling a different aspect of the feedback term.
- The component  $\Lambda$  aggregates these  $l$  representations.  $\mathcal{W}_i$  denotes the relevance score of the document  $D_i$  to the query  $Q$ .
- Finally, the compositionality component  $\odot$  aggregates all the representations obtained from all feedback documents and produces a single real number as the weight of the feedback term  $w$ .

# Relevance Feedback

## Policy Network Architecture

$$\mathcal{R}'(w, Q, F; C, \Omega) = \odot_{i=1}^{|F|} (\Lambda(\phi_1(w, Q, D_i; C), \dots, \phi_l(w, Q, D_i; C)), W_i)$$



# Relevance Feedback

## Example

The top 10 expansion terms added to the query “dinosaur” (topic 14)

- RM3
- RML<sub>MAP</sub>

RM3	Weight	RML <sub>MAP</sub>	Weight
price	0.1611	tyrannosaurus	0.1185
parti	0.1377	stegosaurus	0.1132
regular	0.0788	jurass	0.0982
99	0.0574	triceratop	0.0532
rex	0.0506	skull	0.0512
dino	0.0474	prehistor	0.0488
toy	0.0447	mammal	0.0477
birthday	0.0388	skeleton	0.0467
prehistor	0.0333	bone	0.0392
game	0.0326	plush	0.0372

# Relevance Feedback

## Example

The top 10 expansion terms added to the query “403b” (topic 151).

- $\text{RML}_{\alpha-nDCG@20}$
- $\text{RML}_{MAP}$

<b>Subtopic 1:</b>	What is a 403b plan?		
<b>Subtopic 2:</b>	What is the difference between 401k and 403b retirement plans?		
<b>Subtopic 3:</b>	What are the withdrawal limitations for a 403b retirement plan?		
<b>RML <math>\alpha-nDCG@20</math></b>	<b>Weight</b>	<b>RML <math>MAP</math></b>	<b>Weight</b>
401k	0.3817	choos	0.1645
limit	0.1062	portion	0.1563
portion	0.0973	roth	0.1301
guidanc	0.0743	ira	0.1222
requir	0.0557	rollov	0.0761
ira	0.0468	individu	0.0585
incom	0.0397	retir	0.0492
offer	0.0336	estat	0.0420
rollov	0.0328	organ	0.0318
individu	0.0180	minor	0.0277

# IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models

Jun Wang  
University College London  
j.wang@cs.ucl.ac.uk

Lantao Yu, Weinan Zhang\*  
Shanghai Jiao Tong University  
wnzhang@sjtu.edu.cn

Yu Gong, Yinghui Xu  
Alibaba Group  
renji.xyh@taobao.com

Benyou Wang, Peng Zhang  
Tianjin University  
pzhang@tju.edu.cn

Dell Zhang  
Birkbeck, University of London  
dell.z@ieee.org

## Two Schools of IR Thinking

### Generative models of IR

- **Pros:** theoretically sound and very successful in modelling features
- **Cons:**
  - difficult in leveraging relevancy signals from largely observable data, e.g., links, clicks
  - typically not trainable

### Discriminative models of IR

- **Pros:** able to learn a retrieval ranking function implicitly from labelled data
- **Cons:** lack a principled way of
  - obtaining useful features,
  - gathering helpful signals from the massive unlabeled data available, e.g., text statistics, the collection distribution

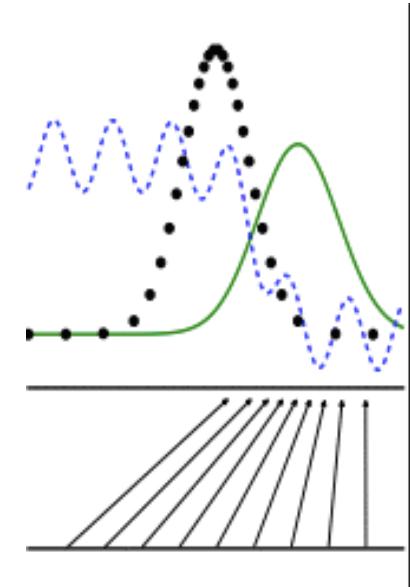
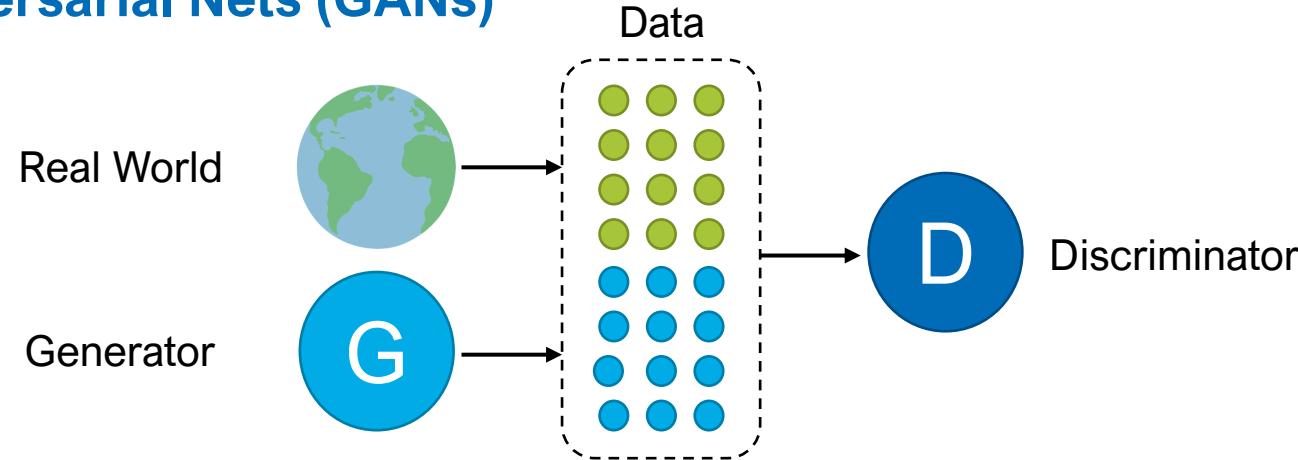
# IRGAN

## To Take Advantage of Both

- The generative model learns to fit the relevance distribution over documents  $p_{\text{true}}(d|q, r)$  via the signal from the discriminative model.
- The discriminative model is able to exploit the unlabelled data selected by the generative model to achieve a better estimation  $f_\phi(q, d)$  for document ranking.

# IRGAN

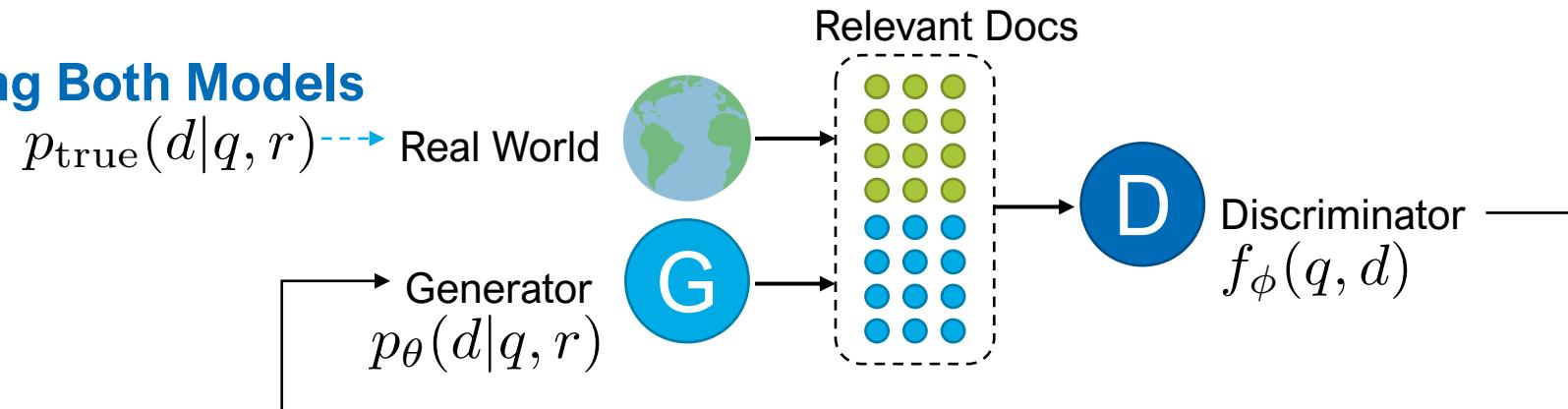
## Generative Adversarial Nets (GANs)



- Minimax game between a **discriminator** & a **generator**:
  - Discriminator (D) tries to correctly distinguish the true data and the fake model-generated data
  - Generator (G) tries to generate high-quality data to fool discriminator
- G & D can be implemented via neural networks
- Ideally, when D cannot distinguish the true and generated data, G nicely fits the true underlying data distribution

# IRGAN

## A Minimax Game Unifying Both Models

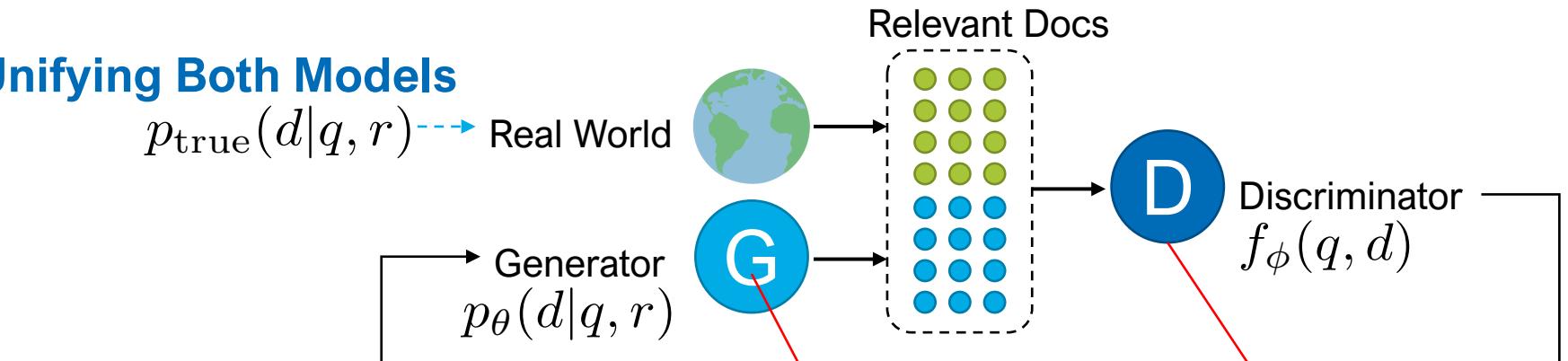


- **Underlying true relevance distribution**  $p_{\text{true}}(d|q, r)$  depicts the user's relevance preference distribution over the candidate documents with respect to his submitted query
  - **Training set:** A set of samples from  $p_{\text{true}}(d|q, r)$
- **Generative retrieval model**  $p_\theta(d|q, r)$ 
  - Goal: approximate the true relevance distribution
- **Discriminative retrieval model**  $f_\phi(q, d)$ 
  - Goal: distinguish between relevant and non-relevant documents

# IRGAN

## A Minimax Game Unifying Both Models

$$p_{\text{true}}(d|q, r) \xrightarrow{\text{Real World}}$$



Objective

$$J^{G^*, D^*} = \min_{\theta} \max_{\phi} \sum_{n=1}^N \left( \mathbb{E}_{d \sim p_{\text{true}}(d|q_n, r)} [\log D(d|q_n)] + \mathbb{E}_{d \sim p_{\theta}(d|q_n, r)} [\log(1 - D(d|q_n))] \right)$$

where  $p_{\theta}(d|q, r) = \frac{\exp(g_{\theta}(q, d))}{\sum_{d'} \exp(g_{\theta}(q, d'))}$

$$D(d|q) = \sigma(f_{\phi}(d, q)) = \frac{\exp(f_{\phi}(d, q))}{1 + \exp(f_{\phi}(d, q))}$$

# IRGAN

## Optimizing Generative Retrieval as Policy Gradient

- Samples documents from the whole document set to fool its opponent

$$\begin{aligned}\theta^* &= \arg \min_{\theta} \sum_{n=1}^N \left( \mathbb{E}_{d \sim p_{\text{true}}(d|q_n, r)} [\log \sigma(f_{\phi}(d, q_n))] + \right. \\ &\quad \left. \mathbb{E}_{d \sim p_{\theta}(d|q_n, r)} [\log(1 - \sigma(f_{\phi}(d, q_n)))] \right) \\ &= \arg \max_{\theta} \sum_{n=1}^N \underbrace{\mathbb{E}_{d \sim p_{\theta}(d|q_n, r)} [\log(1 + \exp(f_{\phi}(d, q_n)))]}_{\text{Generator as Policy}} \underbrace{[\log(1 + \exp(f_{\phi}(d, q_n)))]}_{\text{denoted as } J^G(q_n)} \text{ Reward Term}\end{aligned}$$

- REINFORCE (with Advantage Function)

$$\log(1 + \exp(f_{\phi}(d, q_n))) - \mathbb{E}_{d \sim p_{\theta}(d|q_n, r)} [\log(1 + \exp(f_{\phi}(d, q_n)))]$$

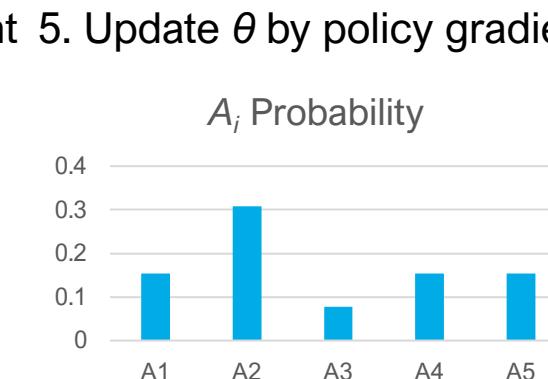
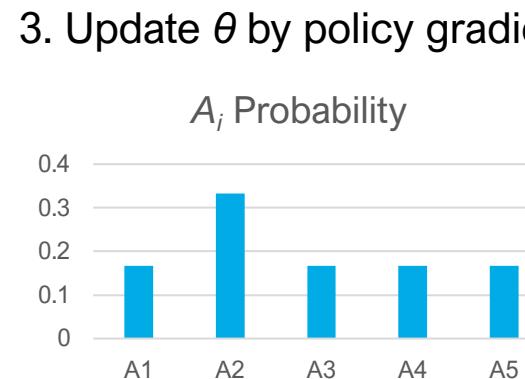
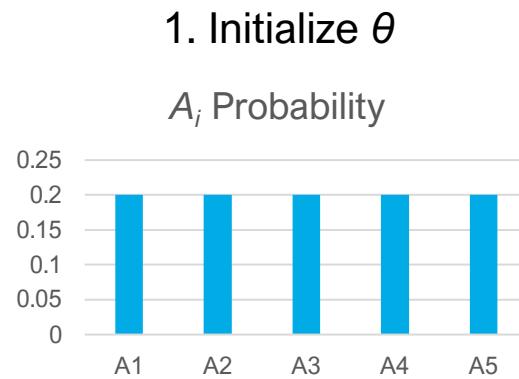
## Optimizing Generative Retrieval as Policy Gradient

Likelihood Ratio  $\nabla_{\theta} J^G(q_n)$

$$\begin{aligned}&= \nabla_{\theta} \mathbb{E}_{d \sim p_{\theta}(d|q_n, r)} [\log(1 + \exp(f_{\phi}(d, q_n)))] \\&= \sum_{i=1}^M \nabla_{\theta} p_{\theta}(d_i|q_n, r) \log(1 + \exp(f_{\phi}(d_i, q_n))) \\&= \sum_{i=1}^M p_{\theta}(d_i|q_n, r) \nabla_{\theta} \log p_{\theta}(d_i|q_n, r) \log(1 + \exp(f_{\phi}(d_i, q_n))) \\&= \mathbb{E}_{d \sim p_{\theta}(d|q_n, r)} [\nabla_{\theta} \log p_{\theta}(d|q_n, r) \log(1 + \exp(f_{\phi}(d, q_n)))] \\&\simeq \frac{1}{K} \sum_{k=1}^K \nabla_{\theta} \log p_{\theta}(d_k|q_n, r) \log(1 + \exp(f_{\phi}(d_k, q_n)))\end{aligned}$$

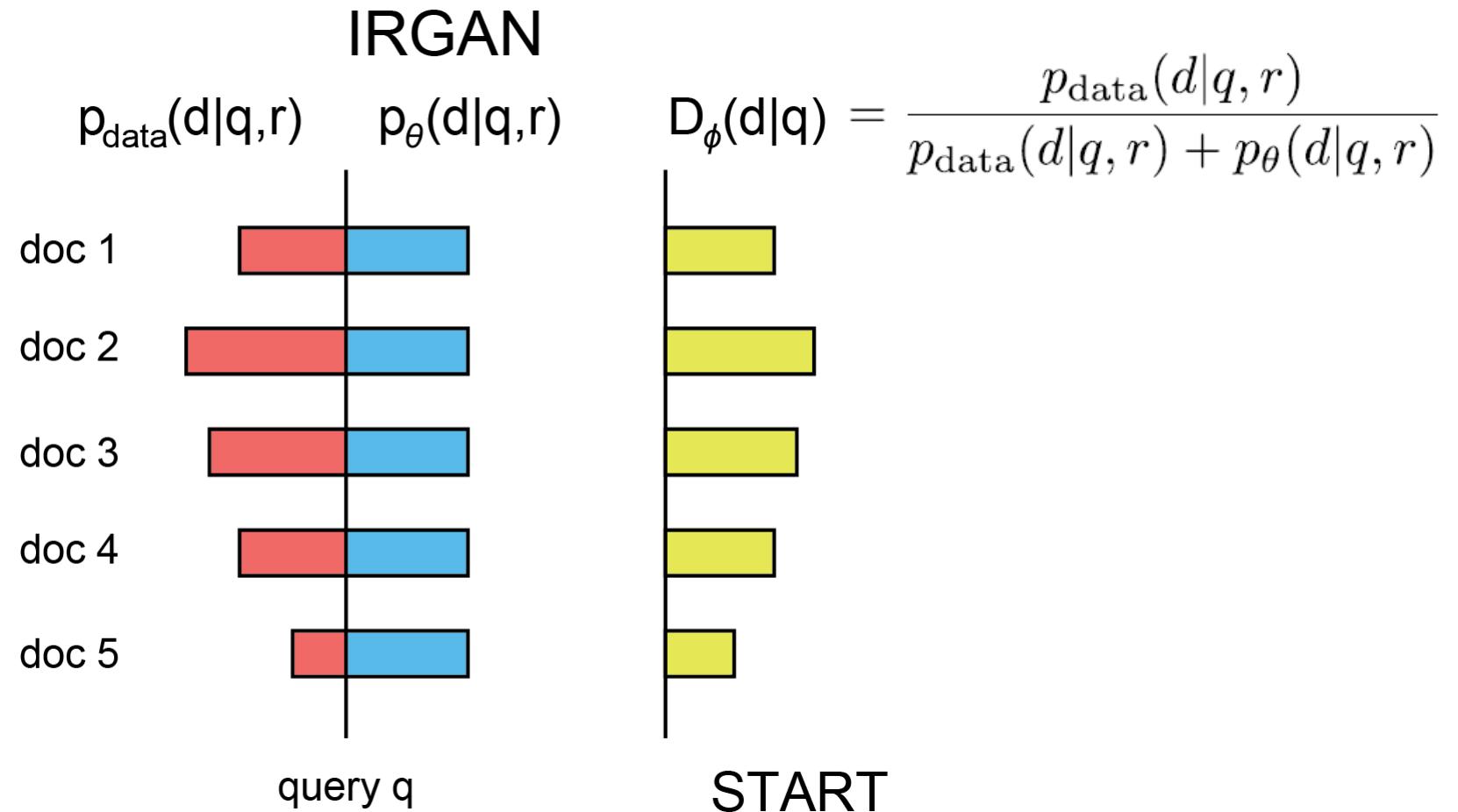
## Optimizing Generative Retrieval as Policy Gradient

- For generative retrieval  $p_\theta(d|q, r)$
- Intuition
  - lower value/reward leads to lower probability of the choice
  - higher value/reward leads to higher probability of the choice
- The one-field 5-category example



# IRGAN

## The Interplay between Generative and Discriminative Retrieval



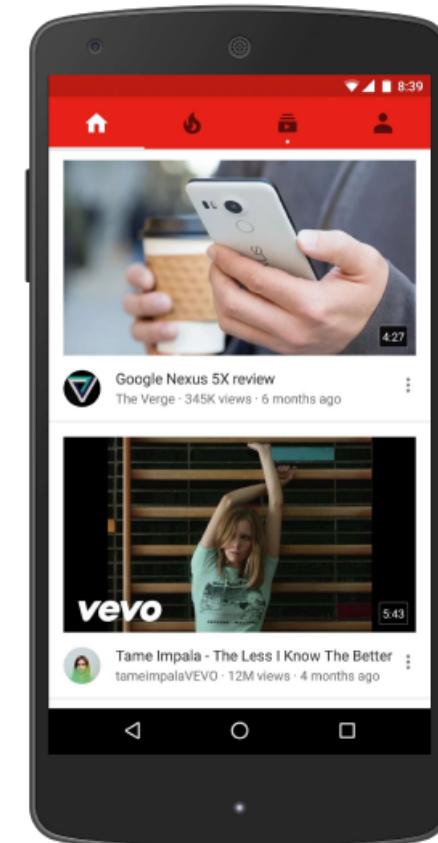
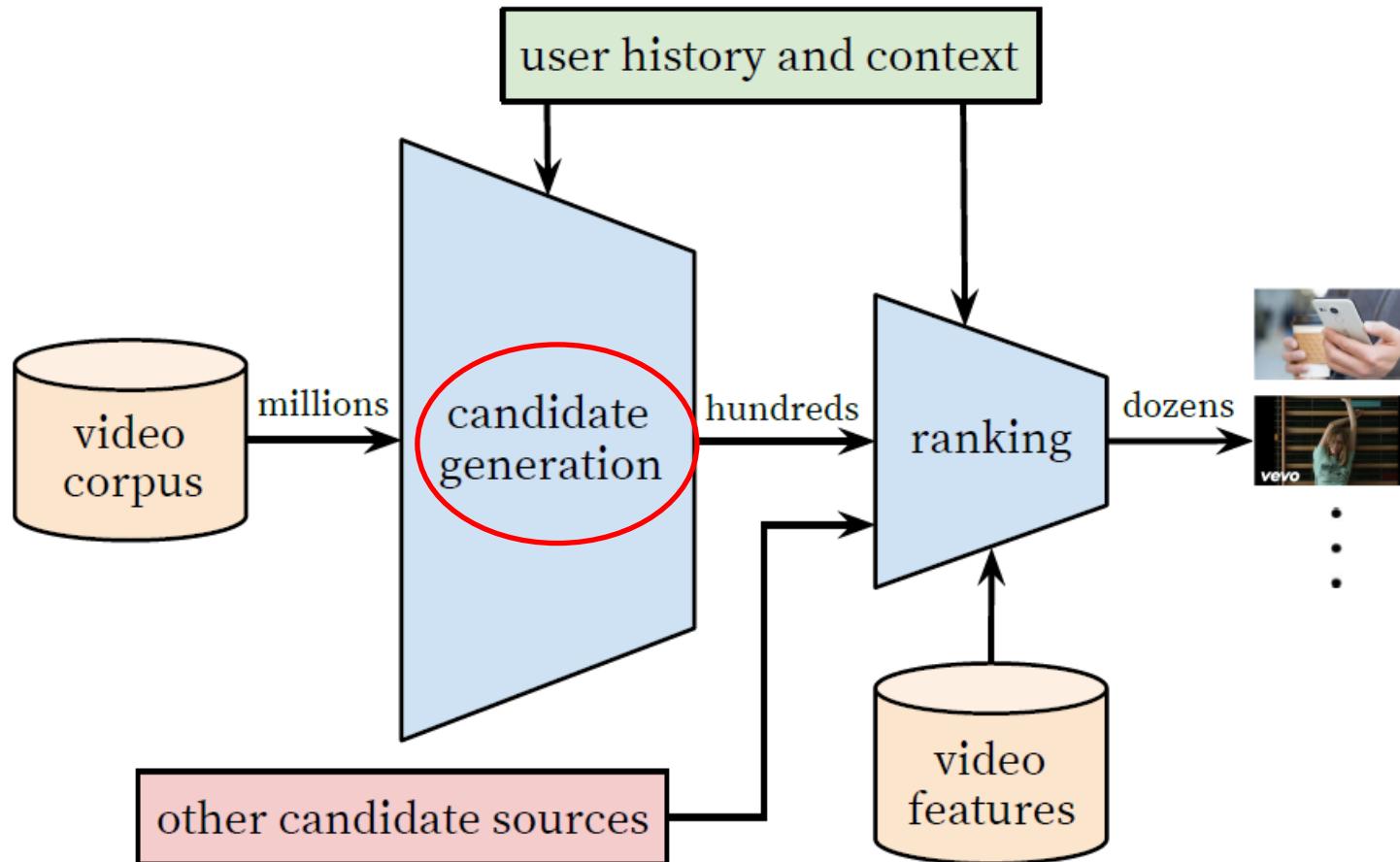
## **Top- $K$ Off-Policy Correction for a REINFORCE Recommender System**

Minmin Chen\*, Alex Beutel\*, Paul Covington\*, Sagar Jain, Francois Belletti, Ed H. Chi  
Google, Inc.  
Mountain View, CA  
minminc,alexbeutel,pcovington,sagarj,belletti,edchi@google.com

# Top-K Off-Policy Correction

## YouTube Recommendations

- The candidate generator is the RL agent



# Top-K Off-Policy Correction

## Data Biases

(target policy  $\pi_\theta \neq$  behaviour policy  $\beta$ )

- A naive policy gradient estimator is no longer unbiased as the gradient requires sampling trajectories from the updated policy  $\pi_\theta$  while the trajectories we collected were drawn from a combination of historical policies  $\beta$ .
  - We collect data with a periodicity of several hours and compute many policy parameter updates before deploying a new version of the policy in production.
  - We learn from batched feedback collected by other recommenders as well, which follow drastically different policies.

# Top-K Off-Policy Correction

- **On-policy:** The agent learned and the agent interacting with the environment is the same.
- **Off-policy:** The agent learned and the agent interacting with the environment is different.



NETFLIX

<https://www.imdb.com/title/tt10048342/>

# Top-K Off-Policy Correction

## On-Policy → Off-Policy

- Importance Sampling

$$\nabla \bar{R}_\theta = E_{\tau \sim \pi_\theta} [R(\tau) \nabla \log \pi_\theta(\tau)]$$

- Problem:  
Use  $\pi_\theta$  to collect data.  
When  $\pi_\theta$  is updated, we have to sample training data again.
- Solution:  
Using the sample from  $\beta$  to train  $\theta$ .  
As  $\beta$  is fixed, we can re-use the sample data.

# Top-K Off-Policy Correction

## On-Policy → Off-Policy

- Importance Sampling

$$E_{x \sim p}[f(x)] \approx \frac{1}{N} \sum_{i=1}^N f(x^i)$$

$x^i$  is sampled from  $p(x)$   
but we only have  $x^i$  sampled from  $q(x)$

$$= \int f(x)p(x)dx = \int f(x) \frac{p(x)}{q(x)} q(x)dx$$

$$= E_{x \sim q}[f(x) \frac{p(x)}{q(x)}]$$

Importance weight

# Top-K Off-Policy Correction

## On-Policy → Off-Policy

- Importance Sampling

$$\nabla \bar{R}_\theta = E_{\tau \sim \beta} \left[ \frac{\pi_\theta(\tau)}{\beta(\tau)} R(\tau) \nabla \log \pi_\theta(\tau) \right]$$

- Sample the data from  $\beta$ .
- Use the data to train  $\pi_\theta$  many times.

# Top-K Off-Policy Correction

On-Policy → Off-Policy

$$\sum_{\tau \sim \pi_\theta} R(\tau) \nabla_\theta \log \pi_\theta(\tau) \approx \sum_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{|\tau|} R_t \nabla_\theta \log \pi_\theta(a_t | s_t) \right]$$

Importance Sampling   $\approx \sum_{\tau \sim \beta} \left[ \sum_{t=0}^{|\tau|} \frac{\pi_\theta(\tau)}{\beta(\tau)} R_t \nabla_\theta \log (\pi_\theta(a_t | s_t)) \right]$

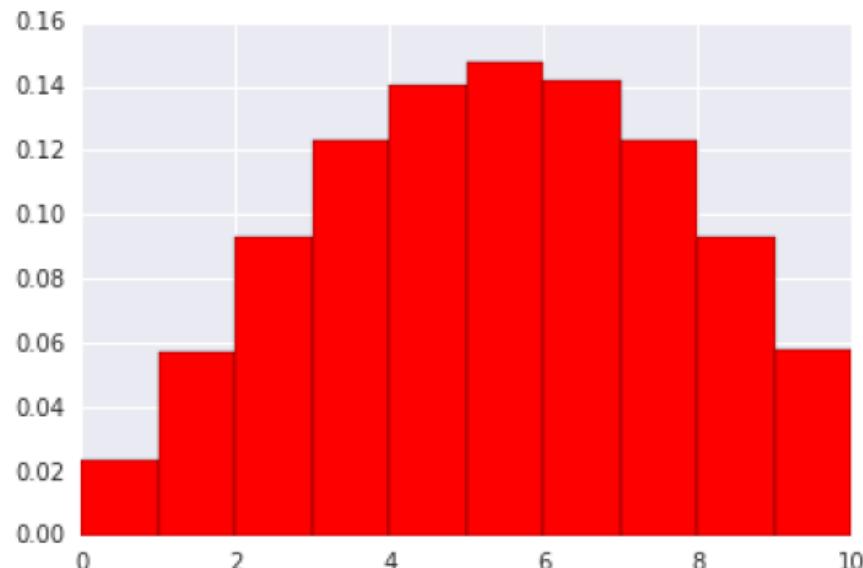
Variance Reduction   $\approx \sum_{\tau \sim \beta} \left[ \sum_{t=0}^{|\tau|} \frac{\pi_\theta(a_t | s_t)}{\beta(a_t | s_t)} R_t \nabla_\theta \log \pi_\theta(a_t | s_t) \right]$

- Ignore the terms after time  $t$
- Take a first-order approximation

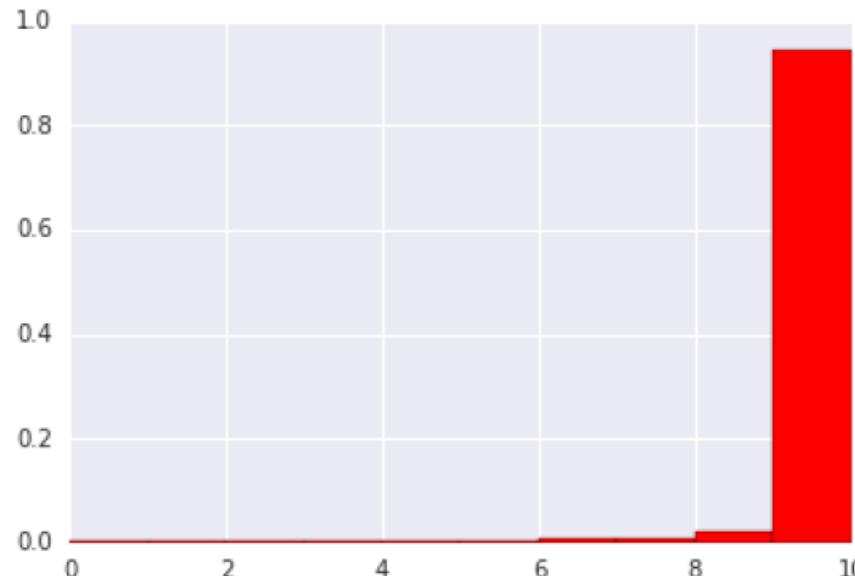
# Top-K Off-Policy Correction

## On-Policy → Off-Policy

- In this simulation, there are 10 items  $\mathcal{A} = \{a_i, i = 1, \dots, 10\}$  and the reward of each one is equal to its index  $r(a_i) = i$ .
- The learned policy  $\pi_\theta$  when behaviour policy  $\beta(a_i) = (11 - i)/55$  is skewed to favour the actions with least reward.



(left) without off-policy correction



(right) with off-policy correction

# Top-K Off-Policy Correction

## Top-1 Correction → Top-K Correction

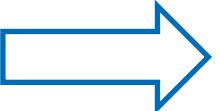
The recommender system need to recommends a page of  $K$  items to users at a time.

- To make the problem tractable, we generate the set action  $A$  by independently sampling each item  $a$  according to the softmax policy  $\pi_\theta$  and then de-duplicate.
- Thus, the gradient update rule can be adapted for set recommendation by replacing  $\pi_\theta$  with

$$\alpha_\theta(a|s) = 1 - (1 - \pi_\theta(a|s))^K$$

# Top-K Off-Policy Correction

Top-1 Correction → Top-K Correction

Top-K Correction 

$$\begin{aligned} & \sum_{\tau \sim \beta} \left[ \sum_{t=0}^{|\tau|} \frac{\pi_\theta(a_t | s_t)}{\beta(a_t | s_t)} R_t \nabla_\theta \log \pi_\theta(a_t | s_t) \right] \\ &= \sum_{\tau \sim \beta} \left[ \sum_{t=0}^{|\tau|} \frac{\alpha_\theta(a_t | s_t)}{\beta(a_t | s_t)} R_t \nabla_\theta \log \alpha_\theta(a_t | s_t) \right] \\ &= \sum_{\tau \sim \beta} \left[ \sum_{t=0}^{|\tau|} \frac{\pi_\theta(a_t | s_t)}{\beta(a_t | s_t)} \frac{\partial \alpha(a_t | s_t)}{\partial \pi(a_t | s_t)} R_t \nabla_\theta \log \pi_\theta(a_t | s_t) \right] \end{aligned}$$

# Top-K Off-Policy Correction

## Top-1 Correction → Top-K Correction

- An additional multiplier needs to be added to the original off-policy correction factor.

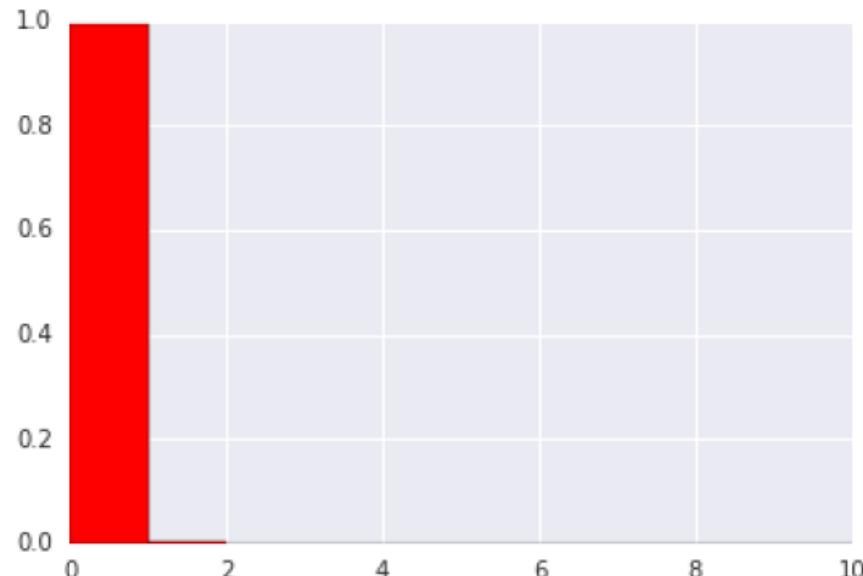
$$\lambda_K(s_t, a_t) = \frac{\partial \alpha(a_t | s_t)}{\partial \pi(a_t | s_t)} = K(1 - \pi_\theta(a_t | s_t))^{K-1}$$

- When the desirable item has a small mass in the policy  $\pi_\theta(\cdot | s)$ , the top- $K$  correction more aggressively pushes up its likelihood than the standard correction.
- Once the policy  $\pi_\theta(\cdot | s)$  casts a reasonable mass on the desirable item (to ensure it is likely to appear in the top- $k$ ), the correction then zeros out the gradient and no longer tries to push up its likelihood. This in return allows other items of interest to take up some mass in the policy.

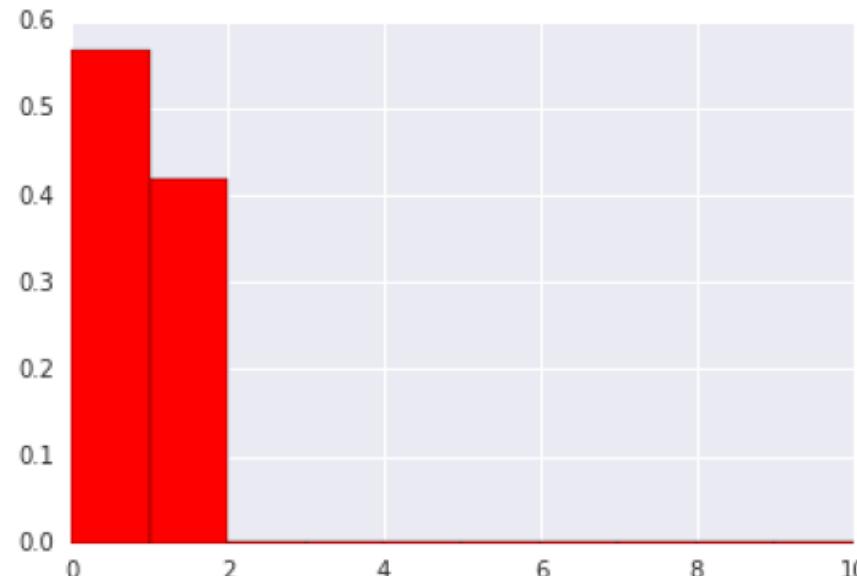
# Top-K Off-Policy Correction

## Top-1 Correction → Top-K Correction

- In this simulation, there are 10 items  $\mathcal{A} = \{a_i, i = 1, \dots, 10\}$  with  $r(a_1) = 10$ ,  $r(a_2) = 9$ , and the remaining items of much lower reward  $r(a_i) = 1, \forall i = 3, \dots, 10$ .
- Here we focus on recommending two items ( $k = 2$ ). The behaviour policy  $\beta$  follows a uniform distribution, i.e., choosing each item with equal chance.



(left) with top-1 correction



(right) with top- $k$  correction