Hierarchical Temporal Convolutional Networks for Dynamic Recommender Systems

발표자: 이동규

at LINE Plus Corporation

June 17, 2019

Table of Contents

Table of Contents

Intro.

추천 시스템이란?

- User가 관심을 가질만한 Item을 추천

Intro.

추천 시스템이란?

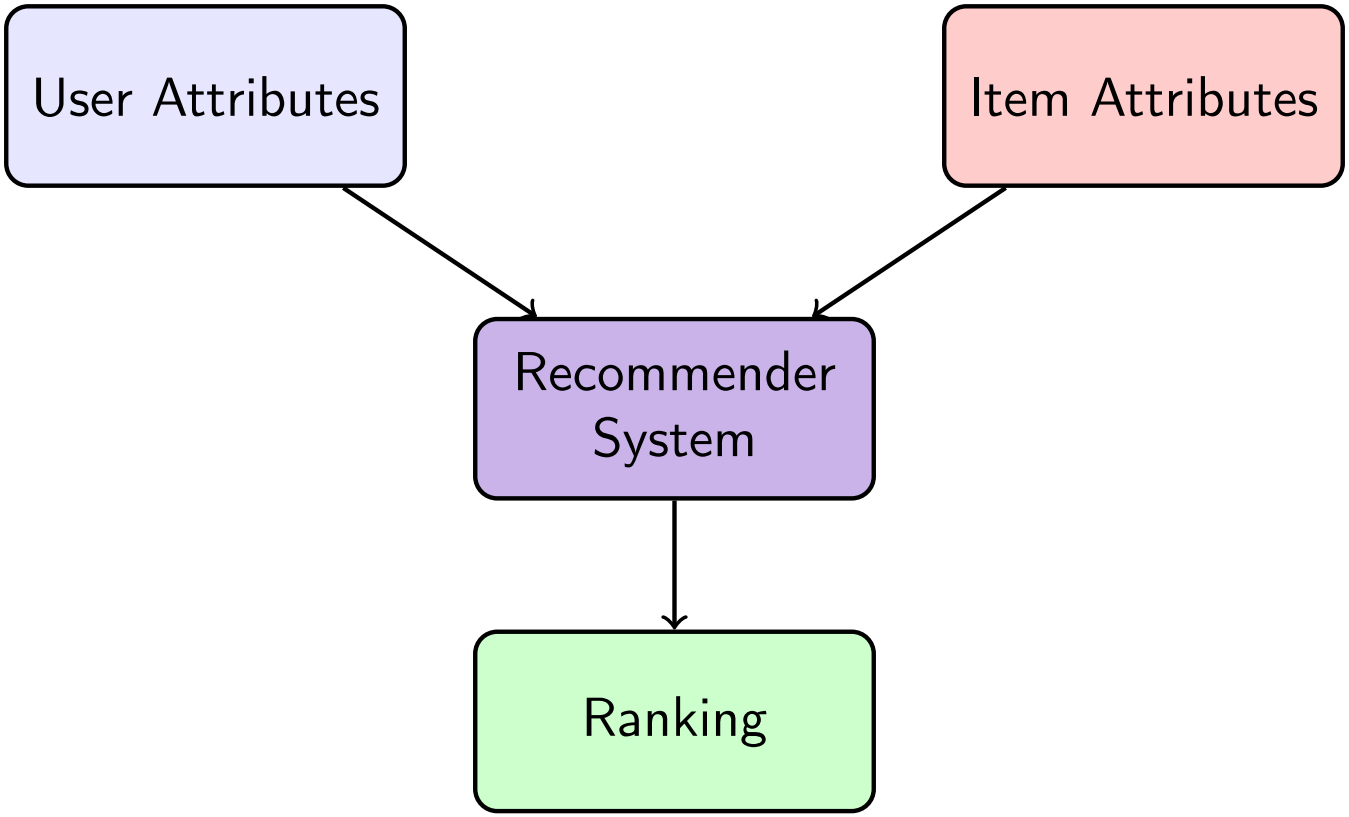
- User가 관심을 가질만한 Item을 추천

왜?

- 판매 (ex 도서, 영화, 음악 등)
- 정보 제공 (ex 검색포털, SNS)
- 평가 (ex 와인, 영화, 꿀팁들)

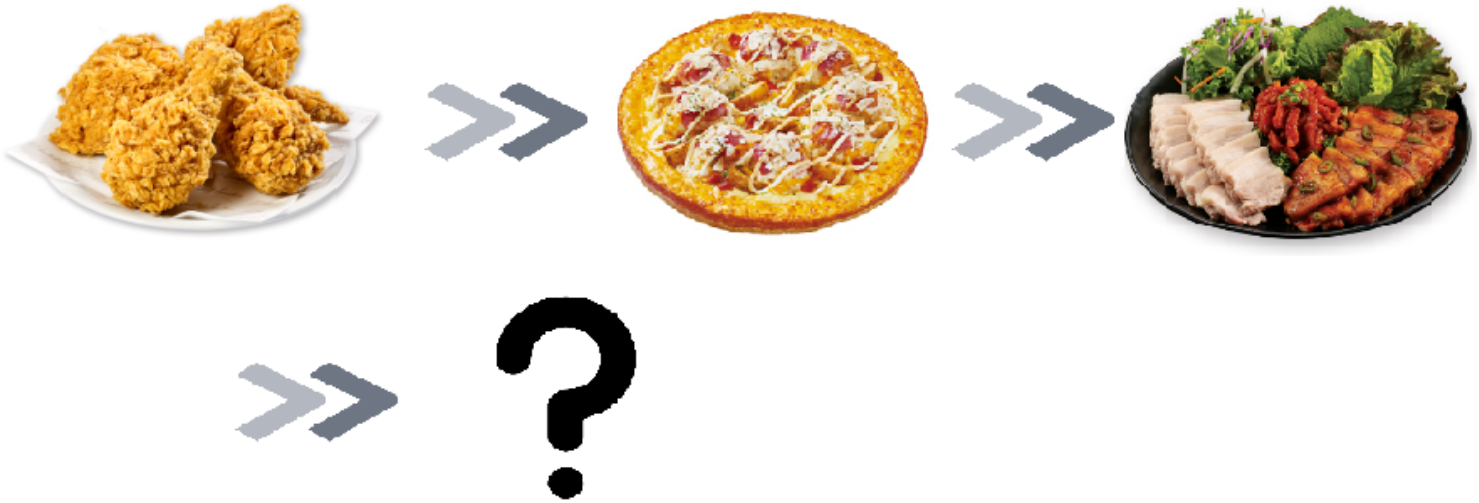
Intro.

추천 시스템의 역할



Intro.

개인화된(personalized) 추천 시스템



- 각 유저의 interest(= context)를 알 수 있을까?

Intro.

User Embedding

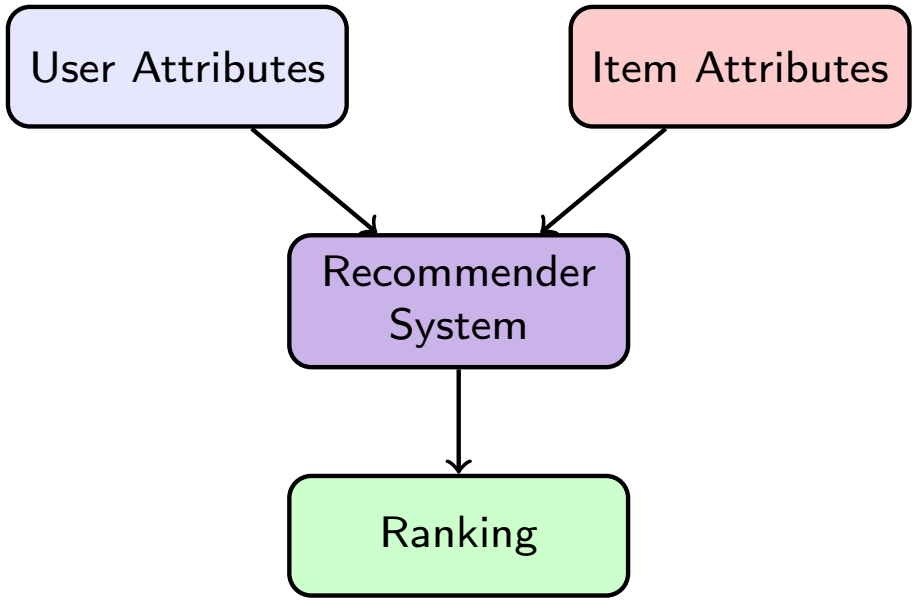
- HierTCN ([this paper], 2019)

Item Embedding

- localized GCN ([36], 2018)

Ranking

- inner product=<user, item>가 높은 순대로 아이템 추천



Intro.

User Embedding

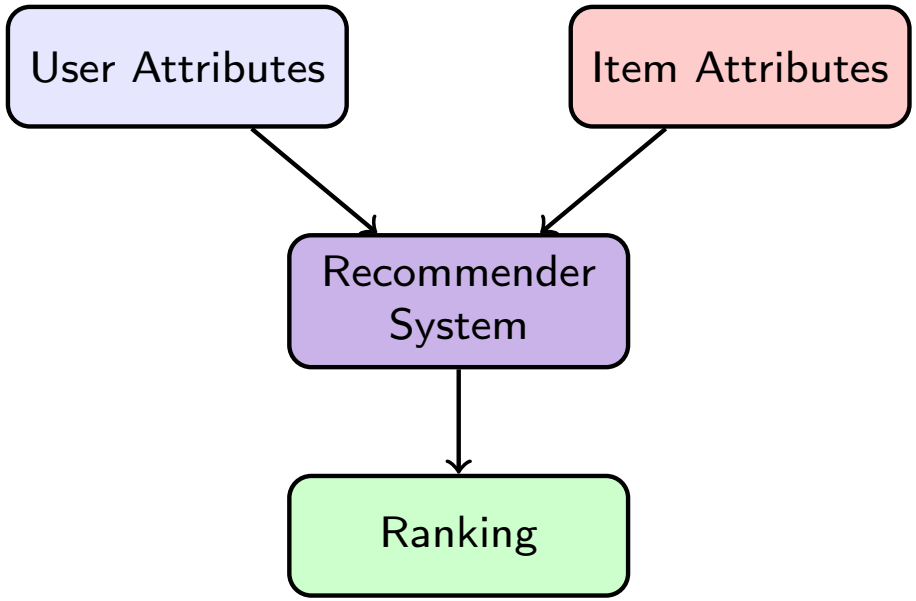
- HierTCN ([this paper], 2019)

Item Embedding

- localized GCN ([36], 2018)

Ranking

- inner product=<user, item>가 높은 순대로 아이템 추천



단, web-scale로!

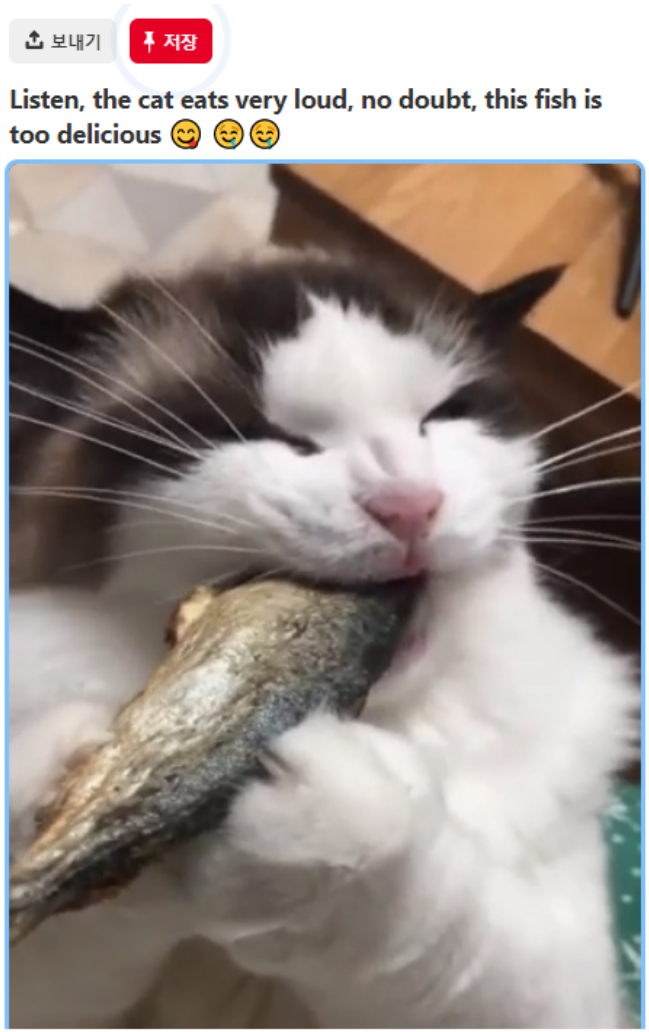
Web-scale

Items

- 약 7400만 개의 아이템
- 아이템 임베딩 ← 75억개의 데이터
- 유저 임베딩 ← 17억개의 데이터
- XING.com 구인공고
- pinterest.com 이미지와 주석

Users

- 약 600만 명의 유저



Web-scale

Item Embedding

- 하루에 한번씩 업데이트
- 빠른 로딩과 대용량 저장을 위해 Linux huge page table 이용

Web-scale

Item Embedding

- 하루에 한번씩 업데이트
- 빠른 로딩과 대용량 저장을 위해 Linux huge page table 이용

User Embedding

- 아이템 임베딩을 고정시킨 후 유저 임베딩에 활용
- 실시간으로 업데이트하여 Ranking 계산
- 유저 **interest**를 수치화하여 저장

Web-scale

Item Embedding

- 하루에 한번씩 업데이트
- 빠른 로딩과 대용량 저장을 위해 Linux huge page table 이용

User Embedding

- 아이템 임베딩을 고정시킨 후 유저 임베딩에 활용
- 실시간으로 업데이트하여 Ranking 계산
- 유저 **interest**를 수치화하여 저장

결국 web-scale 이란..

- 방대한 데이터를 빠르게 학습시키고 효율적으로 저장

Table of Contents

data

session data

user_id	session_id	session_start	click_article_id	click_timestamp
0	1506825423271737	1506825423000	157541	1506826828020
0	1506825423271737	1506825423000	68866	1506826858020
1	1506825426267738	1506825426000	235840	1506827017951
1	1506825426267738	1506825426000	96663	1506827047951
2	1506825435299739	1506825435000	119592	1506827090575
2	1506825435299739	1506825435000	30970	1506827120575
3	1506825442704740	1506825442000	236065	1506827536942
3	1506825442704740	1506825442000	236294	1506827566942
4	1506825528135741	1506825528000	48915	1506826927593
4	1506825528135741	1506825528000	44488	1506826957593

Data

user interacts

$$x = (x_1, x_2, \ldots, x_n)$$

- x_t = (t-번째 시간 아이템의 임베딩)
- 한 명의 유저가 시간에 따라 interact한 아이템들 (클릭, 북마크, 댓글)

Data

user interacts

$$x = (x_1, x_2, \ldots, x_n)$$

- x_t = (t-번째 시간 아이템의 임베딩)
- 한 명의 유저가 시간에 따라 interact한 아이템들 (클릭, 북마크, 댓글)

어떻게 아이템 임베딩을 얻나?

Data

user interacts

$$x = (x_1, x_2, \ldots, x_n)$$

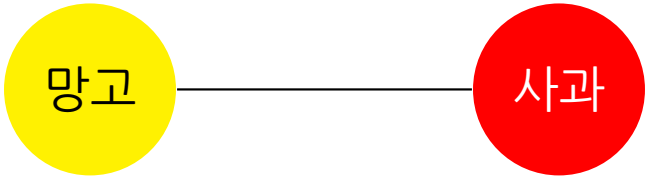
- x_t = (t-번째 시간 아이템의 임베딩)
- 한 명의 유저가 시간에 따라 interact한 아이템들 (클릭, 북마크, 댓글)

어떻게 아이템 임베딩을 얻나?

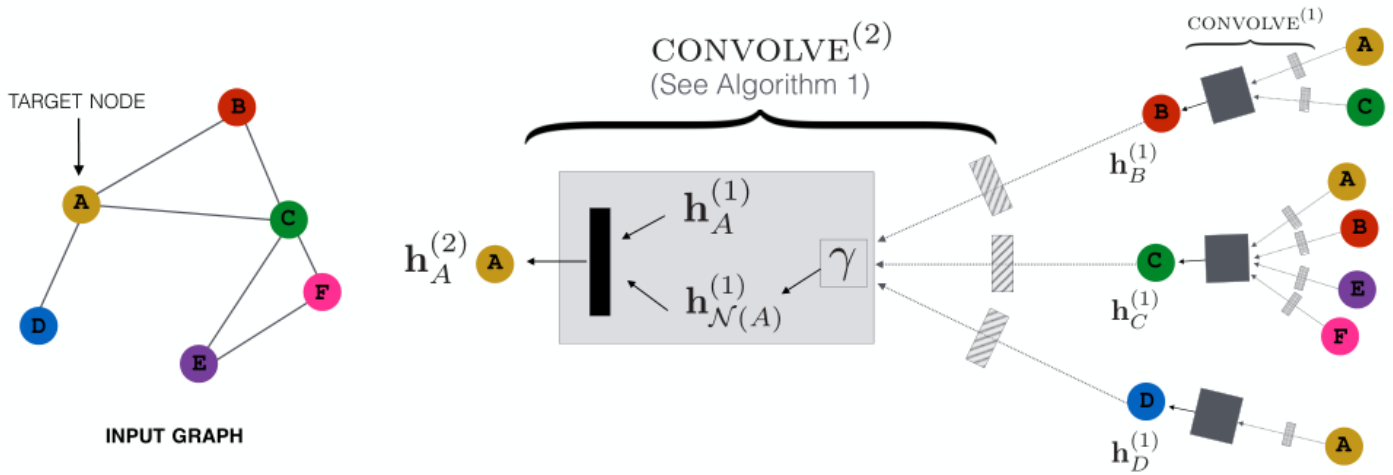
localized Graph Convolutional Networks (R. Ying et al., 2018)

- 유저와 아이템 간의 관계를 **그래프**로 표현
- 그래프의 꼭지점(node)를 하나의 벡터로 임베딩

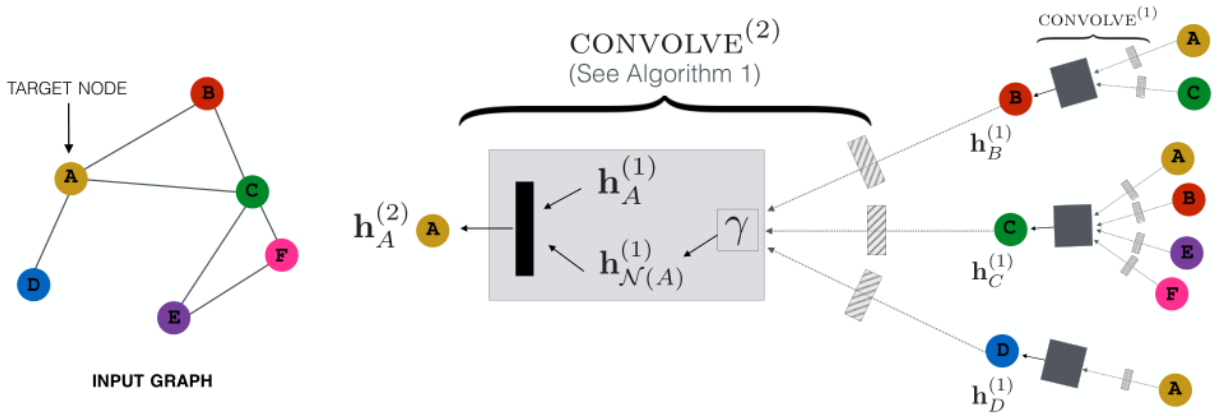
Graph construct



- ▷ 노드 = 아이템
- ▷ 한 유저가 두 아이템에 interact한 경우, 노드 연결



graph construct



update

- $h_{N(u)}^{(l)} = \text{AGG}(\text{ReLU}(Q^{(l)} h_v^{(l)} + q^{(l)}) | v \in N(u))$
- $h_u^{(l+1)} = \text{ReLU}(W^{(l)} \cdot \text{CONCAT}(h_u^{(l)}, h_{N(u)}^{(l)}))$

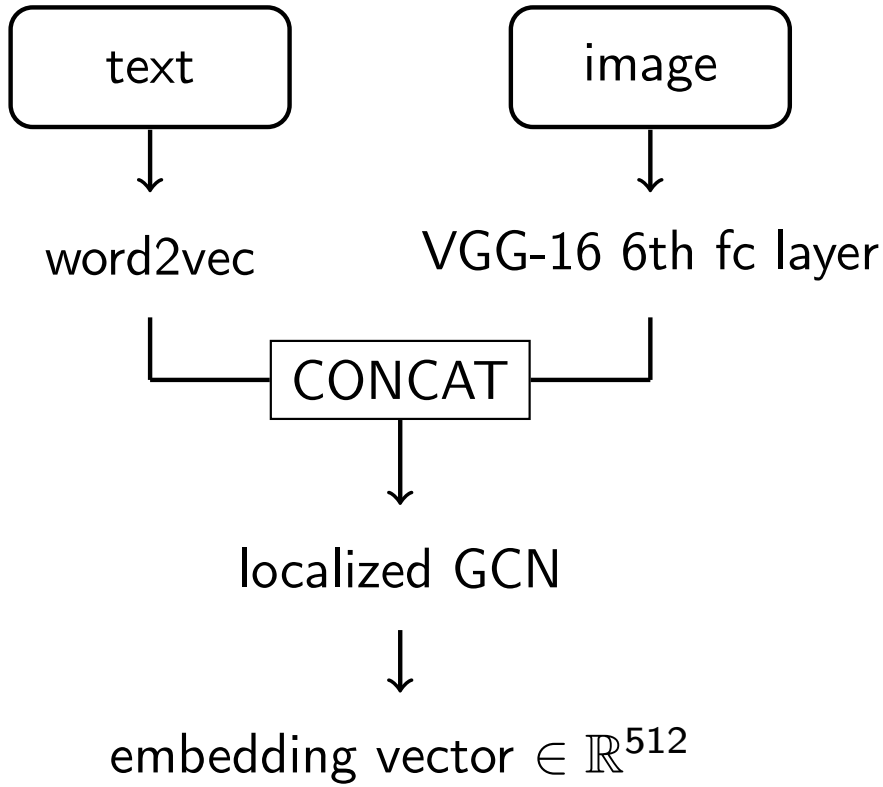
objective function

- Noise Contrastive Estimation (NCE)

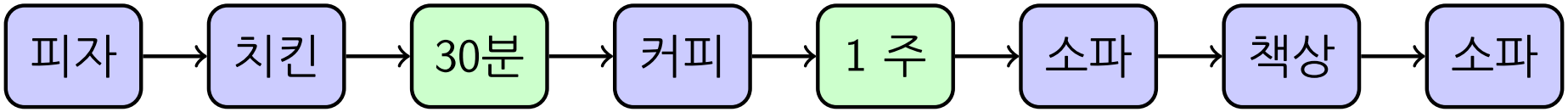
$$-\log(\sigma(h_u^T h_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(u)} \log(\sigma(-h_u^T h_{v_n})), \forall v \in N(u)$$

보내기 저장

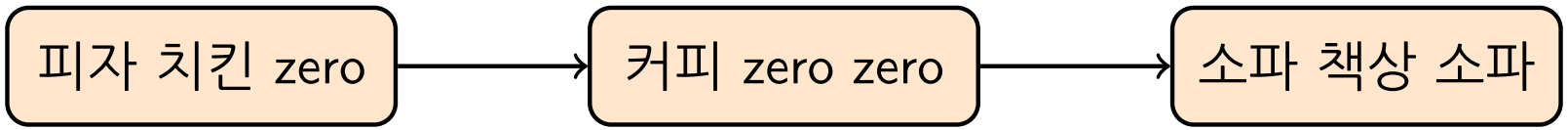
Listen, the cat eats very loud, no doubt, this fish is too delicious 😋 😋😋



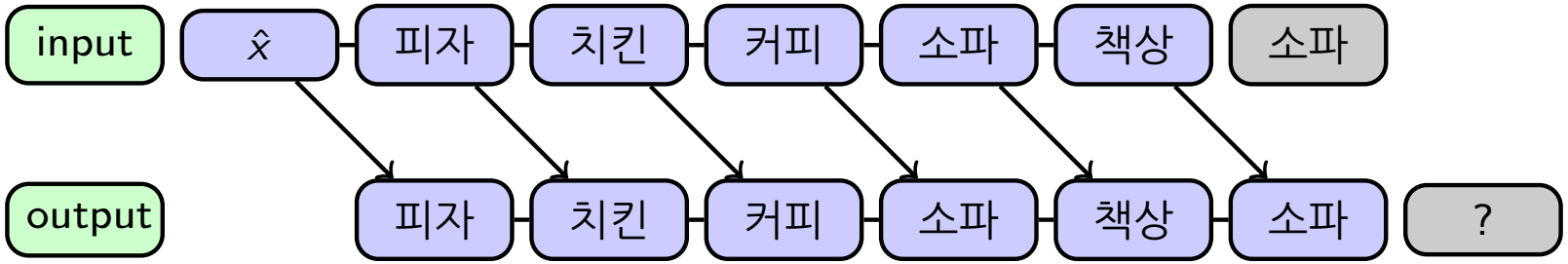
Session

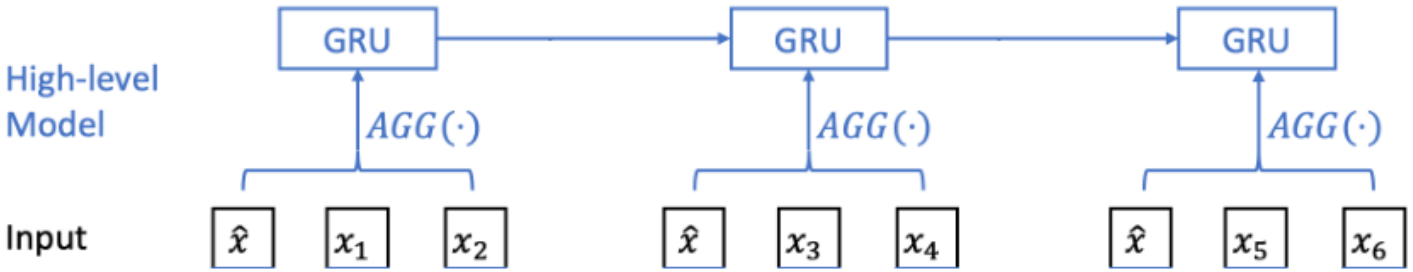


- 30분의 유휴시간 간격으로 session 구분
- 유저마다 session의 개수와 길이가 다양하다
- 모든 session의 길이가 같도록 zero padding
- ex) session 개수 = 3, session 길이 = 3



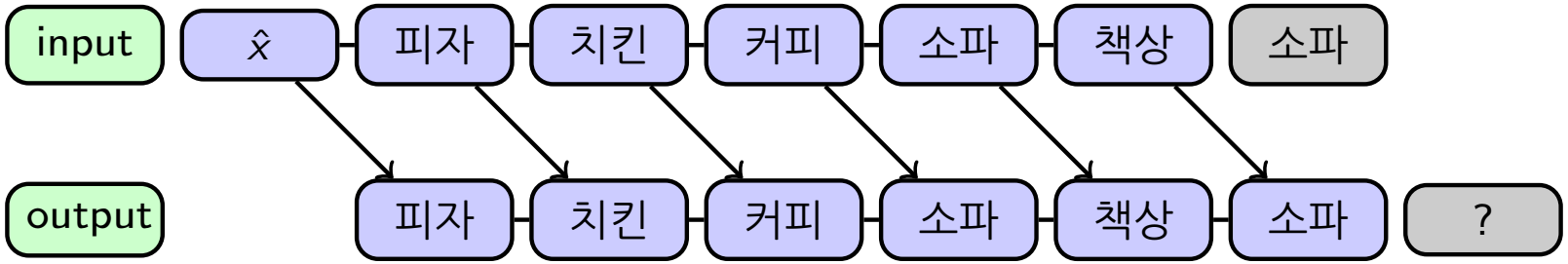
Model

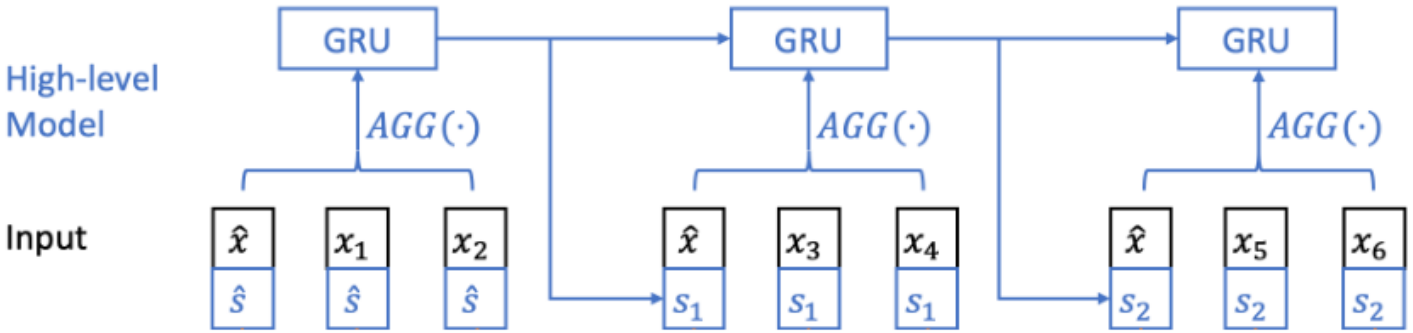

input
$\hat{x}$
피자
치킨
커피
소파
책상
소파
output
피자
치킨
커피
소파
책상
소파
?


High-level Model
GRU
GRU
GRU
$AGG(\cdot)$
$AGG(\cdot)$
$AGG(\cdot)$
Input
$\hat{x}$
x_1
x_2
$\hat{x}$
x_3
x_4
$\hat{x}$
x_5
x_6


Time
t=1
t=2
t=3
t=4
t=5
t=6
Session 1
Session 2
Session 3

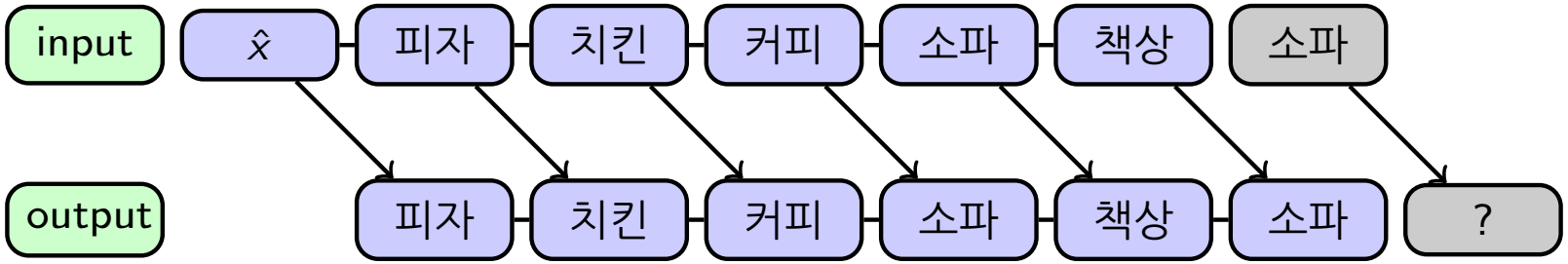
Model

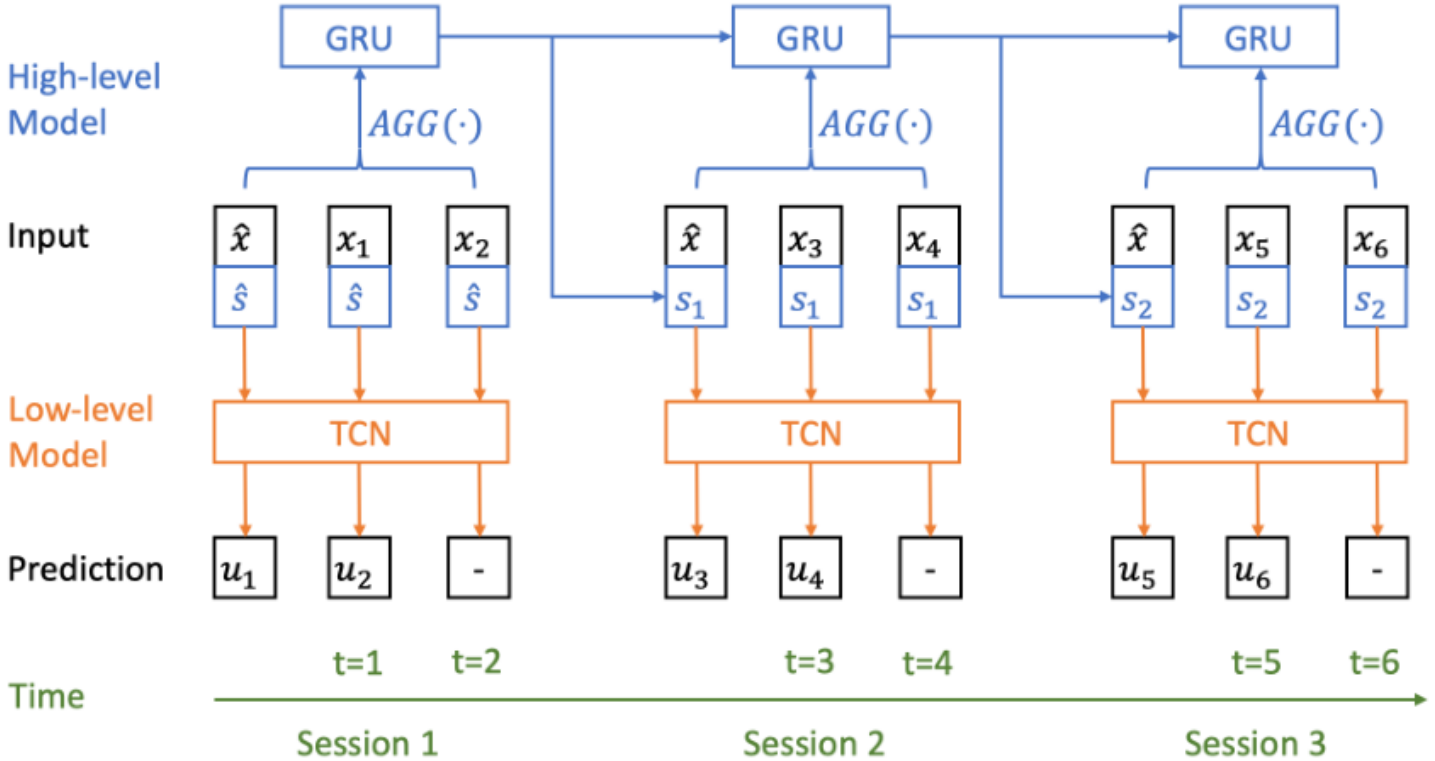

input
$\hat{x}$
피자
치킨
커피
소파
책상
소파
output
피자
치킨
커피
소파
책상
소파
?


High-level Model
GRU
GRU
GRU
$AGG(\cdot)$
$AGG(\cdot)$
$AGG(\cdot)$
Input
$\hat{x}$ x_1 x_2
$\hat{s}$ $\hat{s}$ $\hat{s}$
$\hat{x}$ x_3 x_4
s_1 s_1 s_1
$\hat{x}$ x_5 x_6
s_2 s_2 s_2


Time
t=1
t=2
t=3
t=4
t=5
t=6
Session 1
Session 2
Session 3

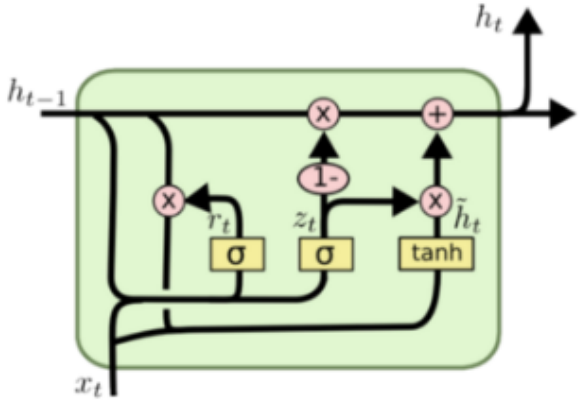
Model


input
output
피자
치킨
커피
소파
책상
소파
피자
치킨
커피
소파
책상
소파
?


High-level Model
Input
Low-level Model
Prediction
Time
GRU
GRU
GRU
AGG(·)
AGG(·)
AGG(·)
TCN
TCN
TCN
t=1
t=2
t=3
t=4
t=5
t=6
Session 1
Session 2
Session 3

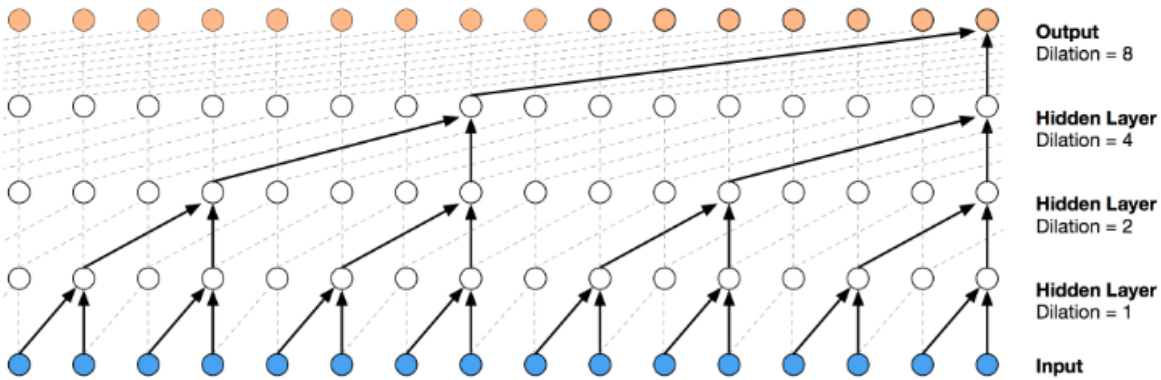
Model

GRU - long term inference



$$z_t = \sigma(W^z \cdot x_t + U^z \cdot h_{t-1})$$
$$r_t = \sigma(W^r \cdot x_t + U^r \cdot h_{t-1})$$
$$\bar{h}_t = \tanh(W \cdot x_t + U \cdot (h_{t-1} * r_t))$$
$$h_t = (1 - z_t) * \bar{h}_t + z_t * h_{t-1}$$

TCN - short term inference



causal 1D conv

+

dilated 1D conv

Web scale again

아이템 임베딩 저장

- 512차원의 interact 데이터를 17억개 저장하려면 몇 십 테러바이트 필요!
- Linux huge page table에 아이템 ID와 아이템 임베딩 미리 저장
- 아이템 ID만 갖고 있다가 필요할 때 look-up하여 로딩 시간을 줄이자

유저 interest 저장

- 유저 ID와 유저 interest 또한 저장
- 매 session이 시작하면 불러오고, 통과가 끝나면 저장

Web scale again

queue-based mini-batch

- HierTCN을 학습시키 위해서는 session의 길이와 크기 고정 필요!
- 사용하지 않은 공간을 모두 zero padding한다면?

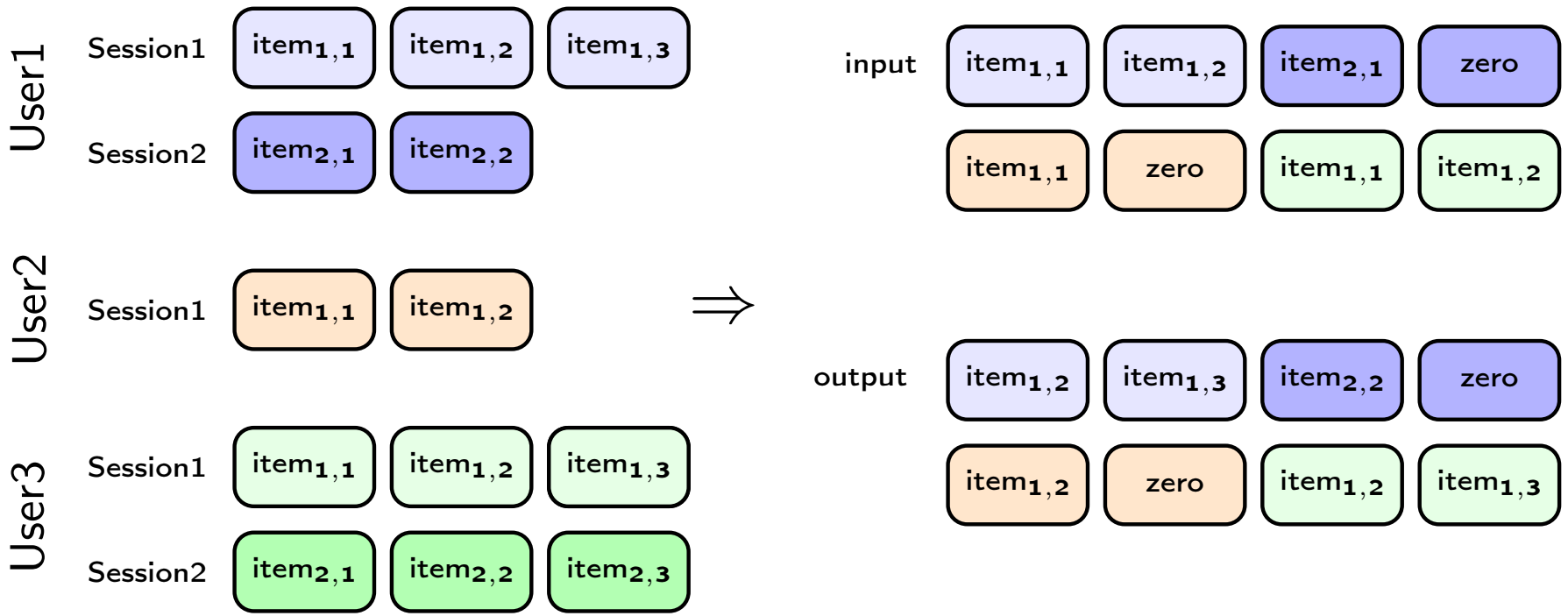


Table of Contents

Result

HierTCN vs Others

Dataset	Metric	**HierTCN**	HierGRU	HRNN [23]	TCN	GRU	MV	MaxItem
XING	Recall@10 (Higher is better)	**0.139**	0.105	0.113	0.108	0.124	0.107	-
	MRR (Higher is better)	**0.071**	0.037	0.040	0.038	0.043	-	-
	MRP (Lower is better)	**0.121**	0.161	0.149	0.198	0.141	-	-
Pinterest	Recall@1	**0.206**	0.184	0.174	0.194	0.195	0.1658	0.1487
	Recall@5	**0.663**	0.639	0.619	0.653	0.655	0.6120	0.5843
	Recall@10	**0.855**	0.839	0.828	0.850	0.852	0.8173	0.7921
	MRR	**0.402**	0.380	0.366	0.391	0.392	0.3593	0.3400
	MRP	**0.304**	0.326	0.338	0.313	0.311	0.3484	0.3724

Recall @ K $= \frac{1}{N}\sum_{i=1}^{N} |\text{relevant}_i \cap \text{top K of retrieved}_i|$, $N =$ (# of Users)

Mean Reciprocal Rank (MRR) $= \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\text{rank}_i}$

Mean Rank Percentile (MRP) $= \frac{1}{N}\sum_{i=1}^{N} \frac{\text{rank}_i}{|\text{cand}_i|}$

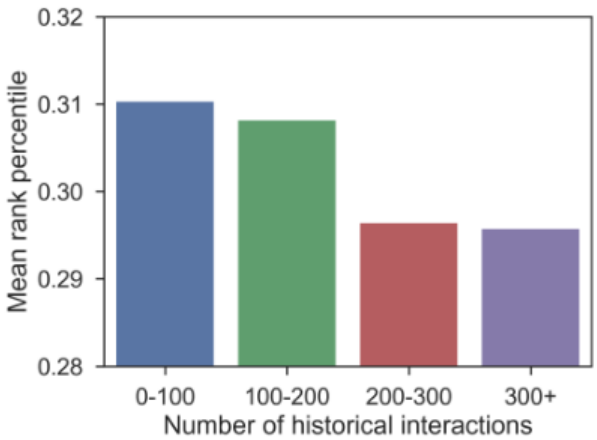
Result

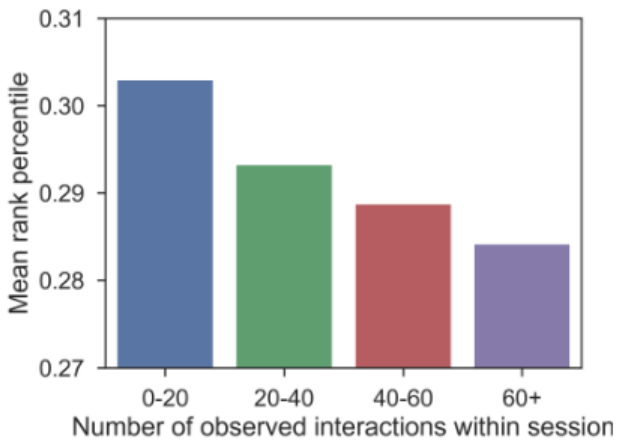
HierTCN vs Others

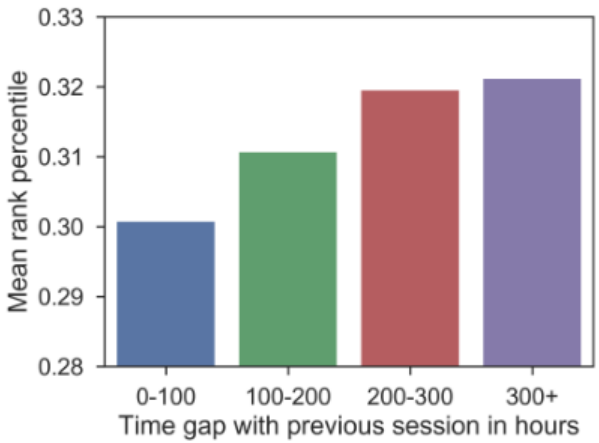
Dataset	Model	Training
Pinterest	**HierTCN**	**295.3s**
	HierGRU	747.8s
	HRNN [23]	693.1s
	TCN	**254.0s**
	GRU	776.8s

Result

Performance (Lower is better)







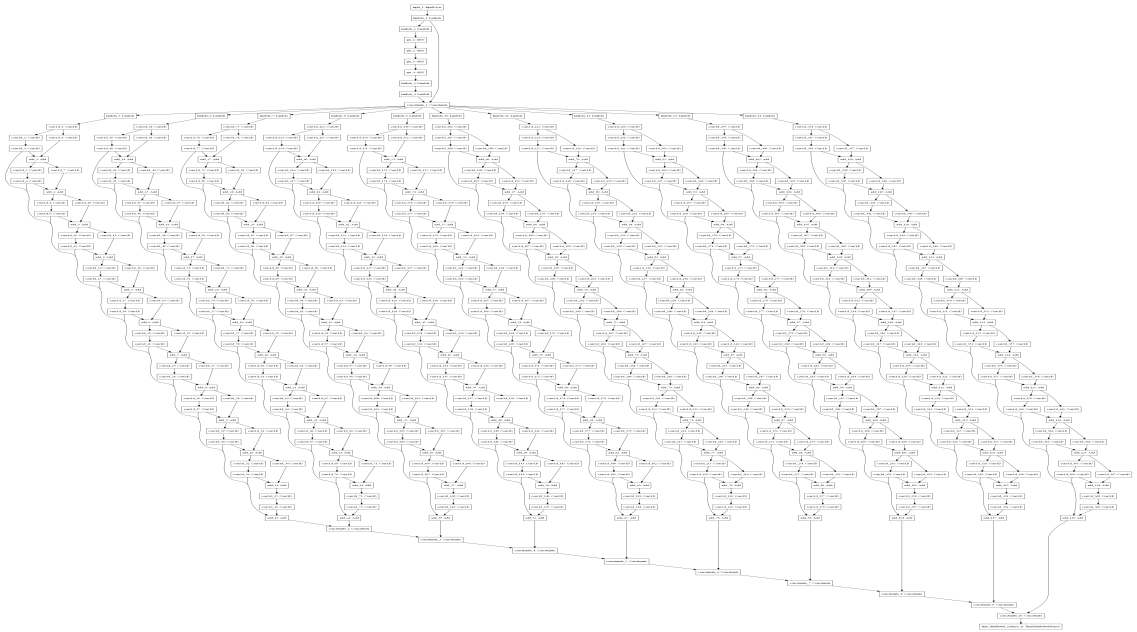
Result

Object functions

Dataset	Metric	L2	NCE	BPR	Hinge
Pinterest	Recall@1	0.167	0.192	0.183	**0.201**
	Recall@5	0.611	0.642	0.632	**0.657**
	Recall@10	0.817	0.838	0.832	**0.851**
	MRR	0.360	0.386	0.377	**0.396**
	MRP	0.349	0.322	0.331	**0.309**

- $\mathrm{L2} = \min ||x_t - u_t||_2$
- $\mathrm{NCE} = \min\{-\log(\sigma(x_t^T \cdot u_t)) - \sum_i -\log(\sigma(-c_{ti}^T \cdot u_t))\}$
- $\mathrm{BPR} = \min\{-\sum_i \log(\sigma(c_{ti}^T u_t - x_t^T u_t))\}$
- $\mathrm{Hinge} = \min\{\sum_i \max\{0, \delta + c_{ti}^T u_t - x_t^T u_t\}\}$

Implementation with Keras



concatenate_10 (Concatenate)	(8, 50, 128)	0	concatenate_9[0][0] add_130[0][0]
time_distributed_1 (TimeDistrib	(8, 50, 512)	66048	concatenate_10[0][0]

Total params: 22,936,576
Trainable params: 22,936,576
Non-trainable params: 0

Result


Session 1
Session 2
Session 3
x_1
x_2
x_3
x_4
x_5
x_6
Past Interactions
Rank=0
Rank=1
Rank=2
Rank=3
Rank=4
Rank=5
MV Predictions
TCN Predictions
HierTCN Predictions

Q & A

감사합니다.