Reinforcement Learning to Rank with Markov Decision Process

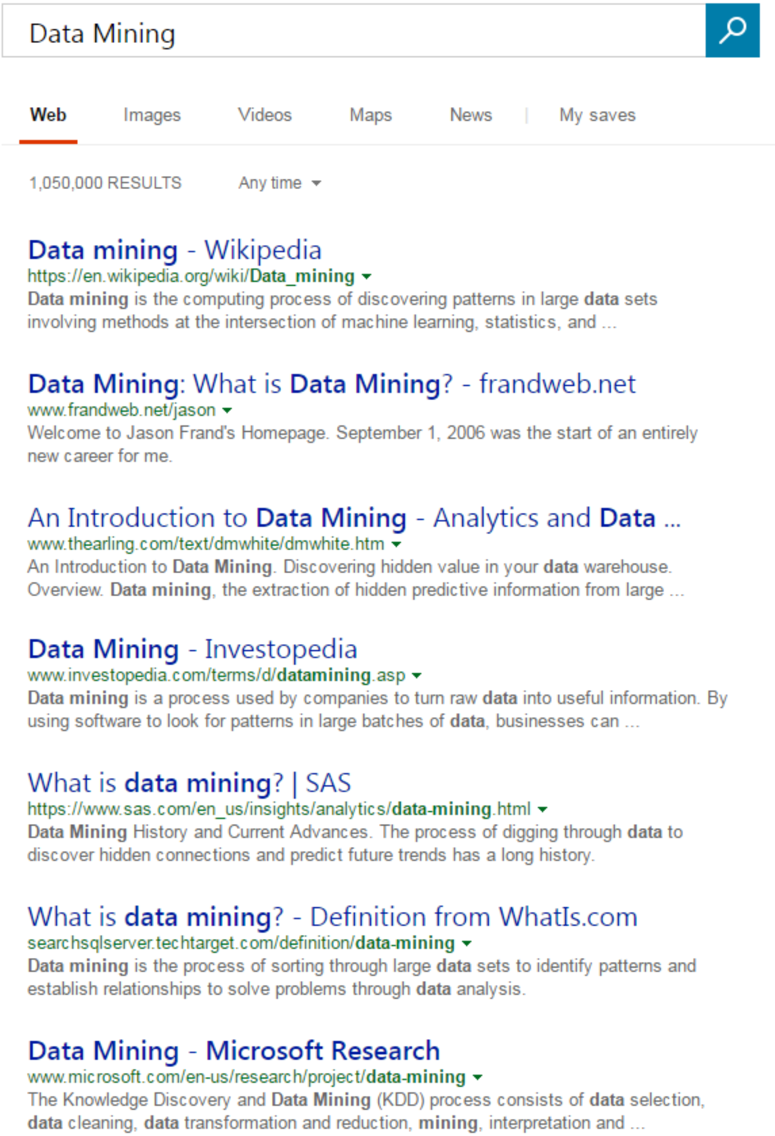
Jun Xu

ICT, CAS

Outline

- Background: learning to rank for IR
- Reinforcement learning to rank
- Summary

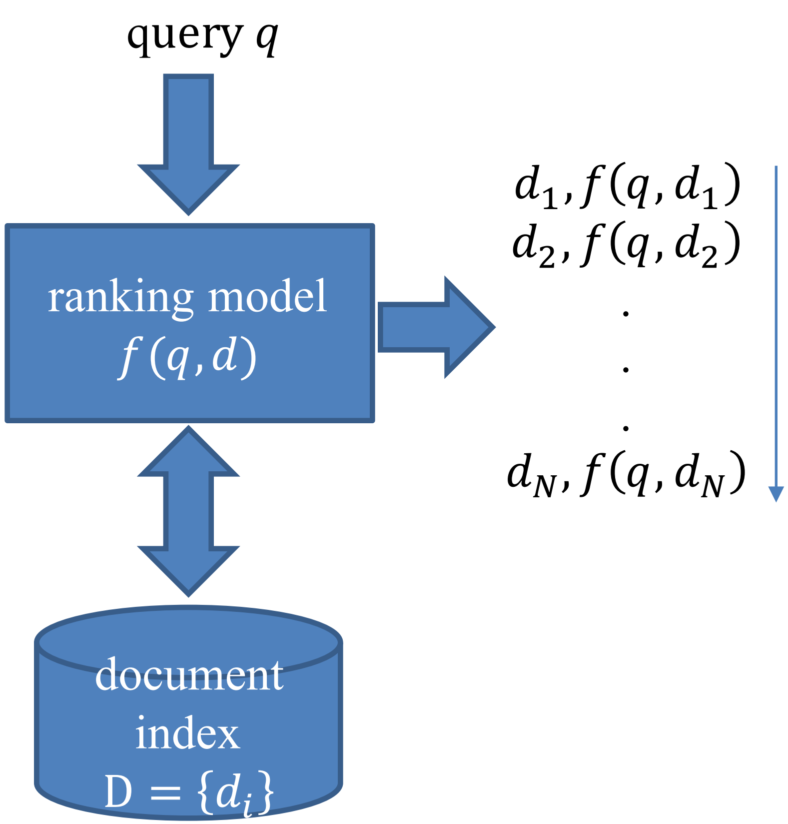
Ranking is Important for Web Search

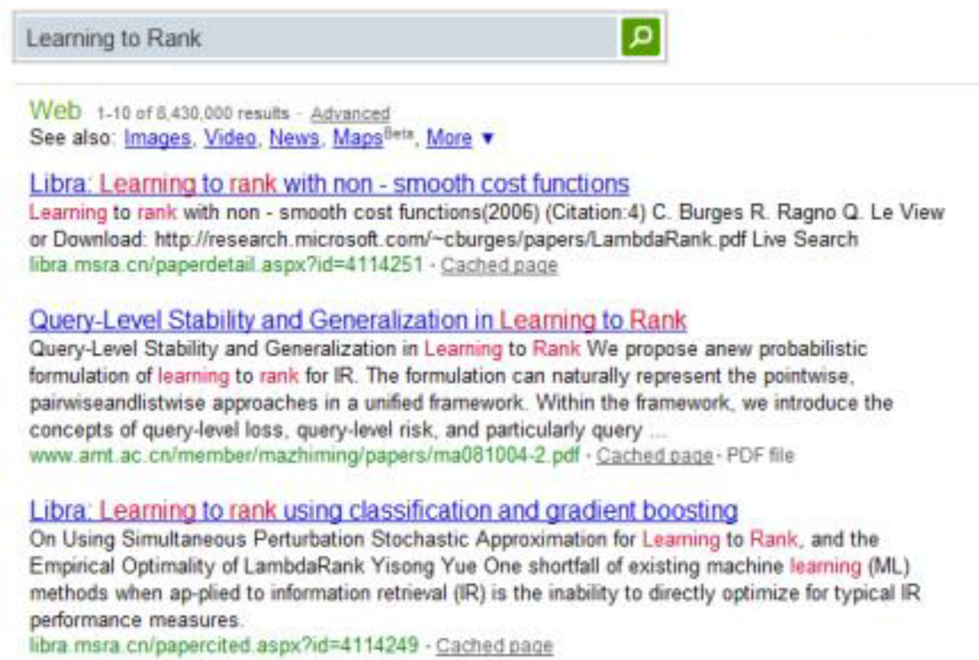


- Criteria
  - Relevance
  - Diversity
  - Freshness

  ……
- Ranking model
  - Heuristic
    - Relevance: BM25, LMIR
    - Diversity: MMR, xQuAD
  - Learning to rank

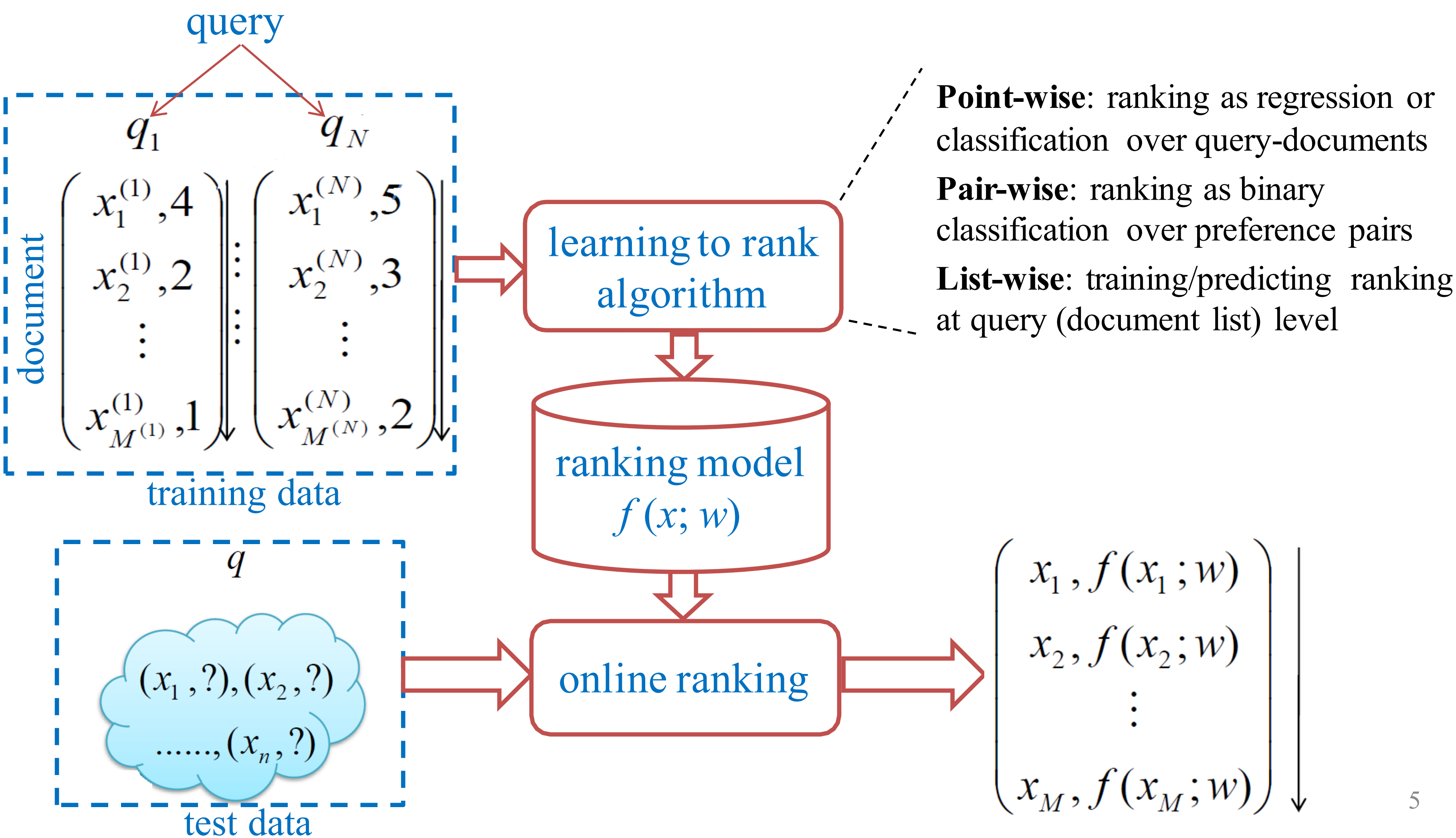
Ranking in Information Retrieval



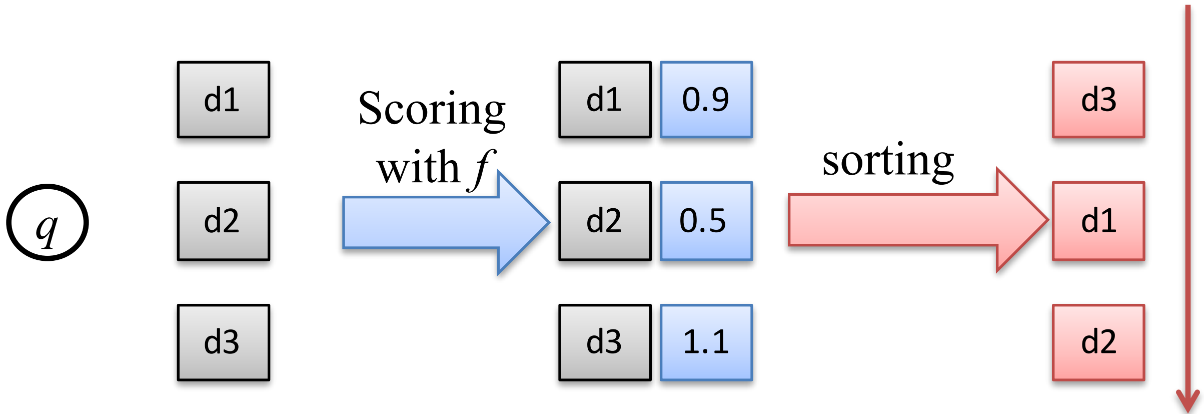


Learning to Rank for Information Retrieval

- Machine learning algorithms for relevance ranking



Independent Relevance Assumption



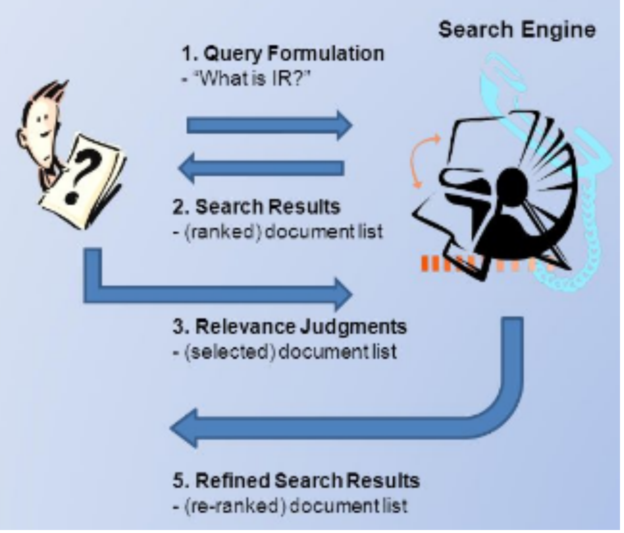
- Utility of a doc is independent of other docs
- Ranking as scoring & sorting
  - Each documents can be scored independently
  - Scores are independent of the rank

Beyond Independent Relevance

- More ranking criteria, e.g., search result diversification
  - Covering as much subtopics as possible with a few documents
  - Need consider the novelty of a document given preceding documents
- Complex application environment, e.g., Interactive IR
  - Human interacts with the system during the ranking process
  - User feedback is helpful for improving the remaining results

Query: Programming language

Good	Bad
Java	Java
C++	Java
Python	Java

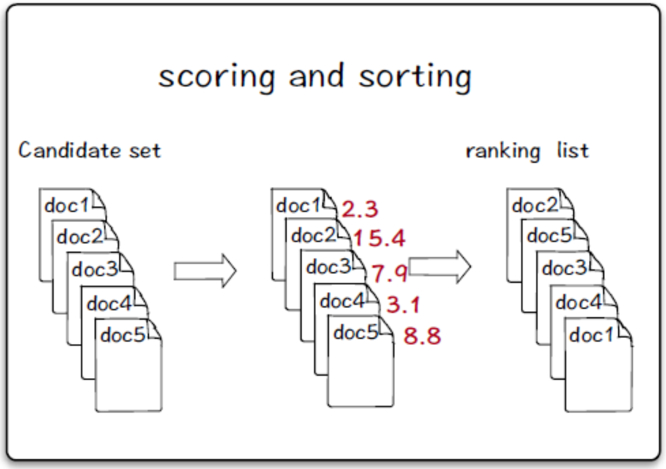


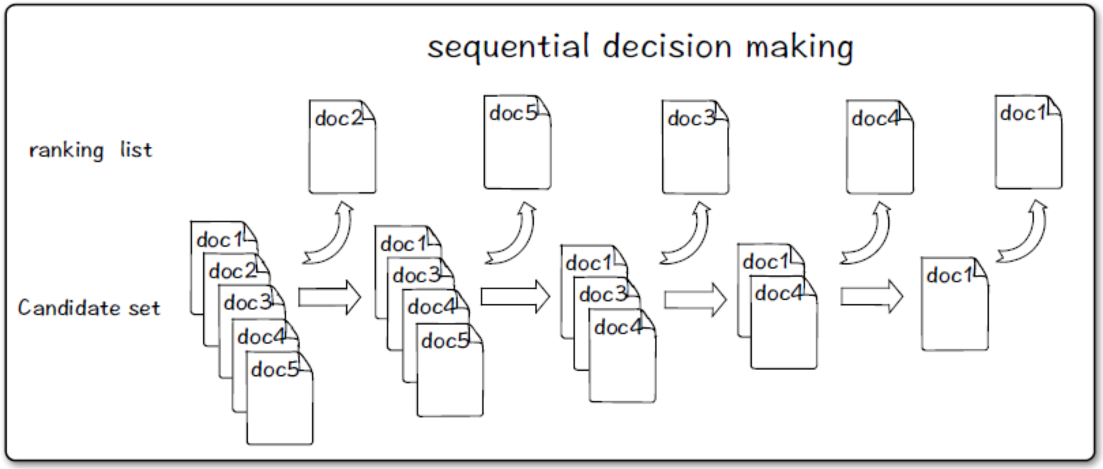
Need more powerful ranking mechanism!

Outline

- Background: learning to rank for IR
- Reinforcement learning to rank
  - Ranking as Markov decision process
  - Adapting MDP for relevance and diverse ranking
- Summary

From Scoring & Sorting to Sequential Decision Making

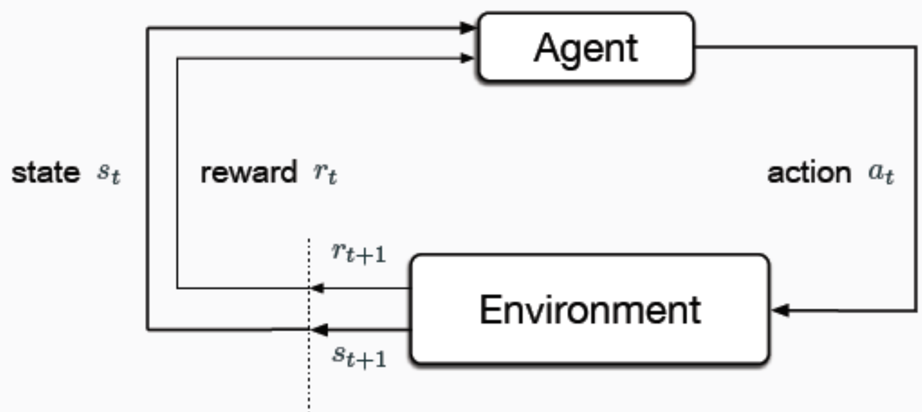




- Advantages: beyond independent relevance
  - Modeling the dependencies between documents
  - Taking the ranking positions into consideration

Markov Decision Process (MDP)

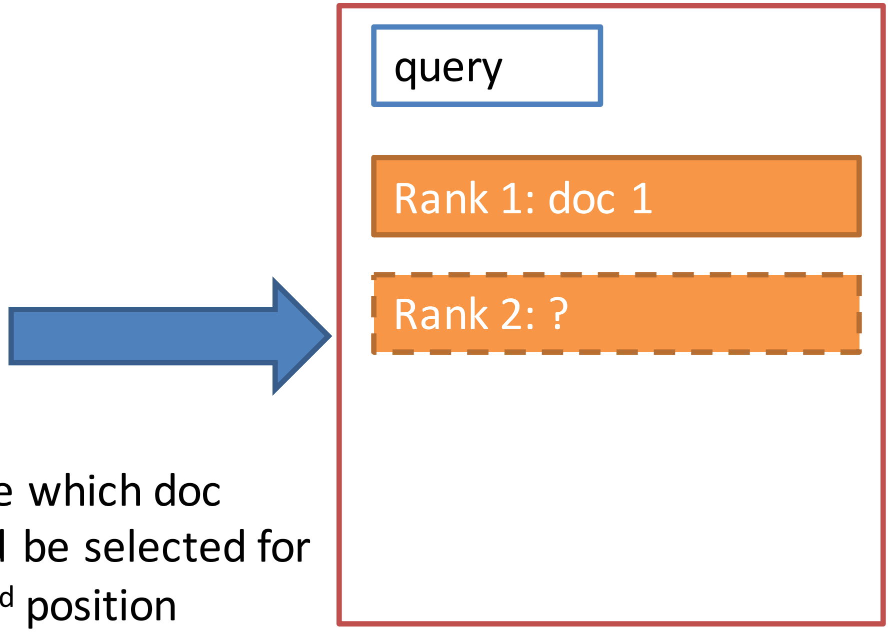
- An MDP is composed by states, actions, rewards, policy, and transitions, and represented by a tuple $\langle S, A, T, R, \pi \rangle$



- **States** S: a set of states.
- **Actions** A: a discrete set of actions that an agent can take.
- **Transition** T: the state transition function $s_{t+1} = T(s_t, a_t)$
- **Reward** $r = R(s, a)$: the immediate reward, also known as reinforcement
- **Policy** $\pi(a|s)$: a probability distribution over the possible actions.

Ranking as Markov Decision Process

Candidate document set



Decide which doc should be selected for the 2nd position

- Time steps: ranks
- State: query, preceding docs, candidates, ……
- Policy: distribution over remaining candidate documents
- Action (Decision): selecting a doc and placing it to current pos
- Reward
  - Additional utility (e.g., the increase of DCG) from the selected doc
  - Calculated based on widely used evaluation measures (e.g., DCG, ERR-IA)

Learning and Online Ranking

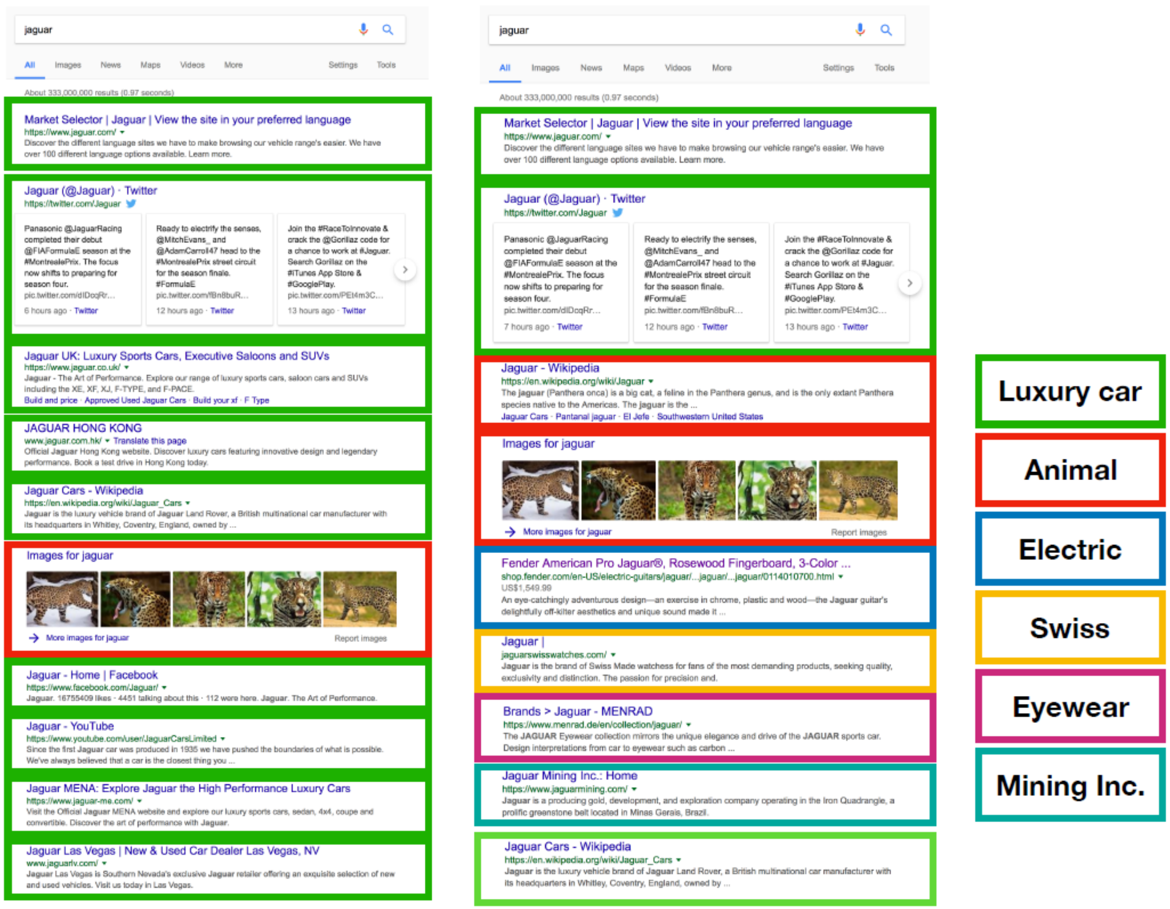
- Learning the parameters
  - Model parameters: policy function, state initialization and transition etc.
  - Reinforcement learning: policy gradient
  - Rewards based on relevance labels as supervision
- Online ranking
  - Without rewards (rewards are based on relevance labels)
  - Fully trust the learned policy

Example 1: Learning for Search Result Diversification

Long Xia, Jun Xu, Yanyan Lan, et al., Adapting Markov Decision Process for Search Result Diversification. Proceedings of SIGIR 2017, pp. 535-544.

Search Result Diversification

Query: jaguar



- Query: information needs are ambiguous and multi-faceted
- Search results: may contain redundant information
- Goal: covering as much subtopics as possible with a few documents

Modeling Diverse Ranking with MDP

- Key points
  - Mimic user top-down browsing behaviors
  - Model dynamic information needs with MDP state
- States $s_t = [Z_t, X_t, \mathbf{h}_t]$
  - Z_t: sequence of t preceding documents, $Z_0 = \phi$
  - X_t: set of candidate documents, $X_0 = X$
  - $\mathbf{h}_t \in R^K$: latent vector, encodes user perceived utility from preceding documents, initialized with the information needs form the query:

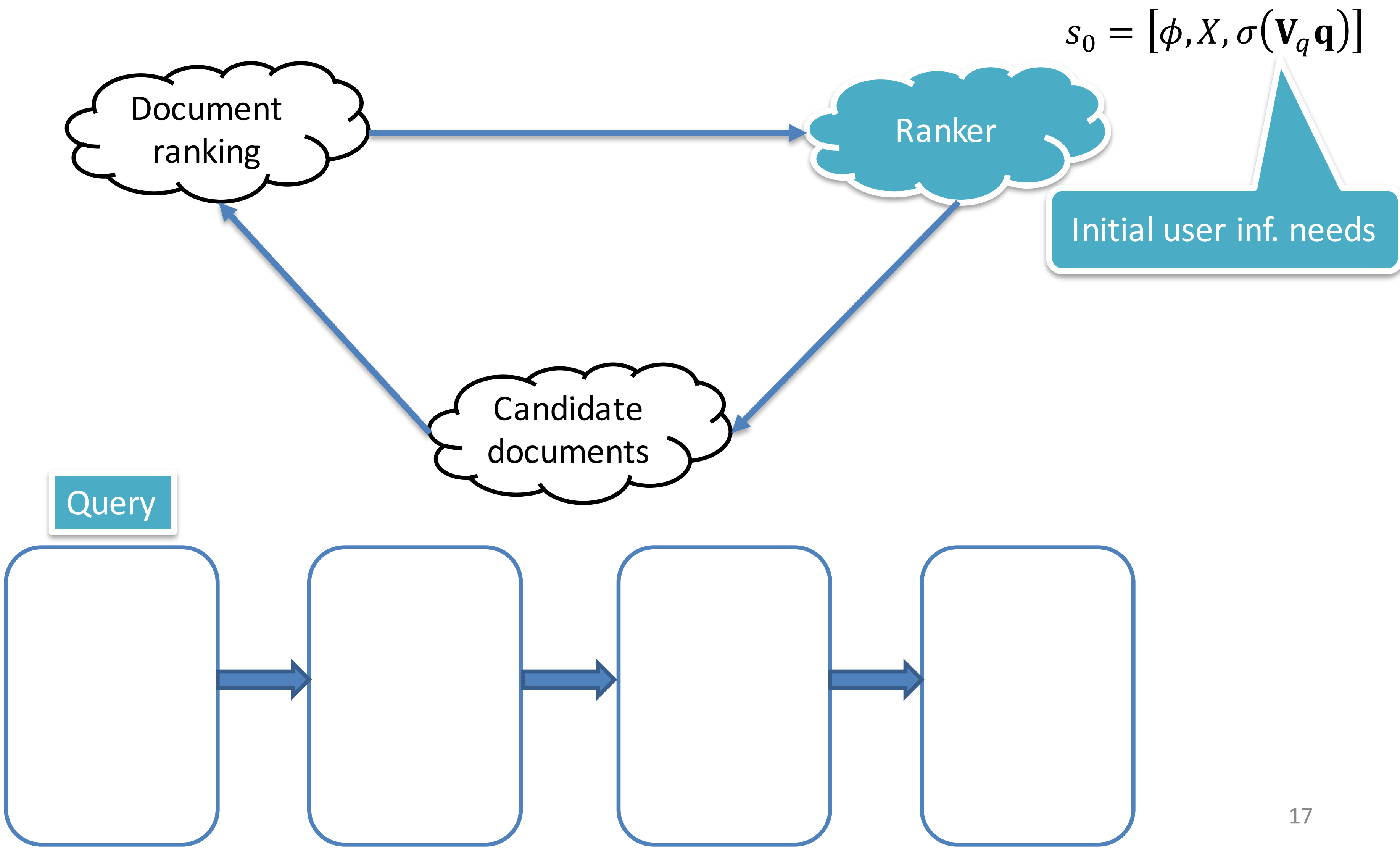
$$\mathbf{h}_0 = \sigma(\mathbf{V}_q \mathbf{q})$$

Modeling Diverse Ranking with MDP

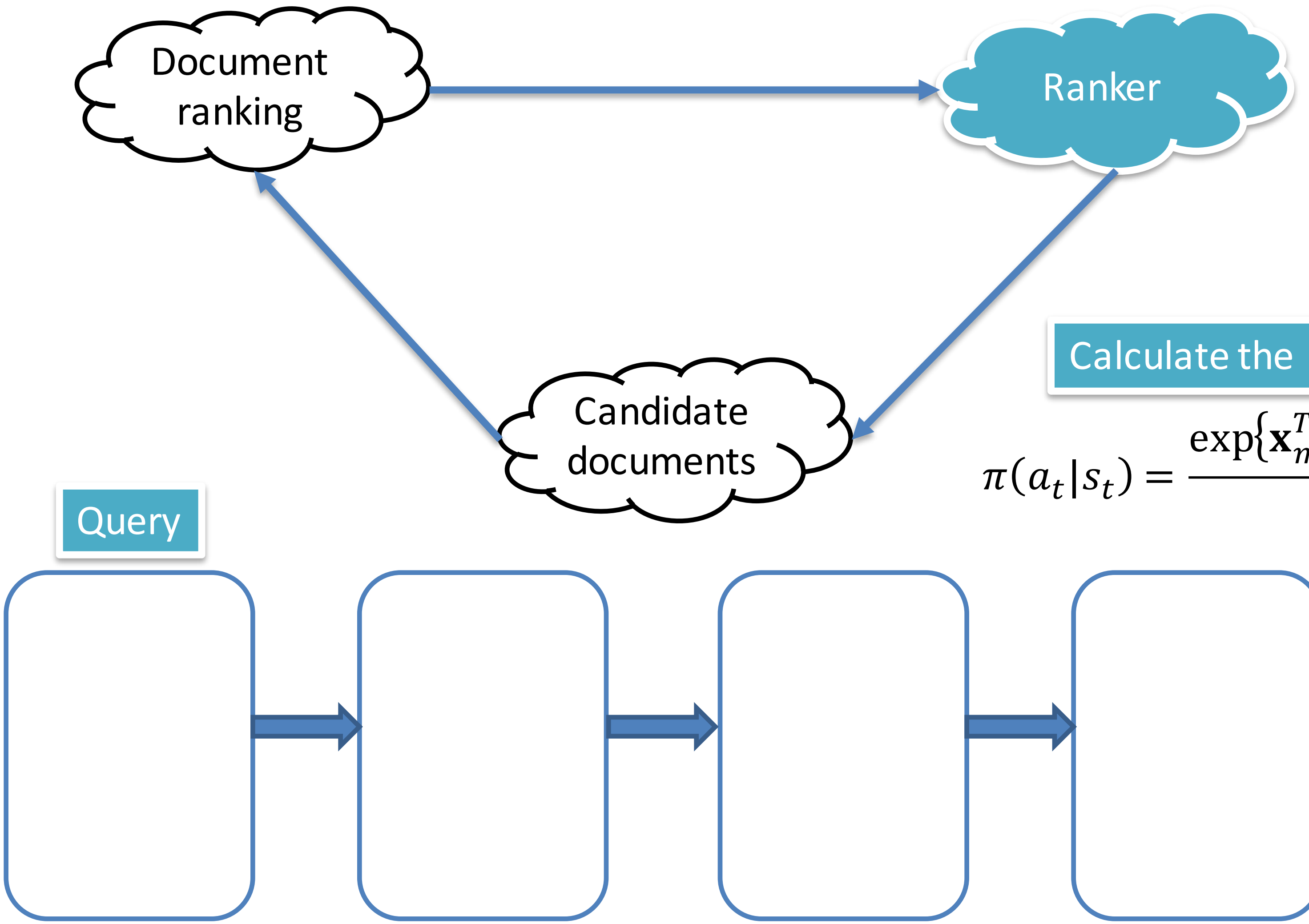
$\mathbf{x}_{m(a_t)}$: document embedding

MDP factors	Corresponding diverse ranking factors
Time steps	The ranking positions
State	$s_t = [Z_t, X_t, \mathbf{h}_t]$
Policy	$\pi(a_t \mid s_t = [Z_t, X_t, \mathbf{h}_t]) = \frac{\exp\{\mathbf{x}_{m(a_t)}^T \mathbf{U}\mathbf{h}_t\}}{Z}$
Action	Selecting a doc and placing it to rank $t+1$
Reward	Based on evaluation measure αDCG, SRecall etc. For example: $R = \alpha\mathrm{DCG}[t+1] - \alpha\mathrm{DCG}[t]$; $R = \mathrm{SRecall}[t+1] - \mathrm{SRecall}[t]$
State Transition	$s_{t+1} = T(s_t = [Z_t, X_t, \mathbf{h}_t], a_t) = \left[Z_t \oplus \{\mathbf{x}_{m(a_t)}\}, X_t \setminus \{\mathbf{x}_{m(a_t)}\}, \sigma(\mathbf{V}\mathbf{x}_{m(a_t)} + \mathbf{W}\mathbf{h}_t)\right]$

Ranking Process: Initialize State

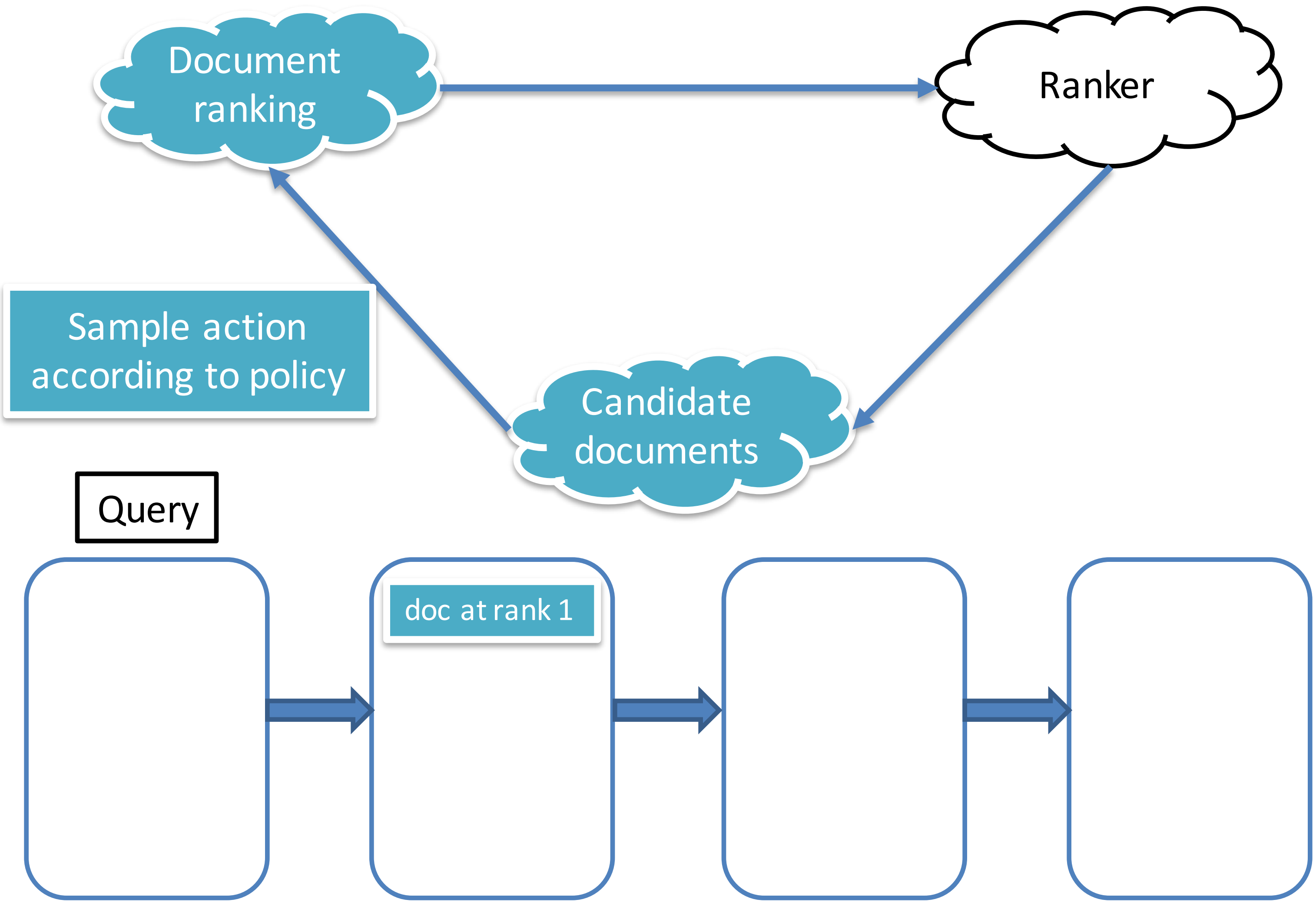


Ranking Process: Policy

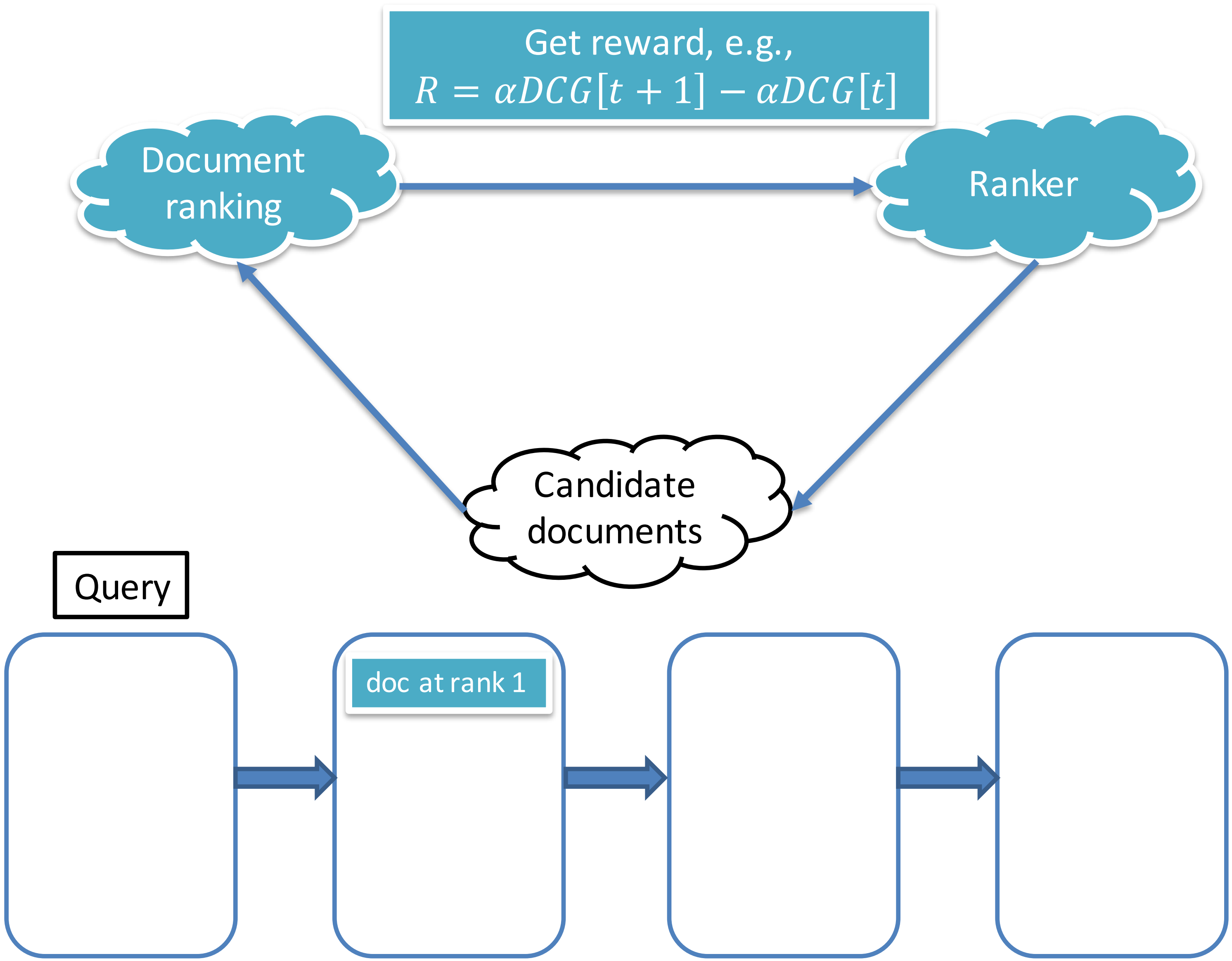


$$\pi(a_t|s_t) = \frac{\exp\{\mathbf{x}_{m(a_t)}^T \mathbf{U}\mathbf{h}_t\}}{Z}$$

Ranking Process: Action



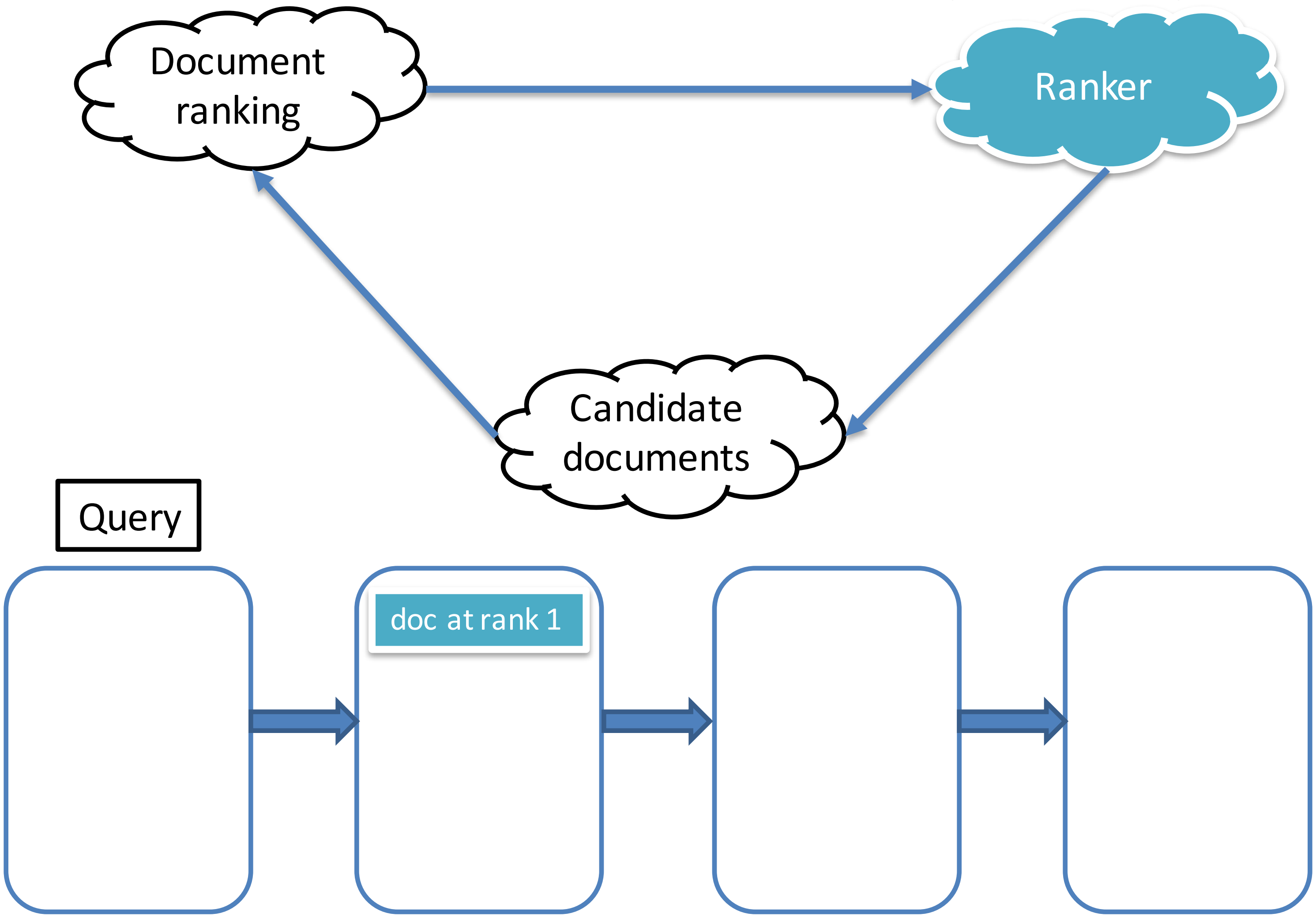
Ranking Process: Reward



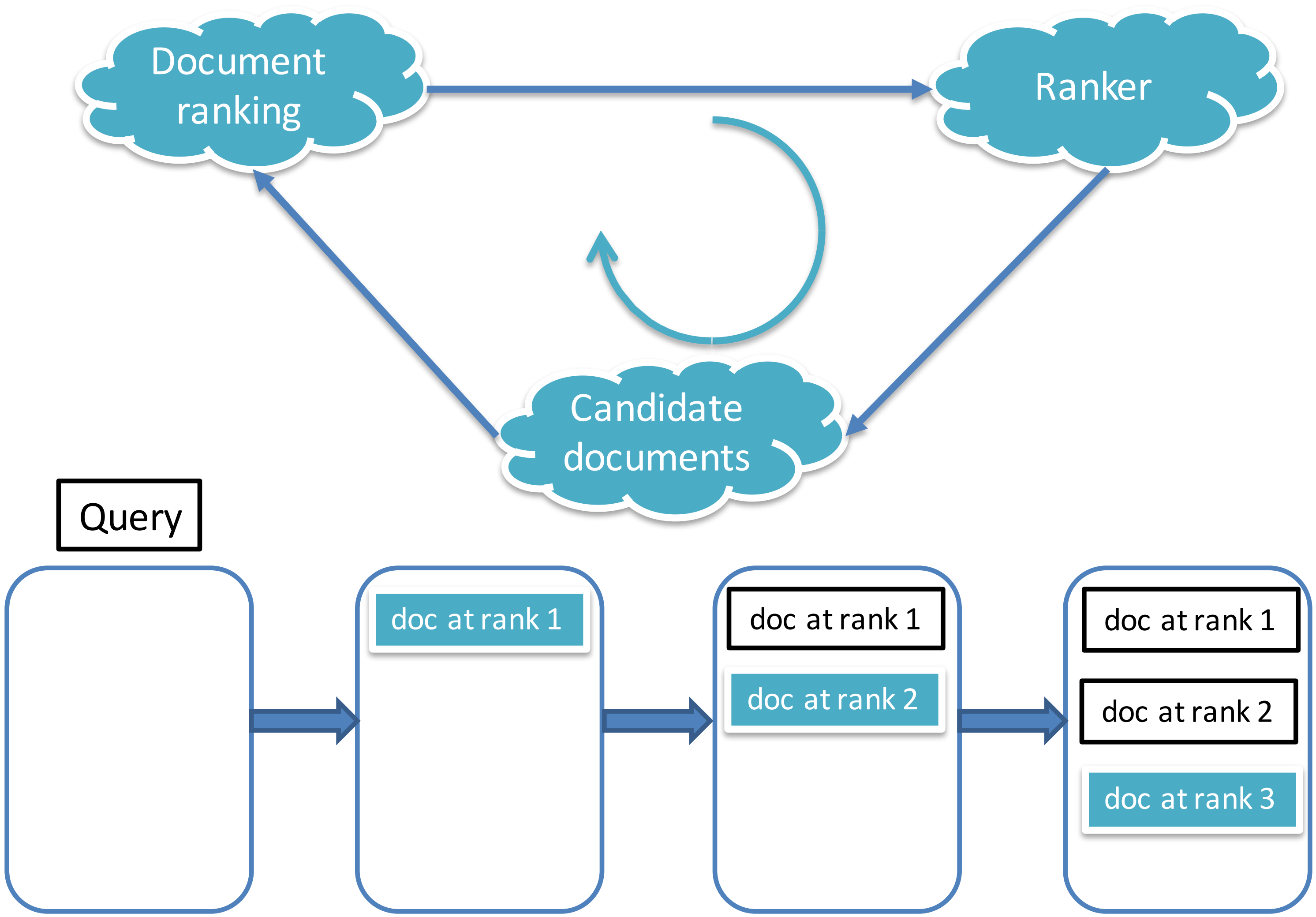
Ranking Process: State Transition

Update ranked list, candidate set, and latent vector

$$s_{t+1} = \left[Z_t \oplus \{\mathbf{x}_{m(a_t)}\}, X_t \backslash \{\mathbf{x}_{m(a_t)}\}, \sigma\left(\mathbf{V}\mathbf{x}_{m(a_t)} + \mathbf{W}\mathbf{h}_t\right)\right]$$



Ranking Process: Iterate



Learning with Policy Gradient

- Model parameters $\boldsymbol{\Theta} = \{\mathbf{V}_q, \mathbf{U}, \mathbf{V}, \mathbf{W}\}$
- Learning objective: maximizing expected return (discounted sum of rewards) of each training query

$$\max_{\boldsymbol{\Theta}} v(\mathbf{q}) = E_\pi G_0 = E_\pi \left[\sum_{k=0}^{M-1} \gamma^k r_{k+1} \right]$$

  - Directly optimizes evaluation measure as $G_0 = \alpha\mathrm{DCG}@M$
- Monte-Carlo stochastic gradient ascent is used to conduct the optimization (REINFORCE algorithm)

$$\widehat{\nabla_{\boldsymbol{\Theta}} v(\mathbf{q})} = \gamma^t G_t \nabla_{\boldsymbol{\Theta}} \log \pi(a_t | s_t; \boldsymbol{\Theta})$$

Analysis

- Optimize general diversity evaluation measures (e.g., α-DCG, S-recall)

discounted sum of the rewards, starting from position 0 (return)

$$\max_{\Theta} V(\mathbf{q}) = \mathbb{E}_{\pi} G_0$$

- Given an episode and time step t

$$\begin{aligned}
\widehat{\nabla_{\Theta} V(\Theta)} &\overset{\text{sample}}{=} \gamma^t \sum_{a \in A(s_t)} \nabla_{\Theta} \pi(a|s_t) Q^{\pi}(s_t, a) \\
&= \gamma^t \sum_{a \in A(s_t)} \pi(a|s_t) \cdot \left(Q^{\pi}(s_t, a) \frac{\nabla_{\Theta} \pi(a|s_t)}{\pi(a|s_t)} \right) \\
&\overset{\text{sample}}{=} \gamma^t Q^{\pi}(s_t, a_t) \frac{\nabla_{\Theta} \pi(a_t|s_t)}{\pi(a_t|s_t)} \\
&\overset{\text{sample}}{=} \gamma^t G_t \nabla_{\Theta} \log \pi(a_t|s_t).
\end{aligned}$$

Maximizing the return starting from position t

The Learning Algorithm

Algorithm 1 MDP-DIV learning

Input: Labeled training set $D = \{(\mathbf{q}^{(n)}, X^{(n)}, J^{(n)})\}^{N}$ … learning rate η, discount factor γ, and reward fun…

Output: $\Theta = \{\mathbf{V}_q, \mathbf{U}, \mathbf{V}, \mathbf{W}\}$

```
Initialize Θ = {V_q, U, V, W} ← random …
repeat
   for all (q, X, J) ∈ D do
      (s_0, a_0, r_1, ···, s_{M−1}, a_{M−1}, r_M) ← S…
      {Algorithm (2), and M = |X|}
      for t = 0 to M − 1 do
         G_t ← Σ_{k=0}^{M−1−t} γ^k r_{t+k+1} {Equati…
         Θ ← Θ + ηγ^t G_t ∇_Θ log π(a_t|s_t; Θ…
      end for
   end for
until converge
return Θ
```

Algorithm 2 SampleEpisode

Input: Parameters $\Theta = \{\mathbf{V}_q, \mathbf{U}, \mathbf{V}, \mathbf{W}\}$, $\mathbf{q}$, X, J, and R

Output: An episode

```
Initialize s ← [∅, X, σ(V_q q)] {Equation (1)}
M ← |X|
E = () {empty episode}
for t = 0 to M − 1 do
   A ← A(s) {Possible actions according to X in state s}
   for all a ∈ A do
      P(a) ← π(a|s; Θ)
   end for
   Sample an action â ∈ A, according to P
   r ← R(s, â) {Calculation on the basis of J}
   Append (s, â, r) to the tail of E
   [Z, X, h] ← s
   s ← [Z ⊕ {x_{m(â)}}, X \ {x_{m(â)}}, σ(V x_{m(â)} + W h)]
end for
return E = (s_0, a_0, r_1, ···, s_{M−1}, a_{M−1}, r_M)
```

Online Ranking Algorithm

- Fully trust the policy

Algorithm 3 MDP-DIV online ranking

Input: Parameters $\Theta = \{\mathbf{V}_q, \mathbf{U}, \mathbf{V}, \mathbf{W}\}$, query $\mathbf{q}$, documents X

Output: Permutation of documents τ

Initialize $s \leftarrow [\emptyset, X, \sigma(\mathbf{V}_q\mathbf{q})]$ {Equation (1)}

$M \leftarrow |X|$

for $t = 0$ **to** $M - 1$ **do**

  $A \leftarrow A(s)$ {Possible actions according to X in state s}

  $\hat{a} \leftarrow \arg\max_{a \in A} \pi(a|s; \Theta)$ {Choosing most possible action}

  $\tau[t+1] \leftarrow m(\hat{a})$ {Document $\mathbf{x}_{m(\hat{a})}$ is ranked at $t+1$}

  $[\mathcal{Z}, X, \mathbf{h}] \leftarrow s$

  $s \leftarrow [\mathcal{Z} \oplus \{\mathbf{x}_{m(\hat{a})}\}, X \setminus \{\mathbf{x}_{m(\hat{a})}\}, \sigma(\mathbf{V}\mathbf{x}_{m(\hat{a})} + \mathbf{W}\mathbf{h})]$

end for

return τ

using max instead of sampling

Experimental Results

Method	α-NDCG@5	α-NDCG@10	S-recall@5	S-recall@10
MMR	0.2753	0.2979	0.4388	0.5151
xQuAD	0.3165	0.3941	0.4933	0.6043
PM-2	0.3047	0.3730	0.4910	0.6012
SVM-DIV	0.3030	0.3699	0.5122	0.6230
R-LTR	0.3498	0.4132	0.5397	0.6511
PAMM(α-NDCG)	0.3712	0.4327	0.5561	0.6612
NTN-DIV(α-NDCG)	0.3962	0.4577	0.5817	0.6872
MDP-DIV(S-recall)	0.4156	0.4734	**0.6123**	**0.7155**
MDP-DIV(α-DCG)	**0.4189**	**0.4762**	0.6102	0.7117

- Based on combination of TREC 2009 ~ 2012 Web Track
- Directly optimize a predefined measure via defining the rewards based on the measure

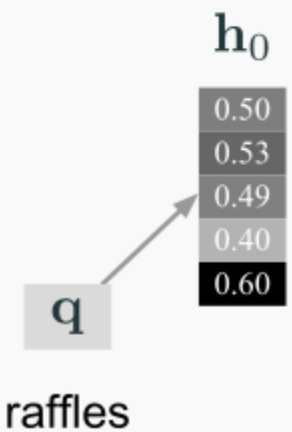
How it works?
Using Query 93 as Example

q

raffles

[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**]

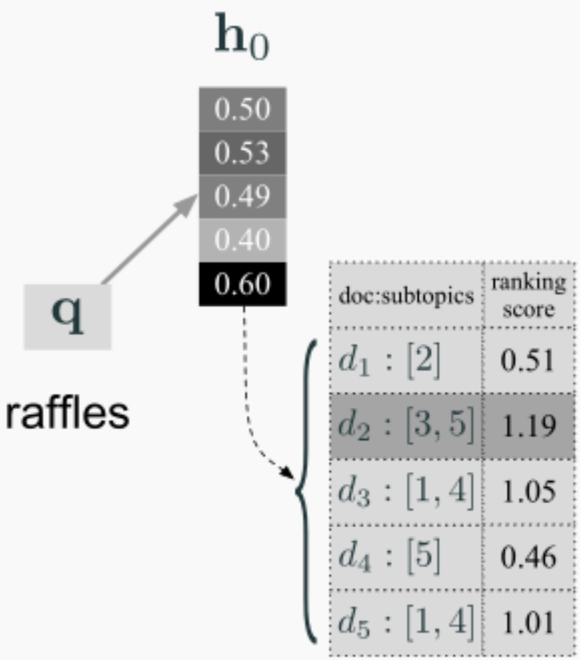
How it works? Using Query 93 as Example



[1] : "Raffles Hotel in Singapore"
[2] : "Sir Stamford Raffles"
[3] : "organizing a raffle"
[4] : "the Raffles hotel in Dubai"
[5] : "car raffles"

d_1 : "Stamford Raffles – Wikipedia, the free encyclopedia" [**2**]
d_2 : "Fundraiser Raffle Ideas" [**3**, **5**]
d_3 : "Luxury Hotel Guide | Raffles Hotels" [**1**, **4**]
d_4 : "National Corvette Museum – Corvette Raffles" [**5**]
d_5 : "Raffles Hotels and Resorts" [**1**, **4**]

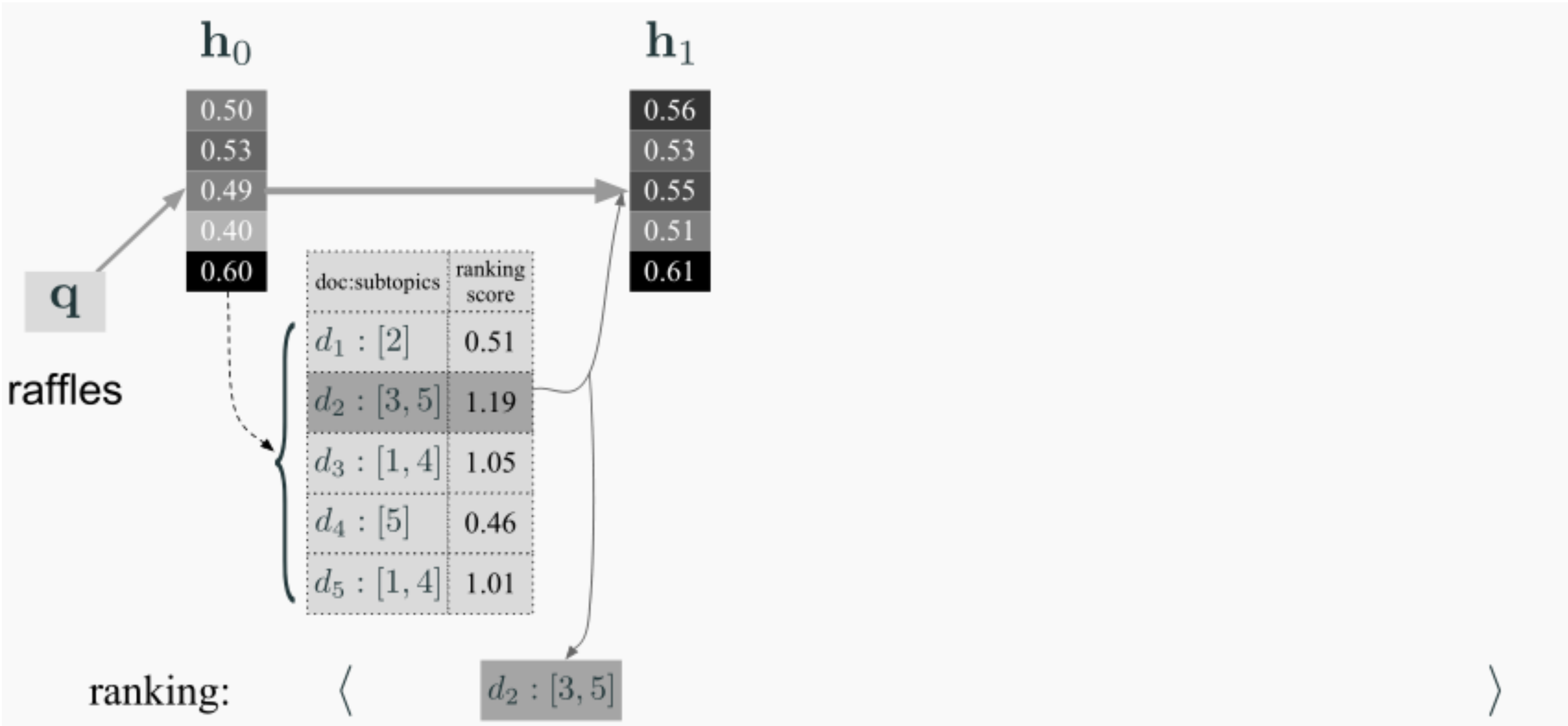
How it works? Using Query 93 as Example



[1] : "Raffles Hotel in Singapore"
[2] : "Sir Stamford Raffles"
[3] : "organizing a raffle"
[4] : "the Raffles hotel in Dubai"
[5] : "car raffles"

d_1 : "Stamford Raffles – Wikipedia, the free encyclopedia" [**2**]
d_2 : "Fundraiser Raffle Ideas" [**3**, **5**]
d_3 : "Luxury Hotel Guide | Raffles Hotels" [**1**, **4**]
d_4 : "National Corvette Museum – Corvette Raffles" [**5**]
d_5 : "Raffles Hotels and Resorts" [**1**, **4**]

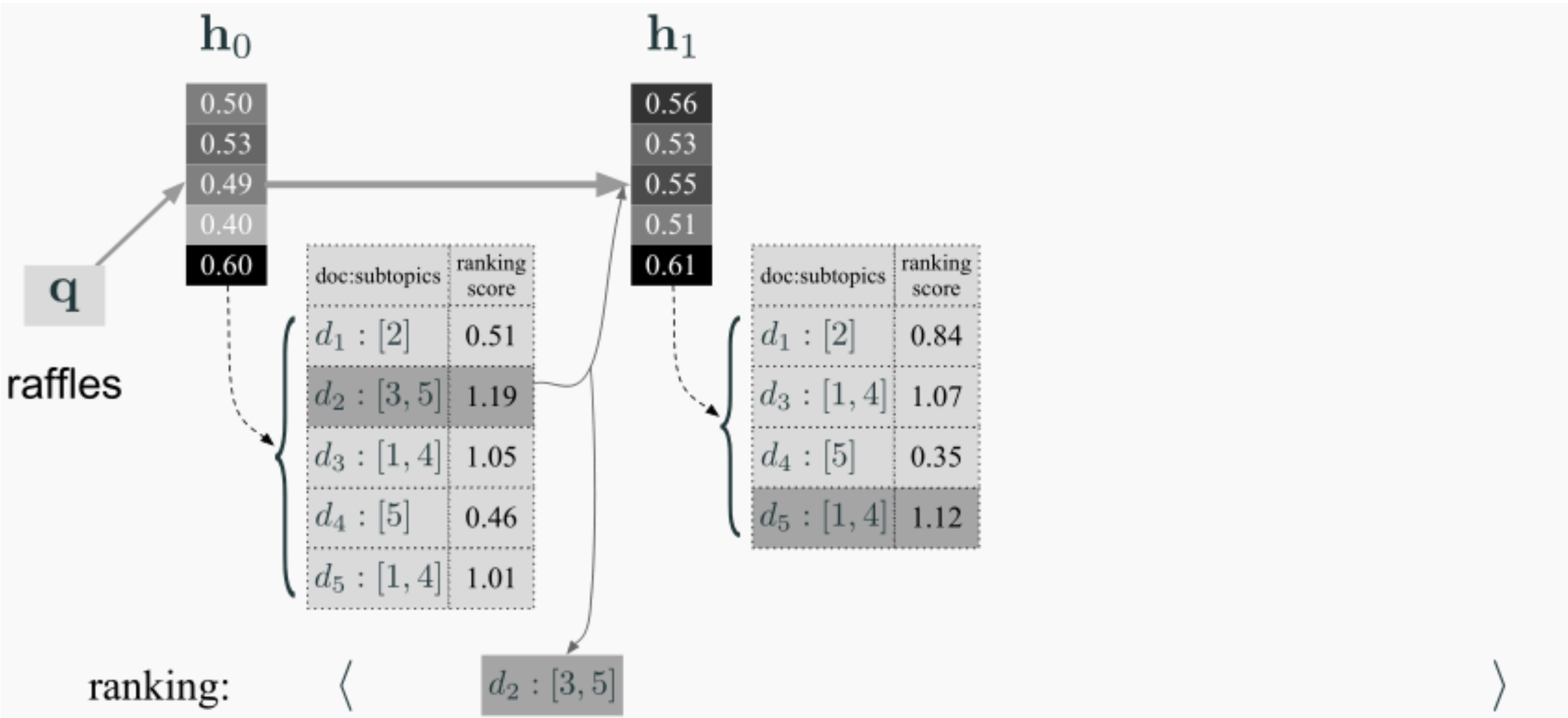
How it works? Using Query 93 as Example



[1] : "Raffles Hotel in Singapore"
[2] : "Sir Stamford Raffles"
[3] : "organizing a raffle"
[4] : "the Raffles hotel in Dubai"
[5] : "car raffles"

d_1 : "Stamford Raffles – Wikipedia, the free encyclopedia" [**2**]
d_2 : "Fundraiser Raffle Ideas" [**3, 5**]
d_3 : "Luxury Hotel Guide | Raffles Hotels" [**1, 4**]
d_4 : "National Corvette Museum – Corvette Raffles" [**5**]
d_5 : "Raffles Hotels and Resorts" [**1, 4**]

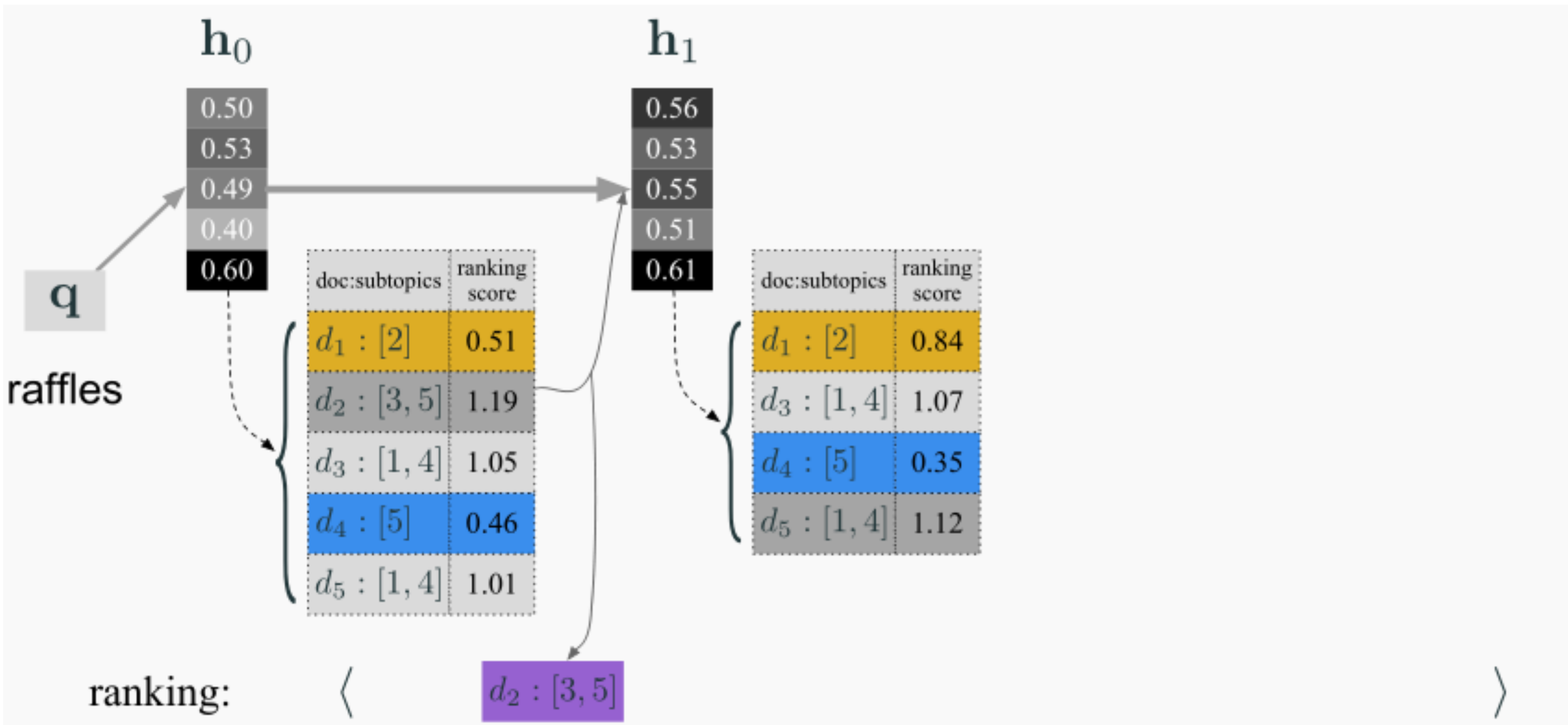
How it works?
Using Query 93 as Example



[1] : "Raffles Hotel in Singapore"
[2] : "Sir Stamford Raffles"
[3] : "organizing a raffle"
[4] : "the Raffles hotel in Dubai"
[5] : "car raffles"

d_1 : "Stamford Raffles – Wikipedia, the free encyclopedia" [**2**]
d_2 : "Fundraiser Raffle Ideas" [**3, 5**]
d_3 : "Luxury Hotel Guide | Raffles Hotels" [**1, 4**]
d_4 : "National Corvette Museum – Corvette Raffles" [**5**]
d_5 : "Raffles Hotels and Resorts" [**1, 4**]

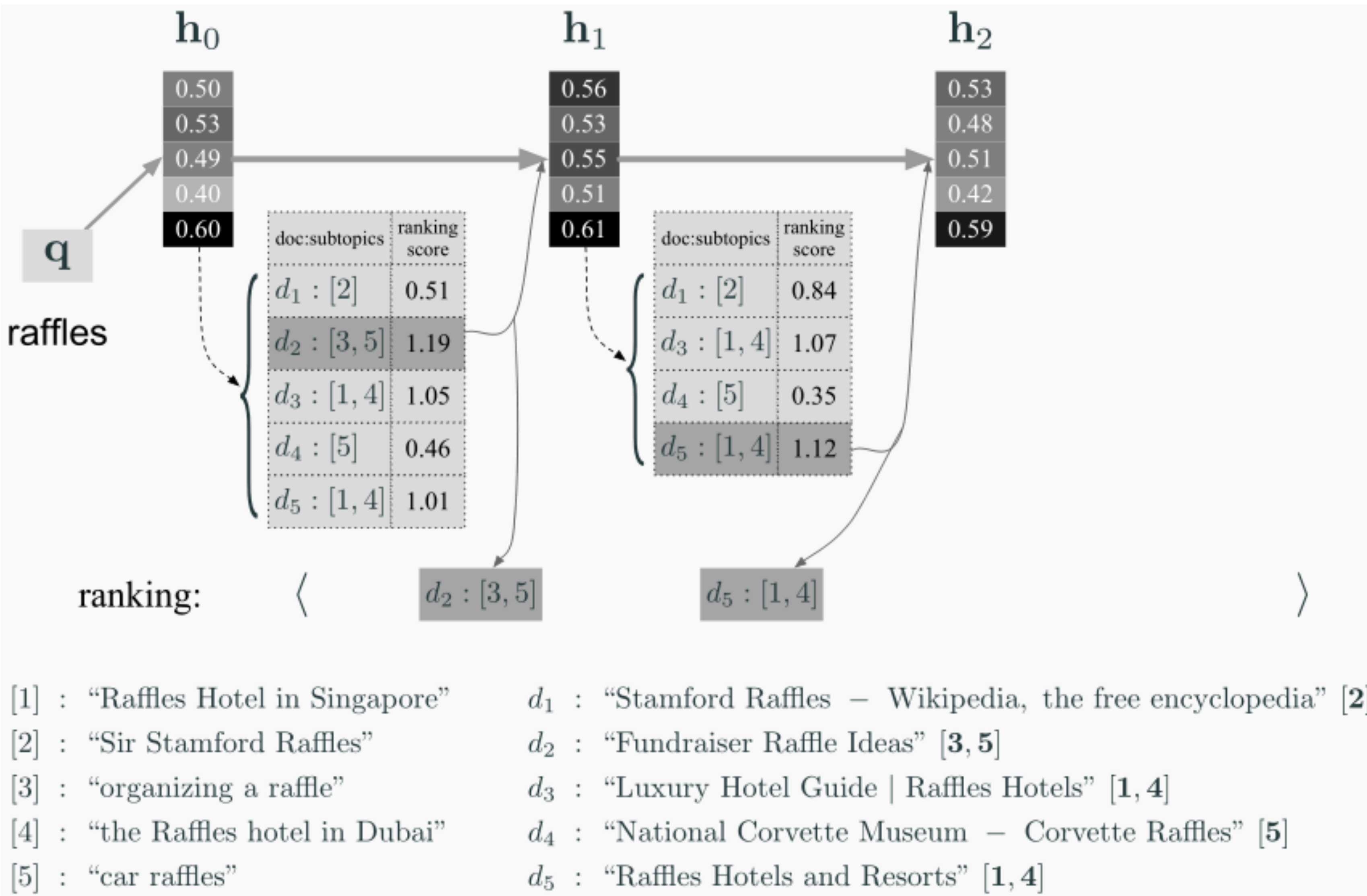
How it works?
Using Query 93 as Example



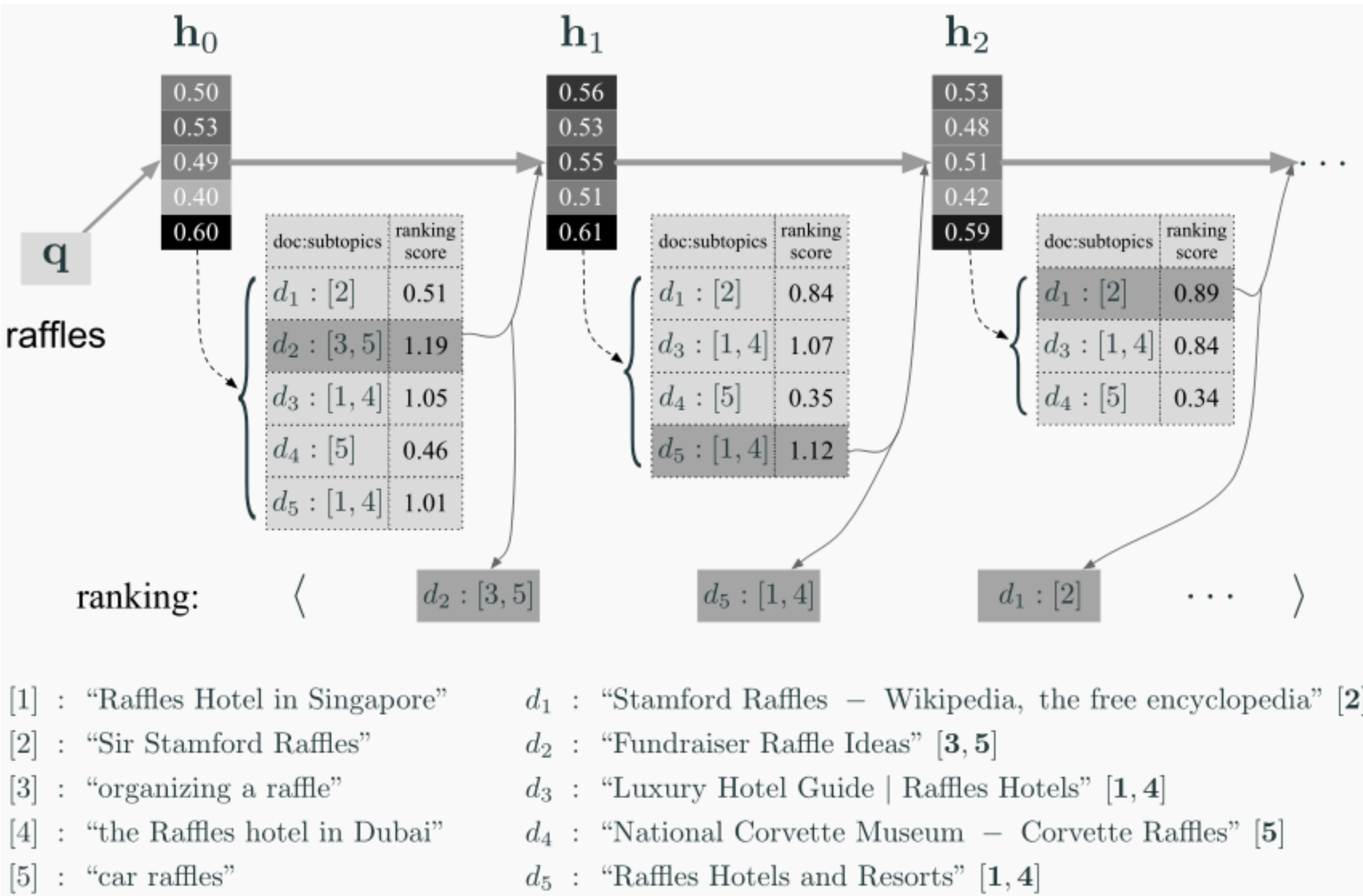
[1] : "Raffles Hotel in Singapore"
[2] : "Sir Stamford Raffles"
[3] : "organizing a raffle"
[4] : "the Raffles hotel in Dubai"
[5] : "car raffles"

d_1 : "Stamford Raffles – Wikipedia, the free encyclopedia" [**2**]
d_2 : "Fundraiser Raffle Ideas" [**3**, **5**]
d_3 : "Luxury Hotel Guide | Raffles Hotels" [**1**, **4**]
d_4 : "National Corvette Museum – Corvette Raffles" [**5**]
d_5 : "Raffles Hotels and Resorts" [**1**, **4**]

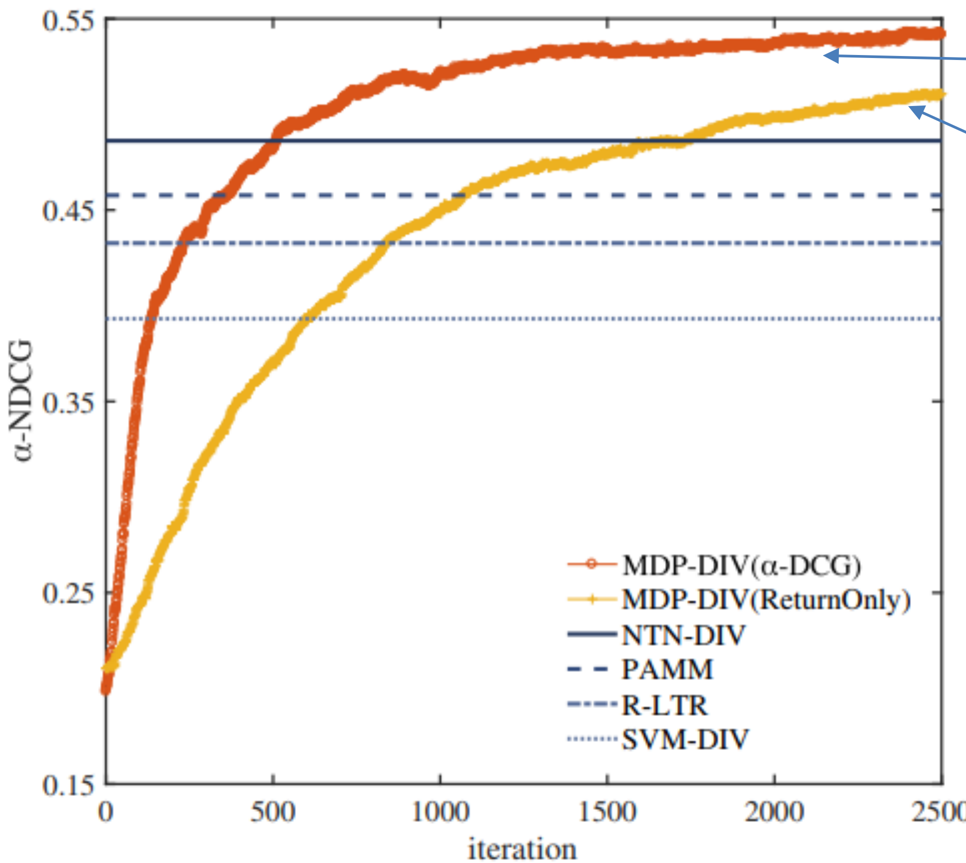
How it works?
Using Query 93 as Example



How it works?
Using Query 93 as Example



Using Immediate Rewards in Training

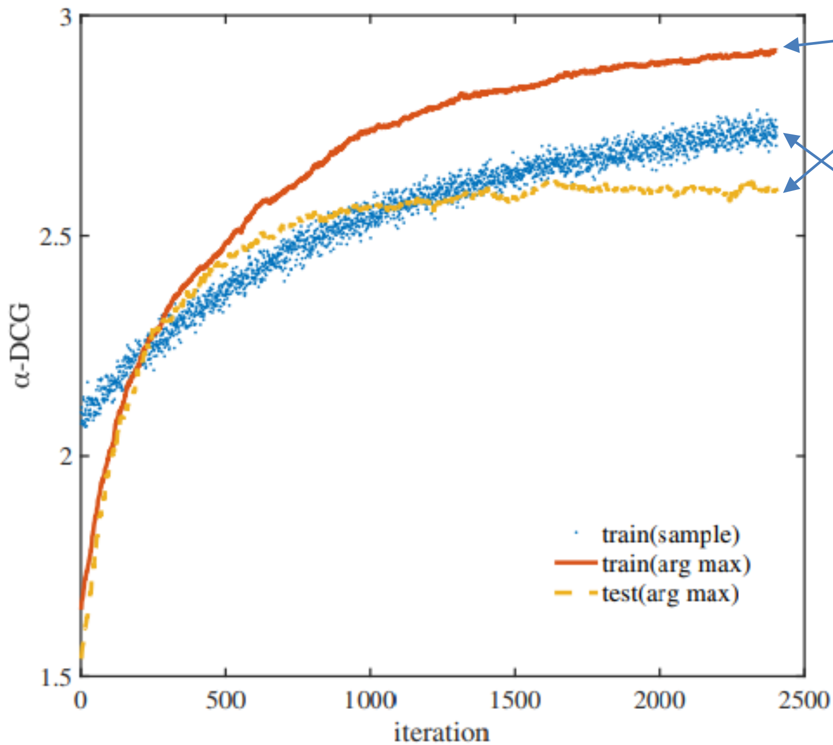


Train with α-DCG@k (k=1, …, M)

Train with α-DCG@M

Figure 4: The performance curves on the test data for MDP-DIV(α-DCG), and the modified MDP-DIV(α-DCG) in which the training only involves the long-term returns. The performances of other baselines are shown as horizontal lines.

Convergence and Online Ranking Criterion



$\hat{a} \leftarrow \arg\max_{a \in A} \pi(a|s;\Theta)$

for all $a \in A$ **do**

$\quad P(a) \leftarrow \pi(a|s;\Theta)$

end for

Sample an action $\hat{a} \in A$, according to P

Figure 5: The performance curves in terms of α-DCG on the training data ("train(arg max)") and the test data ("test(arg max)"). The average performances of the sampled rankings over all training queries are also shown ("train(sample)").

Advantages

- 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

Example 2: Relevance Ranking as an MDP

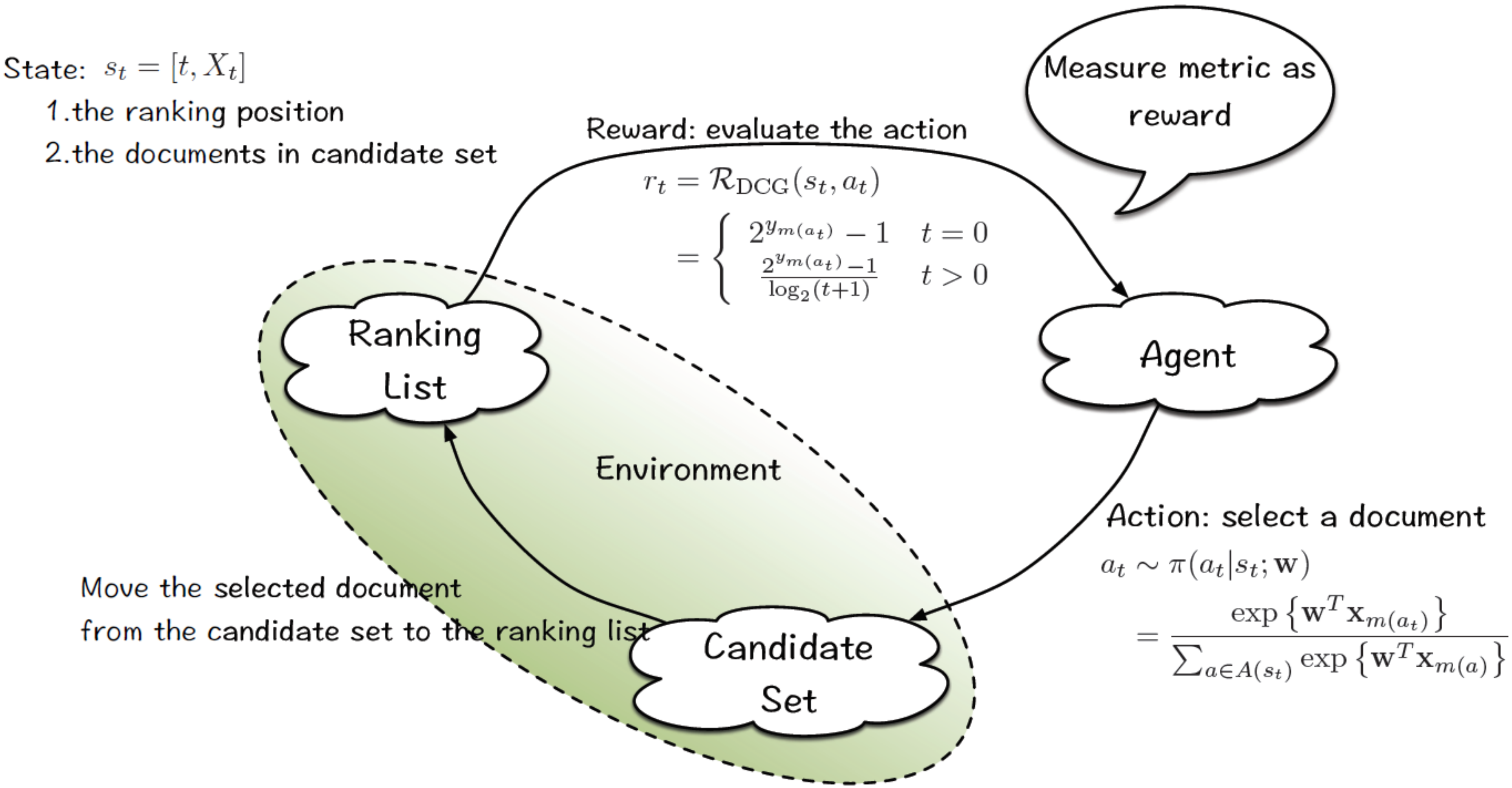
Wei Zeng, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. Reinforcement Learning to Rank with Markov Decision Process. Proceedings of SIGIR 2017, pp. 945-948.

Modeling Relevance Ranking with MDP

$\mathbf{x}_{m(a_t)}$: query-doc relevance features

MDP factors	Corresponding relevance ranking factors
Time steps	The ranking positions
State	$s_t = [t, X_t]$
Policy	$\pi(a_t \mid s_t = [t, X_t]) = \frac{\exp\{\mathbf{w}^T \mathbf{x}_{m(a_t)}\}}{\sum_{a \in A(t)} \exp\{\mathbf{w}^T \mathbf{x}_{m(a)}\}}$
Action	Selecting a doc and placing it to current position
Reward	Based on evaluation measure DCG: $R = \begin{cases} 2^{y(a_t)} - 1 & t = 0 \\ \frac{2^{y(a_t)} - 1}{\log_2(t+1)} & t > 0 \end{cases}$
State Transition	$s_{t+1} = T(s_t = [t, X_t], a_t) = \left[t + 1, X_t \backslash \{\mathbf{x}_{m(a_t)}\},\right]$

The Ranking Process



Learning with Policy Gradient

Algorithm 1 MDPRank learning

Input: Labeled training set $D = \{(q^{(n)}, X^{(n)}, Y^{(n)})\}_{n=1}^{N}$, learning rate η, discount factor γ, and reward function R

Output: $\mathbf{w}$

```
Initialize w ← random values
repeat
    Δw = 0
    for all (q, X, Y) ∈ D do
        (s_0, a_0, r_1, · · · , s_{M−1}, a
        {Algorithm (2), and M =
        for t = 0 to M − 1 do
            G_t ← ∑_{k=1}^{M−t} γ^{k−1} r_{t+
            Δw ← Δw + γ^t G_t ∇
        end for
    end for
    w ← w + ηΔw
until converge
return w
```

Algorithm 2 SampleAnEpisode

Input: Parameters $\mathbf{w}$, q, X, Y, and $\mathcal{R}$

Output: An episode

```
Initialize s_0 ← [0, X], M ← |X|, and episode E ← ∅
for t = 0 to M − 1 do
    Sample an action a_t ∈ A(s_t) ∼ π(a_t | s_t; w) {Equation (2)}
    r_{t+1} ← R(s_t, a_t){Equation (1), calculation on the basis of Y}
    Append (s_t, a_t, r_{t+1}) at the end of E
    State transition s_{t+1} ← [t + 1, X \ {x_{m(a_t)}}]
end for
return E = (s_0, a_0, r_1, · · · , s_{M−1}, a_{M−1}, r_M)
```

Experimental Results

Result on MQ2007 Dataset

Method	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankSVM	0.4045	0.4019	0.4072	0.4383
ListNet	0.4002	0.4091	0.4170	0.4440
AdaRank-MAP	0.3821	0.3984	0.4071	0.4335
AdaRank-NDCG	0.3876	0.4044	0.4102	0.4369
SVMMAP	0.3853	0.3899	0.3983	0.4187
MDPRank	0.4061	0.4101	0.4171	0.4416
MDPRank(return only)	0.4033	0.4059	0.4113	0.4350

Result on OHSUMED Dataset

Method	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankSVM	0.4958	0.4207	0.4164	0.4140
ListNet	0.5326	0.4732	0.4432	0.4410
AdaRank-MAP	0.5388	0.4682	0.4613	0.4429
AdaRank-NDCG	0.5330	0.4790	0.4673	0.4496
SVMMAP	0.5229	0.4663	0.4516	0.4319
MDPRank	0.5925	0.4992	0.4909	0.4587
MDPRank(return only)	0.5363	0.4885	0.4694	0.4591

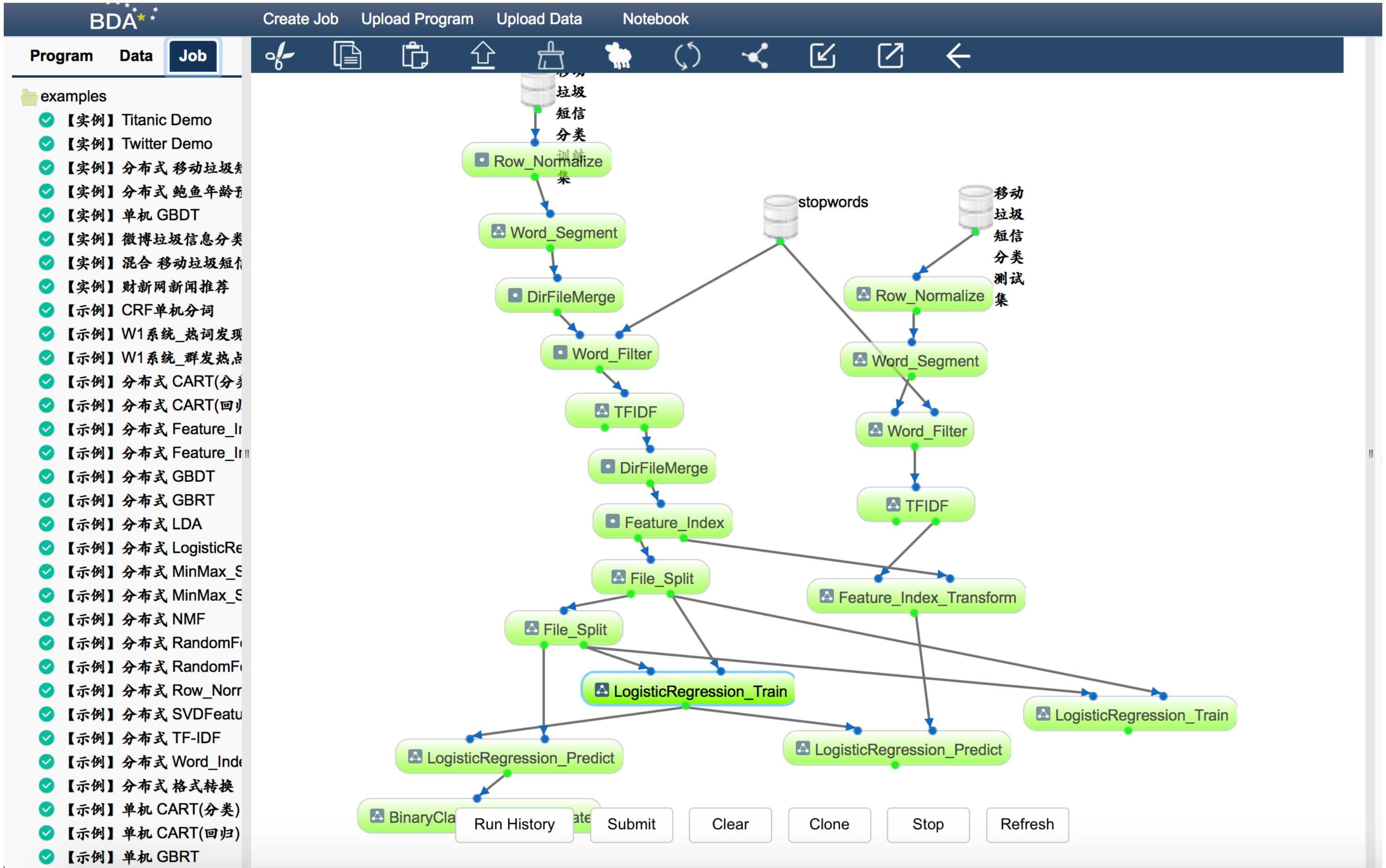
- MDPRank is better because
  - Utilize the IR measures calculated at all the ranking positions as supervision information for training
  - Directly optimizes the IR measure on the training data without any approximation or upper bounding

Outline

- Background: learning to rank for IR
- Reinforcement learning to rank
- Summary

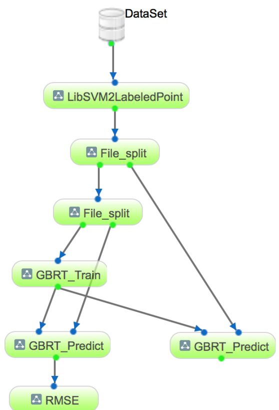
Summary

- Reinforcement learning to rank
  - Ranking as sequential decision making
  - Adapting MDP for the task
  - Learning with policy gradient
- Two examples
  - Diverse ranking
  - Relevance ranking

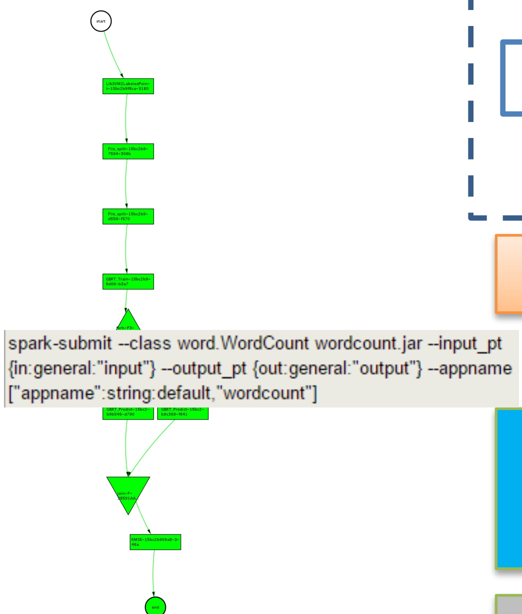


Easy Machine Learning Project

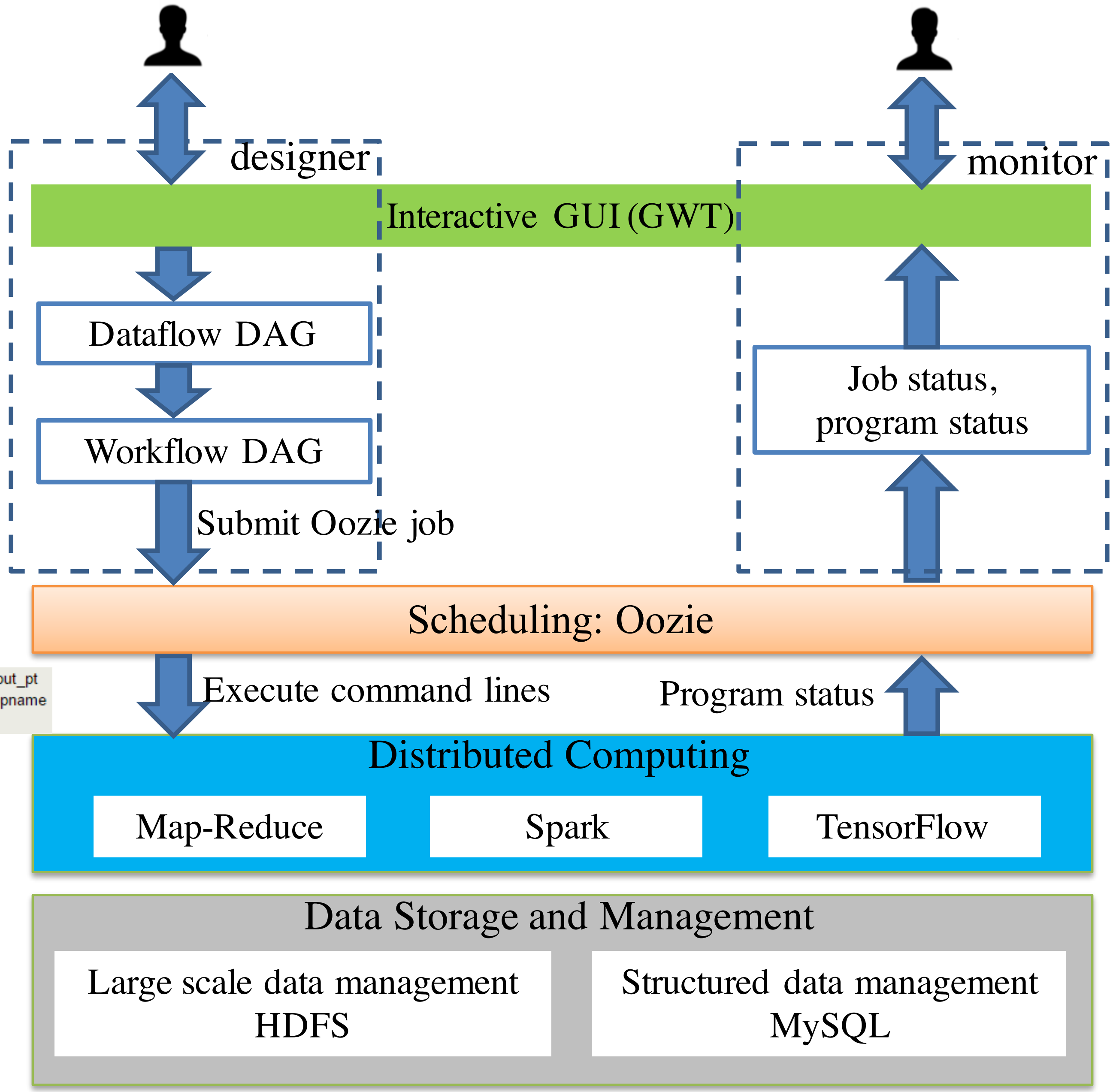
Design of Easy Machine Learning



Node: program / data
Edge: dataflow

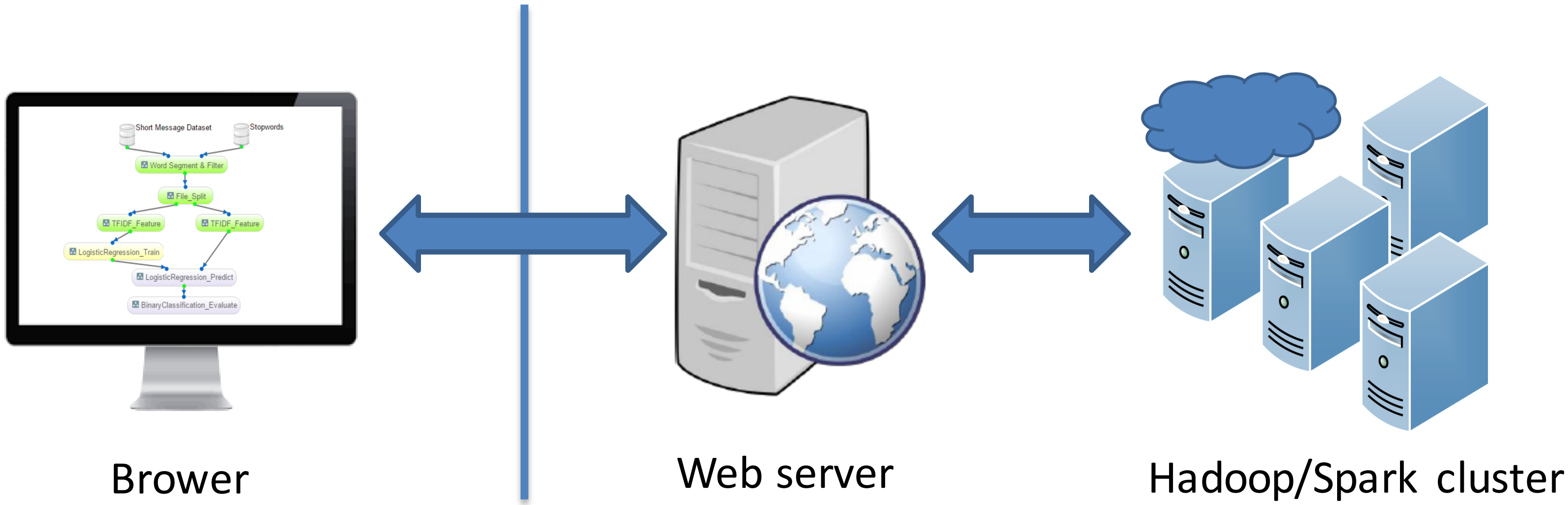


Node: program / start / end / fork / join
Edge: dependency



Deploy as Web Service

http://159.226.40.104:18080/dev

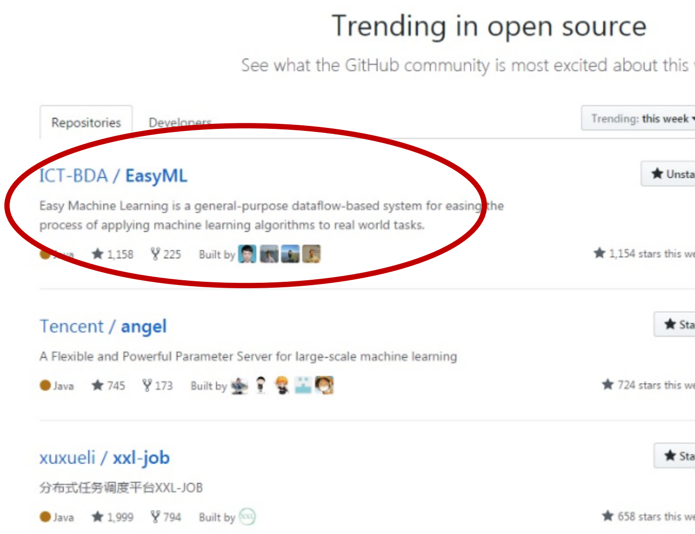


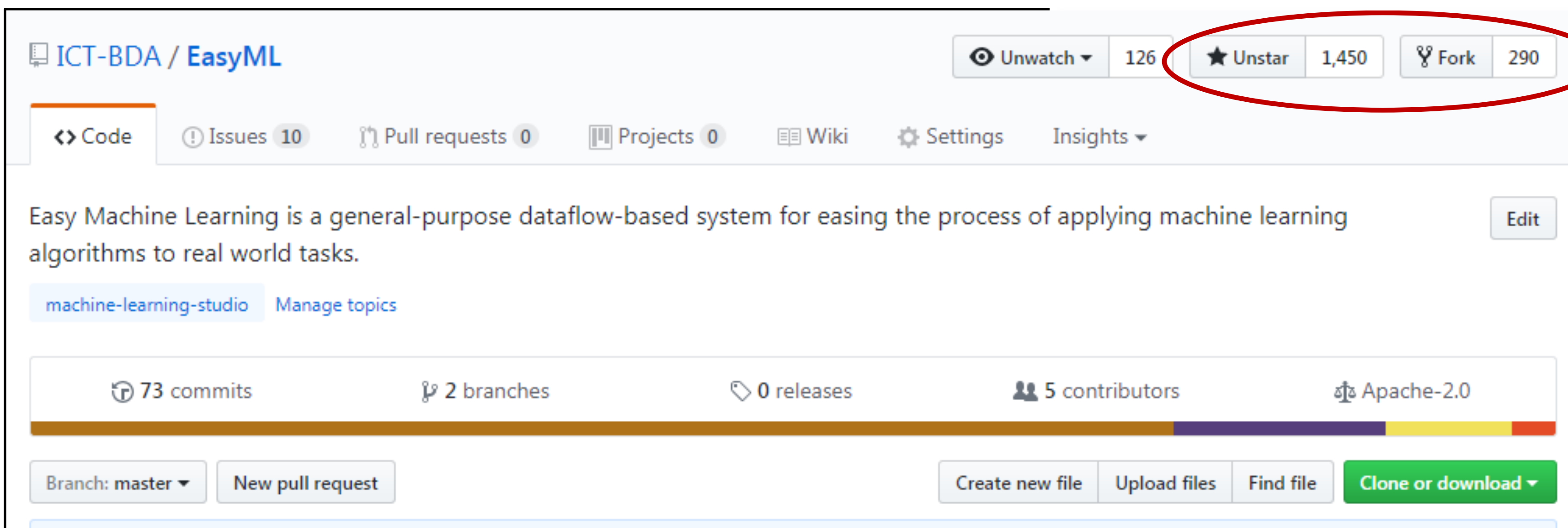
- Advantages
  - **Sharing**: share data/programs/tasks among users
  - **Collaborating**: working together for one task
  - **Mobility**: accessing with web browsers anywhere
  - **Open**: ETL for data import/export; can run third-party programs

Source Shared at Github

https://github.com/ICT-BDA/EasyML

- Top 1 Java project at Github trending for one week
- 1400 + stars and ~300 forks
- CIKM 2016 best demo candidate
  [Guo et al., CIKM ’16]





Thanks!

junxu@ict.ac.cn

http://www.bigdatalab.ac.cn/~junxu