# DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation


Donghua Liu
School of Computer Science
Wuhan University
Wuhan, China
liudonghualdh@whu.edu.cn

Jing Li*
School of Computer Science
Wuhan University
Wuhan, China
leejingcn@whu.edu.cn

Bo Du*
School of Computer Science
Wuhan University
Wuhan, China
remoteking@whu.edu.cn

Jun Chang
School of Computer Science
Wuhan University
Wuhan, China
chang.jun@whu.edu.cn

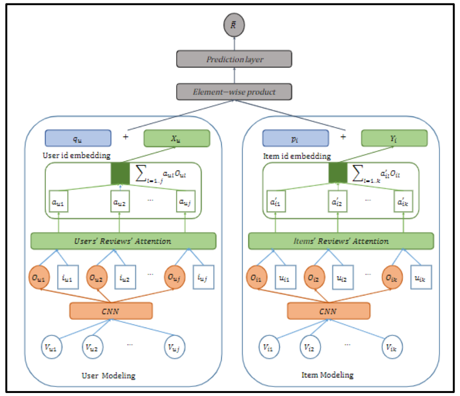
Rong Gao
School of Computer Science
Hubei University of Technology
Wuhan, China
gaorong198149@163.com



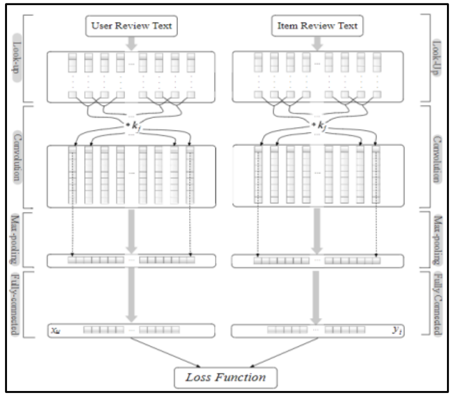
Research Track Paper, ***KDD ’19***, p.344-352, August 4-8, 2019, Anchorage, AK, USA
https://doi.org/10.1145/3292500.3330906

A model **DAML** (**d**ual **a**ttention **m**utual **l**earning between ratings and reviews for item recommendation) was proposed to effectively unify users' ratings and reviews information and realize a dynamic recommender system, which has the ability to extract the latent feature representation of user and item reviews.

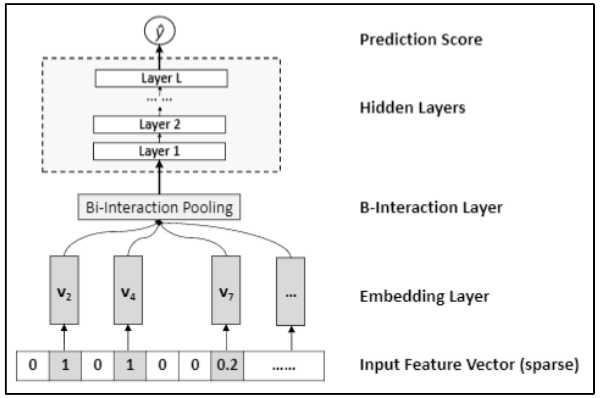
- Mutual attention of the convolutional neural network was used to jointly learn the features of users' reviews.
- A unified neural network model was used to combine the rating features and review features.
- Neural factorization machines are used to process interaction of features to predict rating.




NARRE (2018)[1]: Word Vector Model




CDL (2017) [2]: Item Attributes Representation




Neural Factorization Machines model (2017) [3]

**Traditional recommender system with integrated review and rating information.**

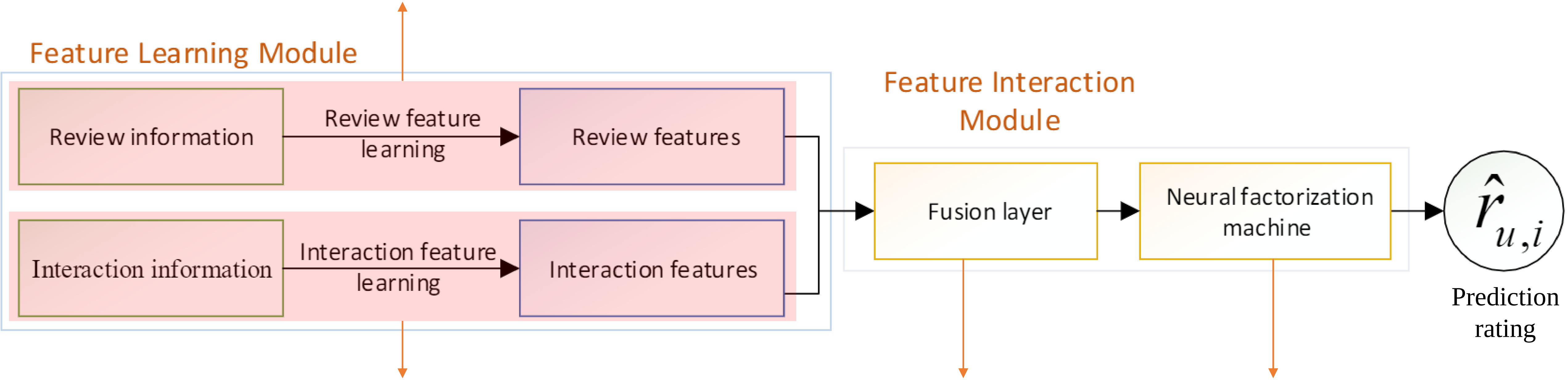
- Learn the latent features of user and item in a static and independent way, which fails to learn the relevance of latent features.
- Without effective framework to unifies ratings and reviews.

DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation.

**Dual attention mutual learning between ratings and reviews for item recommendation (DAML)**

Architecture

- **Attention mechanism** of CNNs is used to learn the correlation between user preferences and item features.




- Map users and items into **low-dimensional dense vectors** to capture the nonlinear interaction of features.

A **unified neural network**, combining two types of feature

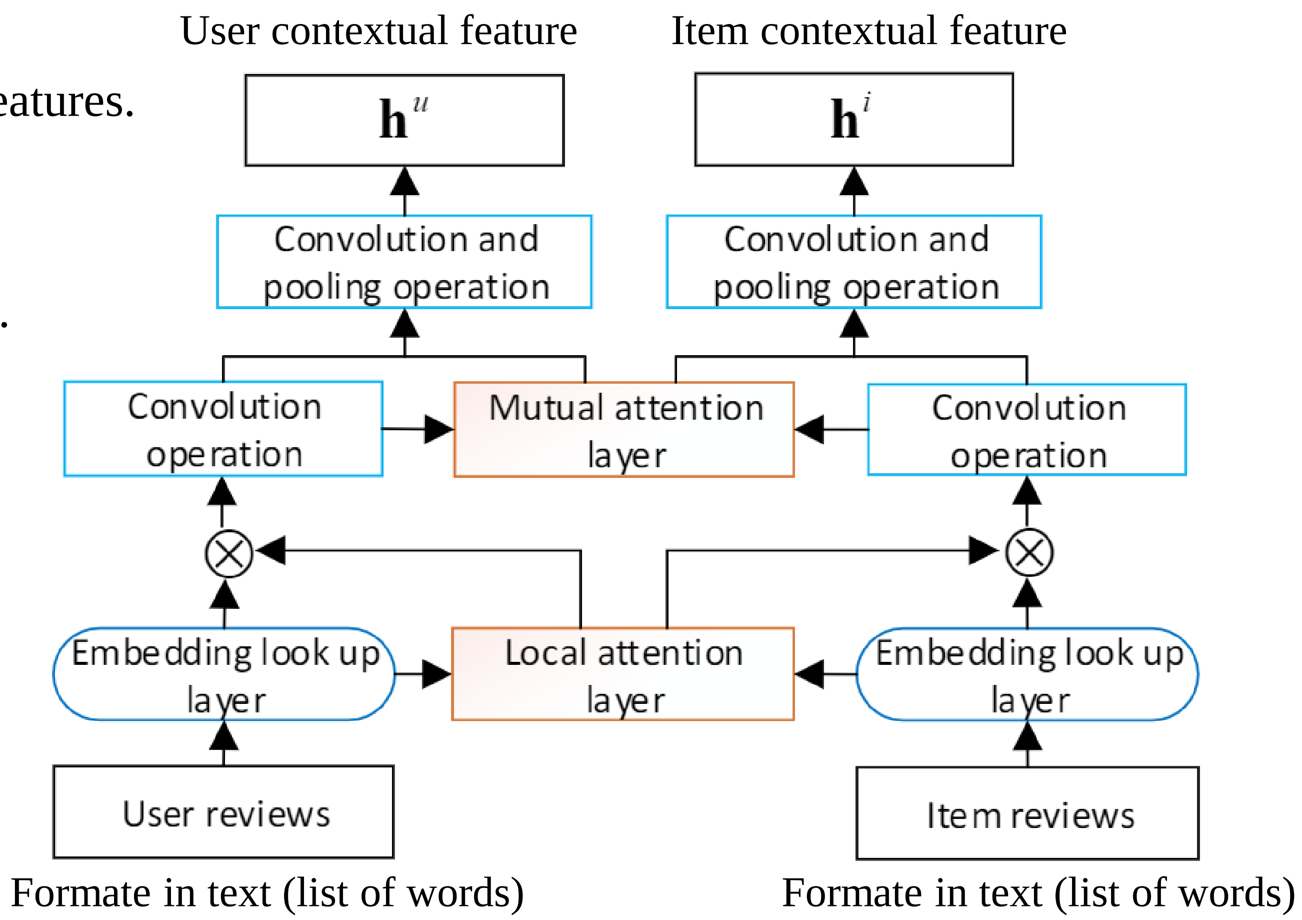
Model the higher-order nonlinear interactions between latent feature vectors.

DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation.

## Dual attention mutual learning between ratingsand reviews for item recommendation (DAML)

Feature learning module

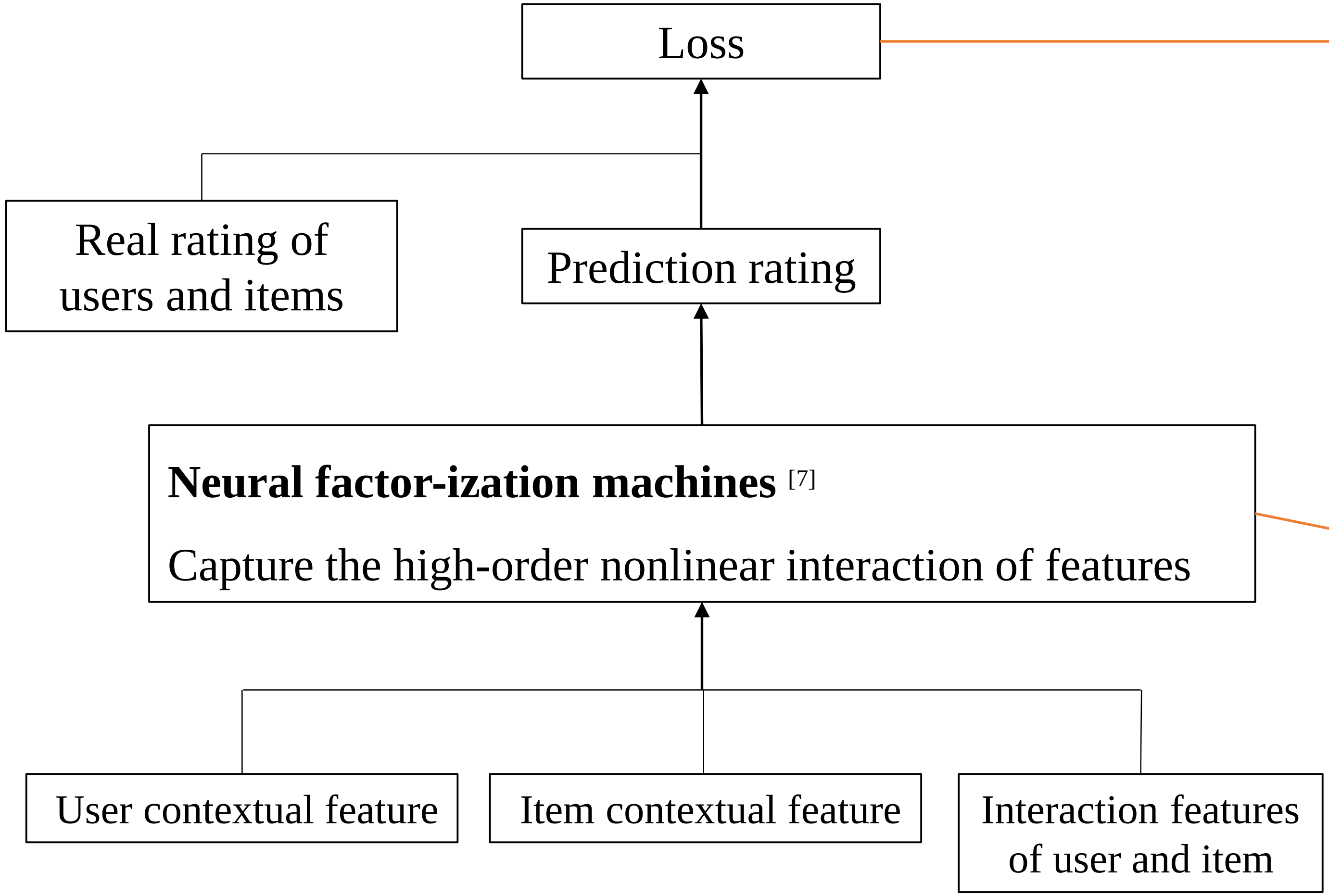
- **Pooling layer** [4]
  Through row-wise averaging to generate more abstract features.
- **Mutual attention layer**
  Measure the correlation between user review and item review. The distance is considered by Euclidean distance.
- **Convolution operation**
  Extract semantic information.
- **Local attention layer** [5]
  Measure the importance of different words in the sentences. Learn the weight of each word in the reviews.
- **Embedding look up layer** [6]
  Word vector model Glove method embed words to a review text matrix.




DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation.

## Dual attention mutual learning between ratingsand reviews for item recommendation (DAML)

Feature Interaction module




Train DAML by the end-to-end paradigm reverse propagation (tochastic gradient descent (SGD) and backpropagation).

Dropout [8] strategy is used to prevent overfitting.

**Adaptive Moment Estimation(Adam)** [9] is used to update DAML feature learning module.

DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation.

# Experiment results

- The number means the Mean Absolute Error (MAE);
- Δ% denotes the performance improvement of DAML over the best baseline;

| Method | Musical Instruments | Office Products | Grocery and Gourmet Food | Video Games | Sports and Outdoors |
|---|---|---|---|---|---|
| PMF | 1.137 | 1.265 | 1.397 | 1.395 | 1.203 |
| NeuMF | 0.7198 | 0.7301 | 0.9434 | 0.8693 | 0.7516 |
| CDL | 0.8336 | 1.062 | 0.9669 | 0.9018 | 0.8522 |
| ConvMF | 0.7860 | 0.7279 | 0.8634 | 0.8993 | 0.8235 |
| DeepCoNN | 0.7590 | 0.7109 | 0.8016 | 0.8752 | 0.7192 |
| D-attn | 0.7420 | 0.7161 | 0.8241 | 0.8422 | 0.7840 |
| NARRE | 0.6949 | 0.6807 | 0.7467 | 0.7991 | 0.6897 |
| CARL | 0.6766 | 0.6469 | 0.7534 | 0.7979 | 0.6864 |
| DAML | **0.6510** | **0.6124** | **0.7354** | **0.7881** | **0.6676** |
| Δ% | 3.78 | 5.33 | 1.51 | 1.23 | 2.74 |

**Performance comparison of 9 methods used in 5 datasets**

- DAML is effective for rating prediction on datasets with different features.
- The gap between DAML & CARL [10] shows that distance metric method of calculating the relevance of features can more intuitively show the correlation between the features.
- Integrating ratings and reviews into a unify network and the high-order non-linear interactions of DAML model can capture more knowledge about users' preferences.

DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation.

# Experiment results

- The number means the Mean Absolute Error (MAE);
- DAML-FM: The DAML model with factorization machines.
- DAML: The DAML with neural factorization machines.

| Method | Musical Instruments | Office Products | Grocery and Gourmet Food | Video Games | Sports and Outdoors |
|---|---|---|---|---|---|
| DAML-FM | 0.6818 | 0.6320 | 0.7437 | 0.7930 | 0.6828 |
| DAML | 0.6510 | 0.6124 | 0.7354 | 0.7881 | 0.6676 |

**The impact of latent feature interactions**

- **High-order non-linear interaction function** can capture the correlation between features in different modalities and improve the recommendation performance.

- Review-outatt: without local and mutual; Review-local: with local, without mutual; Review-mutual: with mutual, without local

| Method | Musical Instruments | Office Products | Grocery and Gourmet Food | Video Games | Sports and Outdoors |
|---|---|---|---|---|---|
| Review-outatt | 0.7095 | 0.7540 | 0.9669 | 0.9018 | 0.8522 |
| Review-local | 0.6985 | 0.7138 | 0.7591 | 0.7958 | 0.6939 |
| Review-mutual | 0.6834 | 0.6945 | 0.7453 | 0.7864 | 0.6884 |
| DAML | 0.6510 | 0.6124 | 0.7354 | 0.7881 | 0.6676 |

**The influence of attention layers**

- The **local** attention layer can effectively distinguish the information words to reduce the noise disturbance and the **mutual** attention layer is able to improve the recommendation performance by identifying the relevance information for user-item pairs.

DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation.

# Deep Reinforcement Learning for List-vise Recommendations

Xiangyu Zhao
Michigan State University
zhaoxi35@msu.edu

Liang Zhang
JD.com
zhangliang16@jd.com

Long Xia
JD.com
xialong@jd.com

Zhuoye Ding
JD.com
dingzhuoye@jd.com

Dawei Yin
JD.com
yindawei@acm.org

Jiliang Tang
Michigan State University
tangjili@msu.edu

***DRL4KDD '19***, August 5, 2019, Anchorage, AK, USA
https://doi.org/10.1145/nnnnnnn.nnnnnnn

A framework **LIRD** (**LI**st-wise **R**ecommendation framework based on **D**eep reinforcement learning) was proposed to realize a dynamic recommender system, which can continuously improve the recommendation strategies during the interactions with users.

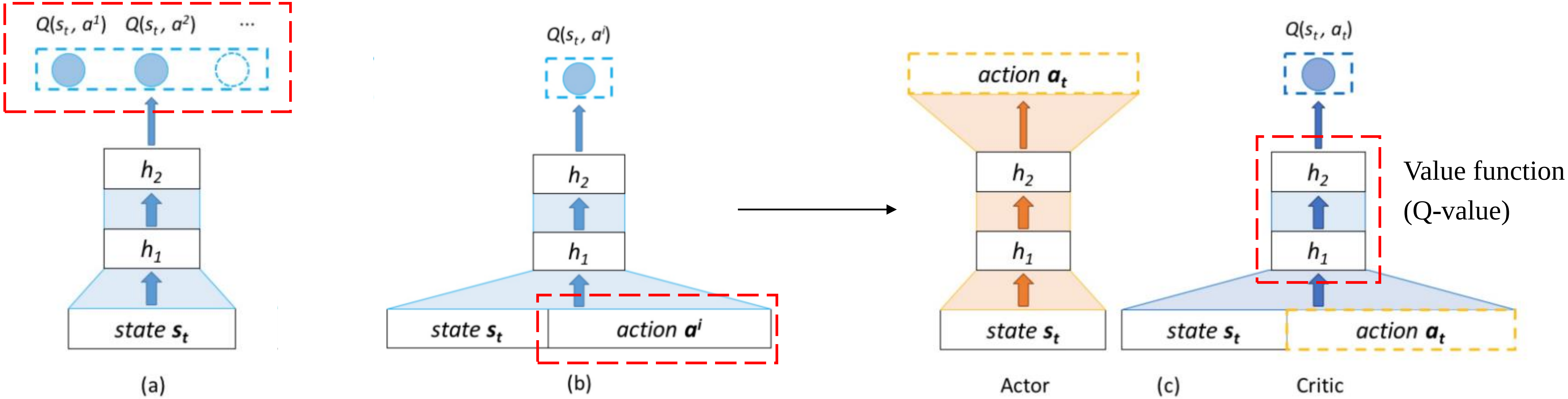
- The recommendation procedure well considered interactions between users and recommender agent.
- The interactions were modeled as a **Markov Decision Process** (MDP) and leverage **Reinforcement Learning (RL)**, to continuously update the strategies during the interactions and maximizing the expected long-term cumulative reward from users.
- **Deep reinforcement learning (DRL)** was leveraged with **artificial neural networks** as the non-linear approximators to estimate the action-value function in RL, which is flexible to support huge amount of items.

Traditional reinforcement learning architectures:

**Deep Q-learning network** [11]

Architectures based on

**Actor-Critic framework**



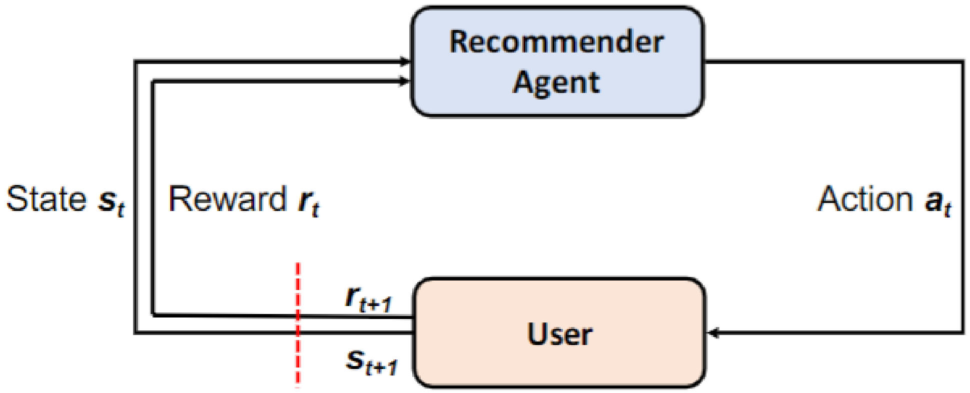

**Cons:** cannot handle large and dynamic **action** space scenario.

**Cons:** temporal complexity.

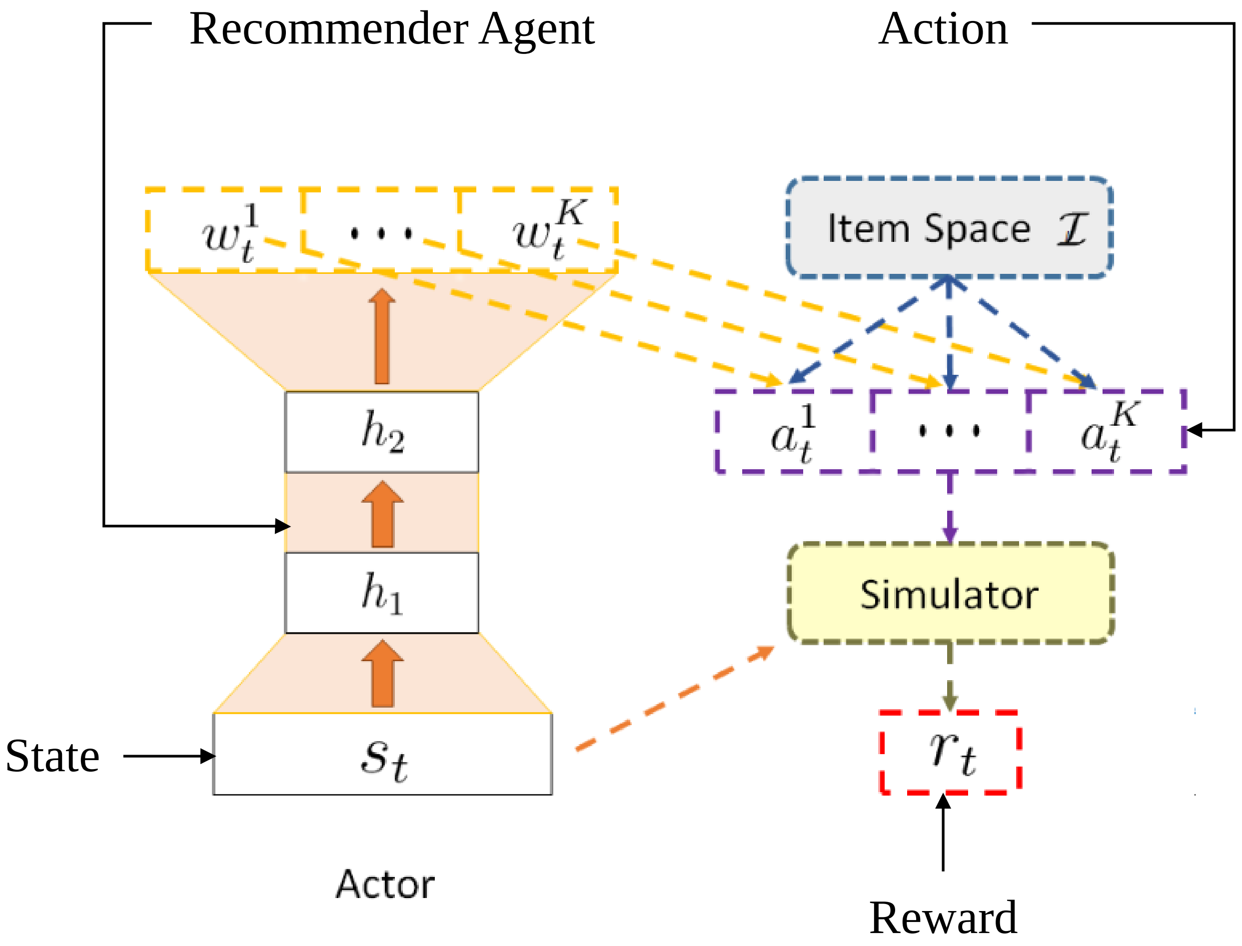
This architecture is suitable for large action space, while can also reduce redundant computation simultaneously.

**LI**st-wise **R**ecommendation framework based on **D**eep reinforcement learning (LIRD)

The Markov Decision Process (MDP)

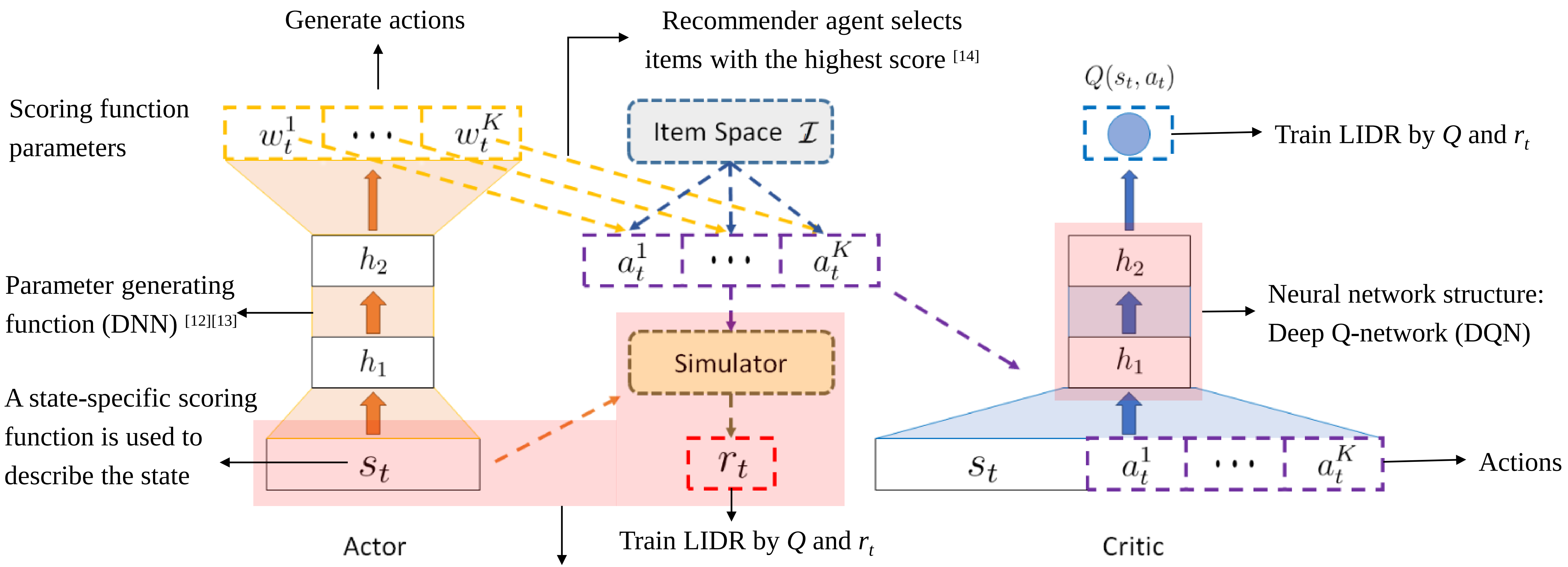



By interacting with user (**state**), the recommender agent takes **actions** to the environment (**user**) to maximize the expected return. Users will give **reward** by different options, like skipping, clicking, or ordering other items. This **reward** will be used to update the recommender agent.




Deep Reinforcement Learning for List-wise Recommendations. Xiangyu Zhao, Liang Zhang, Long Xia, Zhuoye Ding, Dawei Yin, Jiliang Tang.

## **LI**st-wise **R**ecommendation framework based on **D**eep reinforcement learning (LIRD) Structure



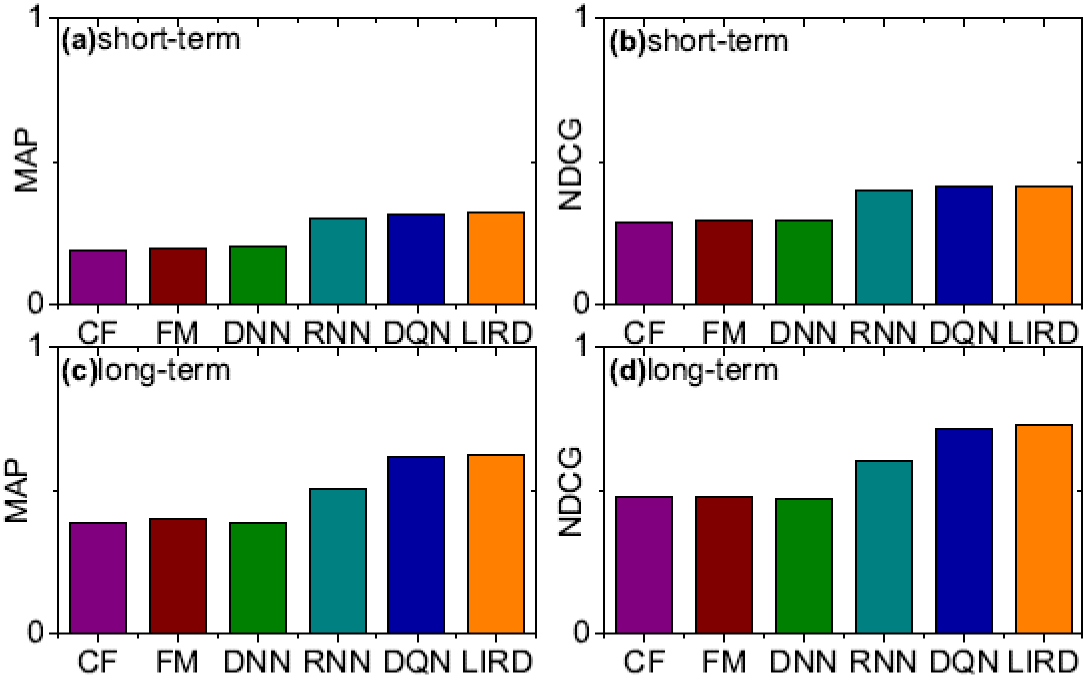

Predict a reward based on current state and a selected action, this process will also update the State.

Experience replay, separated evaluation and target networks, and prioritized sampling strategy are used to train LIRD.

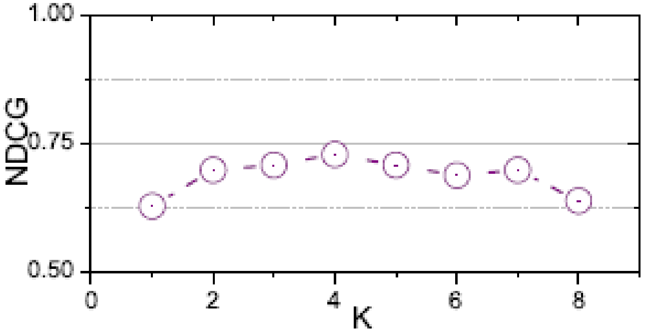
Deep Reinforcement Learning for List-wise Recommendations. Xiangyu Zhao, Liang Zhang, Long Xia, Zhuoye Ding, Dawei Yin, Jiliang Tang.

# Experiment

- The number means the Mean Absolute Error (MAE);
- Normalized Discounted Cumulative Gain (NDCG);

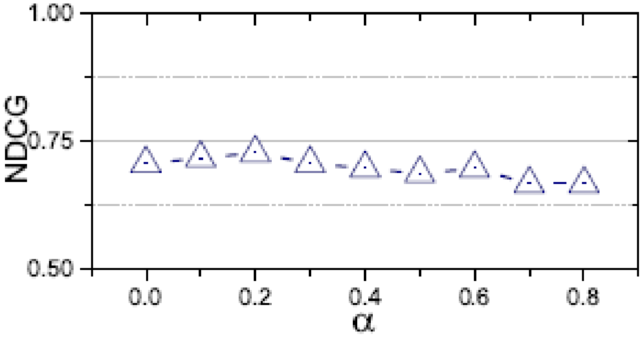



- LIRD framework outperforms most representative baselines in terms of recommendation performance;
- LIRD can be efficiently trained compared to deep Q-network (DQN) [15].




K is the length of state size, which means the number of items in consideration.

- LIRD with a smaller K (recommendation length) could lose some correlations among the items in the same recommendation list; while the proposed framework with a larger K will introduce noises.




α is a parameter controlling the trade-off between state and action similarity in simulator

- LIRD achieves the best performance when α = 0.2. When mapping current state-action pair $p_t(s_t, a_t)$ to a reward, the action-similarity makes more contribution, while state-similarity also influences the reward mapping process.

# Empowering $A*$ Search Algorithms with Neural Networks for Personalized Route Recommendation

Jingyuan Wang*, Ning Wu*
Beijing Advanced Innovation Center for BDBC, School of Computer Science and Engineering, Beihang University, Beijing, China
{jywang,wuning}@buaa.edu.cn

Wayne Xin Zhao†
School of Information, Renmin University of China
Beijing, China
batmanfly@gmail.com

Fanzhang Peng, Xin Lin
MOE Engineering Research Center of ACAT, School of Computer Science and Engineering, Beihang University
Beijing, China
{pengfanzhang,sweeneylin}@buaa.edu.cn

***DRL4KDD '19***, August 4-8, 2019, Anchorage, AK, USA
https://doi.org/10.1145/3292500.3330824

A model **NASR** (**N**euralized **A-S**tar based personalized route **R**ecommendation model) was proposed to automatically learn the cost functions of a classic heuristic algorithm in dealing with personalized route recommendation problem.

- Attention-based **Recurrent Neural Networks (RNN)** was employed to model the cost from the source to the candidate location by incorporating useful context information.
- A **value network** is used to estimate the cost from a candidate location to the destination. The **value network** is built on top of improved **graph attention networks** by incorporating the moving state of a user and other context information. This made it can capture structural characteristics.

## Personalized Route Recommendation (PRR) problem

Aims to generate user-specific route suggestions in response to users' route queries.

**Traditionally use search algorithms by integrating heuristic strategies**

- Require setting the cost functions
- Difficult to use context information

→

**Heuristic algorithm (A* algorithm) with neural networks (RNN)**

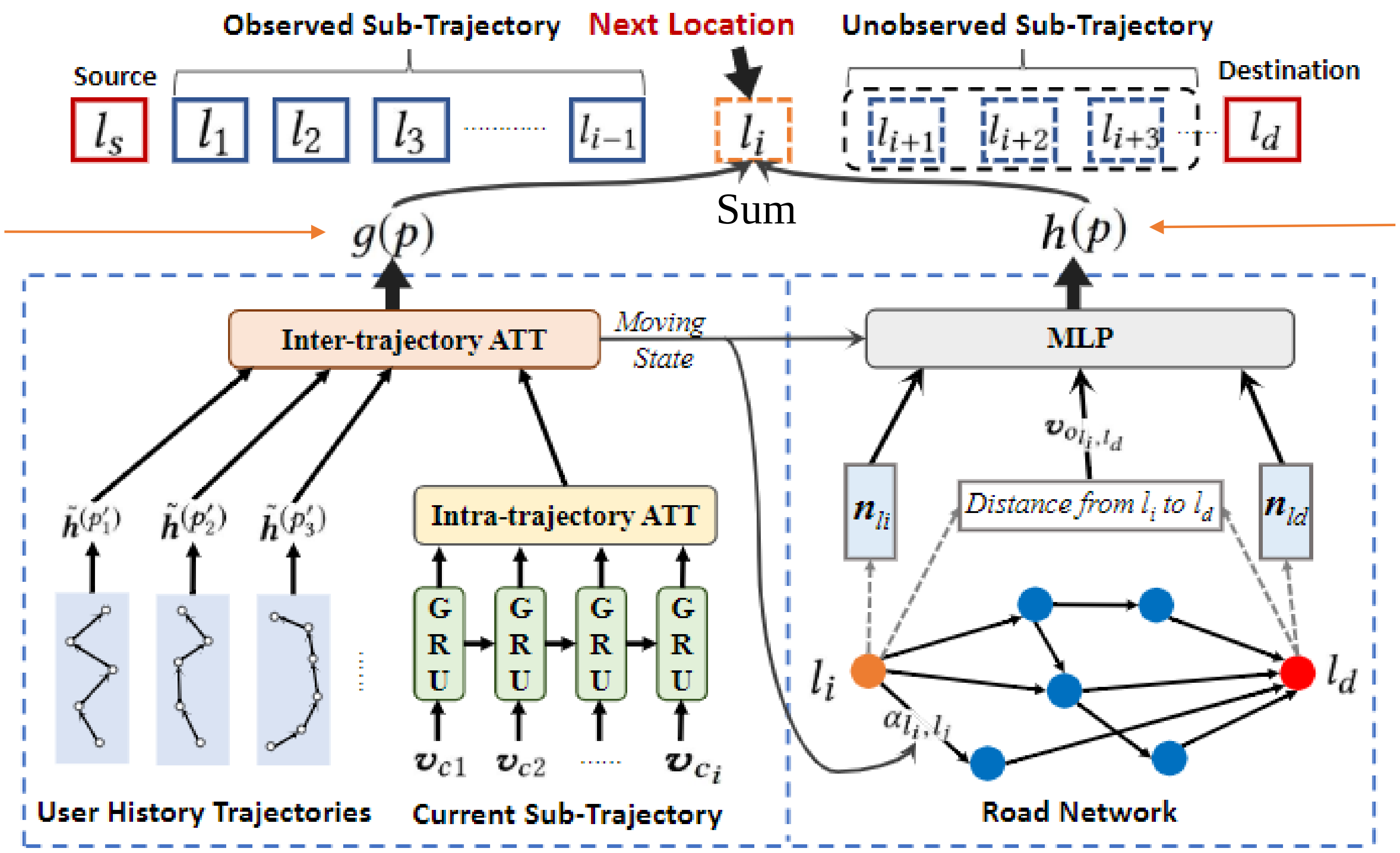
- Can automatically learn the cost functions by time-varying states
- Can make use of useful context information

## Neuralized A-Star based personalized route Recommendation (NASR) model

Use neural networks to automatically learn the cost functions of a classic heuristic algorithm (A* algorithm).

Calculate the cost from the source location to the candidate location

Estimatingthe cost from a candidate location to the destination




**Observable cost calculating part**

Based on RNN

**Estimated cost calculating part**

Based on value network

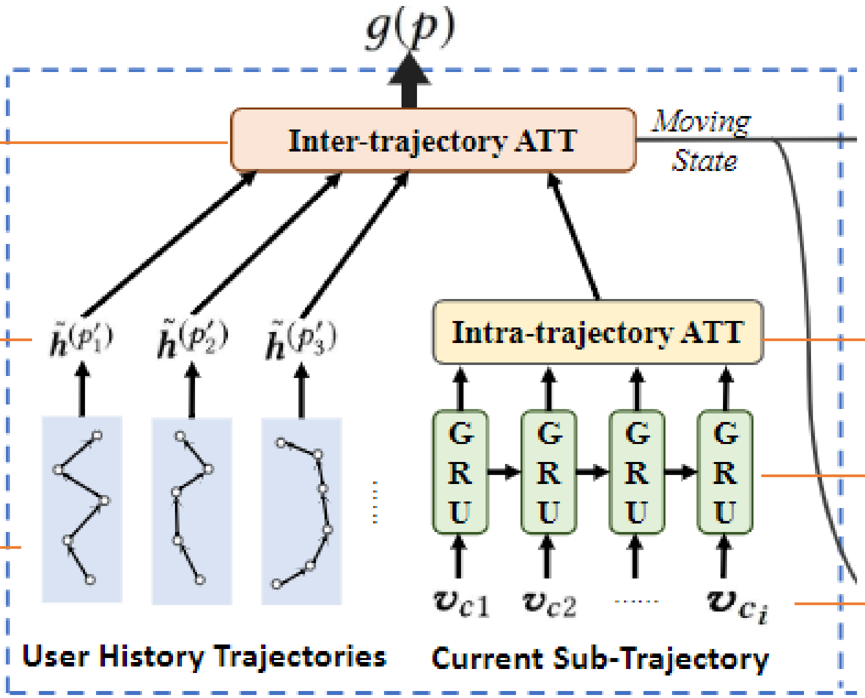
Empowering A* Search Algorithms with Neural Networks for Personalized Route Recommendation

## Neuralized A-Star based personalized route Recommendation (NASR) model

Calculate probability of each location from $l_s$ to $l_i$ to get a loss.

Capture overall moving patterns by incorporating historical trajectories. (also output as moving state to Estimated cost calculating part)

Discover more characteristics by considering the entire trajectory, which can deal with data short and noisy problem.




**Attention Mechanism Method**
Compute the attention between locations in the same trajectory.

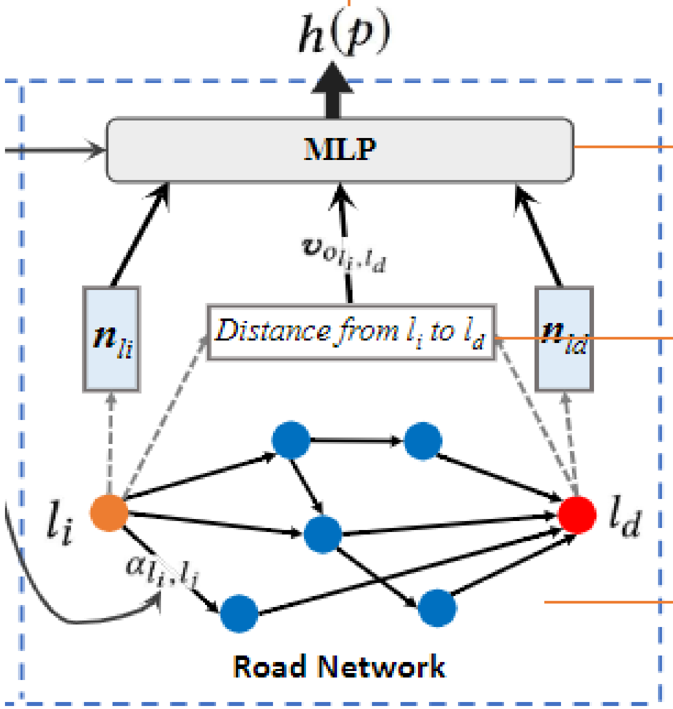
**GRU network** [16] (RNN)

**User's context information** (format in dense vectors)
Contains information for user preference, current location and time.

Users' moving state

**Observable cost calculating part**
Based on Recurrent Neural Networks (RNN)

## Neuralized A-Star based personalized route Recommendation (NASR) model




**Estimated cost calculating part**

Based on value network

- **Temporal Difference** (TD) [17] to value the cost and then consider all the observed trajectories over all users to get the loss.
- **Multi-Layer Perceptron** method to Predicting the Estimated Cost (input contains moving state)
- **Markov Decision Process** (MDP) [18] to optimize the distance
- Value network on top of an improved graph attention network (**Graph ATTention network** (GAT) [19])
  Compute the attention importance of a node on another one.
- Use the **Context-aware Graph Attention** model to describe impacts between road nodes.

# Experiments

The number means the **T-test** value.

RICK [20]
MPR [21]
CTRR [22]
STRNN [23]
DeepMove [24]
NASR [25]

| Datasets | Metric | Precision | | | | | | Recall | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Length | RICK | MPR | CTRR | STRNN | DeepMove | NASR | RICK | MPR | CTRR | STRNN | DeepMove | NASR |
| Beijing Taxi | Short | 0.712 | 0.347 | 0.558 | 0.491 | 0.742 | **0.821** | 0.723 | 0.372 | 0.164 | 0.384 | 0.756 | **0.848** |
| | Medium | 0.638 | 0.253 | 0.276 | 0.446 | 0.642 | **0.757** | 0.651 | 0.261 | 0.067 | 0.350 | 0.654 | **0.773** |
| | Long | 0.586 | 0.169 | 0.194 | 0.359 | 0.562 | **0.684** | 0.589 | 0.173 | 0.045 | 0.214 | 0.575 | **0.709** |
| Porto Taxi | Short | 0.697 | 0.359 | 0.701 | 0.442 | 0.721 | **0.804** | 0.705 | 0.381 | 0.358 | 0.372 | 0.726 | **0.832** |
| | Medium | 0.622 | 0.271 | 0.416 | 0.403 | 0.619 | **0.729** | 0.634 | 0.293 | 0.106 | 0.326 | 0.628 | **0.754** |
| | Long | 0.565 | 0.184 | 0.305 | 0.340 | 0.547 | **0.657** | 0.578 | 0.198 | 0.036 | 0.218 | 0.568 | **0.671** |
| Beijing Bicycle | Short | 0.652 | 0.303 | 0.587 | 0.559 | 0.673 | **0.788** | 0.670 | 0.313 | 0.272 | 0.330 | 0.685 | **0.802** |
| | Medium | 0.568 | 0.217 | 0.603 | 0.461 | 0.582 | **0.715** | 0.574 | 0.226 | 0.142 | 0.304 | 0.589 | **0.724** |
| | Long | 0.503 | 0.129 | 0.613 | 0.297 | 0.487 | **0.641** | 0.519 | 0.139 | 0.045 | 0.206 | 0.492 | **0.663** |

| Datasets | Metric | F1-score | | | | | | EDT | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Length | RICK | MPR | CTRR | STRNN | DeepMove | NASR | RICK | MPR | CTRR | STRNN | DeepMove | NASR |
| Beijing Taxi | Short | 0.717 | 0.359 | 0.253 | 0.431 | 0.749 | **0.834** | 4.594 | 8.287 | 9.082 | 7.551 | 4.362 | **3.376** |
| | Medium | 0.644 | 0.257 | 0.108 | 0.392 | 0.648 | **0.765** | 8.273 | 16.321 | 23.110 | 14.725 | 8.730 | **5.728** |
| | Long | 0.587 | 0.171 | 0.073 | 0.268 | 0.568 | **0.703** | 11.283 | 25.873 | 27.493 | 22.705 | 12.059 | **8.314** |
| Porto Taxi | Short | 0.701 | 0.370 | 0.474 | 0.404 | 0.723 | **0.818** | 4.801 | 8.104 | 6.935 | 8.790 | 4.496 | **3.563** |
| | Medium | 0.628 | 0.282 | 0.169 | 0.360 | 0.623 | **0.741** | 8.619 | 15.032 | 18.294 | 13.368 | 8.930 | **5.949** |
| | Long | 0.571 | 0.191 | 0.065 | 0.266 | 0.557 | **0.687** | 11.379 | 21.349 | 31.745 | 19.603 | 12.297 | **8.572** |
| Beijing Bicycle | Short | 0.661 | 0.308 | 0.372 | 0.414 | 0.679 | **0.795** | 5.183 | 8.924 | 7.784 | 7.092 | 4.629 | **3.719** |
| | Medium | 0.571 | 0.221 | 0.229 | 0.367 | 0.585 | **0.720** | 8.972 | 17.497 | 20.966 | 14.503 | 9.039 | **6.253** |
| | Long | 0.511 | 0.134 | 0.084 | 0.243 | 0.489 | **0.671** | 11.891 | 22.028 | 57.997 | 21.324 | 12.692 | **8.794** |

Performance comparison using four metrics on three datasets. All the results are better with larger values except the EDT measure.

- **Heuristic search methods** are competitive to solve the PRR task, especially when suitable heuristics are used and context information is utilized.
- **Deep learning** is also able to improve the performance by leveraging the powerful modeling capacity.
- **NASR** combines both the benefits of heuristic search and neural networks, and hence it performs best among the comparison methods.

**Experiments**: Based on the result of NASR on F1-score metric and Beijing Taxi dataset

Variants of the **attention mechanism**

- **NA**: without attention
- **IA**: only intra-trajectory attention
- **BA**: both intra- and inter- trajectory attention
- **BA-S**: no moving state for the h(·) function

Variants for the **value network**

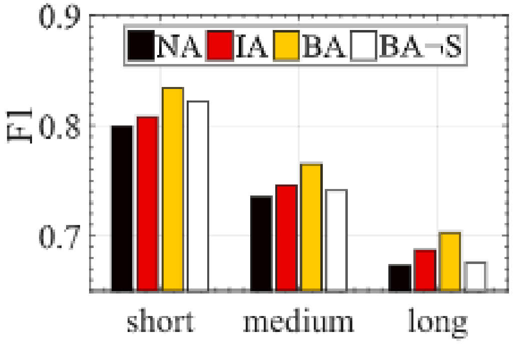
- **ED**: Euclid distance as heuristics
- **SP**: the scalar product between the embeddings of the candidate and destination locations
- **O-GAT**: origin graph attention networks
- **i-GAT**: improved GAT

Variants for the **learn model**

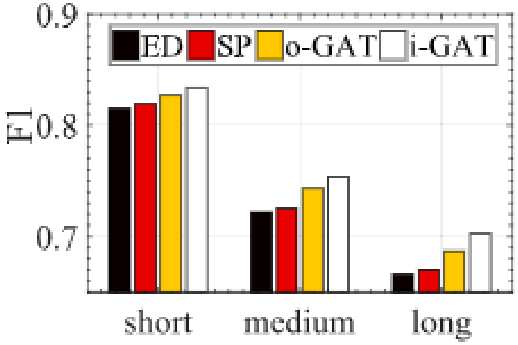
**SL**: directly learns the distance in supervised way

**MC** [25]: Monte Carlo method

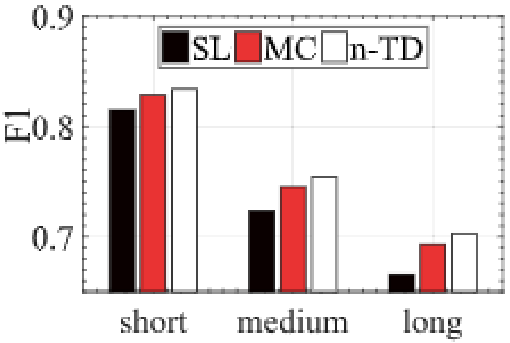
**N-TD**: method used in NASR




(a) Examining the RNN component.




(b) Examining the value network.




(c) Examining the TD learning method.

**Result**: Both inter- and intra-trajectory attention are important to improve the performance of the PRR task. And the moving state is also useful.

**Result**: Simple heuristics may not work well in this task (like ED). Graph attention networks are more effective to capture structural characteristics from graphs.

**Result**: Simplest supervised learning method performs worst. Since the prediction involves multi-step moving process, it is not easy to directly fit the distance using traditional supervised learning methods.

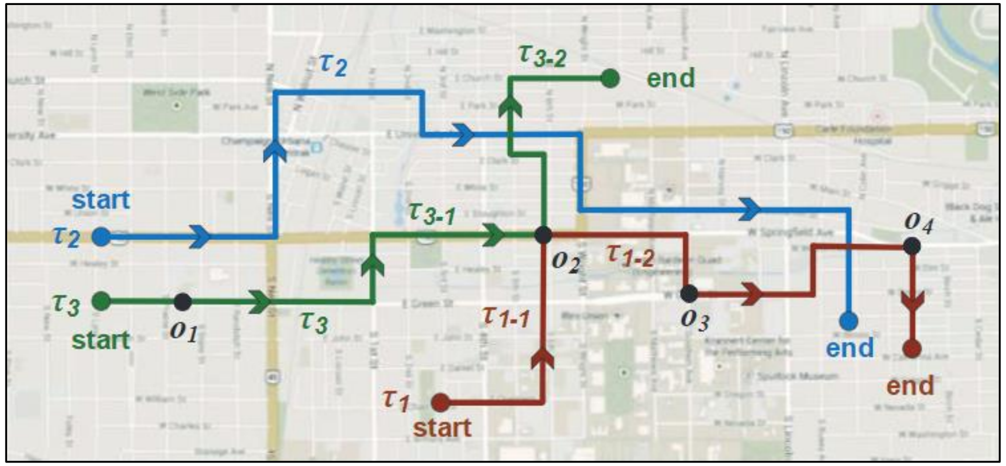
# Effective and Efficient Reuse of Past Travel Behavior for Route Recommendation

Lisi Chen[1], Shuo Shang[2,*], Christian S. Jensen[3], Bin Yao[4], Zhiwei Zhang[5], Ling Shao[6]
[2]UESTC, China, [1,2,6]Inception Institute of Artificial Intelligence, UAE, [3]Aalborg University, Denmark
[4]Shanghai Jiao Tong University, China, [5]Hong Kong Baptist University
[1]lisi.chen@inceptioniai.org, [2]jedi.shang@gmail.com, [3]csj@cs.aau.dk
[4]yaobin@cs.sjtu.edu.cn, [5]cszwzhang@comp.hkbu.edu.hk, [6]ling.shao@inceptioniai.org

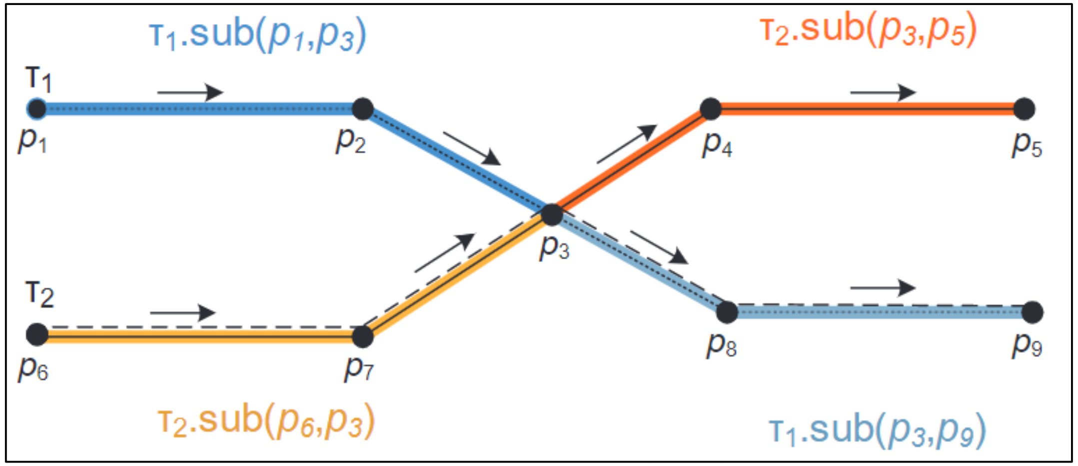
***DRL4KDD ’19***, August 4-8, 2019, Anchorage, AK, USA
https://doi.org/10.1145/3292500.3330835

Two algorithms **FSPS** (The **F**ully-**S**plit **P**arallel **S**earch Algorithm) and **GSPS** (The **G**roup-**S**plit **P**arallel Algorithm), which are **p**arallel **s**plit-and-**c**ombine approaches, were proposed to efficiently and effectively address massive route data and deal with **r**oute **s**earch by **l**ocation problem (**RSL-Psc**).

- **Network expansion** and **exploit spatial similarity bounds** were used to prune route data.
- The algorithms split candidate routes into **sub-routes** and combine them to construct new routes, which would be an efficient way to process route data.
- Processes in **GSPS** are independent and are performed in parallel, which can reduce the burden of calculation.




$\tau_2$ will be the top-1 result, but it is of low quality (i.e., relatively far away from the query locations)




The split and combine policy for two roughts with at least one same point.

## Traditional way to deal with RSL problem

- The quality of query results can not be guaranteed due to insufficient data.

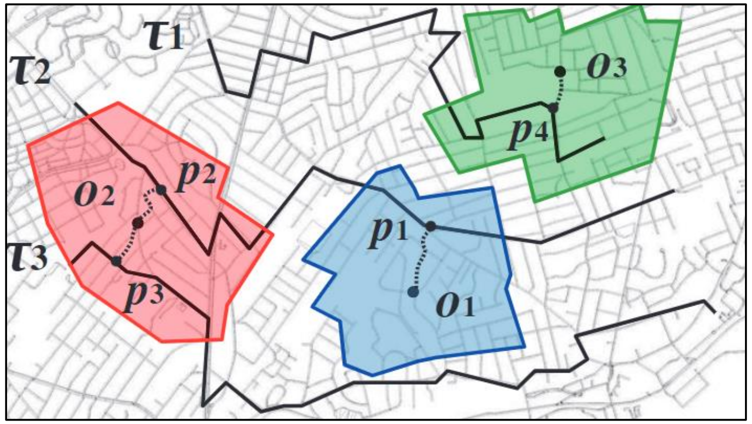
## Split-and-combine approach to deal with the RSL problem

- This approach can be efficient and effective.

# The Fully-Split Parallel Search (FSPS) algorithm

Query locations

Trajectories




**Network expansion (Dijkstra's algorithm [26])**
Explore the spatial network and retrieve route candidates close to each center.

**Route Candidates**
Select routes with high similarity to query locations (i.e. "close to" query locations).
Presented by tuples.

**Artitioning and query location group**
Disjoint query locations are seen as a group. The final route is generated from a set of qualified route candidates from each group.

**Route combination**
Combine two route candidates from two different query location groups as aroute candidate in the next level. Then continuely find the final route.

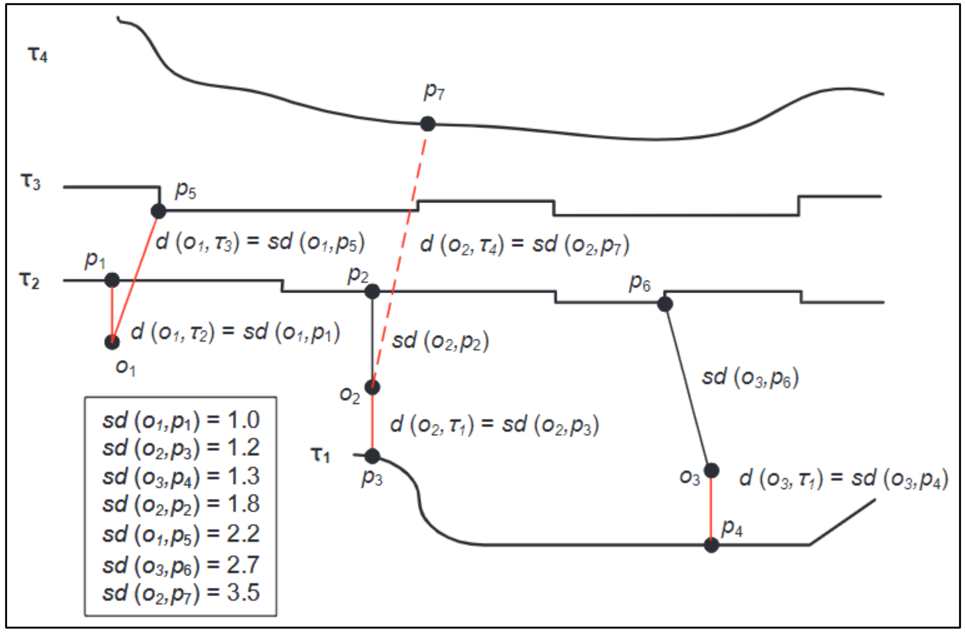
Final route

**Cons:** computing route candidate results in a very large (exponential) number of combination possibilities, witch makes the process computationally expensive.

## The Group-Split Parallel Search (GSPS) algorithm

Query locations → Groups

Find an ideal splid batch size




**Group-based network expansion**

Network expansion is performed individually for each query location, with a newly considered similarity upper bound for each group.

**Route candidate filtering (CTFilter)**

Reduce the unqualified candidates

**Find Route Candidates for each group**

Similar with FSPS

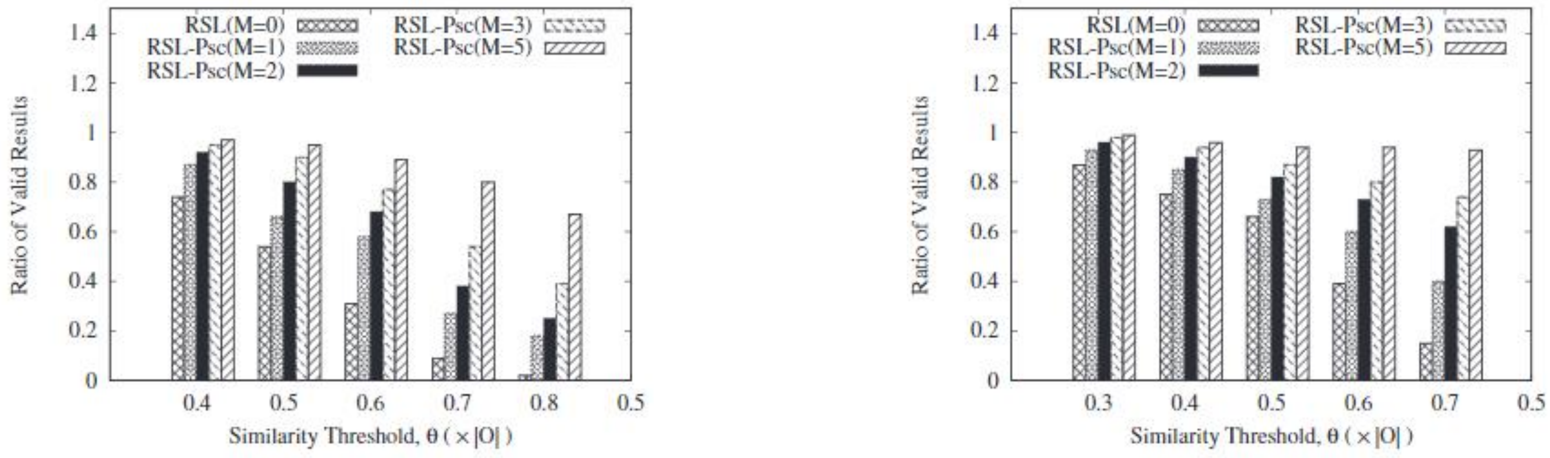
**Route combination**

Combine and evaluate the route candidates in the same partitioning

Final route

# Experiments

- **Radio of Valid Results**: the ratio of queries that return valid results
- **Similarity Threshold**: the lowest similarity to be validated
- **M**: the number of route combinations to be queried
- **Beijing Road Network (BRN)** and **New York Road Network (NRN)** are two road networks.




(a) BRN (b) NRN

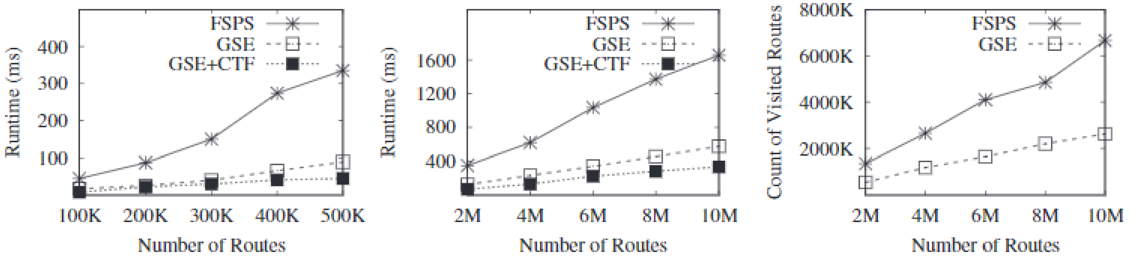
**Result**: The **RSL-Psc** query, even with a small number of route combinations, demonstrates superiority over the **RSL** query (without route combination) with regards to the probability of returning a valid result route.

# Experiments

- **FSPS:** The Fully-Split Parallel Search algorithm
- **GSE**: Group-Split Parallel Search without CTFilter (GSPS Expansion only)
- **GSE+CTF**: Group-Split Parallel Search (GSPS Expansion +CTFilter)

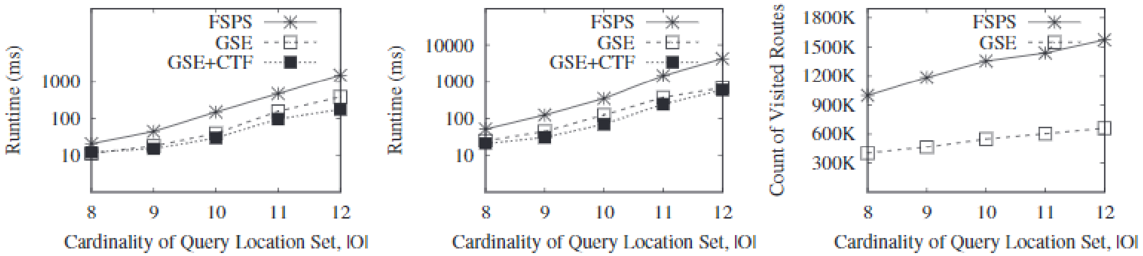
Large **Cardinality of query location set** means:

- A larger search space with more routes to be accessed and evaluated
- A larger number of possible partitions and split-and-combine sub-tasks




(a) BRN-time (b) NRN-time (c) NRN-count

**Effect of the number of routes**



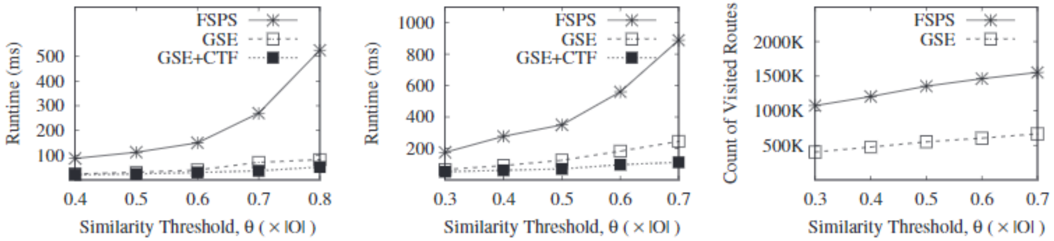

(a) BRN-time (b) NRN-time (c) NRN-count

**Effect of cardinality of query location set**

**Result**:
Both the CPU time and the count of visited routes are expected to increase for all three algorithms.
Both Group-Split Parallel Search and CTFilter are helpful for CPU time.

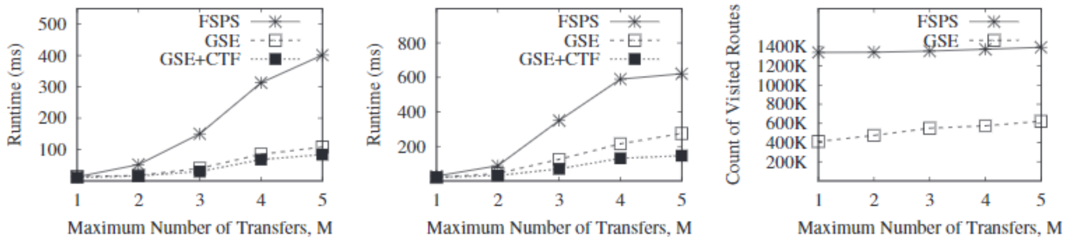
**Result**:
CPU time increases less for all algorithms, which may mean that some qualified routes require only 0, 1, or 2 transfers.

# Experiments



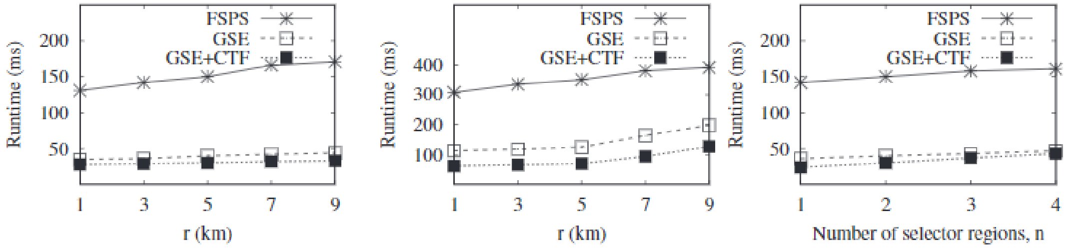

(a) BRN-time (b) NRN-time (c) NRN-count

**Effect of Similarity Threshold**



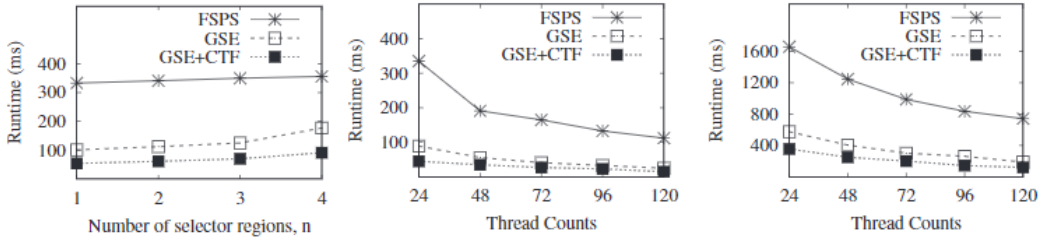

(a) BRN-time (b) NRN-time (c) NRN-count

**Effect of Maximum Number of Transfers**




(a) BRN-time (b) NRN-time (a) BRN-time

**Effect of the radius of query location selector region**




(b) NRN-time (a) BRN-time (b) NRN-time

**Effect of thread counts**

**Result**:

- A larger value of $\theta$ leads to higher pruning effectiveness, which may improve the efficiency;
- A larger value of $\theta$ may postpone the termination of the algorithms;
- When we increase M, the CPU time increases for all algorithms;
- The performance of route counts is relatively consistent as we increase M;
- The CPU time and the count of visited routes for all three algorithms increase with the number of query location selector regions;
- GSE+CTF outperforms FSPS and GSE.

# Environment Reconstruction with Hidden Confounders for Reinforcement Learning based Recommendation


Wenjie Shang
National Key Laboratory for Novel Software Technology
Nanjing University
shangwj@lamda.nju.edu.cn

Yang Yu
National Key Laboratory for Novel Software Technology
Nanjing University
yuy@nju.edu.cn

Qingyang Li
AI Labs, Didi Chuxing
qingyangli@didiglobal.com

Zhiwei Qin
AI Labs, Didi Chuxing
qinzhiwei@didiglobal.com

Yiping Meng
AI Labs, Didi Chuxing
mengyipingkitty@didiglobal.com

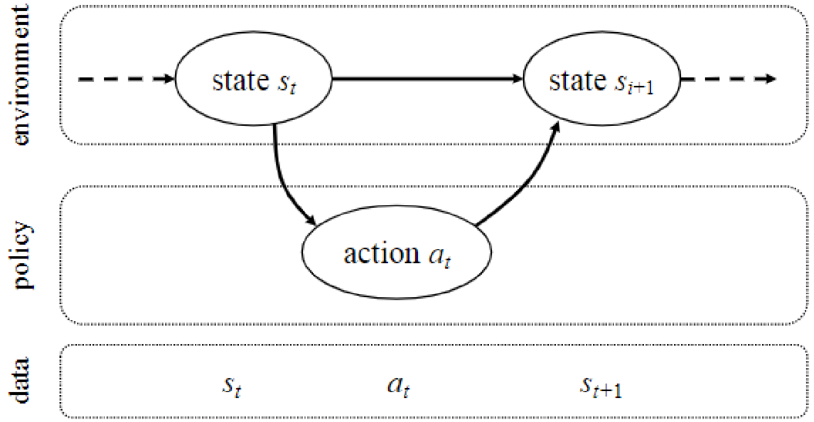
Jieping Ye
AI Labs, Didi Chuxing
yejieping@didiglobal.com

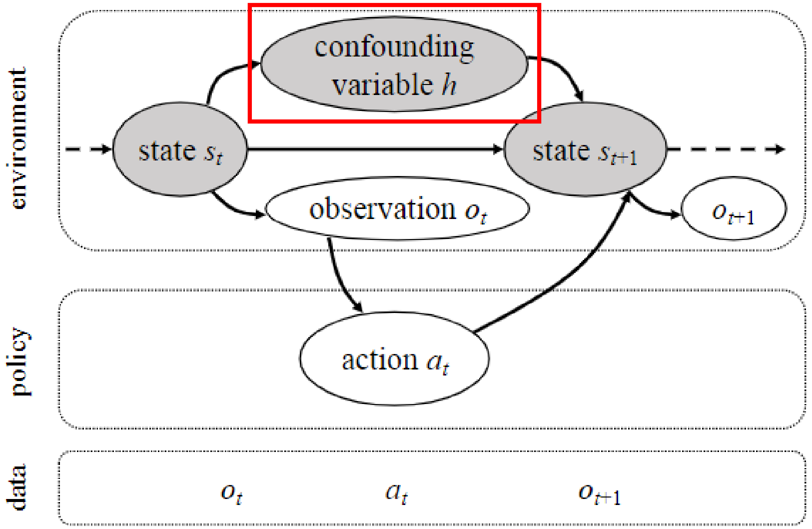


***DRL4KDD '19***, August 4-8, 2019, Anchorage, AK, USA
https://doi.org/10.1145/3292500.3330933

A approach **DE**confounded **M**ulti-agent **E**nvironment **R**econstruction (**DEMER)** was proposed to learn the environment with hidden confounders.

- **Multi-agent generative adversarial imitation learning framework** was adopted in **DEMER** to introduce the confounder embedded policy;
- A **compatible discriminator** is used for training the policies.




**Traditional recommendation method** (e.g. MAIL) under the assumption that the whole world consists of two agents only (policy & environment).

In real world situation, the scenario is too complex to offer a fully observable environment, which means that it might exist the **hidden confounders**.

# Deconfounded multi-agent environment reconstruction (DEMER)

## Interaction between agents and hidden confounders

**A (Policy agent)**
$o_A$: observation
$\pi_a$: policy
$a_A = \pi_A(o_A)$: action

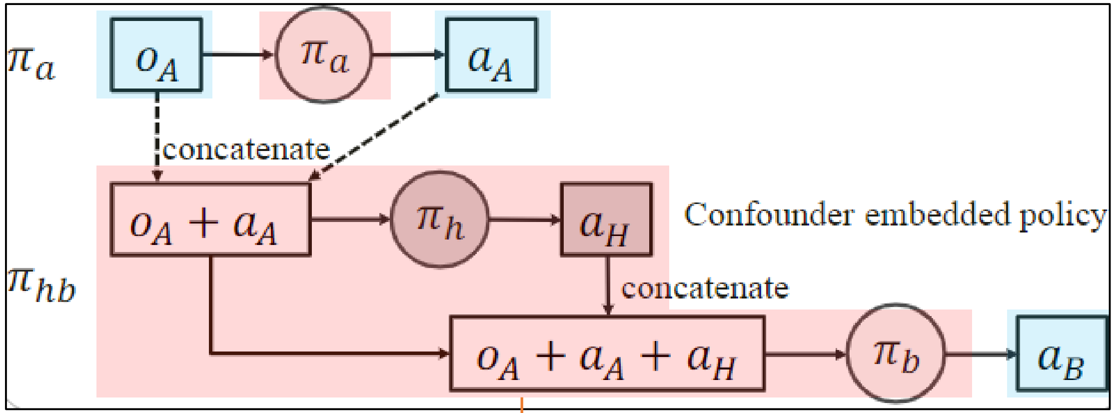
Interaction

**B (Environment)**
$o_B$: observation
$\pi_b$: policy
$a_B = \pi_b(o_B)$: action

**H (Hidden confounders)**
$o_H$: observation
$\pi_h$: policy (dynamic)
$a_H = \pi_h(o_H)$: action

## The generator of DEMER

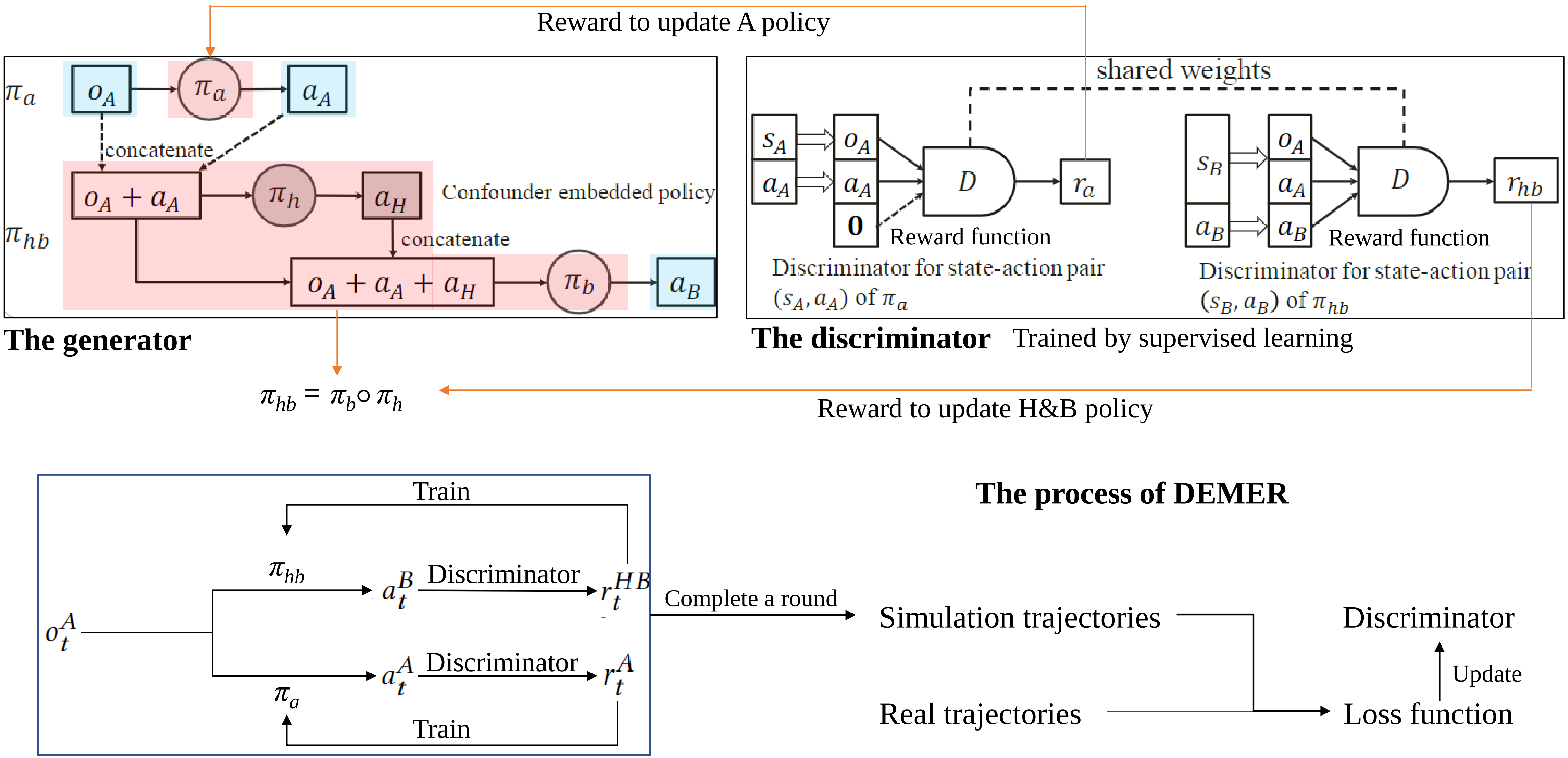
Structure in Generative adversarial networks (GANs)




$$\pi_{hb} = \pi_b \circ \pi_h$$

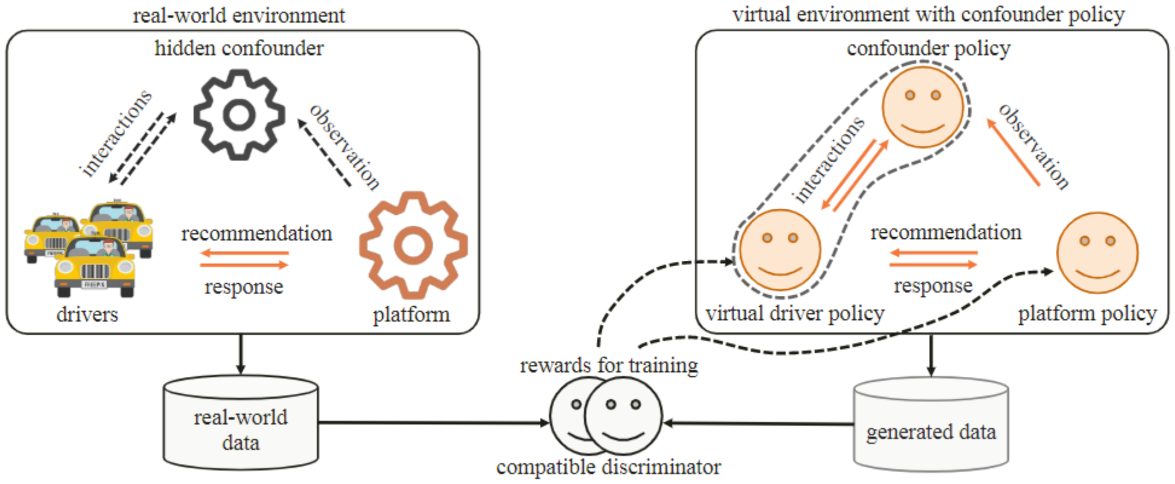
- Blue means ‘observed’;
- Red means ‘to be trained’, the rewards for updating come from the discriminator.

# Deconfounded multi-agent environment reconstruction (DEMER)



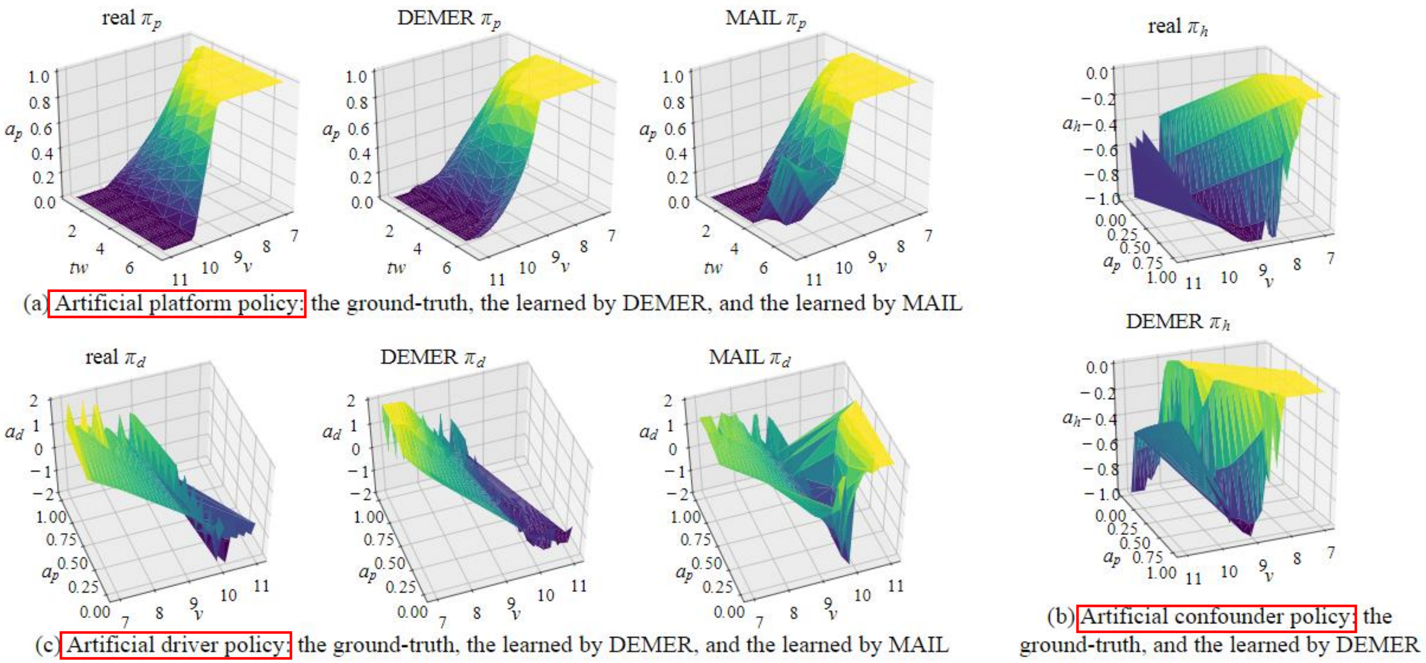

Environment Reconstruction with Hidden Confounders for Reinforcement Learning based Recommendation.

# DEMER framework applied in the driver program recommendation




- In **real world**, we can collect the data of drivers and the platform, while the hidden confounder can not be observed;
- In the **virtual environment** we can observe three policies, including platform, drivers and the confounder policy.

## Experiments




(a) Artificial platform policy: the ground-truth, the learned by DEMER, and the learned by MAIL

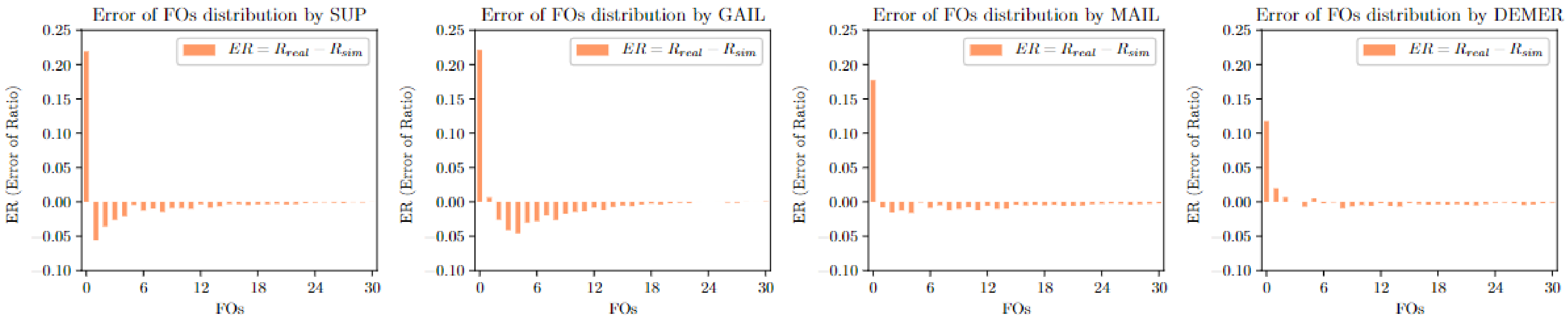
(b) Artificial confounder policy: the ground-truth, and the learned by DEMER

(c) Artificial driver policy: the ground-truth, the learned by DEMER, and the learned by MAIL

- $tw$ : timestamp indicator, keep in cycling in running sequences
- $v$ : a key variant (denotes the driver's response)
- $a_p$ : **a**ction by **p**latform policy
- $a_d$ : **a**ction by **d**river policy
- $\pi_p$ : artificial **p**latform policy
- $\pi_d$ : artificial **d**river policy
- MAIL: Multi-agent adversarial imitation learning, without modeling the hidden confounder

**Result**:
- Policies produced by **DEMER** are more similar to the real function space than those by **MAIL**, since MAIL ignore hidden confounders;
- It is still hard to match **confounder policy** since it is fully unobservable.

# Experiments




- **FOs**: number of finished orders
- **SUP**: Supervised learning of the driver policy with historical state-action pairs, i.e., behavioral cloning
- **GAIL**: GAIL to learn the driver policy, given the historical record of program recommendation as a static environment
- **MAIL**: Multi-agent adversarial imitation learning, without modeling the hidden confounder
- **DEMER**: The proposed method in this study

**Result:**

- The simulation distributions by SUP and GAIL are biased apparently when FOs is low, since that these two methods use whole or partial real data directly for building simulators;
- FOs distribution by DEMER is exactly closer to the real than by MAIL, where the confounder setting makes difference explicitly.

| Methods | Training set | Testing set |
|---|---|---|
| SUP | 17.09 | 18.00 |
| GAIL | 18.43 | 17.85 |
| MAIL | 15.27 | 14.52 |
| DEMER | **21.74** | **21.21** |

Comparison of test **log-likelihood** on real data.

| Methods | FOs | TDIs |
|---|---|---|
| SUP | -0.0213 | 0.0010 |
| GAIL | 0.4987 | 0.4252 |
| MAIL | **0.8129** | 0.7861 |
| DEMER | 0.7945 | **0.8596** |

Comparison of **Pearson correlation coefficients** on FOs and TDIs trend lines.

- **FOs**: number of finished orders
- **TDIs**: **t**otal **d**river **i**ncomes

**Result:**

- **DEMER** got the best performance in log-likelihood testing, which means the confounder setting plays a positive role;
- **DEMER** and **MAIL** achieve high correlations to the real with Pearson correlation coefficient.

# 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

***DRL4KDD '19***, August 4-8, 2019, Anchorage, AK, USA
https://doi.org/10.1145/3292500.3330832

An architecture **G**raph **A**ttention **N**etworks (**GAttN**) was proposed to tackle a NP-hard problem: Exact-K recommendation problem. GattN can end-to-end learn the joint distribution of items and generate an optimal card as recommendation.

- **Exact-K** recommendation problem is a novel type of recommendation problem, compared with traditional **Top-K** recommendation problem. **GAttN** was designed mainly focus on **Exact-K** problem.;
- By graph embedding technology, Exact-K problem can be transferred to **M**aximum **C**lique **O**ptimization (**MCO**) problem;
- A **Multi-head Self-attention** encoder and a decoder with attention mechanism are designed in **GAttN** to process data;
- Approach of **R**einforcement **L**earning from **D**emonstrations (**RLfD**) was used to train **GattN**.

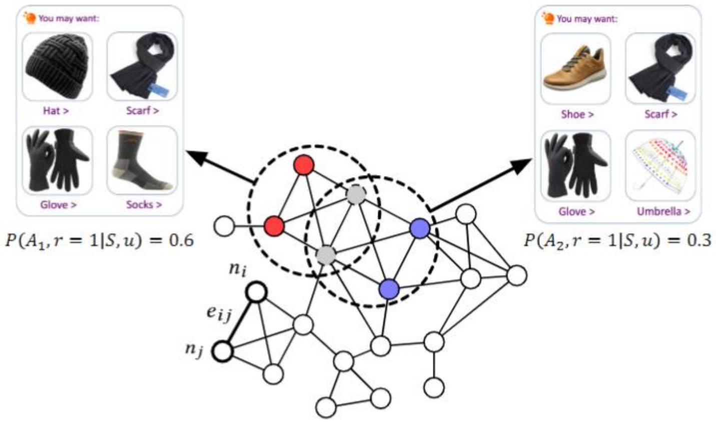
## Compare Top-K Recommendation problem and Exact-K Recommendation problem.

**Top-K Recommendation Problem**

- Top-K recommendation problem is a ranking optimization problem which assumes that “better” items should be put into top positions;
- Top-K recommendation problem focus on rank items and put better items into top position. which means that maximum the chance for user to click exact items.

**Exact-K Recommendation Problem**

- Exact-K recommendation problem is a (constrained) combinatorial optimization problem which tries to maximize the joint probability of the **set** of items in a card;
- Exact-K recommendation problem focus on combinatorial optimization, recommend a whole set of K items.




- **P**: The probability of this **card** to be clicked/satisfied
- $\boldsymbol{A}_{\mathbf{i}}$: Card
- **S**: Candidate items set
- **u**: User
- $\boldsymbol{n}_{\mathbf{i}}$: Node, donates an item
- $\boldsymbol{e}_{\mathbf{ij}}$: Constraint between nodes

**Illustration for the Exact-K Recommendation problem.**

In the situation of illustration, we can suppose that card A1 takes more chance to satisfy users than A2.

The Exact-K problem means finding a card with the highest P.

Construct undirected graph of items

↓ Encoder

Representations of input nodes in graph

↓ Decoder

Selected clique

**RLfD** $\xrightarrow{\text{Train}}$ **GAttN** ← Encoder / Decoder

**Structure for GAttN to deal with Exact-K Recommendation problem**

## Graph Attention Networks (GAttN): Encoder

Transform original representations of nodes in graph to embedding representations of these nodes, considering graph constructure information.

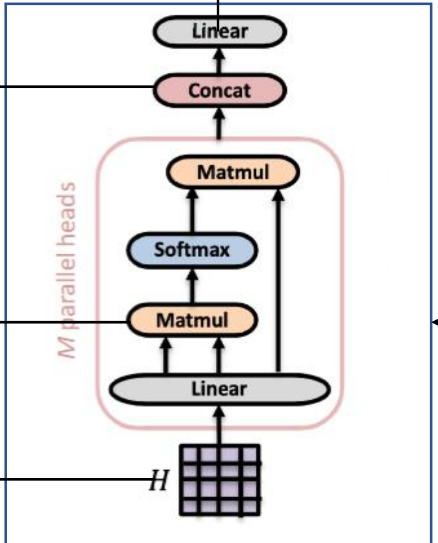
**Parameterized linear transformation**

- Produce the sub-layer

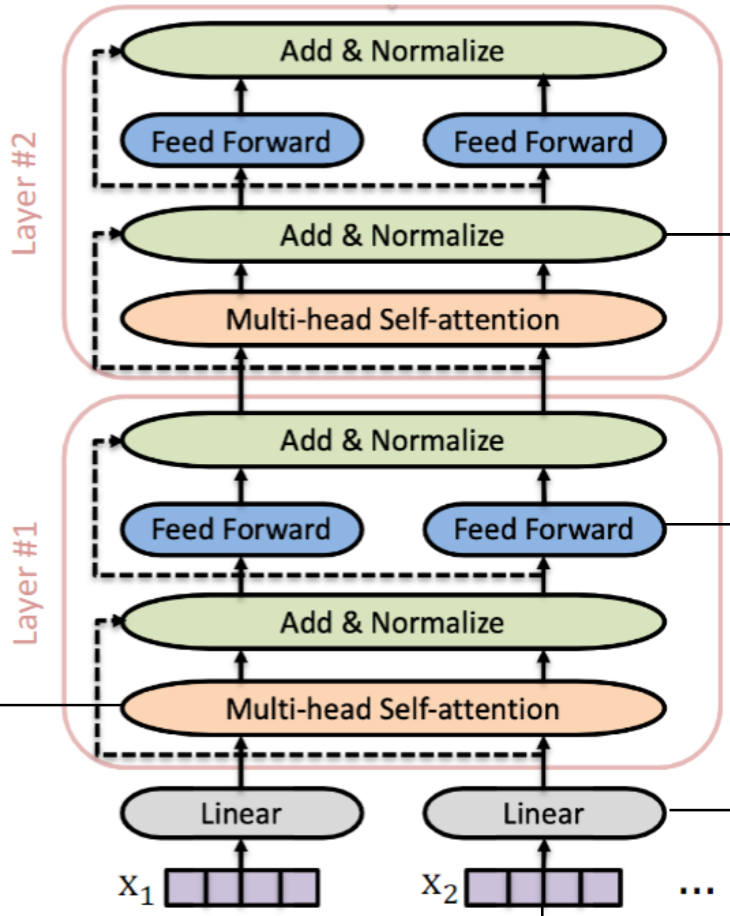
**Concat**: Concatenate results from each head

**Matul**: numpy.matmul()

A matrix of input d-dimensional embedding vectors




**Multi-head Self-attention**




**Encoder**

**Skip-connection** and **layer normalization**

Two linear transformations with a **ReLU** [28] activation between them.

**Node embedding** [27]

- A learned linear projection.

Feature vectors for nodes.

## **G**raph **A**ttention **N**etworks (**GAttN**): **Decoder**

Receives embedding representations of nodes in graph from encoder, elects clique of K nodes with attention mechanism.

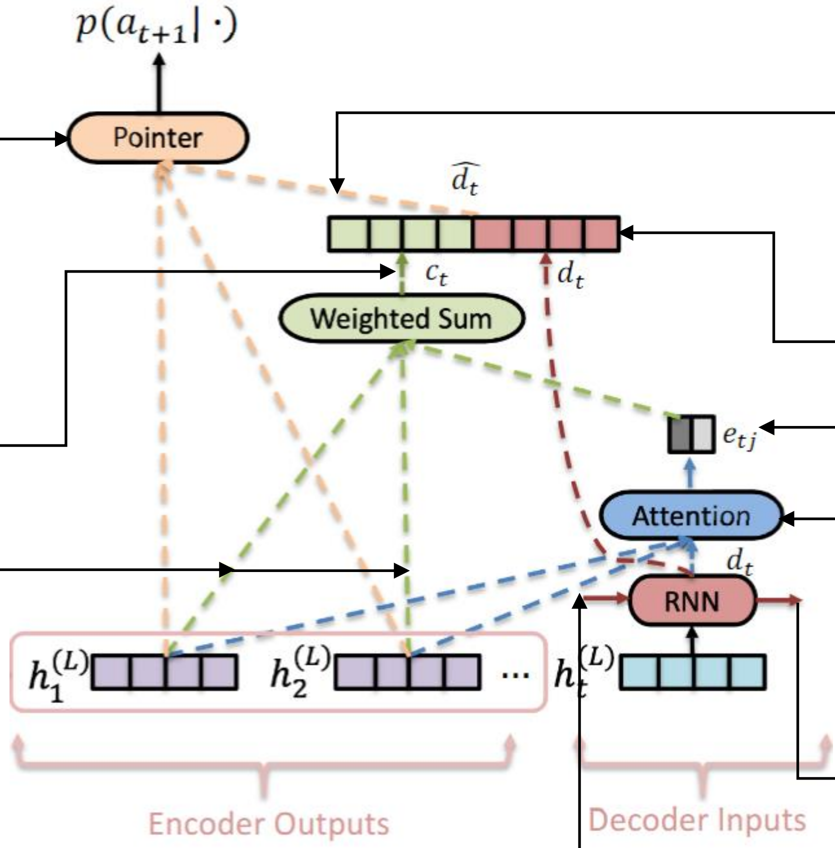
**Generate feasible clique from graph**

(Softmax and technique of **beam search** [30])

- Expand the search space and try more combinations of nodes in a clique (a.k.a items for a card) to get a most optimal solution

Softmax

Glimpses the whole encoder outputs [29]




**Decoder**

Mask nodes if disobey the clique constraint rule among the decoded subgraph.

**State vectors** generated by non-linear function, combines the previous state and previous output.

Representation of decoding

Attend to each node in the encoded graph and calculate the attention scores

Demonstrations Loss

Rewards Loss

Learn by Policy-sampling

Learn by Hill-climbing

User feedback data

- CTR (Click-Through-Rate)

For **RNN** here:

- Encoder can be seen as State
- Decoder can be seen as Policy

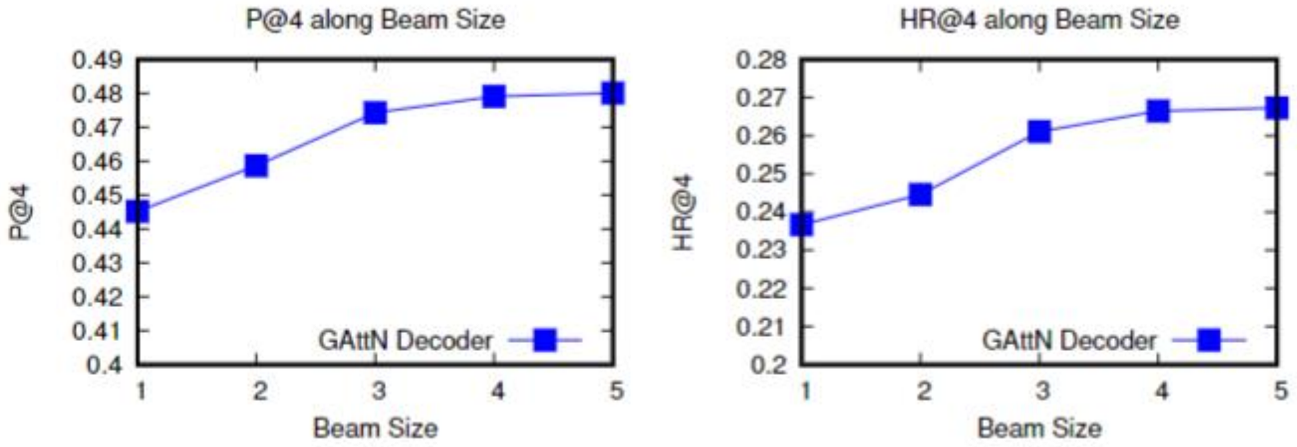
# Experiments

| 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% |

- **DeepRank**: applies DNNs
- **BPR**: optimized MF
- **Listwise-GRU**: listwise based on GRU
- **Listwise-MHSA**: listwise model based on Multi-head Self-attention
- **N**: number of items
- **K**: recommendation set size
- **P@K**: precision
- **HR@K**: hit ratio
- **Impv.**: improvement ratio

**Result:**

- It would be effective to apply MHSA method for encoding the candidate items;




**Performance of P@4 and HR@4 with different beam size**

**Result:**

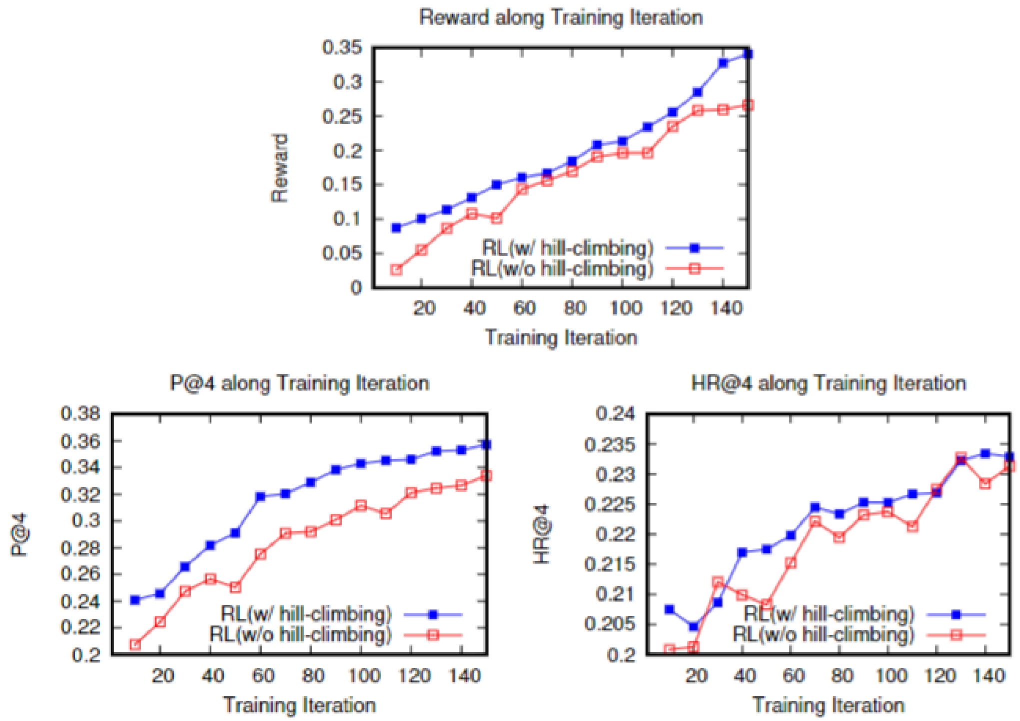
- Larger beam size can lead to better performances on both P@K and HR@K
- Beam size larger than 3 has minor improvement to P@K and HR@K

# Experiments

- **RL:** Learning from Rewards
- **SL:** Learning from Demonstrations
- **w/**: with
- **w/o**: without

| | 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** |

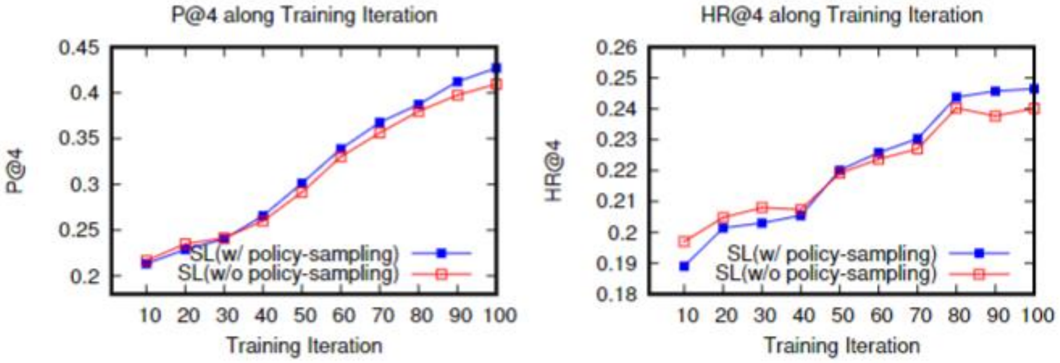
**1. Performance for different settings in RLfD**




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

**Result (1. 2.)**:

- Hill-climbing help the training be more stable and steady in improving the performance, and achieving a better solution;
- The designed define reward function is effective in problem.

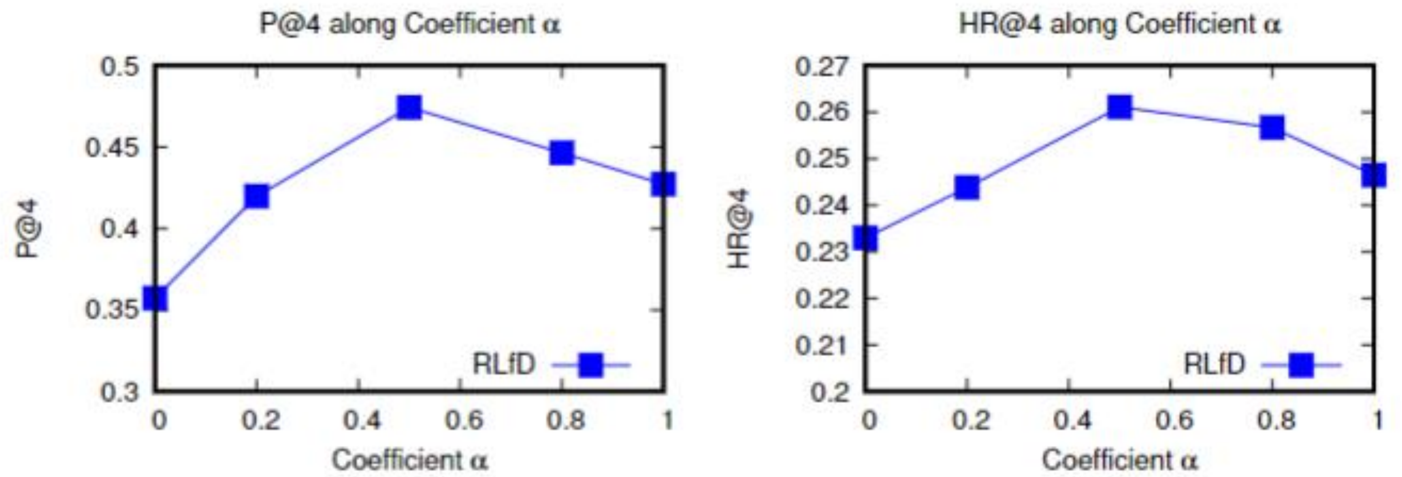



**3. Learning curves respect to P@4 and HR@4 for SLwith (w/) or without (w/o) policy-sampling**

**Result (3.)**:

- The first steps of training procedure the learned policy can be poor;
- With the training goes on, SL with policy-sampling will converge better for revising the inconsistency between training and inference of policy, finally achieve better performances.

## Experiments




**Performance of P@4 and HR@4 with different co-efficients α**

**α :** A hyper-parameter in combining the loss of *Learning from Demonstrations* and *Learning from Rewards*, ***which*** represents **trade-off** for applying **SL** and **RL** in training process.

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

Loss of *Learning from Demonstrations*

Loss of *Learning from Rewards*

**Result**:

- Properly combining SL and RL losses can result in the best solution;
- Only use SL, we will get a preliminary sub-optimal policy;
- Involving some degree of RL will achieve more optimal solutions.

# Hydra: A Personalized and Context-Aware Multi-Modal Transportation Recommendation System

Hao Liu[1], Yongxin Tong[2], Panpan Zhang[1], Xinjiang Lu[1], Jianguo Duan[1], Hui Xiong[1,3*]
[1]The Business Intelligence Lab, Baidu Research,
National Engineering Laboratory of Deep Learning Technology and Application, Beijing, China,
[2]SKLSDE Lab, Beihang University, Beijing, China, [3]Rutgers University
[1]{liuhao30, zhangpanpan04, luxinjiang, duanjianguo}@baidu.com,
[2]yxtong@buaa.edu.cn, [3]xionghui@gmail.com

***DRL4KDD '19***, August 4-8, 2019, Anchorage, AK, USA
https://doi.org/10.1145/3292500.3330660

A recommendation system **Hydra** was proposed to offer multi-modal transportation planning and is adaptive to various situational context.

- **Two-level framework** that integrates uni-modal and multi-modal routes as well as heterogeneous urban data is used in the intelligent multi-modal transportation recommendation;
- Latent representations of users, origin-destination (OD) pairs and transportation modes were learned by users' feedbacks;
- A **gradient boosting tree** based model was used to improve the recommendation result;
- Hydra supports real-time, large-scale route query and recommendation.

**Limitations of current transportation recommendation solutions**

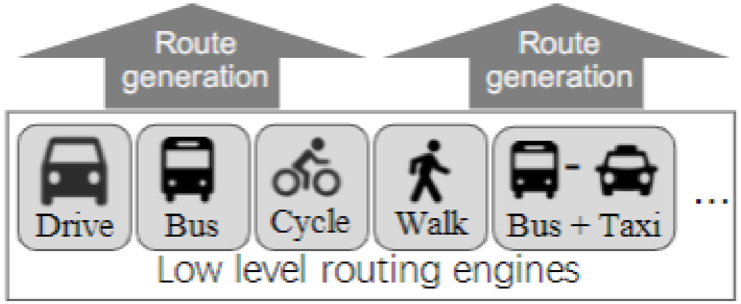
- Ignorance of situational context (e.g. it would be hard to call a taxi when a big concert lets out);
- Uni-modal transportation recommendation (e.g. a cost-effective transportation recommendation may be more attractive if the trip is not emergency).

→

**Hydra framework**

- Integrates route plans in different transportation modes and heterogeneous user data;
- Learns the latent representations of users, origin-destination (OD) pairs and transportation modes.

## Hydra Overview: Route generation, feature construction, and recommendation model




**Route generation**

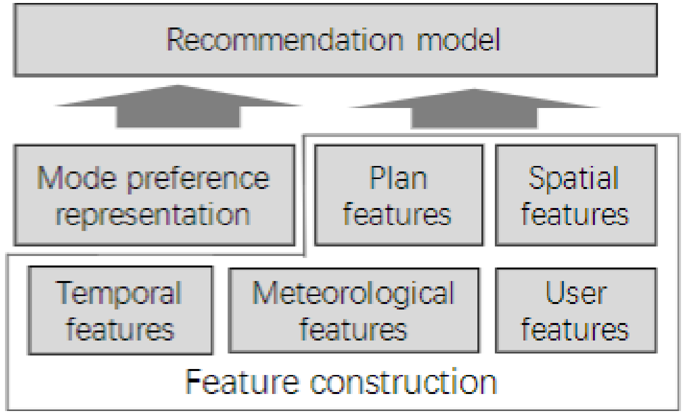
1. **Station binding process**: bind origin and destination locations to validate start and end points;
2. **Bidirectional shortest-path search** [31]:
   - Uni-modal transport mode: contraction hierarchy (CH) was used to reduce latency;
   - Multi-modal transport mode [32]: multi-modal transportation network was built to get the result;
3. **An internal rule based ranking model** is applied in each transport mode to filter routes.

**Recommendation model**

- **Gradient boosting tree model** was used in recommendation model, which is suitable for data mining with sparse and high dimensional features.

**Mode preference representation**

- Learn high order collaborative relationship among users, OD pairs, and transport modes.

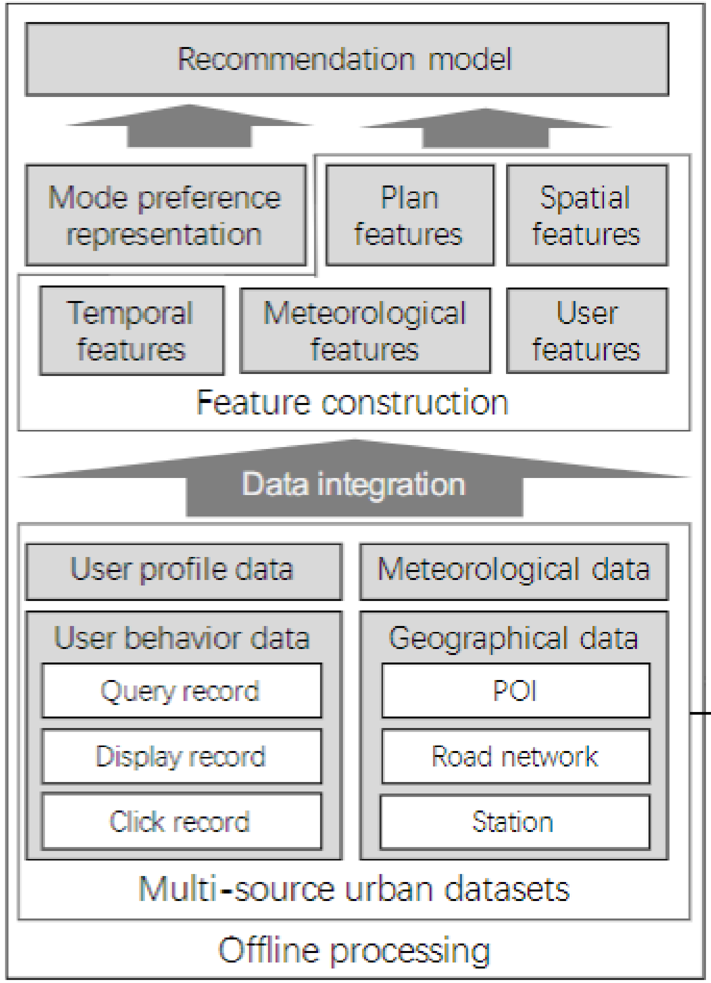



**Feature construction**

1. **Plan features**: cost of a plan are part of considerations; road network distance, route distance, price, transfer count, transfer model count would be extracted;
2. **Spatial features**: district, point-of-interest (POI) category and the relationship between trip distance and transport modes would be extracted;
3. **Temporal features**: Hour, Minute, Day of week, Day of month and Workday would be considered;
4. **Meteorological features**: Weather, Temperature, AQI, Wind speed and Wind direction would be considered;
5. **User features**: Demographic attribute, Social attribute and User historical mode distribution would be considered.

# Hydra Overview: data pipline

**XGBoost library is used to train this model**




**Extra datasets from log system**

↓

**Integrate dataset**

↓

**Construct features from integrated datasets**

↓

**Replace missing data with default data or average data**

↓

**Scale feature value to [0,1]**

↓

**One-hot encoding is used to each categorical feature**

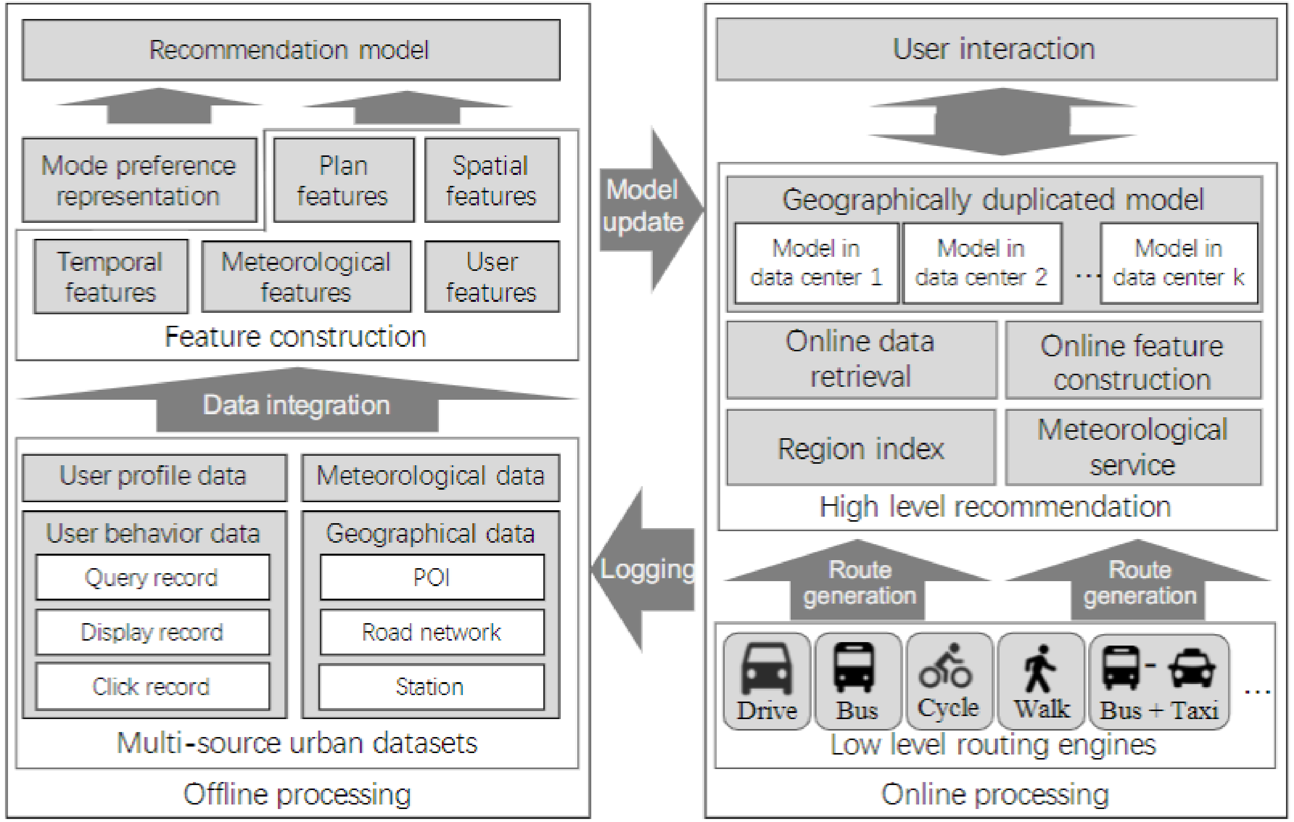
↓

**Features and labels are combined into a two dimensional array**

**Data Processing**

- **Behavior data**: update daily
- **Geographical data and other social data**: update monthly
- **Meteorological data**: update hourly

# Hydra Overview




Divide processing area into fine-grained grids based on coordinates with a unique grid id, allocate **regions** to the corresponding grids.

↓

**Retrieve** geographical information, meteorological data, user profile data in parallel and integrate them with raw route plans.

↓

Execute the online feature engineering process by leveraging the metadata generated in the **offline data pipeline**.

↓

Feed the processed feature vector into the model, sort each mode by model score and return the transport mode with the highest score to the use.

# Experiments

- **BEIJING, SHANGHAI:** two cities' datasets
- **NDCG:** NDCG metrics
- **PREC:** weighted precision metrics
- **REC:** recall metrics
- **F1:** F1 matrics
- **UHP**: based on fraction of user historical preference
- **ODHP**: based on the fraction of origin-destination (OD) historical preference
- **LR**: logistic regression model
- **RF**: Random Forest
- **LTR**: LambdaMart [33]
- **Trans2vec**: state-of-the-art transportation mode recommendation method [34]

| | Algorithm | NDCG | PREC | REC | F1 |
|---|---|---|---|---|---|
| BEIJING | UHP | 0.29 | 0.159 | 0.207 | 0.18 |
| | ODHP | 0.343 | **0.478** | 0.229 | 0.31 |
| | LR | 0.802 | 0.255 | 0.681 | 0.371 |
| | RF | 0.754 | 0.329 | 0.448 | 0.379 |
| | LTR | 0.798 | 0.258 | 0.673 | 0.373 |
| | Trans2vec | 0.462 | 0.26 | 0.282 | 0.271 |
| | Hydra | **0.815** | 0.271 | **0.72** | **0.396** |
| SHANGHAI | UHP | 0.288 | 0.162 | 0.188 | 0.174 |
| | ODHP | 0.367 | **0.454** | 0.253 | 0.325 |
| | LR | 0.789 | 0.262 | 0.652 | 0.374 |
| | RF | 0.747 | 0.336 | 0.423 | 0.37 |
| | LTR | 0.794 | 0.265 | 0.653 | 0.377 |
| | Trans2vec | 0.46 | 0.266 | 0.258 | 0.262 |
| | Hydra | **0.819** | 0.274 | **0.685** | **0.391** |

**Overall performance for 7 algorithms**

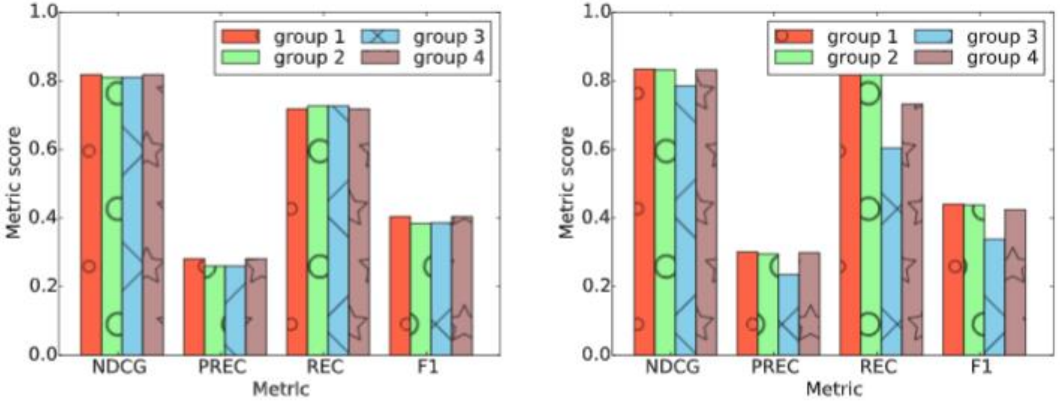
**Result**:

- Hydra achieves better performance than six baselines over all metrics except PREC, but Hydra achieves better balance between PREC and REC
- Situational context information and tailored feature engineering is curial for multi-modal transportation recommendation
- Large proportion of cold-start users will affect Trans2vec's performance

# Experiments

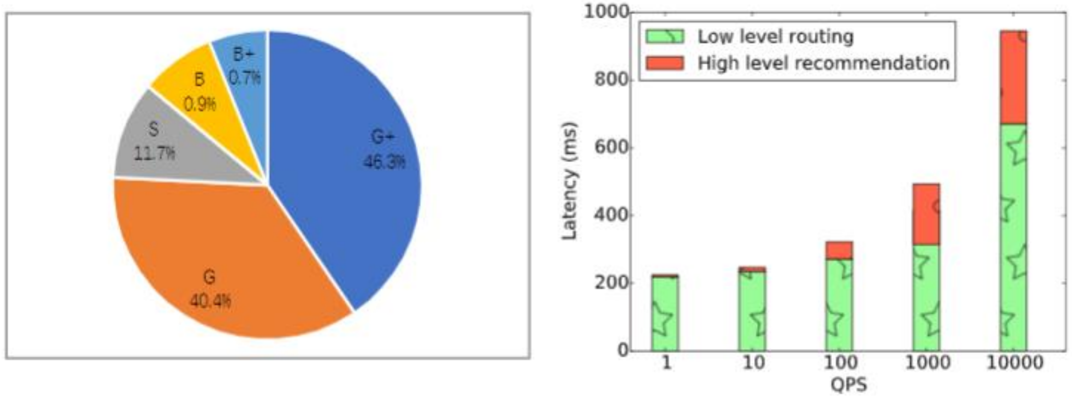
- **Group 1:** women and age lower than 35
- **Group 2:** men and age lower than 35
- **Group 3:** women and age older than 35
- **Group 4:** men and age older than 35

- **G+: Good plus**
- **G: Good**
- **S: same with previous model**
- **B+: Bad plus**
- **B: Bad**




(a) Group by users (b) Group by OD pairs

**(2). Robustness check on the BEIJING dataset**




(a) User satisfaction (b) Latency

**(3). Results of the online service**

| Rank | Feature name | Relative gain |
|---|---|---|
| 1 | Walk ETA | 1 |
| 2 | Bus-cycle ETA | 0.803 |
| 3 | Bus ETA | 0.577 |
| 4 | Taxi-bus ETA | 0.451 |
| 5 | User walk percentage | 0.295 |
| 6 | Consumption level | 0.213 |
| 7 | Origin station count | 0.162 |
| 8 | Primary POI category | 0.096 |
| 9 | Hour | 0.092 |
| 10 | Spherical distance | 0.051 |

**(1). Top-10 features ranked by information gain**

**Result**:

- **(1)** Travel time is the major consideration in the transport mode choice;
- **(1)** User attributes especially user social attributes also make significant contribution for prediction;
- **(1)** The spatial and temporal dependency influences the transport mode choice;
- **(2a, 2b)** Results are strongly stable on four metrics, except the third in OD group, which need future optimization;
- **(2b)** The variation from the OD prole perspective is more significant;
- **(3a)** Hydra provides better recommendations in terms of user experience;
- **(3b)** The low level routing is the major bottleneck and can be further optimized.

# MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation

Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, Sehee Chung
Knowledge AI Lab., NCSOFT Co., South Korea
{hoyeoplee,jinbae,swjang90,dakgalbi,seheechung}@ncsoft.com

***DRL4KDD ’19***, August 4-8, 2019, Anchorage, AK, USA
https://doi.org/10.1145/3292500.3330859

A meta-learning-based recommendation system **MeLU** was proposed to alleviate the cold-start problem that can estimate user preferences based on only a small number of items.

- **Meat-learning** can help the recommendation system adopt new task with a few examples rapidly;
- An **evidence candidate selection strategy** was provided to determine distinguishing items for customized preference estimation.

**Limitations of previous recommendation system**

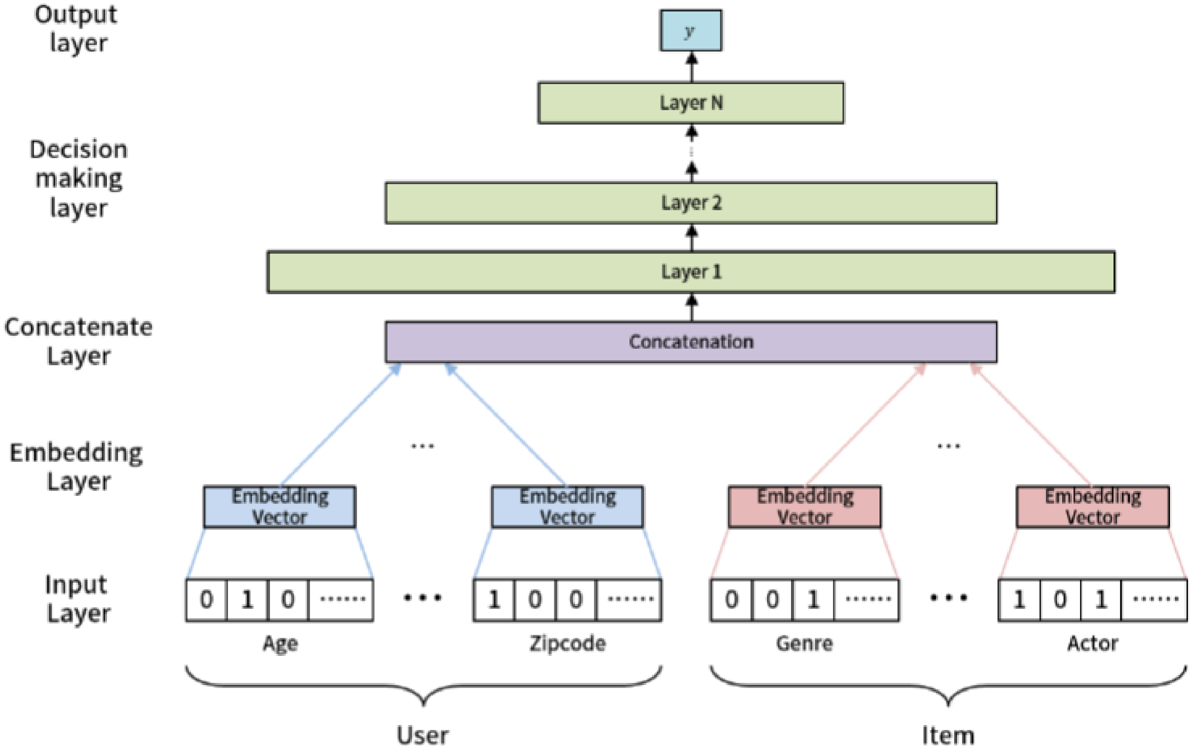
- The users who consumed a few items have poor recommendations;
- Inadequate evidence candidates are used to identify user preferences.

→

**MeLU**

- **Meta-learning** focuses on improving classification or regression performance by learning with only a small amount of training data;
- **Model-Agnostic Meta-Learning (MAML)** algorithm can allow estimation of customized preference directly based on an individual user's few item-consumption history.
- **Evidence candidate selection strategy** can substantially enhance the initial recommendation performance for new users by selecting distinguishing items for customized preference estimation.

## MeLU user preference estimator




**User preference estimator**

Estimates user preferences
(ratings, implicit feedback [35], or dwell times [36])

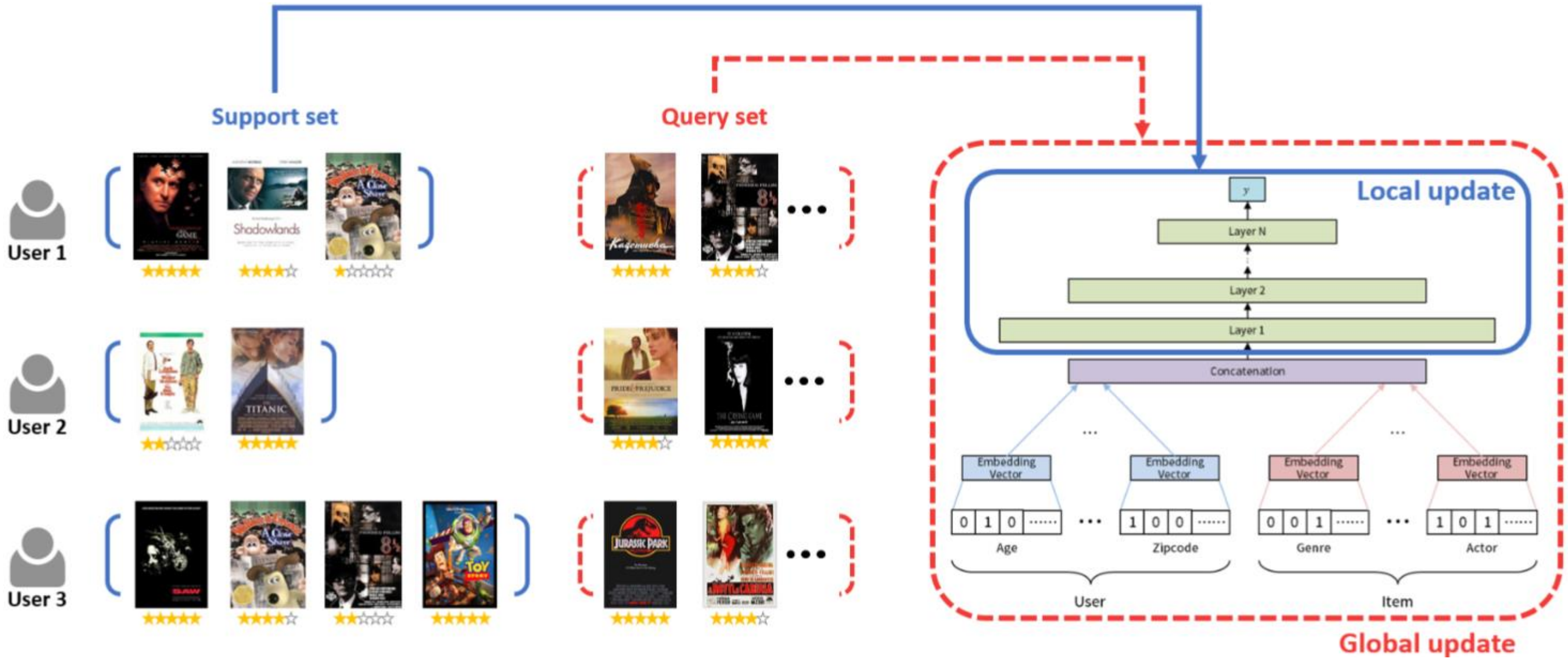
Model the decision-making process by means of a multilayered neural network (**N-layer fully-connected neural network**), ReLU [37] is also used here.

Embedding method is the same method with **Wide & Deep learning**

- Extract useful features to estimate user preferences from the contents,

**User content and item content**

# MeLU




User's item-consumption history and preferences

**Local update**: updates the parameters in the decision-making layers and the output layer backpropagating by loss function:

$$\mathcal{L}_i = \frac{1}{|H_i|} \sum_{j \in H_i} (y_{ij} - \hat{y}_{ij})^2$$

***i***: user number

***j***: item number

***Hi***: set of items consumed by user ***i*** (support set)

$y_{ij}$: user ***i***'s actual preference to item ***j***

$\hat{y}_{ij}$: user ***i***'s predicted preference to item ***j***

- **Local update** can be considered to be an iteration for personalization, which can be repeated several times;
  The reason why do not use support set to update embedding is to ensure the stability of the learning process;
- **Global update** aims to find the desirable parameters that achieve good recommendation performance after a few local updates for all users, which use the query set and the same loss function with local update.

MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation

# Experiments

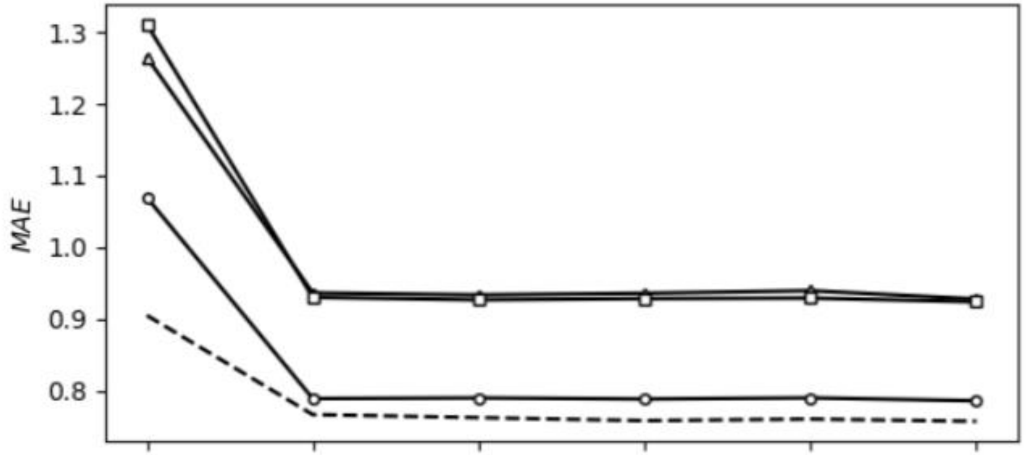
| Type | Method | MovieLens | | | Bookcrossing | | |
|---|---|---|---|---|---|---|---|
| | | *MAE* | $nDCG_1$ | $nDCG_3$ | *MAE* | $nDCG_1$ | $nDCG_3$ |
| Recommendation of existing items for existing users | PPR | **0.1820** | **0.9796** | **0.9831** | 3.8092 | 0.8242 | 0.8494 |
| | Wide & Deep | 0.9047 | 0.9090 | 0.9117 | 1.6206 | 0.9012 | 0.9172 |
| | MeLU-1 | 0.7661 | 0.8866 | 0.8904 | **0.7799** | **0.9563** | **0.9572** |
| | MeLU-5 | 0.7567 | 0.8870 | 0.8919 | 0.7955 | 0.9546 | 0.9552 |
| Recommendation of existing items for new users | PPR | 1.0748 | 0.8299 | 0.8468 | 3.8430 | 0.8201 | 0.8434 |
| | Wide & Deep | 1.0694 | 0.8559 | 0.8639 | 2.0457 | 0.8238 | 0.8515 |
| | MeLU-1 | 0.7884 | 0.8799 | 0.8810 | **1.8701** | **0.8265** | 0.8527 |
| | MeLU-5 | **0.7854** | **0.8803** | **0.8812** | 1.8767 | 0.8263 | **0.8532** |
| Recommendation of new items for existing users | PPR | 1.2441 | 0.7289 | 0.7632 | 3.6821 | 0.8115 | 0.8367 |
| | Wide & Deep | 1.2655 | 0.7420 | 0.7721 | 2.2648 | 0.8190 | 0.8437 |
| | MeLU-1 | 0.9361 | **0.7715** | 0.7990 | **2.1047** | **0.8202** | **0.8441** |
| | MeLU-5 | **0.9275** | 0.7697 | **0.8005** | 2.1236 | 0.8190 | 0.8440 |
| Recommendation of new items for new users | PPR | 1.2596 | 0.7292 | 0.7634 | 3.7046 | 0.8171 | 0.8381 |
| | Wide & Deep | 1.3114 | 0.7680 | 0.7874 | 2.3088 | 0.8160 | 0.8405 |
| | MeLU-1 | 0.9299 | **0.7760** | **0.8011** | **2.1475** | **0.8184** | 0.8410 |
| | MeLU-5 | **0.9235** | 0.7752 | 0.8008 | 2.1721 | **0.8184** | **0.8422** |

- **MovieLens:** dataset
- **Bookcrossing:** dataset
- **MAE:** mean absolute error
- **nDCG:** normalized discounted cumulative gain
- **PPR:** Pairwise Preference Regression, recommendation system
- **Wide & Deep:** recommendation system
- **MeLU-l:** MeLU model with l local updates

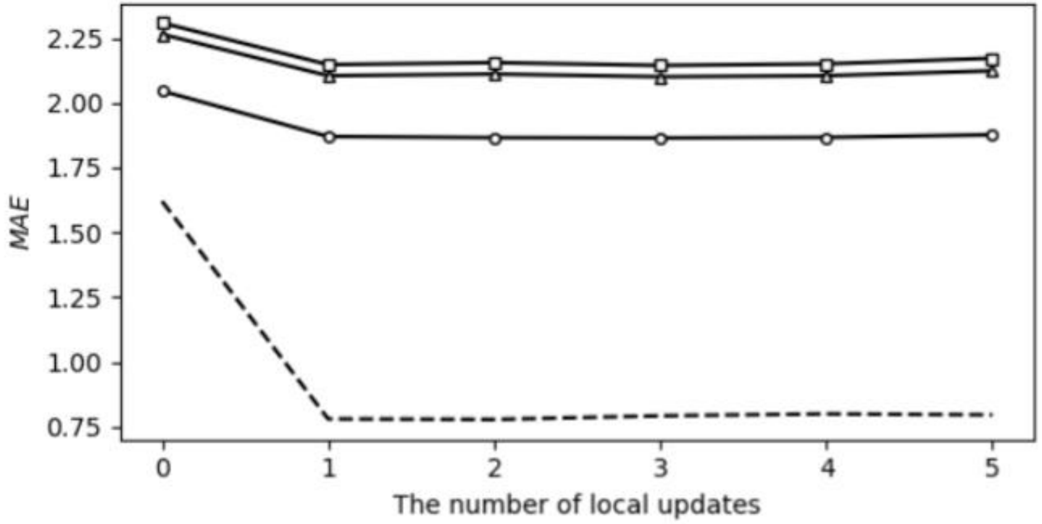
**Result**:

- MeLU outperforms two comparative methods in three types of cold-start scenarios in two datasets;
- The performance gaps of PPR between the non-cold-start scenario and cold-start scenarios come from the overfitting when sparsity is low;
- MeLU per-formed well when minimal information about users is available (as for Bookcrossing dataset)

# Experiments




**MovieLens datasets**

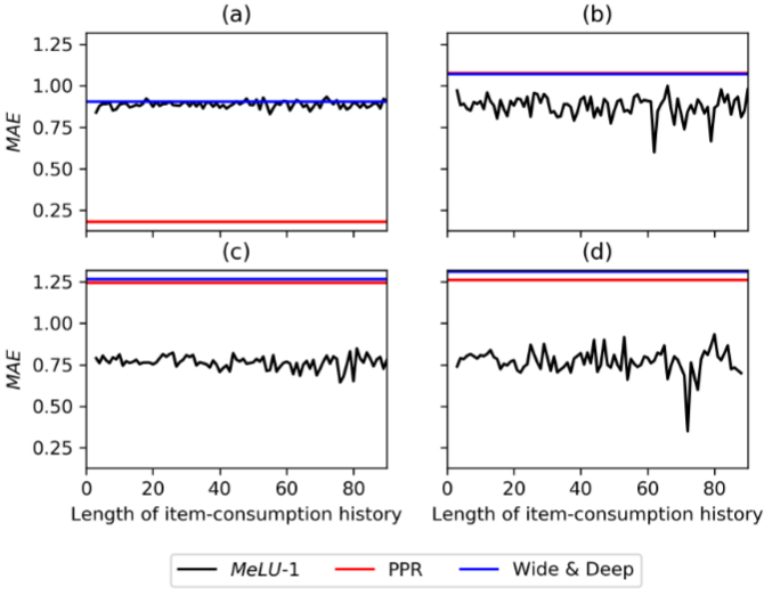



**Bookcrossing datasets**

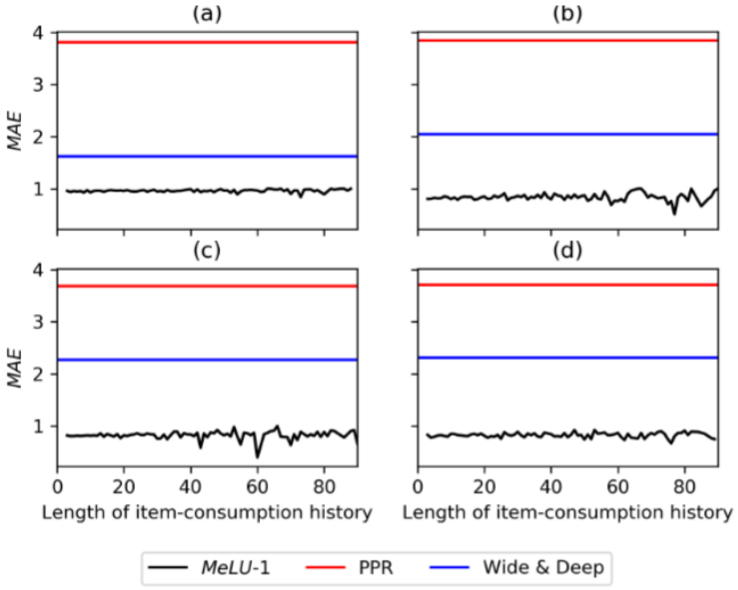

**Result**:

- MAE decreased rapidly after the single iteration for all dataset, while slight differences were observed for increasing the number of local updates;
- **MeLU** can adapt quickly to users because a single local update is sufficient. The rapid adaptation allows the proposed method to be applied to online recommendation based on user ratings.

# Experiments




**MovieLens datasets**




**Bookcrossing datasets**

- **(a)** Recommendation of existing items for existing users
- **(b)** Recommendation of existing items for new users
- **(c)** Recommendation of new items for existing users
- **(d)** Recommendation of new items for new users.

**Result**:

- MeLU shows good robust performance with regard to the length of the item-consumption history.
- Longer item-consumption history means smaller number of people, in this time, the result would be unstable because of insufficient sample size.

**Other results about user study**:

- For item recommendation, MeLU got higher average of rating, number of selected items and nDCG$_1$, which means MuLU strategy provides reliable evidence candidates and can quickly identify individual preferences of new users for the items.

# References

[1] Chong Chen, Min Zhang, Yi qun Liu, and Shao ping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In Proceedings of the 2018World Wide Web Conference. International World Wide Web Conferences Steering Committee, 1583–1592.

[2] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. ACM, 425–434.

[3] Xian gnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 355–364.

[4] Wen peng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. Transactions of the Association for Computational Linguistics4 (12 2015).

[5] Sung yong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction. In Proceedings of the Eleventh ACM Conference on Recommender Systems. ACM, 297–305.

[6] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 conference one mpirical methods in natural language processing. 1532–1543.

[7] Xian gnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 355–364.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J. Mach. Learn. Res.15, 1 (Jan. 2014), 1929–1958.

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization.arXiv preprint arXiv:1412.6980(2014).

[10] Li bing Wu, Cong Quan, Chen liang Li, Qian Wang, Bo long Zheng, and Xiang yang Luo. 2019. A Context-Aware User-Item Representation Learning for Item Recommendation. ACM Trans. Inf. Syst.37, 2 (Jan. 2019), 1–29.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing a tari with deeper inforcement learning. arXiv preprint arXiv:1312.5602(2013).

[12] Long-Ji Lin. 1993.Reinforcement learning for robots using neural networks. Technical Report. Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

[13] Andrew W Moore and Christopher G Atkeson. 1993. Prioritized sweeping: Reinforcement learning with less data and less time. Machine learning13, 1(1993), 103–130.

[14] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015.Gated feedback recurrent neural networks. In ICML. 2067–2075.

[15] Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. Machine learning3, 1 (1988), 9–44.

[16] Richard S Sutton and Andrew G Barto. 2018.Reinforcement learning: An intro-duction. MIT press

[17] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, PietroLio, and Yoshua Bengio. 2017. Graph attention networks. arXiv preprintarXiv:1710.109031, 2 (2017).

[18] Ling Yin Wei, Yu Zheng, and Wen Chih Peng. 2012. Constructing popular routes from uncertain trajectories. In SIGKDD. 195–203.

[19] Zai ben Chen, Heng Tao Shen, and Xiaofang Zhou. 2011. Discovering popular rroutes from trajectories. In ICDE. 900–911

# References


[20] Ge Cui, Jun Luo, and Xin Wang. 2018. Personalized travel route recommendation using collaborative filtering based on GPS trajectories.IJED11, 3 (2018), 284–307.

[21] Qiang Liu, Shu Wu, Liang Wang, and Tie niu Tan. 2016. Predicting the next location: a recurrent model with spatial and temporal contexts. In AAAI. 194–200

[22] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fan chao Meng, Ang Guo, and Depeng Jin. 2018. DeepMove: Predicting Human Mobility with Attentional Recurrent Networks. In WWW. 1459–1468.

[23] Richard S Sutton and Andrew G Barto. 2018.Reinforcement learning: An intro-duction. MIT press.

[24] E. W. Dijkstra. 1959. A note on two problems in connection with graphs. Nu-merische Math1 (1959), 269–271.

[25] Kaiming He, Xiangyu Zhang, Shao qing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In CVPR.

[26] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. In arXiv.

[27] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015. Order matters: Sequence to sequence for sets. In arXiv

[28] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In CVPR

[29] Andrew V Goldberg and Chris Harrelson. 2005. Computing the shortest path: A search meets graph theory. In Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms. 156–165.

[30] Julian Dibbelt et al.2016. Engineering Algorithms for Route Planning in Multi-modal Transportation Networks. Transportation(2016).

[31] Chris J.C. Burges. 2010.From Rank Net to Lambda Rank to Lambda MART: An Overview. Technical Report.

[32] Hao Liu, Ting Li, Renjun Hu, Yanjie Fu, Jingjing Gu, and Hui Xiong. 2019. Joint Representation Learning for Multi-Modal Transportation Recommendation. In AAAI

[33] Diane Kelly and Jaime Teevan. 2003. Implicit feedback for inferring user preference: a bibliography. In SIGIR Forum, Vol. 37. 18–28.

[34] Xing Yi, Liangjie Hong, Erheng Zhong, Nan than Nan Liu, and Suju Rajan. 2014.Beyond clicks: dwell time for personalization. In RecSys. 113–120.

[35] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In ICML. 807–814