Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks

Srijan Kumar, Xikun Zhang, Jure Leskovec

組員:

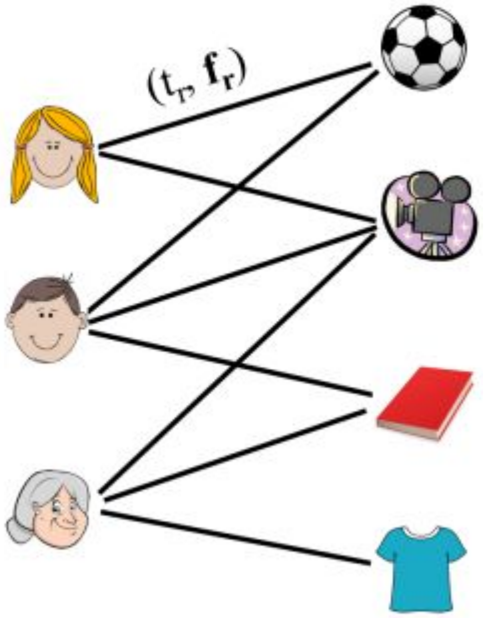
M11015072 王樸
M11015084 魏向晨
M11015094 李彥霖
M11015109 李冠霆
B10701015 黃鈺淇
B10715012 梁俊彥

Outline

1. Introduction
2. Related Work
3. JODIE: Joint Dynamic User-Item Embedding Model
4. Experiments
5. Conclusion

Introduction

- Users interact sequentially with items in many domains.
  - e-commerce (e.g., a customer purchasing an item)
  - education (a student enrolling in a MOOC course)
  - social and collaborative platforms (a user posting in a group in Reddit)
- The same user may interact with different items over a period of time and these interactions change over time.



Introduction

- Representation learning
  - Representation learning, or learning low-dimensional embeddings of entities
  - represent the evolution of users' and items' properties.
- Four fundamental challenges:
  - How to accurately predict the embedding trajectories of users/items as time progresses.
  - It is essential to consider stationary properties that do not change over time and time evolving properties.
  - Methods are required that can recommend items in near-constant time.
  - Methods are needed that can be trained with batches of data to generate embedding trajectories.

Introduction

JODIE: Joint Dynamic User-Item Embedding Model

- Learns to generate embedding trajectories of all users and items from temporal interactions.
- The embeddings of the user and item are updated when a user takes an action and a projection operator predicts the future embedding trajectory of the user.

Related Work

- Deep recurrent recommender models
  - Recent methods, such as Time-LSTM and LatentCross learn how to incorporate features into the embeddings
- Dynamic co-evolution models
  - Methods that jointly learn representations of users and items have recently been developed using point-process modeling and RNN-based modeling, such as DeepCoevolve
- Temporal network embedding models
  - Several models have recently been developed that generate embeddings for the nodes (users and items) in temporal networks, such as CTDNE

JODIE: Joint Dynamic User-Item Embedding Model

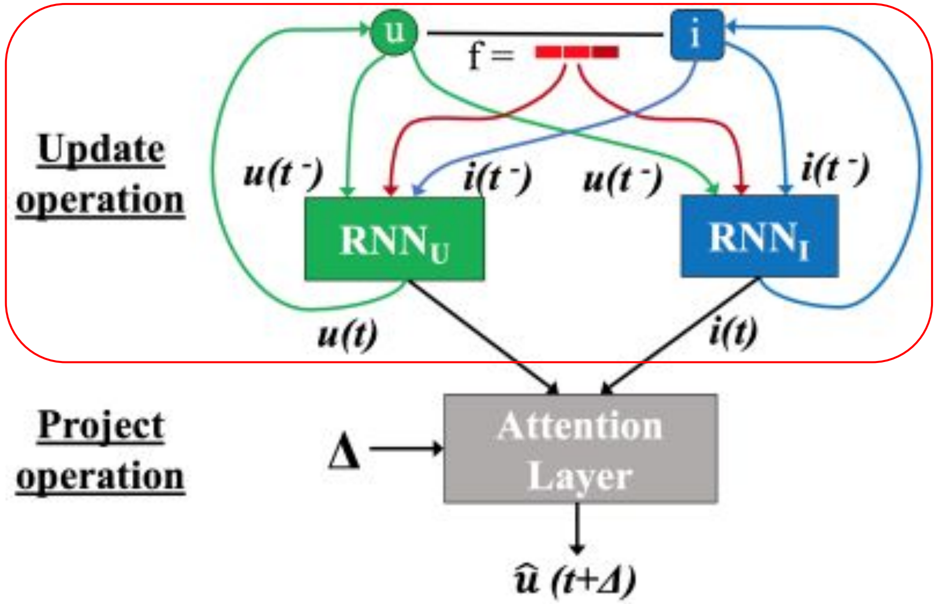
- A method to learn embedding trajectories of users and items from an ordered sequence of temporal user-item interactions $S = (u, i, t, f)$

- Type of Embeddings
  - Static Embeddings : Used to express stationary properties such as the long-term interest of users
  - Dynamic Embeddings : Embeddings change over time to model their time-varying behavior and properties

JODIE

Embedding update operation

- Dynamic embedding of an item reflects the item's current state and leading to more meaningful dynamic user embeddings and easier training



$$u(t) = \sigma(W_1^u u(t^-) + W_2^u i(t^-) + W_3^u f + W_4^u \boxed{\Delta_u})$$

$$i(t) = \boxed{\sigma}(W_1^i i(t^-) + W_2^i u(t^-) + W_3^i f + W_4^i \boxed{\Delta_i})$$

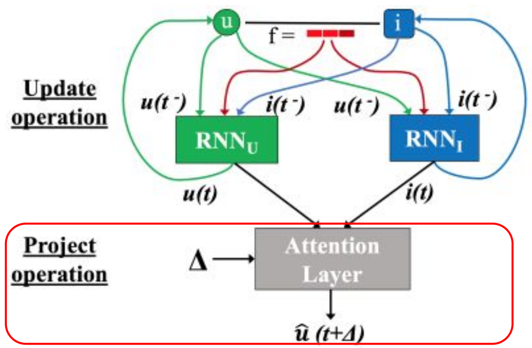
time since previous interaction

non-linearity sigmoid function

Symbol	Meaning
$u(t)$ and $i(t)$	Dynamic embedding of user u and item i at time t
$u(t^-)$ and $i(t^-)$	Dynamic embedding of user u and item i before time t
$\bar{u}$ and $\bar{i}$	Static embedding of user u and item i
$\hat{u}(t)$	Projected embedding of user u at time t
$\tilde{j}(t)$	Predicted item j embedding

JODIE

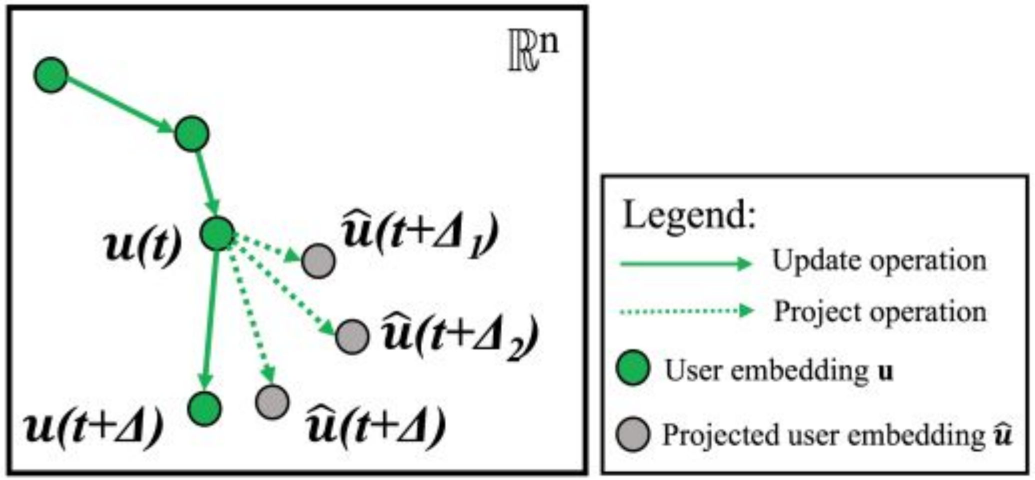
Embedding projection operation



- Predicts the future embedding trajectory of the user by projecting the embedding of the user at a future time
- Two inputs are required for the projection operation: u’s embedding at time t and the elapsed time Δ
- Projected Embedding :

$$\hat{\boldsymbol{u}}(t+\Delta) = (1 + \boxed{\boldsymbol{w}}) * \boldsymbol{u}(t)$$

time-context vector



JODIE

Training to predict next item embedding

- Training JODIE to make the prediction of item which user will interact with using user’s projected embedding
- JODIE directly outputs an item embedding vector instead of an interaction probability between user and item

$$\tilde{j}(t+\Delta) = W_1\widehat{\boldsymbol{u}}(t+\Delta) + W_2\overline{\boldsymbol{u}} + W_3\boldsymbol{i}(t+\Delta^{-}) + W_4\overline{\boldsymbol{i}} + \boxed{B}$$

bias vector

- JODIE is trained to minimize the L2 distance between the predicted item embedding and the ground truth item’s embedding at every interaction

$$Loss = \sum_{(u,j,t,f)\in S} ||\tilde{j}(t) - [\bar{j}, j(t^{-})]||_2 + \lambda_U ||\boldsymbol{u}(t) - \boldsymbol{u}(t^{-})||_2 + \lambda_I ||\boldsymbol{j}(t) - \boldsymbol{j}(t^{-})||_2$$

JODIE

t-Batch: Training data batching

- Parallelizing the training of JODIE by using a training data batching algorithm

- Two requirements for creating the training batches
  - All interactions in each batch should be processed in parallel
  - Processing the batches in increasing order of their index should maintain the temporal ordering of the interactions

- t-Batch creates each batch by selecting independent edge sets of the interaction network in two steps :
  - Select step : a new batch is created by selecting the maximal edge set
  - Reduce step : the selected edges are removed from the network

Experiment

- Future interaction prediction
  - mean reciprocal rank (MRR) and recall@10
- User state change prediction
  - area under the curve metric (AUC)
- Runtime experiment
- Robustness to the proportion of training data
- Embedding size

Experiment

Datasets

- Reddit
- Wikipedia
- LastFM
- MOOC course activity

Data	Users	Items	Interactions	State Changes	Action Repetition
Reddit	10,000	984	672,447	366	79%
Wikipedia	8,227	1,000	157,474	217	61%
LastFM	980	1,000	1,293,103	-	8.6%
MOOC	7,047	97	411,749	4,066	-

Experiment

Baselines

- Deep recurrent recommender models
  - RRN, LatentCross, Time-LSTM, standard LSTM
- Dynamic co-evolution models
  - DeepCoevolve
- Temporal network embedding models
  - CTDNE

Experiment

Future interaction prediction

Which item will user *u* interact with at time *t* ?

- Using Interaction Data:
  - 80% train
  - 10% validation
  - 10% test
- Measurement:
  - MRR
  - Recall@10

Method	Reddit		Wikipedia		LastFM		Minimum % improvement of *JODIE* over method	
	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10
LSTM [52]	0.355	0.551	0.329	0.455	0.062	0.119	104.5%	54.6%
Time-LSTM [52]	0.387	0.573	0.247	0.342	0.068	0.137	87.6%	48.7%
RRN [45]	0.603	0.747	0.522	0.617	0.089	0.182	20.4%	14.1%
LatentCross [8]	0.421	0.588	0.424	0.481	0.148	0.227	31.8%	35.2%
CTDNE [33]	0.165	0.257	0.035	0.056	0.01	0.01	340.0%	231.5%
DeepCoevolve [11]	0.171	0.275	0.515	0.563	0.019	0.039	44.8%	46.0%
JODIE (proposed)	**0.726**	**0.852**	**0.746**	**0.822**	**0.195**	**0.307**	-	-

Experiment

User state change prediction

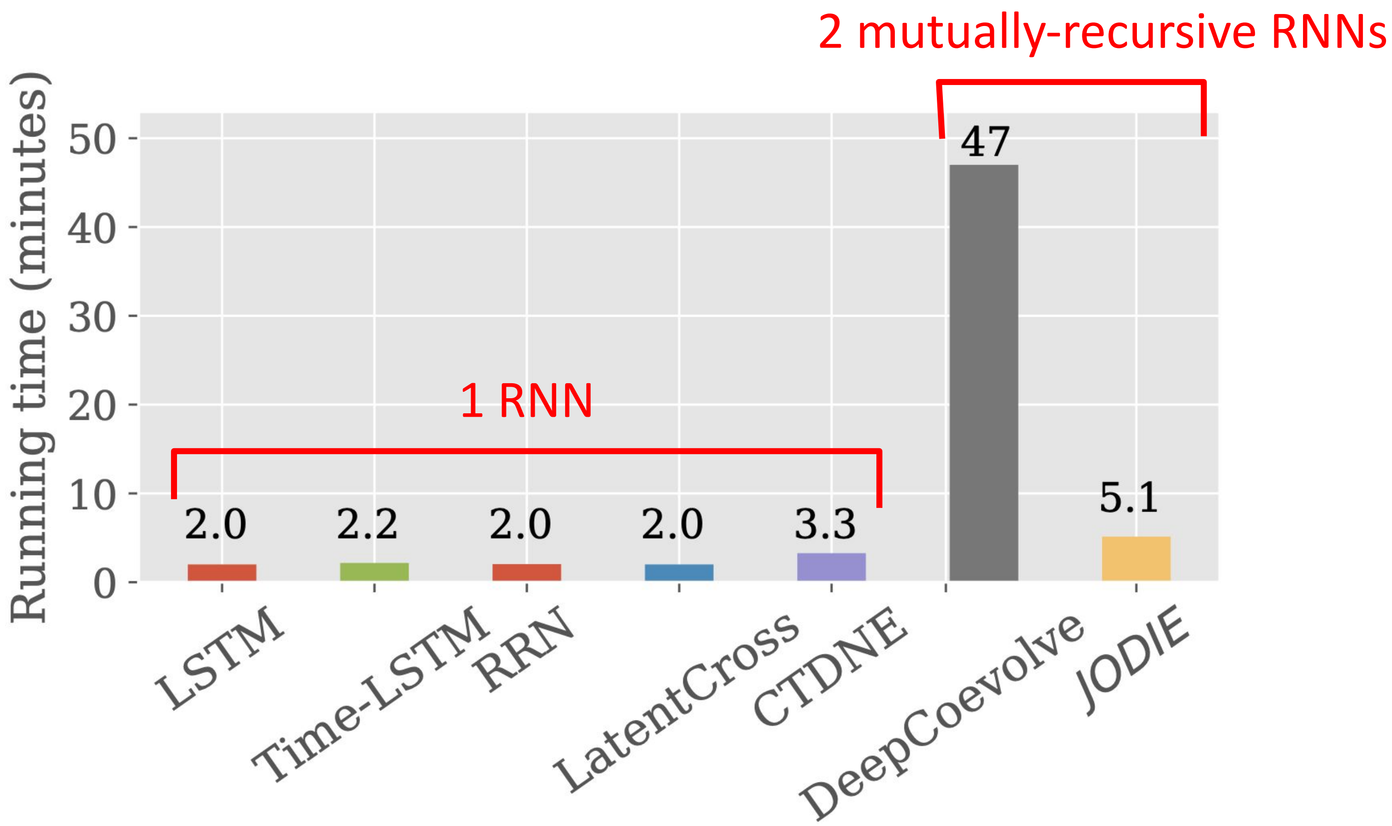
To predict if an interaction will lead to a state change in user.

- Using Interaction Data:
  - 60% train
  - 20% validation
  - 20% test
- Measurement:
  - AUC
- Baseline:
  - Logistic regression classifier

Method	Reddit	Wikipedia	MOOC	Mean improvement of *JODIE*
LSTM	0.523	0.575	0.686	23.08%
Time-LSTM	0.556	0.671	0.711	12.63%
RRN	0.586	0.804	0.558	13.69%
LatentCross	0.574	0.628	0.686	15.62%
DeepCoevolve	0.577	0.663	0.671	13.94%
JODIE	**0.599**	**0.831**	**0.756**	-

Experiment

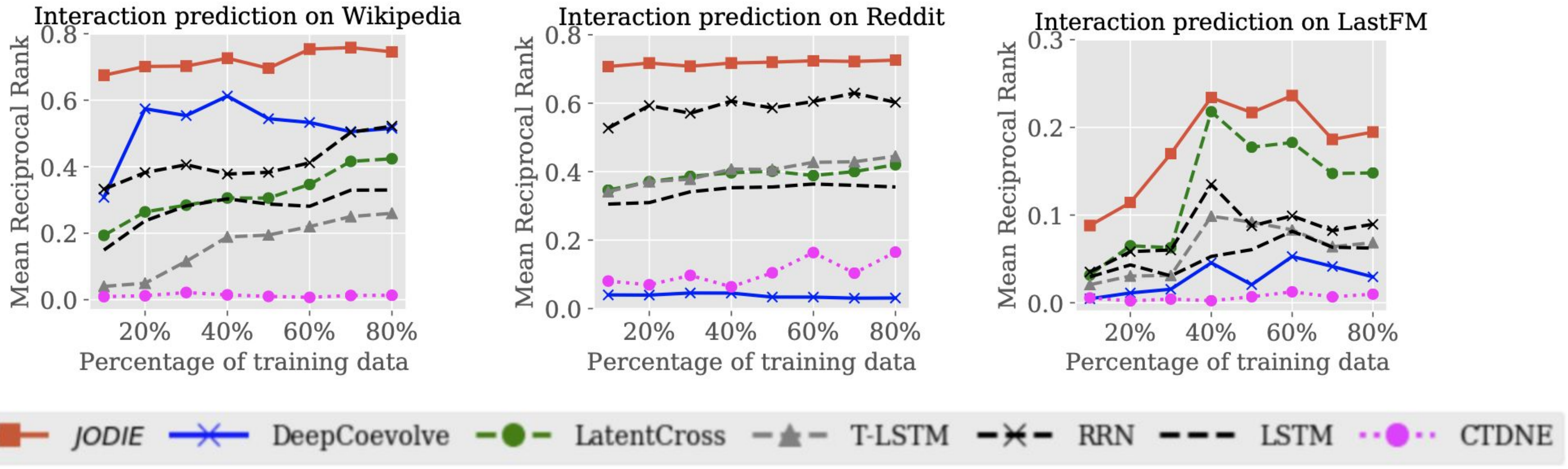
Runtime experiment



Experiment

Robustness to the proportion of training data

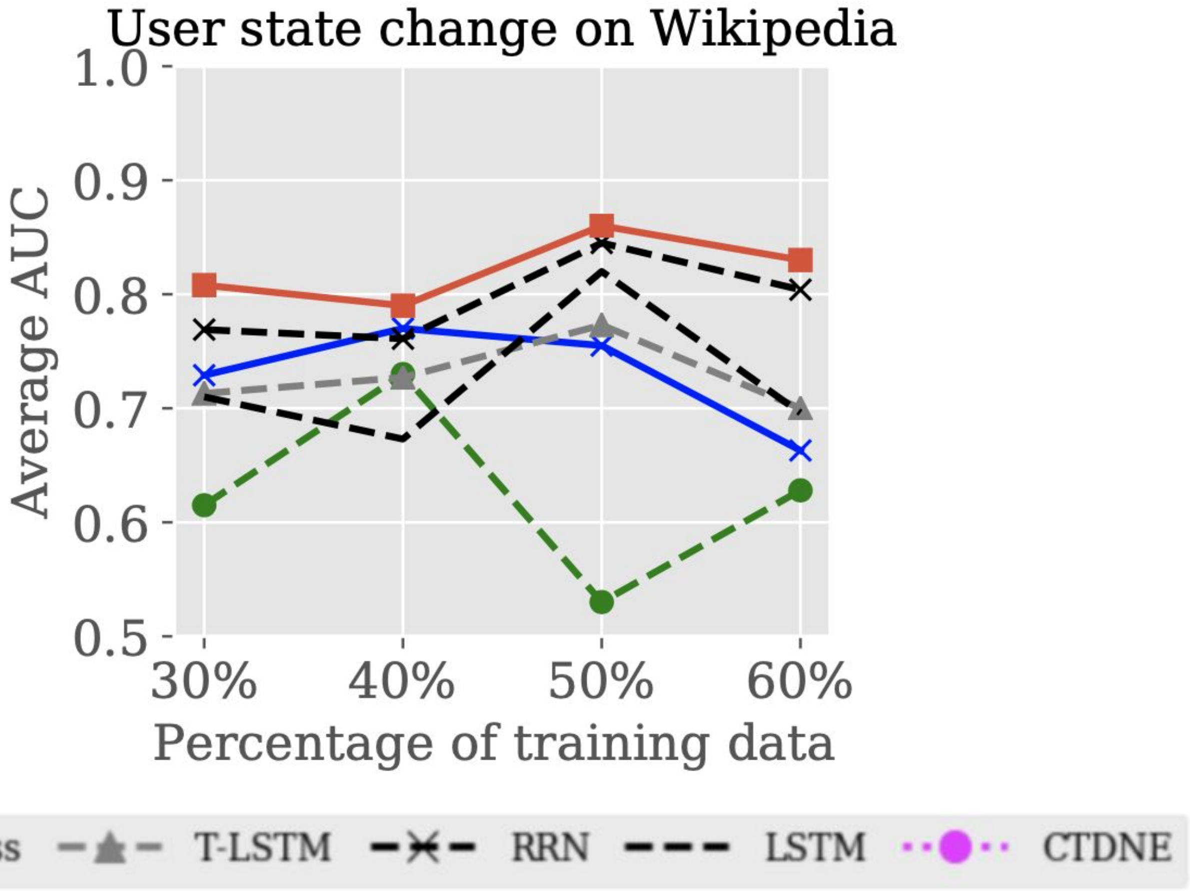
- Future interaction prediction
  - Mean reciprocal rank



Experiment

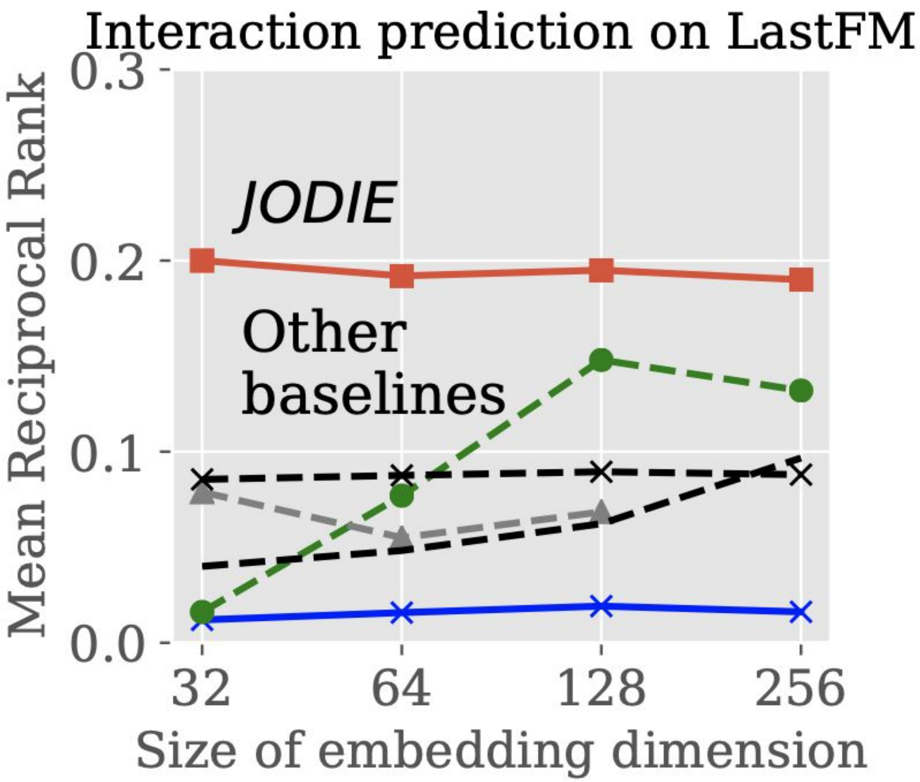
Robustness to the proportion of training data

- User state change prediction
  - Average AUC metrics



Experiment

Embedding size



- Dynamic embedding dimension
  - From 32 to 256
- JODIE uses both the static and the dynamic embedding for prediction

Conclusion

- JODIE gives better prediction performance of future user-item interactions and change in user state.
- Future Works:
  - Learn trajectories for groups of users or items to reduce the number of parameters
  - Characterizing the trajectories to cluster similar entities
  - Design new items based on missing predicted items that many users are likely to interact with

實作報告

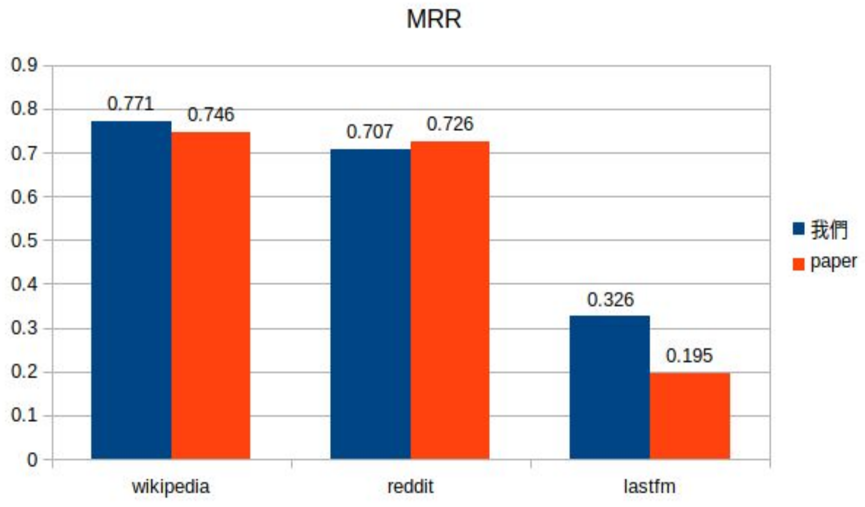
目錄

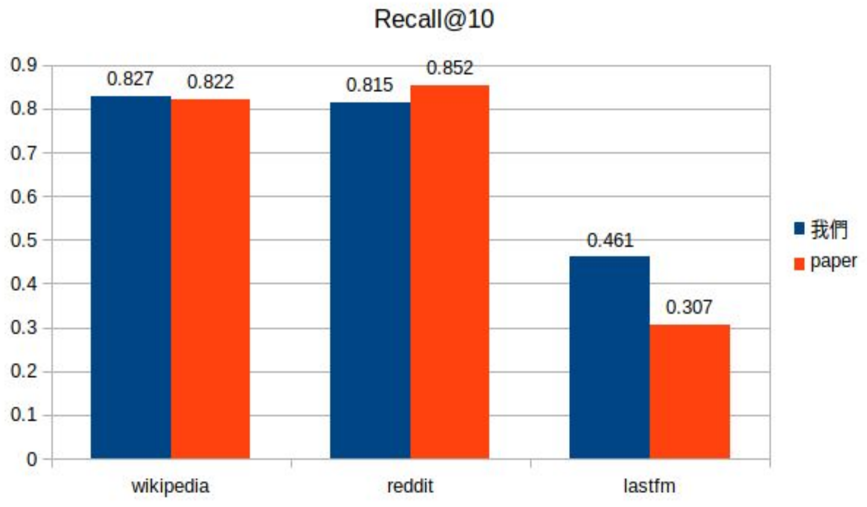
- 與論文結果比較及分析
- 我們的改動
- 改動結果比較及分析

與論文結果比較及分析

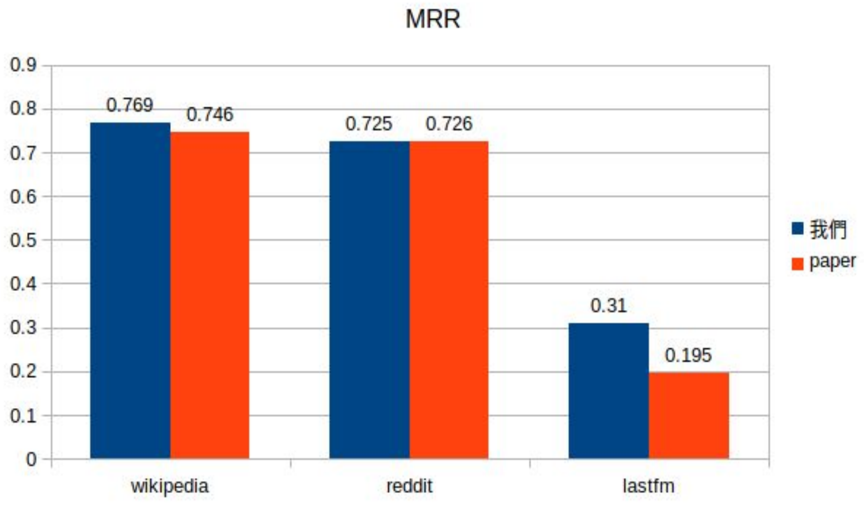
interaction

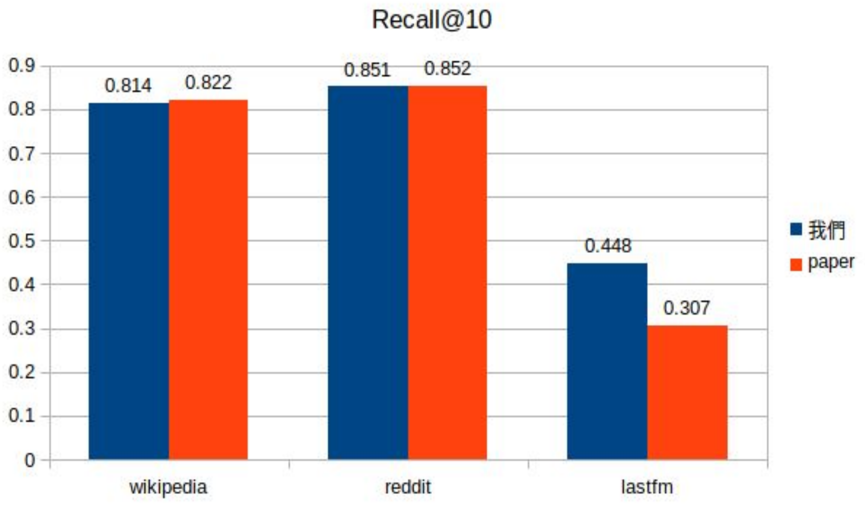
Validation





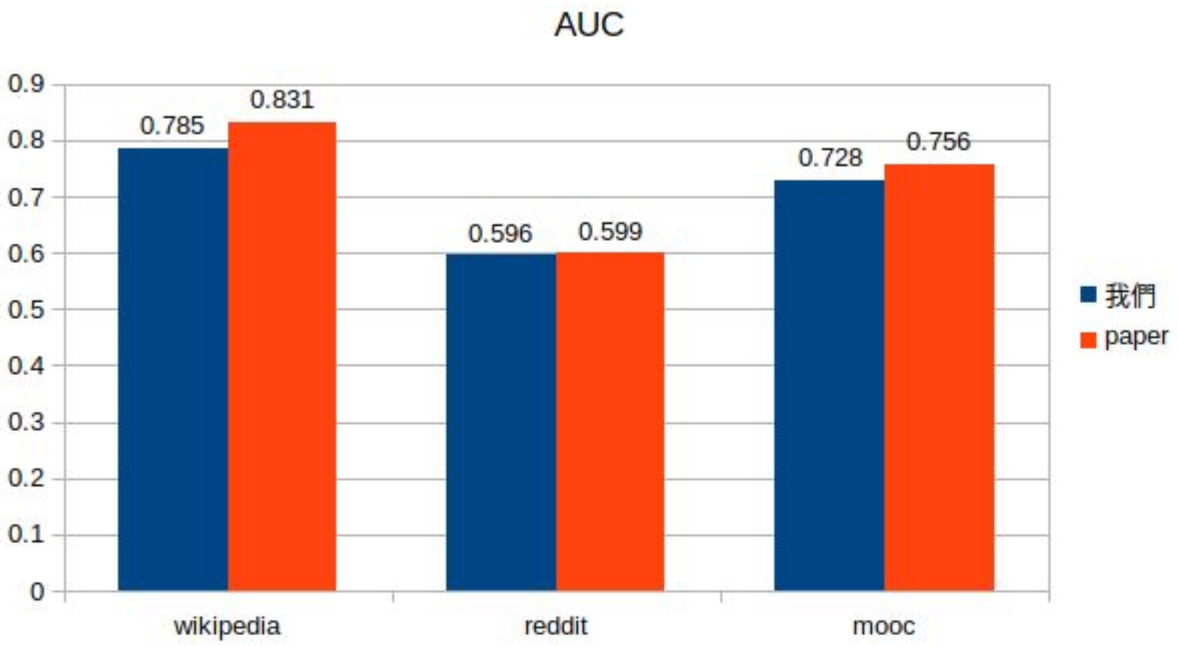
Test



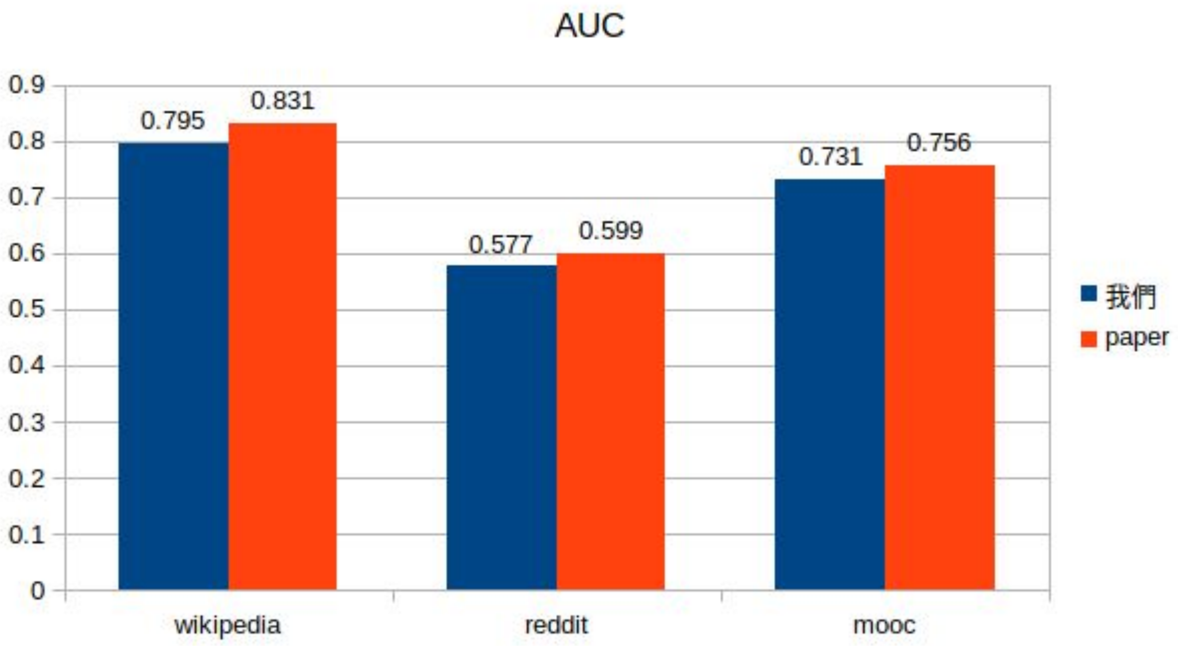


user state

Validation



Test



我們的改動

改動一：將RNNCELL改GRUCELL

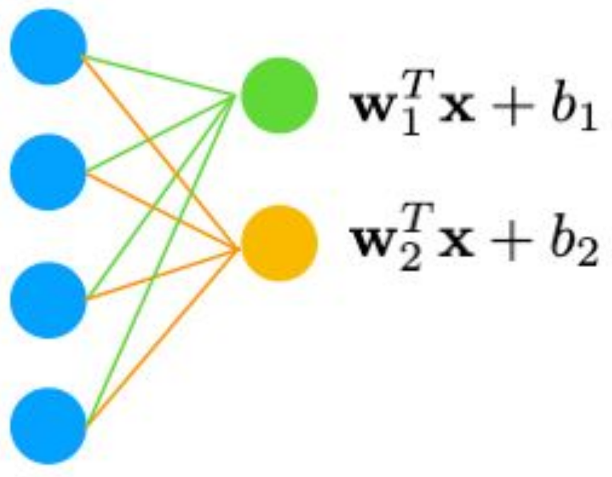
- GRU比RNN記得更久之前的資訊
- 可以解決梯度消失問題

```python
self.item_rnn = nn.RNNCell(rnn_input_size_users, self.embedding_dim)
self.user_rnn = nn.RNNCell(rnn_input_size_items, self.embedding_dim)
```

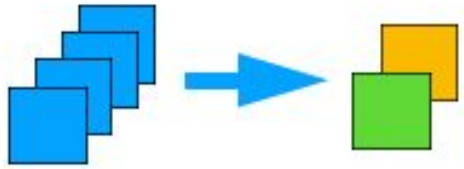
↓

```python
self.item_gru = nn.GRUCell(gru_input_size_users, self.embedding_dim)
self.user_gru = nn.GRUCell(gru_input_size_items, self.embedding_dim)
```

改動二：卷積層取代全連接層



Fully connected layer



Or, we can concatenate the inputs into 1x1 images with 4 channels and then use 2 kernels (remember, each kernel then also has 4 channels)

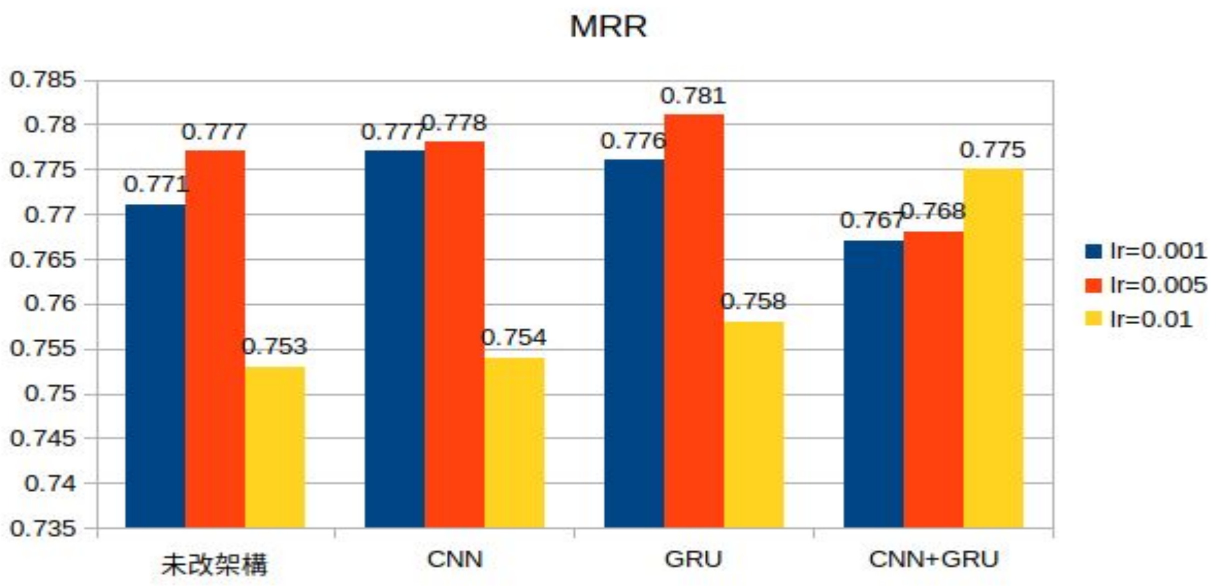
```python
self.linear_layer1 = nn.Linear(self.embedding_dim, 50)
self.linear_layer2 = nn.Linear(50, 2)
self.prediction_layer = nn.Linear(self.user_static_embedding_size + self.item_static_embedding_size + self.embedding_dim * 2, self.item_static_embedding_size + self.embedding_dim)
```

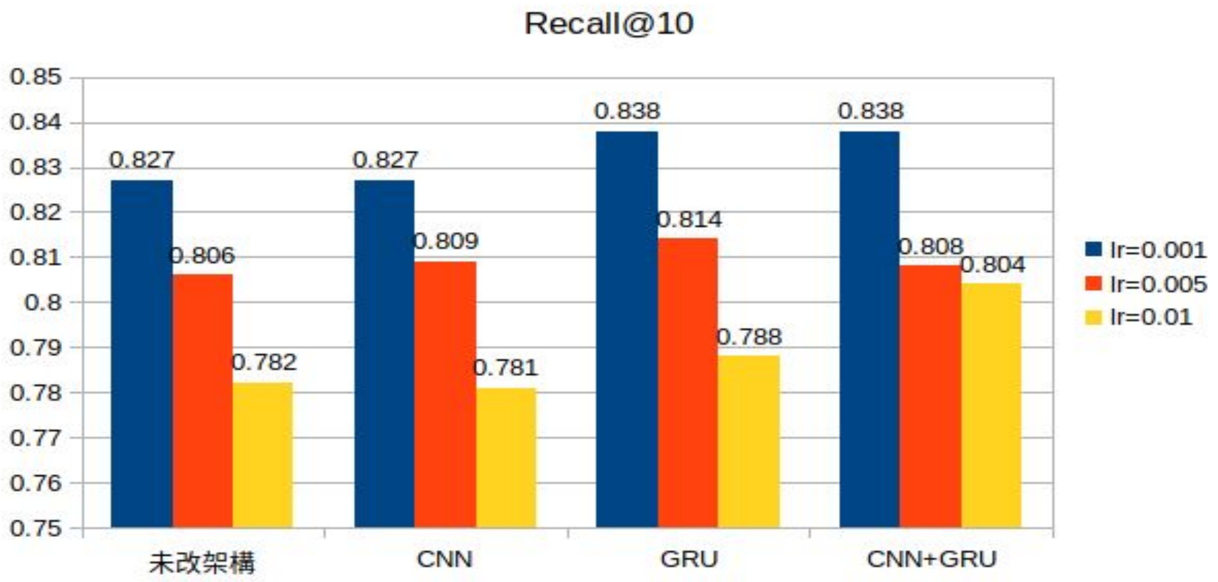
↓

```python
self.conv_layer1 = torch.nn.Conv2d(in_channels=self.embedding_dim,  out_channels=50,kernel_size=(1, 1))
self.conv_layer2 = torch.nn.Conv2d(in_channels=50,out_channels=2,kernel_size=(1, 1))
self.prediction_layer  = torch.nn.Conv2d(in_channels=self.user_static_embedding_size + self.item_static_embedding_size + self.embedding_dim * 2, out_channels=self.item_static_embedding_size + self.embedding_dim,kernel_size=(1, 1))
```

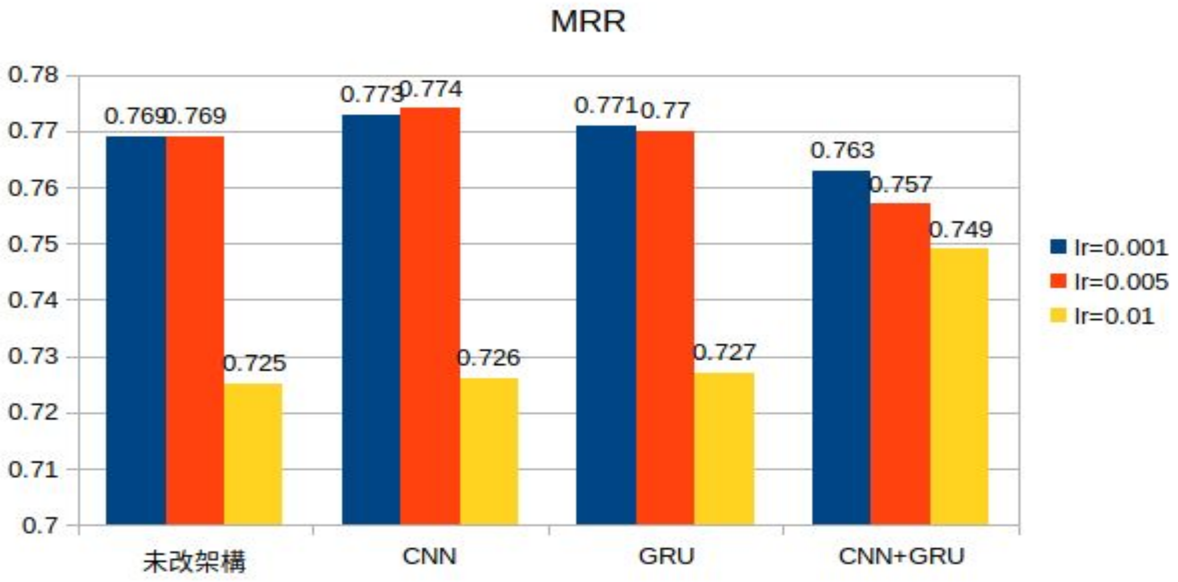
改動結果比較及分析

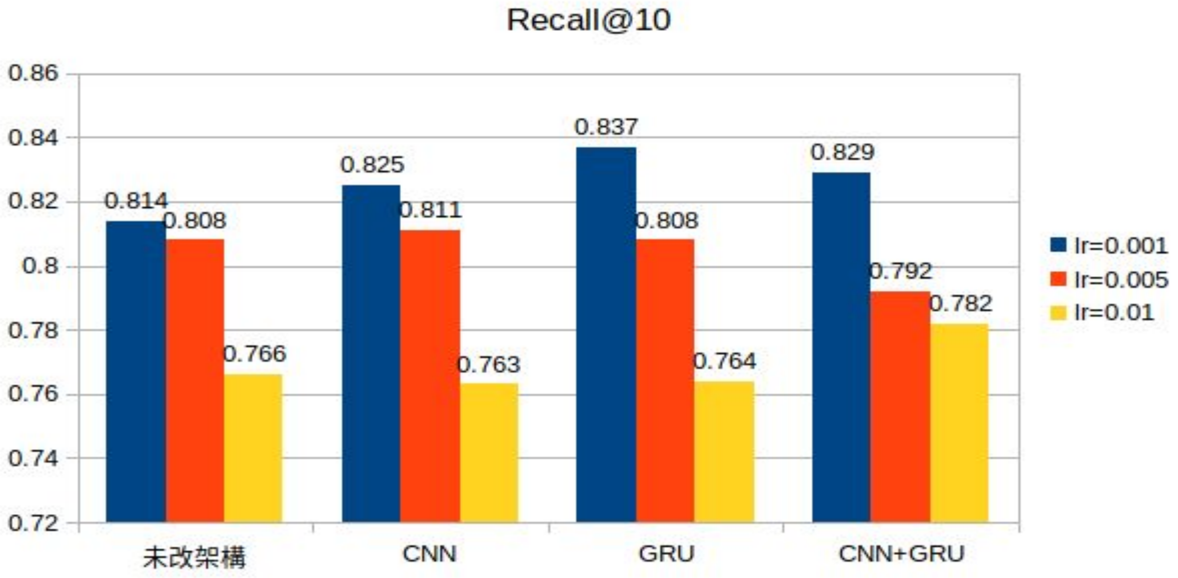
interaction(validation)



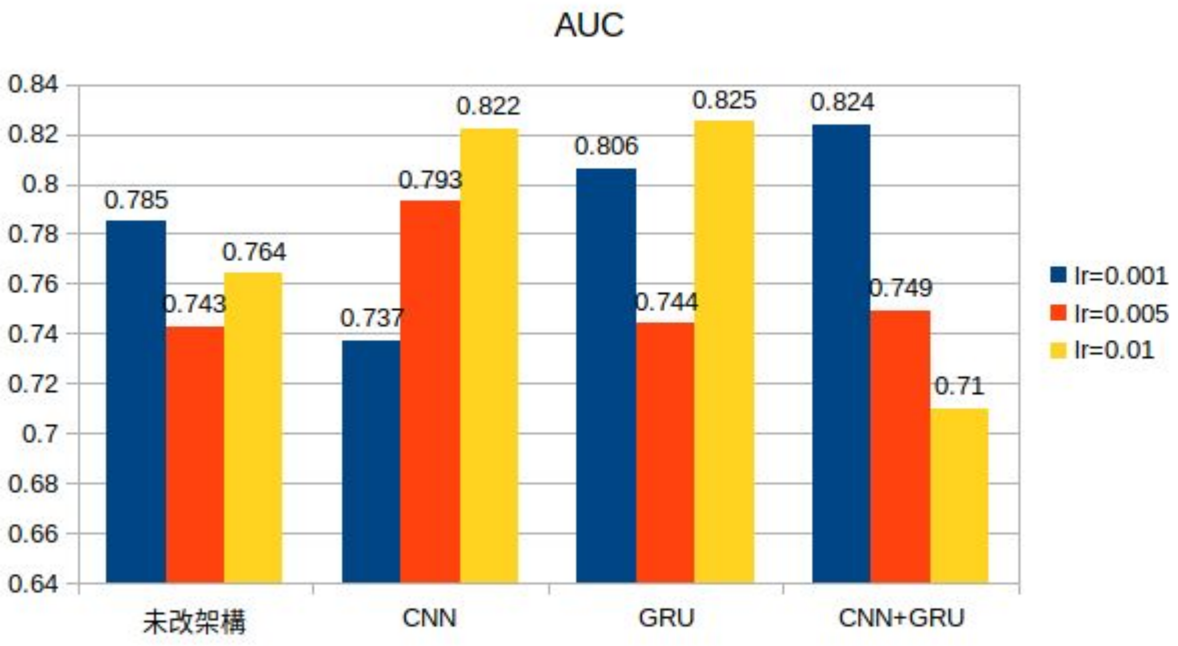


interaction(test)

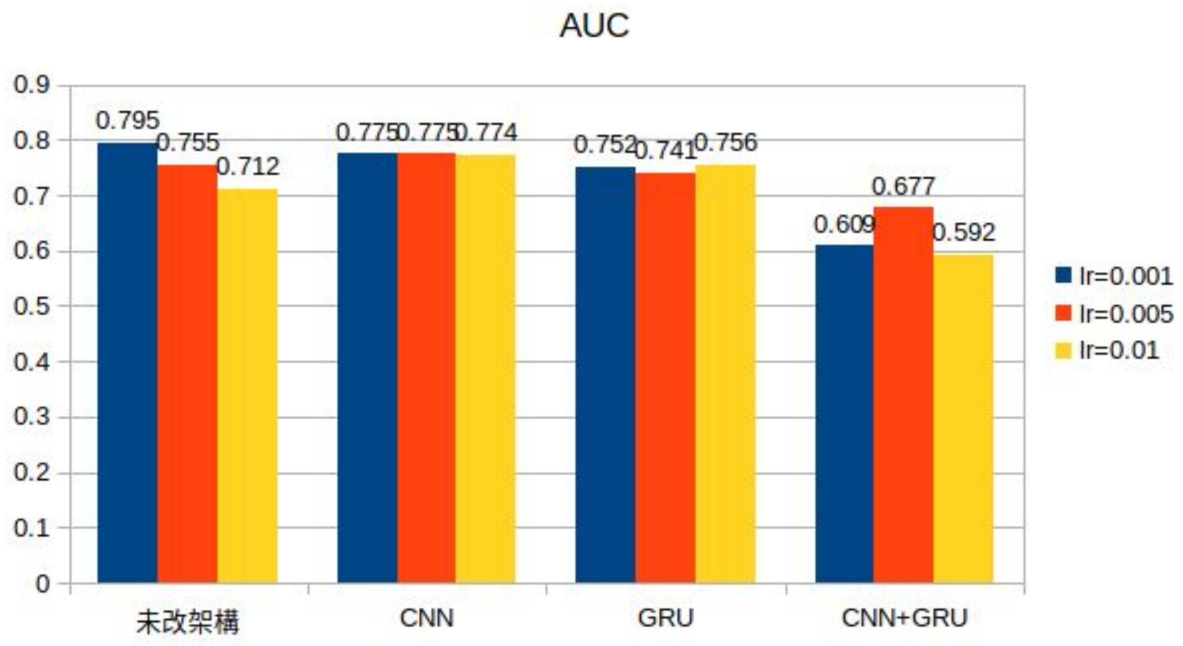




user state(validation)



user state(test)



Q & A