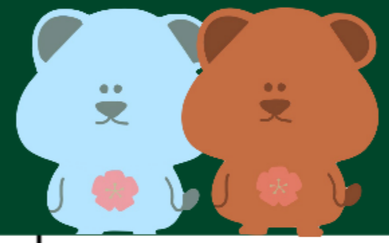
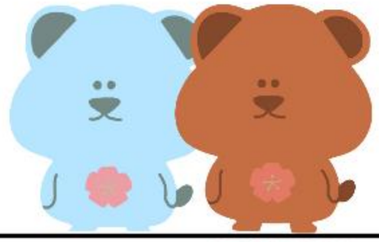


BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer(CIKM 2019)

손소현

EWHA EURON

목차



EWHA EURON

#논문으로 들어가기 전에

알리 익스프레스, 타오바오 등을 소유한 기업인 알리바바에서 2019년에 CIKM에서 발표한 논문

말 그대로 구글의 BERT를 추천 시스템에 결합한 논문
: BERT 원 논문은 크게 MLM (Masked Language Model)과 NSP (Next Sentence Prediction) 두 가지 테스크를 진행하는데, MLM만 사용. 후자를 사용하지 않는 이유는 뒤에서 설명 예정

BERT
: Bidirectional Encoder Representations from Transformers
(양!방향, 트랜스포머!) 논문은 왼쪽 인코더 부분을 차용함

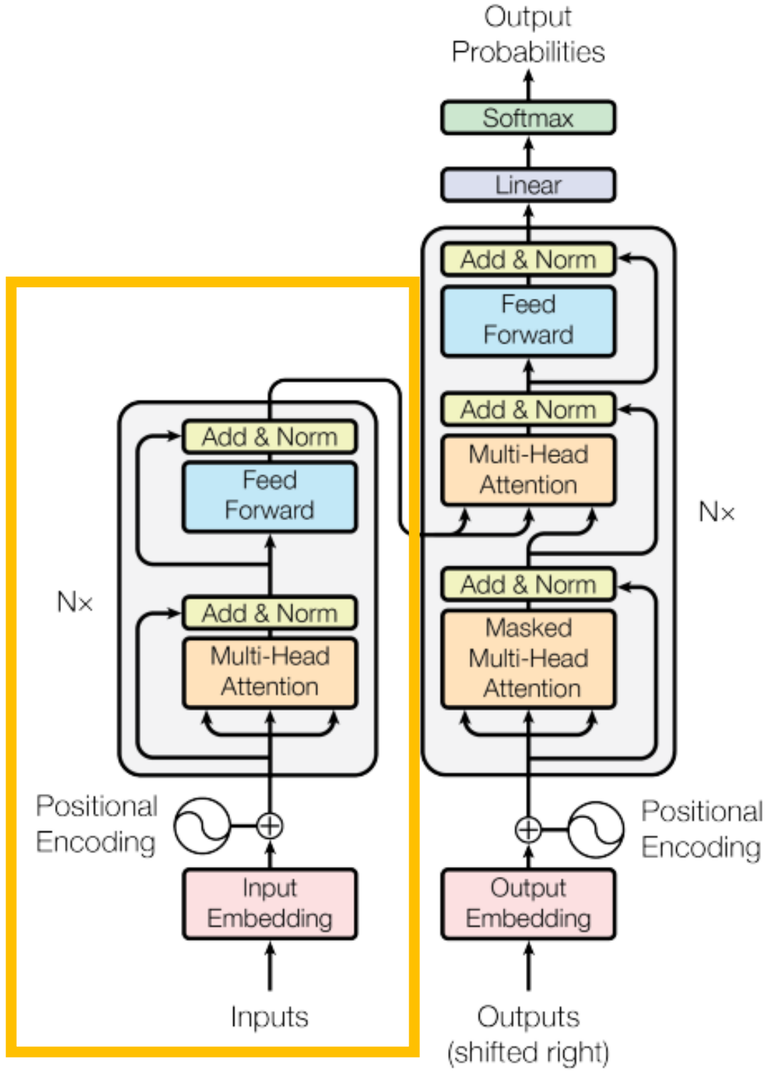


Figure 1: The Transformer - model architecture.

#논문으로 들어가기 전에

트랜스포머 구조의 특징
- 인코더, 디코더 구조로 되어있음
- 셀프 어텐션을 사용 (Query, Key, Value가 들어옴)
- multi head attention이 있음
- residual connection과 layer normalization을 사용

아래의 트랜스포머 식이 본 논문에서 거의 그대로 적용됨

$$Attention(Q, K, V) = softmax_k(\frac{QK^T}{\sqrt{d_k}})V$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

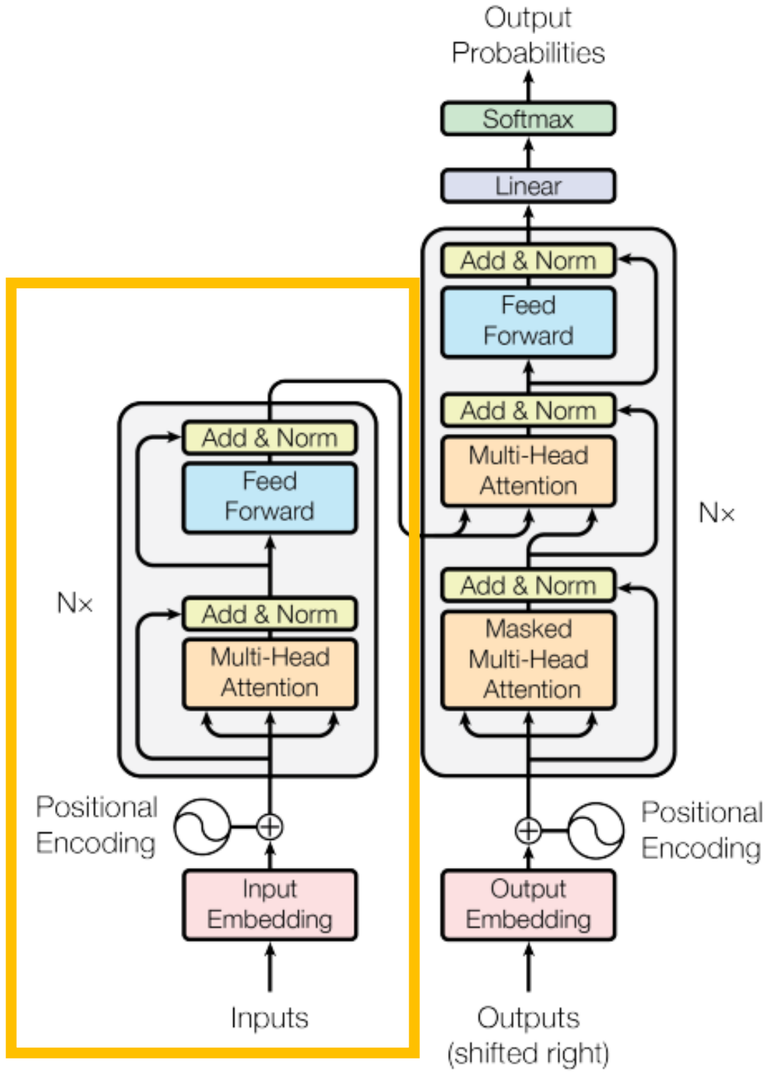
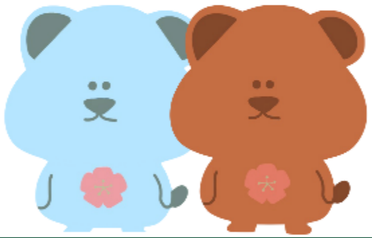


Figure 1: The Transformer - model architecture.

#00 Abstract



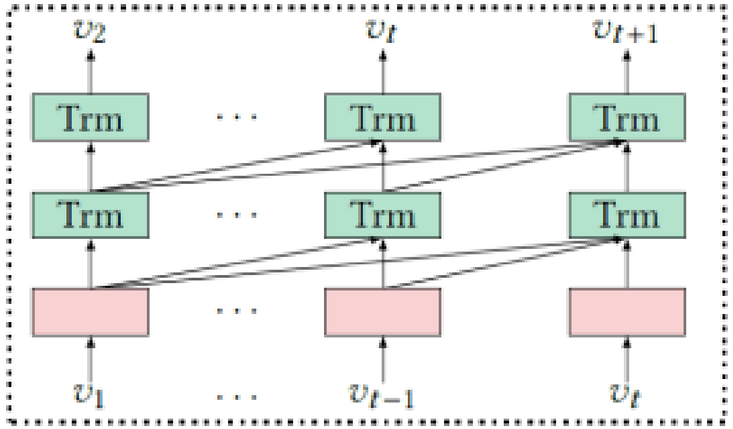
EWHA EURON

#00 Abstract

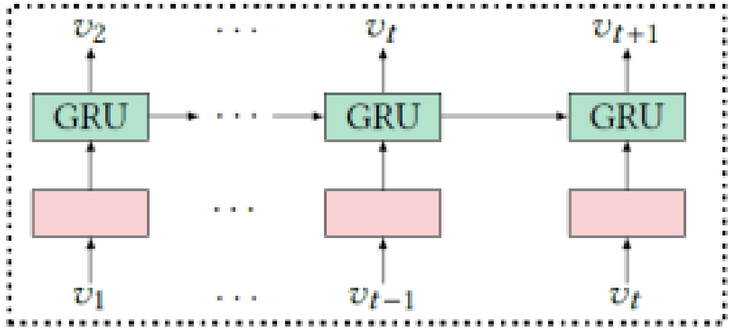
1. 유저의 과거의 행동에서 다이나믹한 선호도를 찾는 것은 추천시스템에서 중요
2. 이전의 방법은 left to right(단방향, unidirectional)으로 'sequential' 신경망을 사용. 하지만 이런 단방향 방법은 한계가 있음
   - 단방향 아키텍처는 유저 행동 시퀀스의 hidden representaion의 성능을 제한
   - 종종 순서를 엄격하게 지키게 하는데, 그것은 항상 실용적인 것은 아님
3. 이런 한계를 처리하기 위해, BERT4Rec라고 하는 **양방향** sequential 추천시스템을 제안
4. 정보 누출(information leakage)을 막고, 효율적으로 양방향 모델을 훈련시키기 위해 , **Cloze** 를 채택
   - Cloze란? 랜덤하게 가려진(masked) 아이템을 예측하는 방법

#00 Abstract

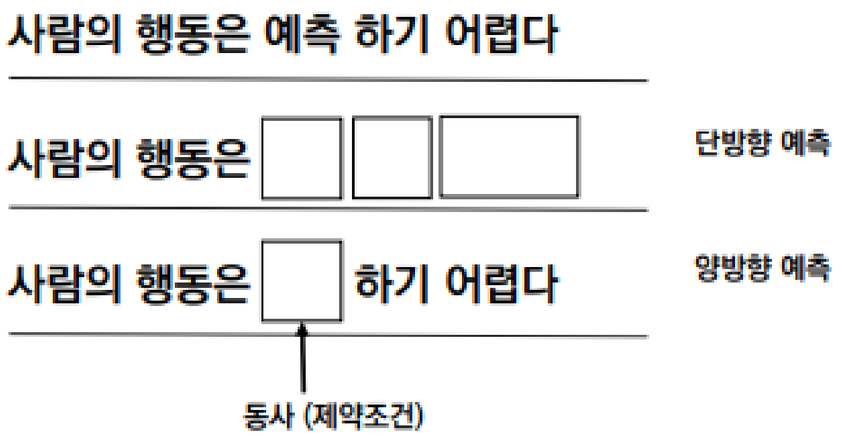
unidirectional vs bidirectional



(c) SASRec model architecture.



(d) RNN based sequential recommendation methods.



사람의 행동은? → 제약이 없고, 관측되지 않은 외부 요소 존재

양방향 및 복잡한 관계 학습 가능한 모델 필요

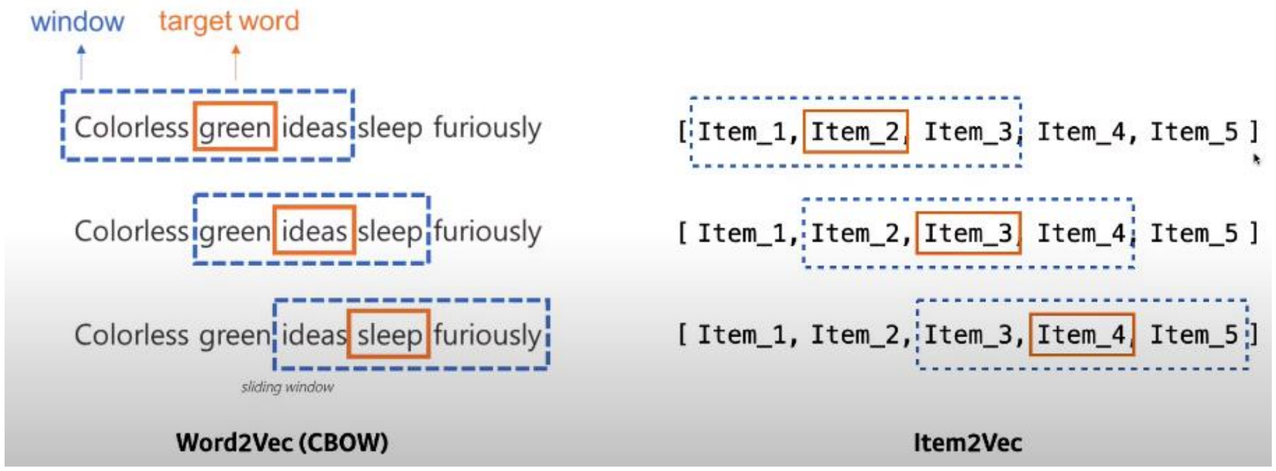
사진출처 : 고려대 DSBA 연구실 세미나 발표자료

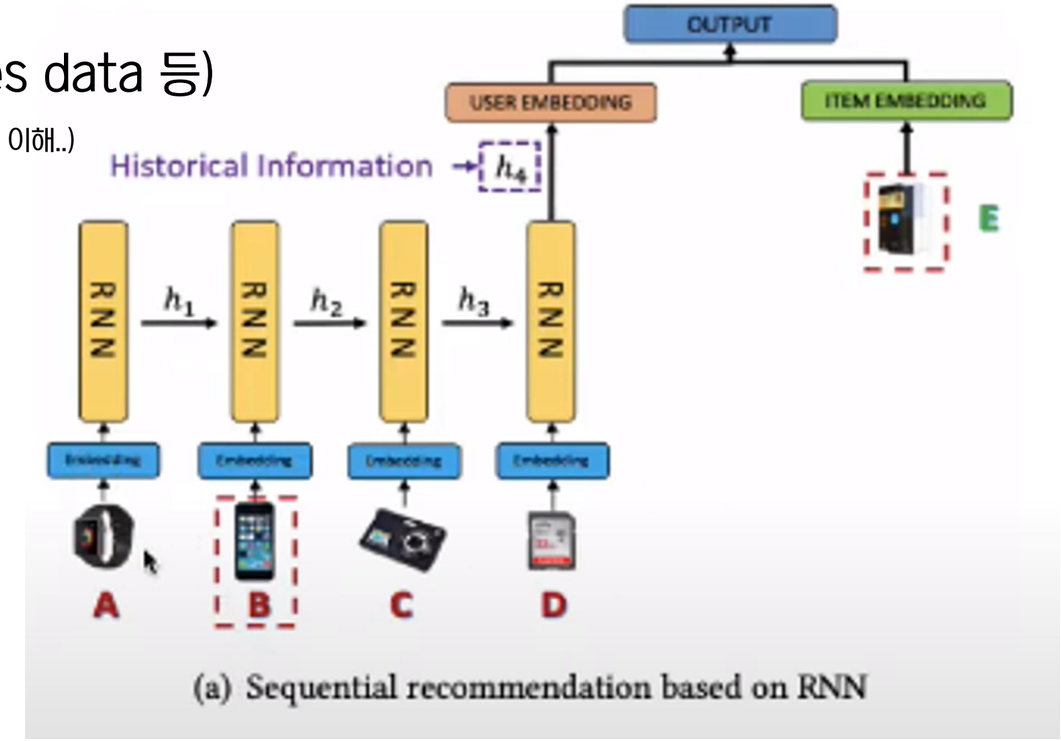
#01 Introduction

EWHA EURON

#01 Introduction

- ✓ 사용자의 과거행동은 선호도에 많은 영향을 줌.
  ex) 사용자가 과거에 아이폰을 산 이력이 있으면,
  그 사용자는 아이폰 악세사리를 살 확률이 높지, 갤럭시 악세사리를 살 확률은 낮음.
- ✓ 그래서 사용자의 과거이력을 고려하여 아이템을 추천하기 위한, 많은 sequential recommendation 방법들이 고안되어 왔음
- ✓ 최근에는, **RNN**과 결합된 형태 : 유저 과거 이력을 하나의 벡터에 인코딩
  - 그치만 RNN도 왼→ 오, 단방향모델
  - hidden representation 성능을 제한함
  - RNN 자체는 natural order를 위해 고안된 것 (text, series data 등)
  - 관찰할 수 없는 외부요소들에 영향을 많이 받음 (소비유행 같은거라고 이해..)

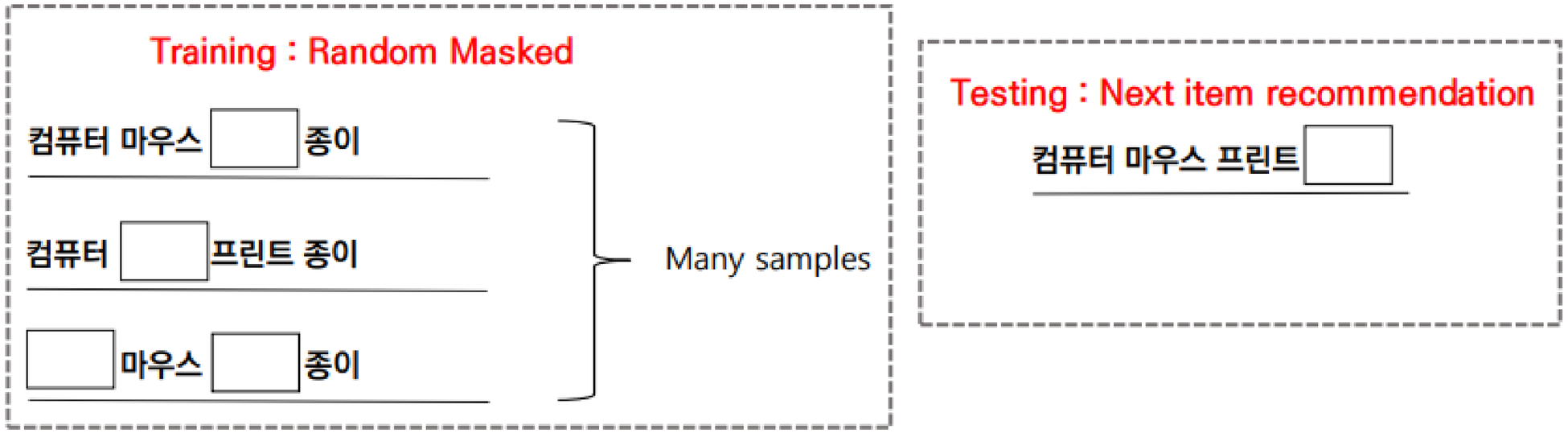




#01 Introduction

✓ 그래서 양방향 모델을 개발할 필요가 있음

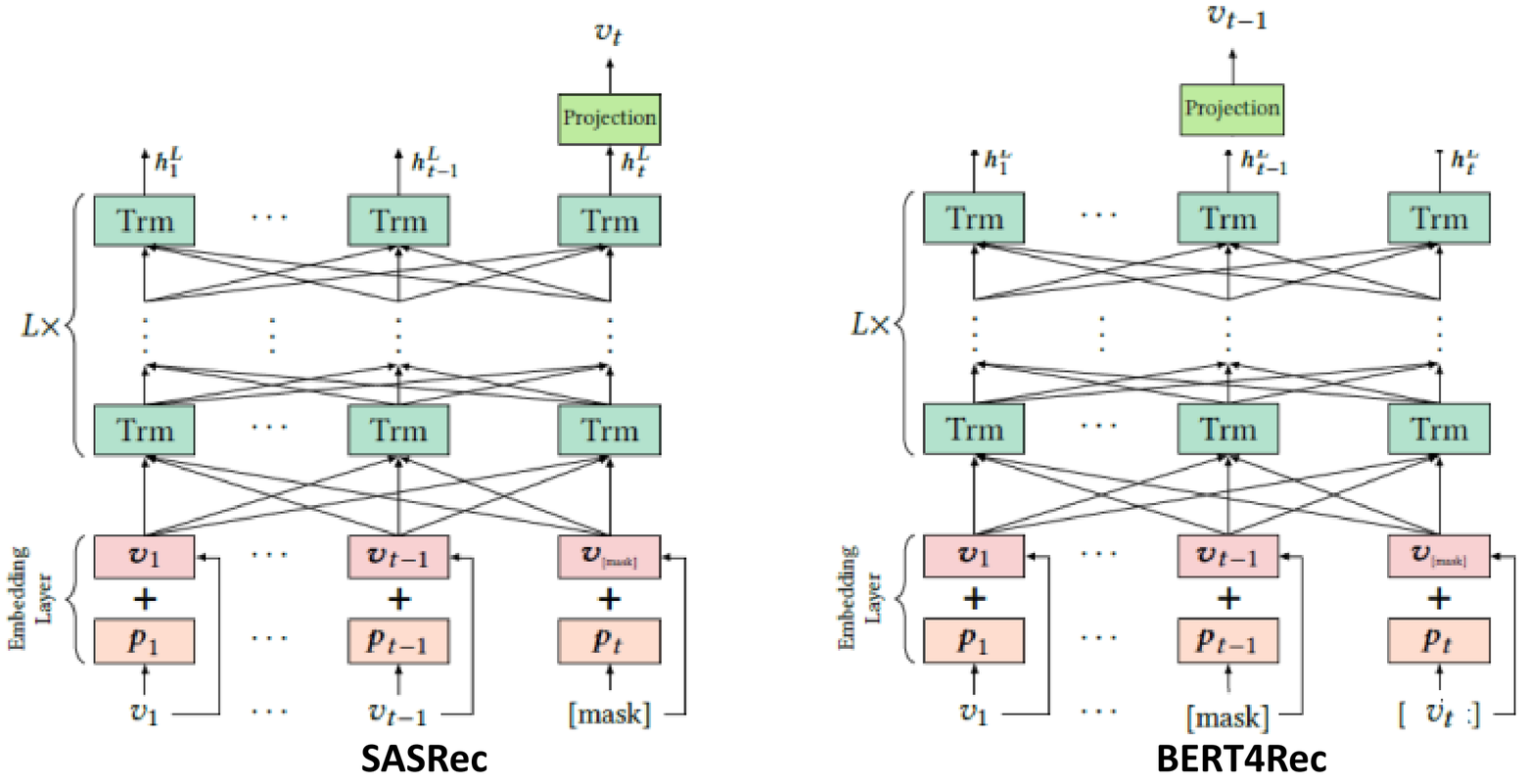
✓ 근데 조심해야 할 점은 conventional sequential 추천모델은 보통 **input받은 순서**에 따라 각 자리에서 다음 자리에 올 것이 무엇인지 예측하는 식으로 훈련되어 옴. 그림과 같이 양방향 모델은 정보 누출(leakage)이 있을 수 있음. (각 아이템이 간접적으로 타겟 아이템을 봄으로써). 이렇게 되면 모델이 쓸모없게 만들 수 있음
😲개인적으로 이때의 '정보누출'이 ' 양방향' 모델링과 같은거 아닌가? 양방향으로 분석하게 되면 정보누출은 생길 수 밖에 없나 싶었는데, 아래 그림을 보고 이해했음.



사진출처 : 고려대 DSBA 연구실 세미나 발표자료

#01 Introduction

- ✓ 이런 게 논문에서 택한 방법인 **Cloze task!**
  - 랜덤하게 아이템을 [mask]토큰으로 숨기고, surrounding context 기반으로 item의 id를 예측하는 학습 방법
    →양방향으로 융합하여 학습하는 효과, 학습데이터가 늘어나는 효과,
  - 그러나 이 방법은 sequential추천의 최종 목표와 일치하지 않음. 따라서 테스트 단계에서는 [mask]토큰을 입력의 마지막에 추가하여 다음에 나타날 아이템을 예측하도록



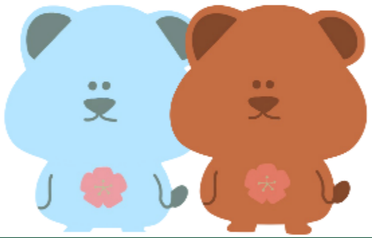
사진출처 : 고려대 DSBA 연구실 세미나 발표자료

#01 Introduction

contribution:

- 사용자 행동 sequence를 Cloze task 기반으로 양방향 셀프 어텐션을 활용하여 인코딩
- 4개의 벤치마크 데이터셋에 대해 SOTA모델과 BERT4Rec 성능 비교
- BERT4Rec 의 핵심 요소들이 성능에 미치는 영향 분석

#02 Related Work



EWHA EURON

#2.1 General Recommendation

- 협업 필터링 CF : 과거 행동을 기반으로 추천 제공
  - 행렬 분해 MF

- Item-based neighborhood : 과 상호작용한 아이템 간의 유사도 행렬을 바탕으로 추천 제공

- Deep learning based recommendation
  - 추가적인 정보 (텍스트, 이미지 등)를 CF모델에 결합
  - MF 대체
    - Neural Collaborative Filtering (NCF): MF에서 inner product 대신 MLP로 사용자 선호도 추정
    - AutoRec, CDAE : 오토 인코더 기반으로

#2.2 Sequantial Recommendation

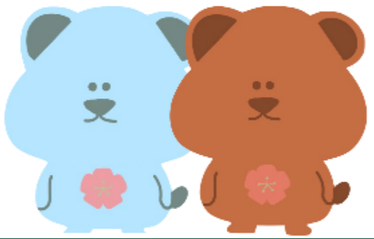
- Markov Chains (MCs) : 과거의 상호작용으로부터 sequential 패턴을 포착하여 추천 제공
  - Markov Decision Processes (MDPs)
  - Factorizing Personalized Markov Chains (FPMC) : MCs와 MF를 결합하여 상호작용의 순서와 일반적인 선호도를 고려

- Deep Learning models
  - Recurrent Neural Networks (RNN)
    - Gated Recurrent Unit (GRU)
      - session-based GRU with ranking loss (GRU4Rec), user-based GRU, attention-based GRU (NARM), improved GRU4Rec with new loss function (BPR-max, TOP1-max)
    - Long Short-Term Memory
  - Convolutional Sequence Model (Caser) convolution 필터를 활용하여 상호작용의 순서에 존재하는 패턴 학습, 이미지를 고려한 추천에 유용 ex. 옷 추천
  - Memory Netork
  - STAMP MLP와 attention을 활용하여 사용자의 일반적인 관심사와 현재 관심사를 포착

#2.3 Attention Mechanism

- attention mechanism into GRU
- purely attention-based neural networks
  - SASREC 2개의 Transformer decoder layer를 사용 : attention mask를 사용하여 여전히 unidirectional model ⇒ BERT4Rec은 bi-directional model로 Cloze task를 활용해 사용자의 행동을 encoding

#03 BERT4Rec



EWHA EURON

#3.1~3.2

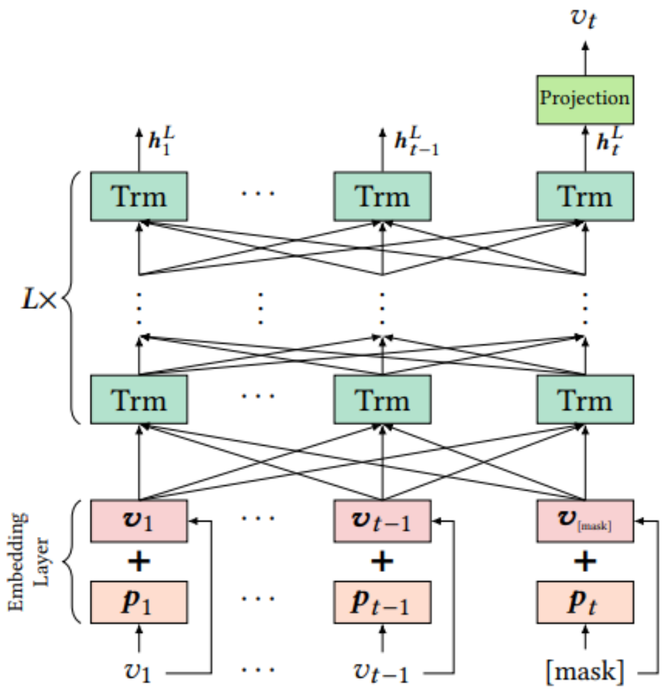
$$U = [u1, u2, ..., u_{|U|}]$$

$$V = [v1, v2, ..., v_{|V|}]$$

$$Su = [v_1^{(u)}, ..., v_t^{(u)}, ..., v_{n_u}^{(u)}]$$

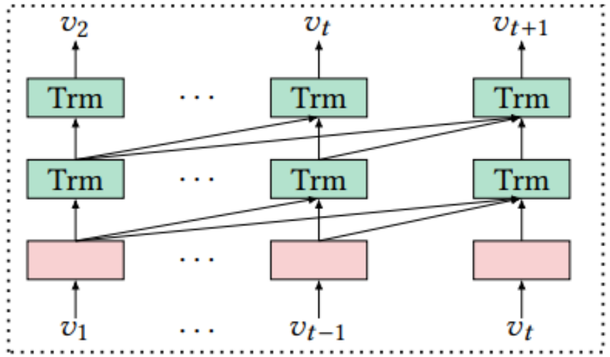
➡ $$p(v_{n_u+1}^{(u)} = v | S_u)$$

U는 유저
V는 아이템 벡터
S는 순서(시퀀스)
즉 P는, 유저가 V(아이템)을 좋아할 확률
(Positive일 확률)

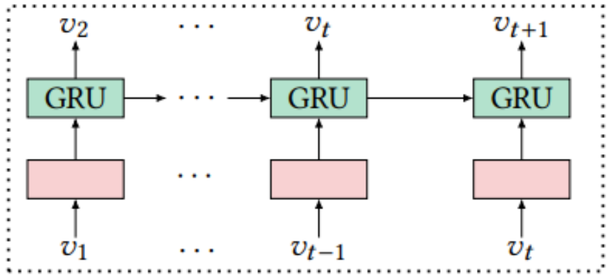


(b) BERT4Rec model architecture.

왼쪽에서 나온 P를 입력으로 받아서,
벡터에 임베딩, 트렌스포머 레이어를
L개 통과, 그리고 헤드에 임베딩하여
loss 계산



(c) SASRec model architecture.



(d) RNN based sequential recommendation methods.

단방향인 학습과정

EWHA EURON

#3.3 Transformer Layer

Multi-head self attention

- Head 별로 각기 다른 어텐션이 가능하도록 Transpose 후 각각 Query, Key, Value attention에 통과
- 다시 Transpose한 후 모든 head들의 attention 결과를 합침

$$\mathrm{MH}(\boldsymbol{H}^l) = [\mathrm{head}_1; \mathrm{head}_2; \ldots; \mathrm{head}_h]\boldsymbol{W}^O$$

$$\mathrm{head}_i = \mathtt{Attention}(\boldsymbol{H}^l\boldsymbol{W}_i^Q, \boldsymbol{H}^l\boldsymbol{W}_i^K, \boldsymbol{H}^l\boldsymbol{W}_i^V) \quad (1)$$

$$\mathtt{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d/h}}\right)\boldsymbol{V} \quad (2)$$

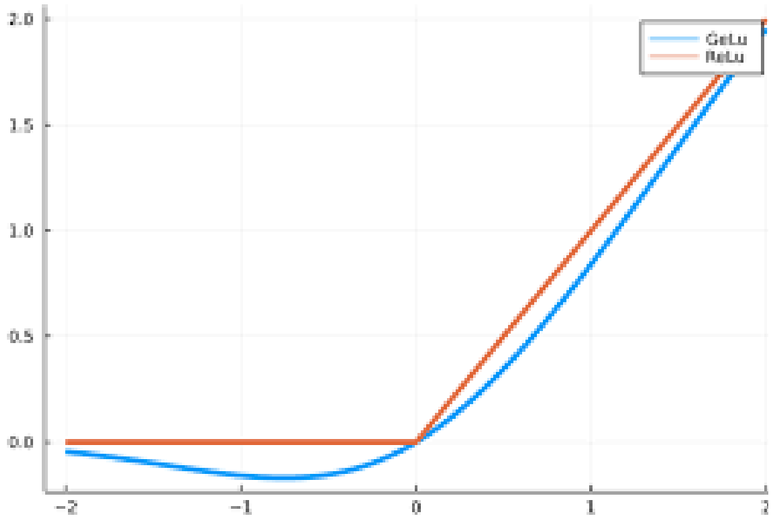
Position-wise Feed Forward Network(PFFN)

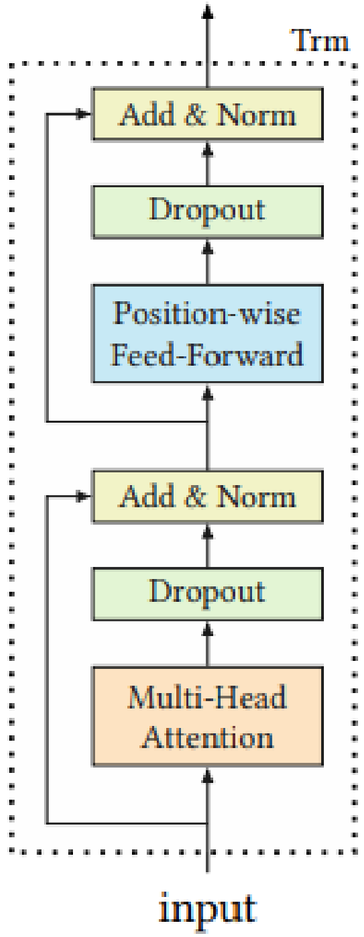
- 쉽게 말해, 앞에 멀티헤드로 각 특성을 학습시켰는데, 각 헤드의 정보를 섞어주는 역할
- BERT에서 제시된 GELU를 사용

$$\mathrm{PFFN}(\boldsymbol{H}^l) = \left[\mathrm{FFN}(\boldsymbol{h}_1^l)^\top; \ldots; \mathrm{FFN}(\boldsymbol{h}_t^l)^\top\right]^\top$$

$$\mathrm{FFN}(\boldsymbol{x}) = \mathrm{GELU}(\boldsymbol{x}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)})\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)}$$

$$\mathrm{GELU}(x) = x\Phi(x)$$





(a) Transformer Layer.

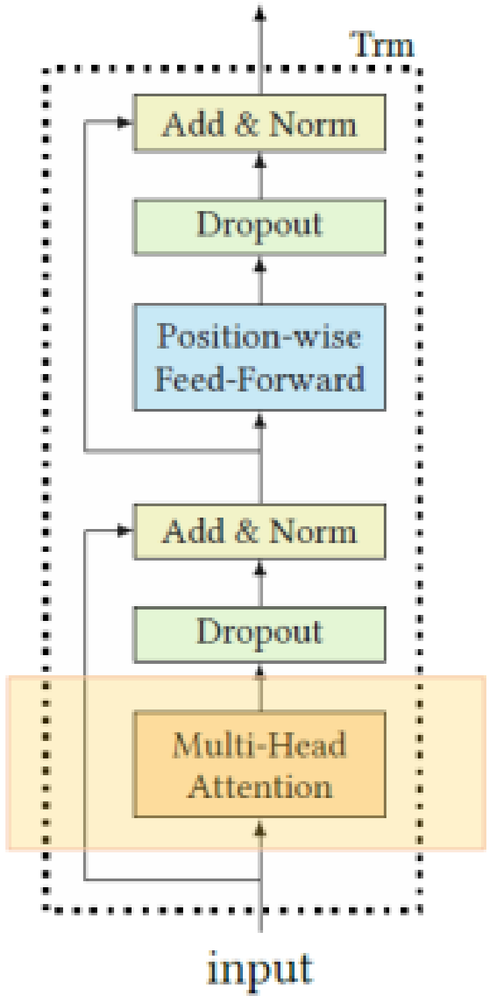
#3.4~3.5

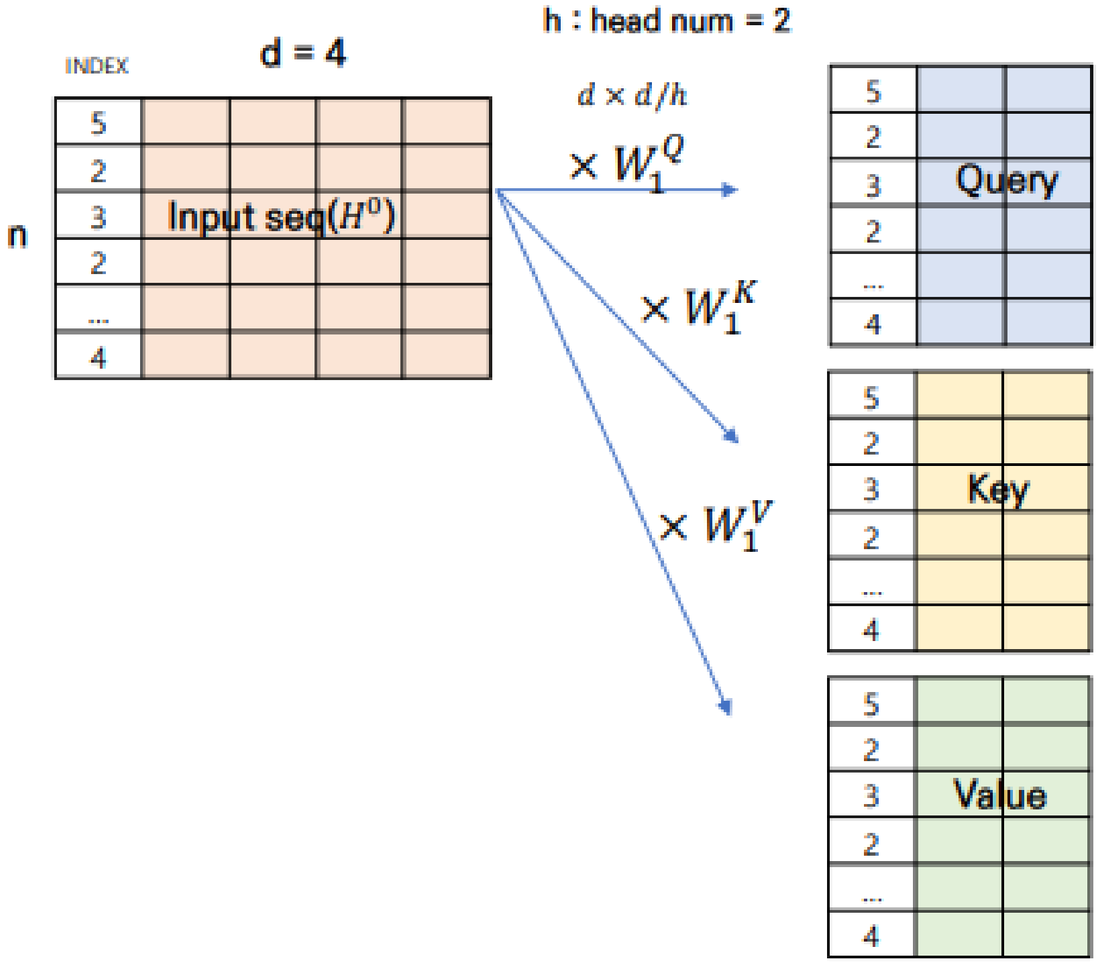
Embedding layer
- 아직 인풋 시퀀스를 모르는 상태. 순서 정보를 알려주기 위해 트렌스포머 레이어 마지막에 positional embedding

Output Layer
FNN과 비슷한 모양새. 다른점은 x대신에 h. 그리고 두번째 weight대신에 임베딩

$$P(v) = \text{softmax}\big(\text{GELU}(\boldsymbol{h}_t^L \boldsymbol{W}^P + \boldsymbol{b}^P)\boldsymbol{E}^\top + \boldsymbol{b}^O\big) \tag{7}$$

#정리

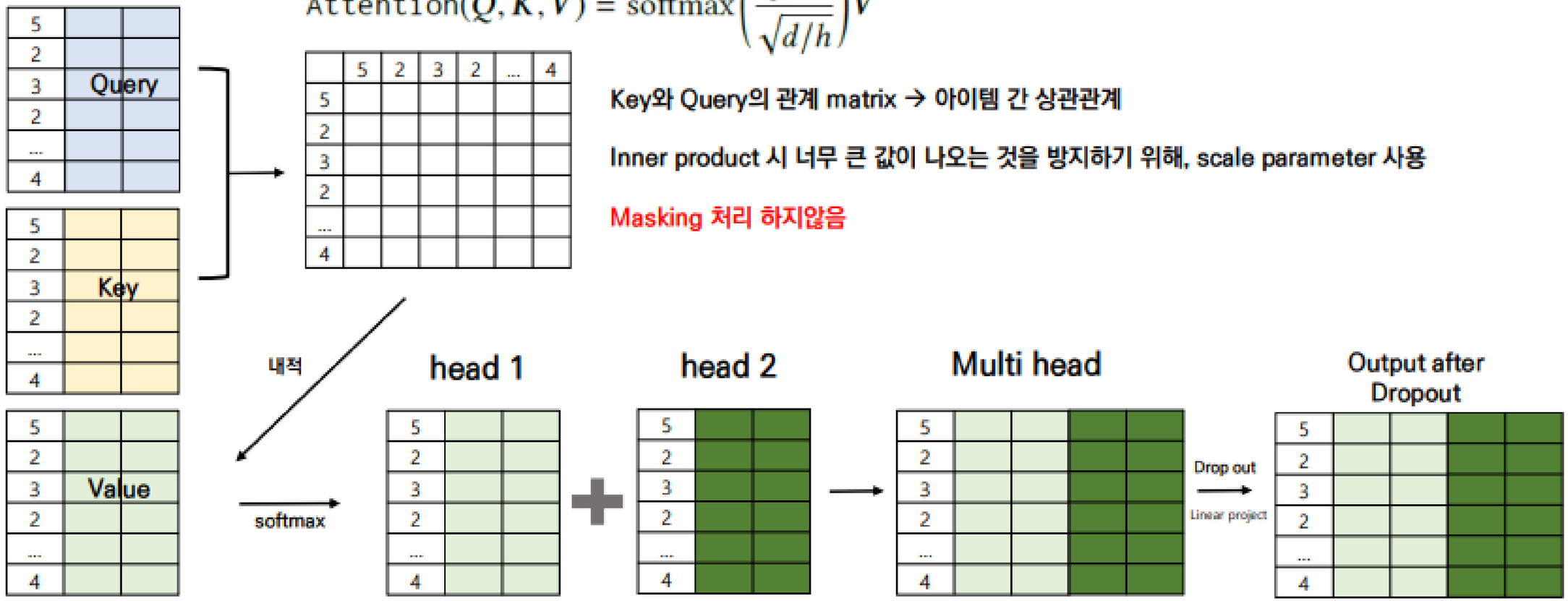




사진출처 : 고려대 DSBA 연구실 세미나 발표자료

#정리

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d/h}}\right)V$$



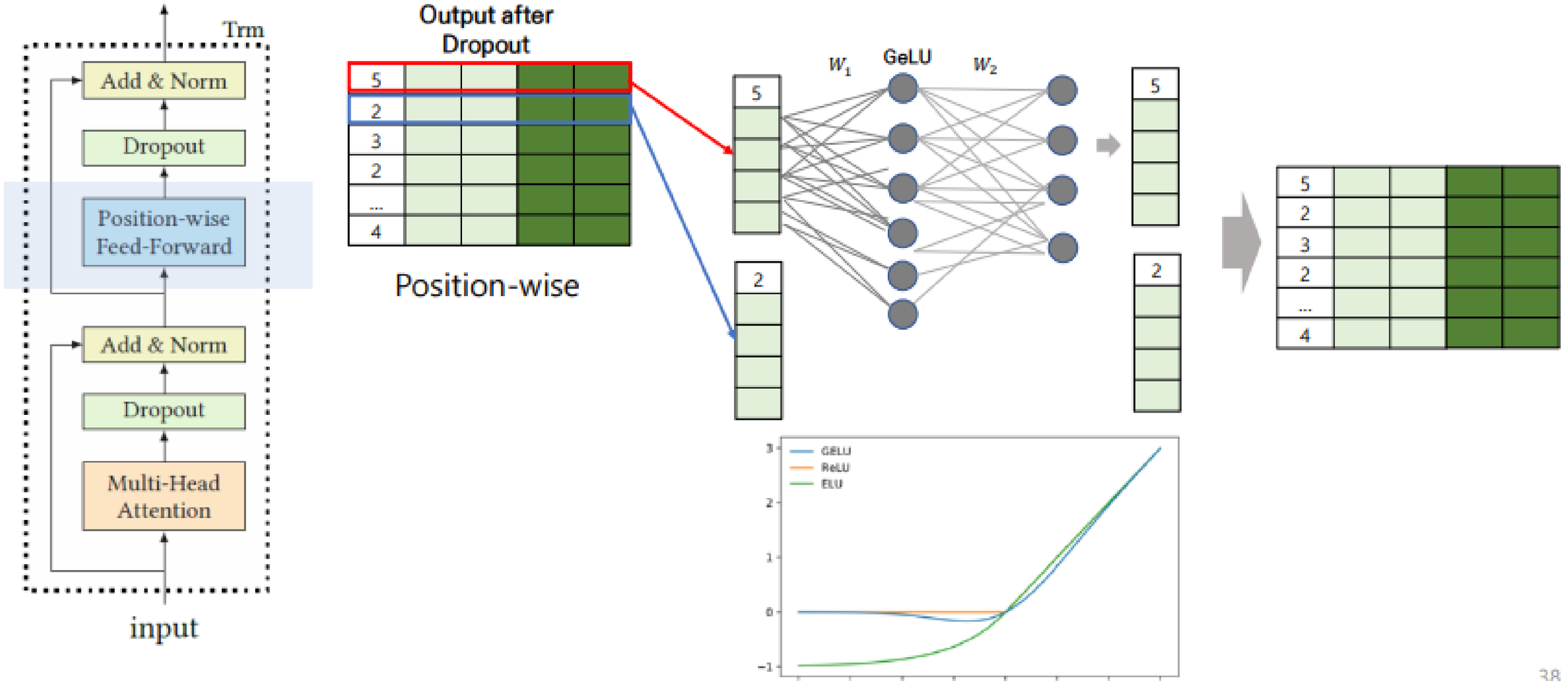
Key와 Query의 관계 matrix → 아이템 간 상관관계

Inner product 시 너무 큰 값이 나오는 것을 방지하기 위해, scale parameter 사용

Masking 처리 하지않음

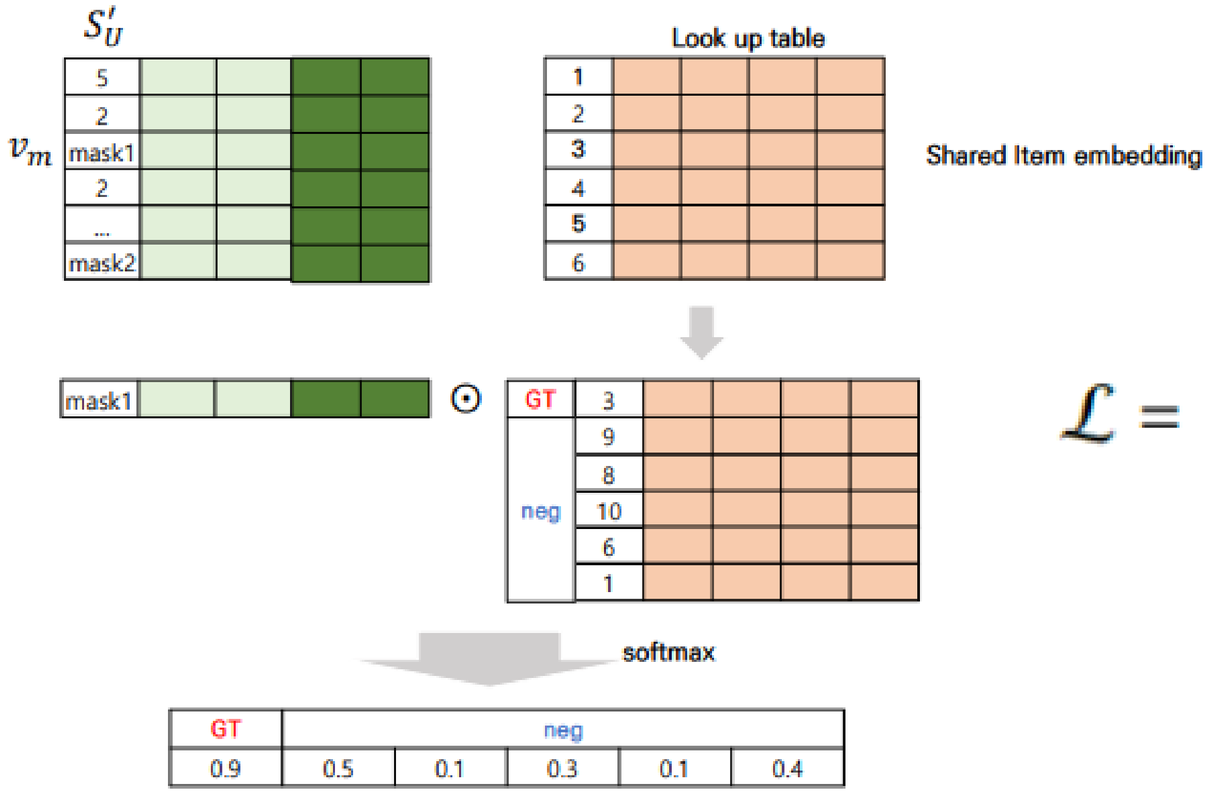
사진출처 : 고려대 DSBA 연구실 세미나 발표자료

#정리



사진출처 : 고려대 DSBA 연구실 세미나 발표자료

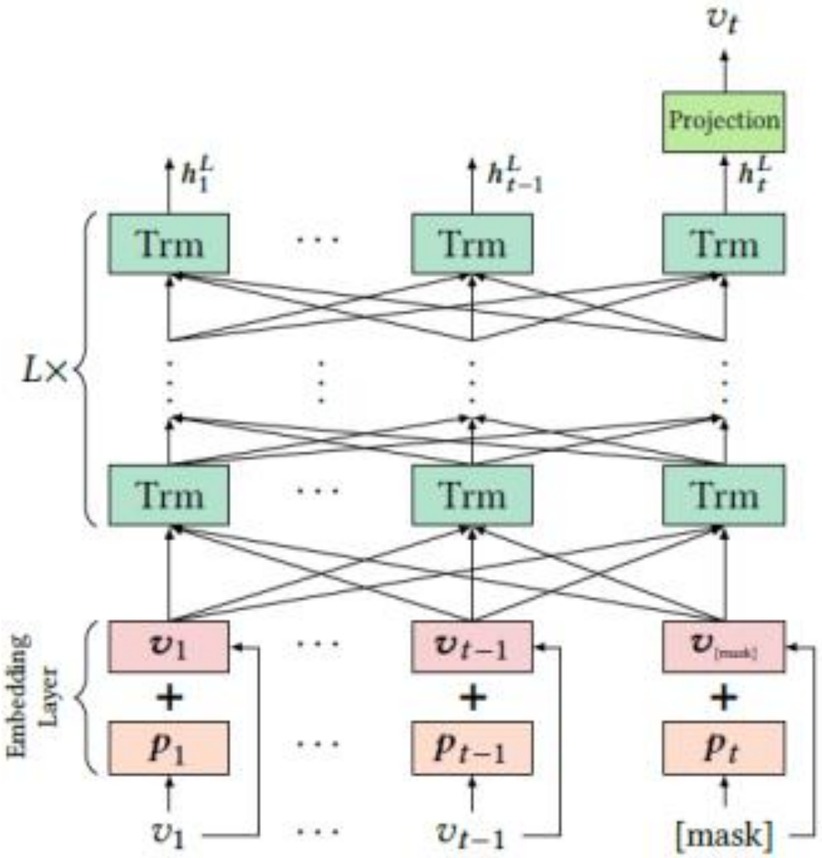
#정리



$$\mathcal{L} = \frac{1}{|S_u^m|} \sum_{v_m \in S_u^m} -\log P(v_m = v_m^* | S'_u)$$

사진출처 : 고려대 DSBA 연구실 세미나 발표자료

#정리



사진출처 : 고려대 DSBA 연구실 세미나 발표자료

#3.6 Model learning

Training

- ✓ Cloze task 사용. p(proportion)로 얼마만큼 마스킹 할건지 비율을 정함
- ✓ 다른 모델들은 1개씩 마스킹해서, nC1 = n개의 훈련샘플을 만들지만,
  이 방법은 k개씩 마스킹해서 nCk개를 만들기 때문에, 훈련샘플이 많아져서 성능이 좋아진다.

Test

- ✓ Cloze는 현재데이터 중에서 중간에 마스킹된 거를 맞추는 것인 반면 sequential 모델의 목표는 미래 값을 맞추는 거

-> 테스트를 위해서는 유저행동 시퀀스 맨 끝에 [mask]라는 토큰을 넣음

-> 그래서 성능 좋아지라고 훈련 값에 (nCk 뿐만 아니라) 끝에 값을 마스킹한 것도 추가함.

#3.7 Discussion

SASRec

- 트랜스포머 사용. 단방향. 그냥 어텐션 마스킹을 사용(n개)
- 시퀀스의 각 위치에 대한 다음 아이템을 예측
- 반면, BERT4Rec은 Cloze를 사용하여 시퀀스의 마스킹된 아이템을 예측

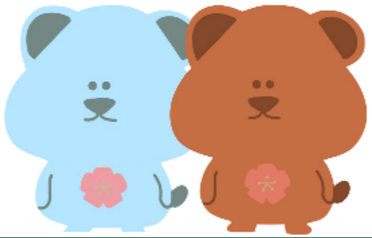
CBOW & SG

- CBOW : 컨텍스트의 모든 단어 벡터의 평균을 사용하여 목표 단어를 예측
- SG : 하나를 제외한 모든 아이템을 마스킹

BERT(와 BERT4Rec의 차이라고 생각하면 될 듯)

- 대용량 말뭉치(언어데이터!)를 이용한 문장 표현 모델 → 추천모델 목표와 맞지 않음
- BERT4Rec : 추천 데이터로 학습. next sentence loss 와 세그먼트 임베딩 제거
  - next sentence loss란? 다음 '문장' 에 대한 예측, 추천시스템은 다름. 한 유저와 그 다음 유저는 완전히 다른 사람임. 문법처럼 모든 문장에 적용되는 규칙을 사용할 수 없음

#04 Experiment



EWHA EURON

#4.1 Datasets

Table 1: Statistics of datasets.

Datasets	#users	#items	#actions	Avg. length	Density
Beauty	40,226	54,542	0.35m	8.8	0.02%
Steam	281,428	13,044	3.5m	12.4	0.10%
ML-1m	6040	3416	1.0m	163.5	4.79%
ML-20m	138,493	26,744	20m	144.4	0.54%

Sparse

Dense

- (1-5평점, 리뷰 유무 등) implicit feedback이 유무로 변경. 있으면 1로
- timestamps에 따라 정렬
- 5개 이상 피드백 있는 사용자데이터만 사용

#4.3 Baseline + 4.4 Overall Performances

- POP : 그냥 인기 TOP K로 추천. 생각보다 꽤 성능 좋음
- 그리고 앞에서 설명한 SOTA 모델들

Datasets	Metric	POP	BPR-MF	NCF	FPMC	GRU4Rec	GRU4Rec$^+$	Caser	SASRec	BERT4Rec	Improv.
Beauty	HR@1	0.0077	0.0415	0.0407	0.0435	0.0402	0.0551	0.0475	0.0906	**0.0953**	5.19%
	HR@5	0.0392	0.1209	0.1305	0.1387	0.1315	0.1781	0.1625	0.1934	**0.2207**	14.12%
	HR@10	0.0762	0.1992	0.2142	0.2401	0.2343	0.2654	0.2590	0.2653	**0.3025**	14.02%
	NDCG@5	0.0230	0.0814	0.0855	0.0902	0.0812	0.1172	0.1050	0.1436	**0.1599**	11.35%
	NDCG@10	0.0349	0.1064	0.1124	0.1211	0.1074	0.1453	0.1360	0.1633	**0.1862**	14.02%
	MRR	0.0437	0.1006	0.1043	0.1056	0.1023	0.1299	0.1205	0.1536	**0.1701**	10.74%
Steam	HR@1	0.0159	0.0314	0.0246	0.0358	0.0574	0.0812	0.0495	0.0885	**0.0957**	8.14%
	HR@5	0.0805	0.1177	0.1203	0.1517	0.2171	0.2391	0.1766	0.2559	**0.2710**	5.90%
	HR@10	0.1389	0.1993	0.2169	0.2551	0.3313	0.3594	0.2870	0.3783	**0.4013**	6.08%
	NDCG@5	0.0477	0.0744	0.0717	0.0945	0.1370	0.1613	0.1131	0.1727	**0.1842**	6.66%
	NDCG@10	0.0665	0.1005	0.1026	0.1283	0.1802	0.2053	0.1484	0.2147	**0.2261**	5.31%
	MRR	0.0669	0.0942	0.0932	0.1139	0.1420	0.1757	0.1305	0.1874	**0.1949**	4.00%
ML-1m	HR@1	0.0141	0.0914	0.0397	0.1386	0.1583	0.2092	0.2194	0.2351	**0.2863**	21.78%
	HR@5	0.0715	0.2866	0.1932	0.4297	0.4673	0.5103	0.5353	0.5434	**0.5876**	8.13%
	HR@10	0.1358	0.4301	0.3477	0.5946	0.6207	0.6351	0.6692	0.6629	**0.6970**	4.15%
	NDCG@5	0.0416	0.1903	0.1146	0.2885	0.3196	0.3705	0.3832	0.3980	**0.4454**	11.91%
	NDCG@10	0.0621	0.2365	0.1640	0.3439	0.3627	0.4064	0.4268	0.4368	**0.4818**	10.32%
	MRR	0.0627	0.2009	0.1358	0.2891	0.3041	0.3462	0.3648	0.3790	**0.4254**	12.24%
ML-20m	HR@1	0.0221	0.0553	0.0231	0.1079	0.1459	0.2021	0.1232	0.2544	**0.3440**	35.22%
	HR@5	0.0805	0.2128	0.1358	0.3601	0.4657	0.5118	0.3804	0.5727	**0.6323**	10.41%
	HR@10	0.1378	0.3538	0.2922	0.5201	0.5844	0.6524	0.5427	0.7136	**0.7473**	4.72%
	NDCG@5	0.0511	0.1332	0.0771	0.2239	0.3090	0.3630	0.2538	0.4208	**0.4967**	18.04%
	NDCG@10	0.0695	0.1786	0.1271	0.2895	0.3637	0.4087	0.3062	0.4665	**0.5340**	14.47%
	MRR	0.0709	0.1503	0.1072	0.2273	0.2967	0.3476	0.2529	0.4026	**0.4785**	18.85%

#4.4 Overall Performances

- **Question1 : BERT4Rec의 이 성능이 양방향에서 오는지 Cloze에서 오는지 모름**
  - Cloze가 여러 개를 mask하는 거였으니, 하나만 마스크 해보자.
  - 위에 두 행을 보면 알겠지만, 그래도 SASRec보다 나음. 하지만 원래대로 Cloze(여러개 마스킹)하는게 성능이 제일 좋긴 함

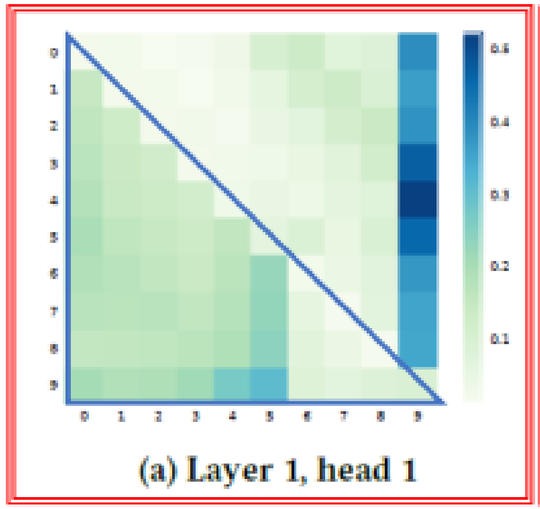
Table 3: Analysis on bidirection and Cloze with $d = 256$.

Model	Beauty			ML-1m		
	HR@10	NDCG@10	MRR	HR@10	NDCG@10	MRR
SASRec	0.2653	0.1633	0.1536	0.6629	0.4368	0.3790
BERT4Rec (1 mask)	0.2940	0.1769	0.1618	0.6869	0.4696	0.4127
BERT4Rec	0.3025	0.1862	0.1701	0.6970	0.4818	0.4254

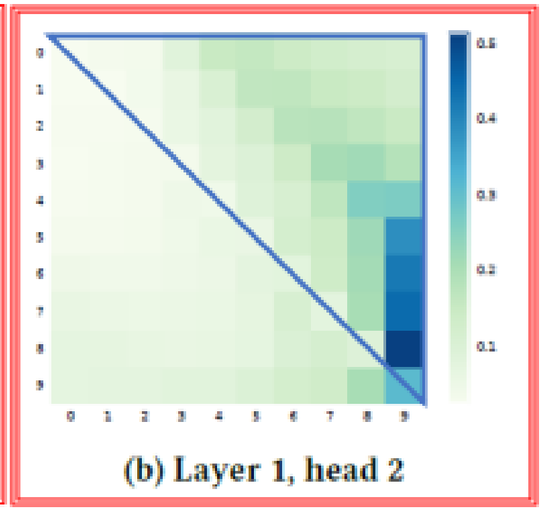
#4.4 Overall Performances

Question2 : 왜 양방향이 단방향 보다 성능이 좋을까

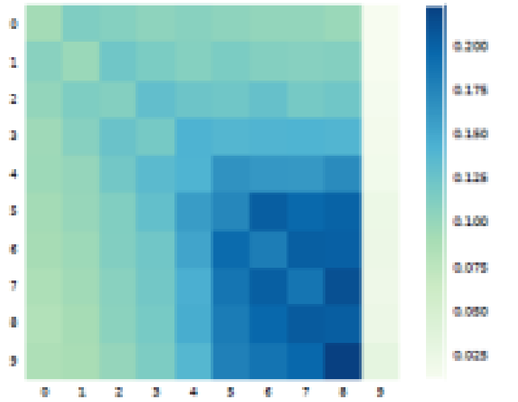
- attention weight을 히트맵으로 그려봄



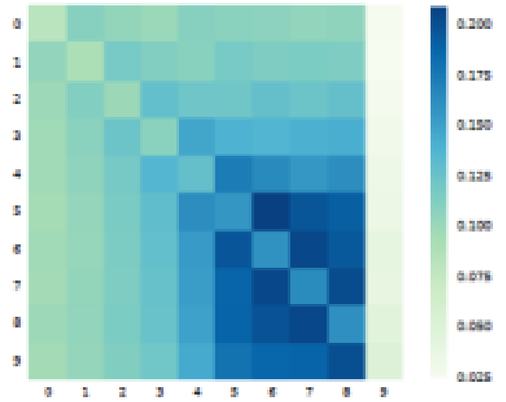
(a) Layer 1, head 1



(b) Layer 1, head 2



(c) Layer 2, head 2



(d) Layer 2, head 4

Head 1 : Item 간의 관계를 학습 시, 왼쪽 부분에 더 집중

Head 2 : Item 간의 관계를 학습 시, 오른쪽 부분에 더 집중

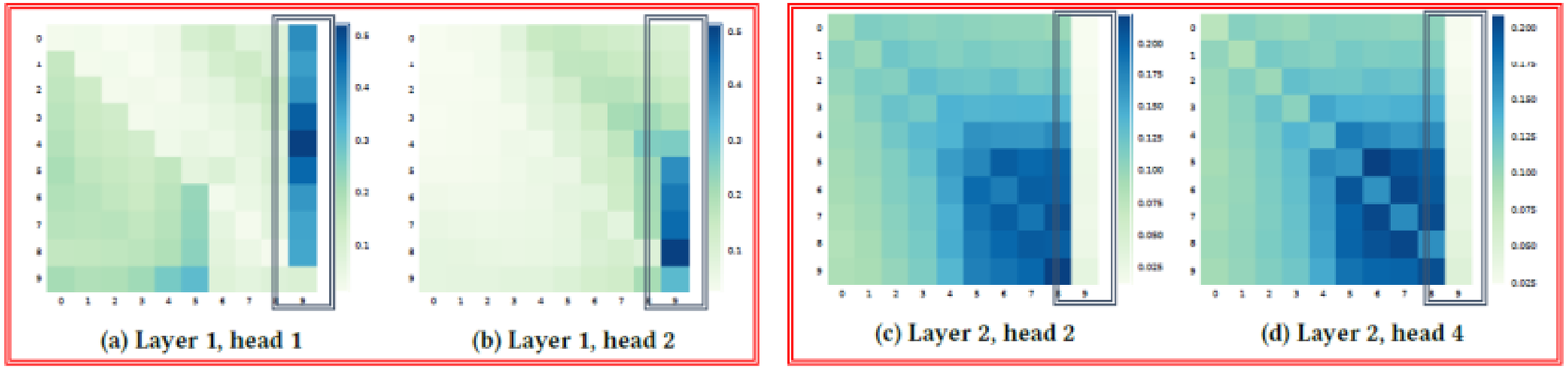
CNN Filter map 이 서로 다른 특징을 잡아 내는것과 유사

사진출처 : 고려대 DSBA 연구실 세미나 발표자료

#4.4 Overall Performances

Question2 : 왜 양방향이 단방향 보다 성능이 좋을까

- attention weight을 히트맵으로 그려봄



Layer 1 : Masked Item 과의 관계에 좀 더 집중

Layer 2 : Masked 와의 관계 정보를 기반으로, 예측에 필요한 최근 아이템에 좀 더 집중 (Prediction layer 와 직접연결)

CNN Layer 에서의 원리와 유사하며, 예측에 영향을 미치는 중요(세부) 정보는 최근 아이템이 때문

사진출처 : 고려대 DSBA 연구실 세미나 발표자료

#4.5 Impact of hidden dimensionality d

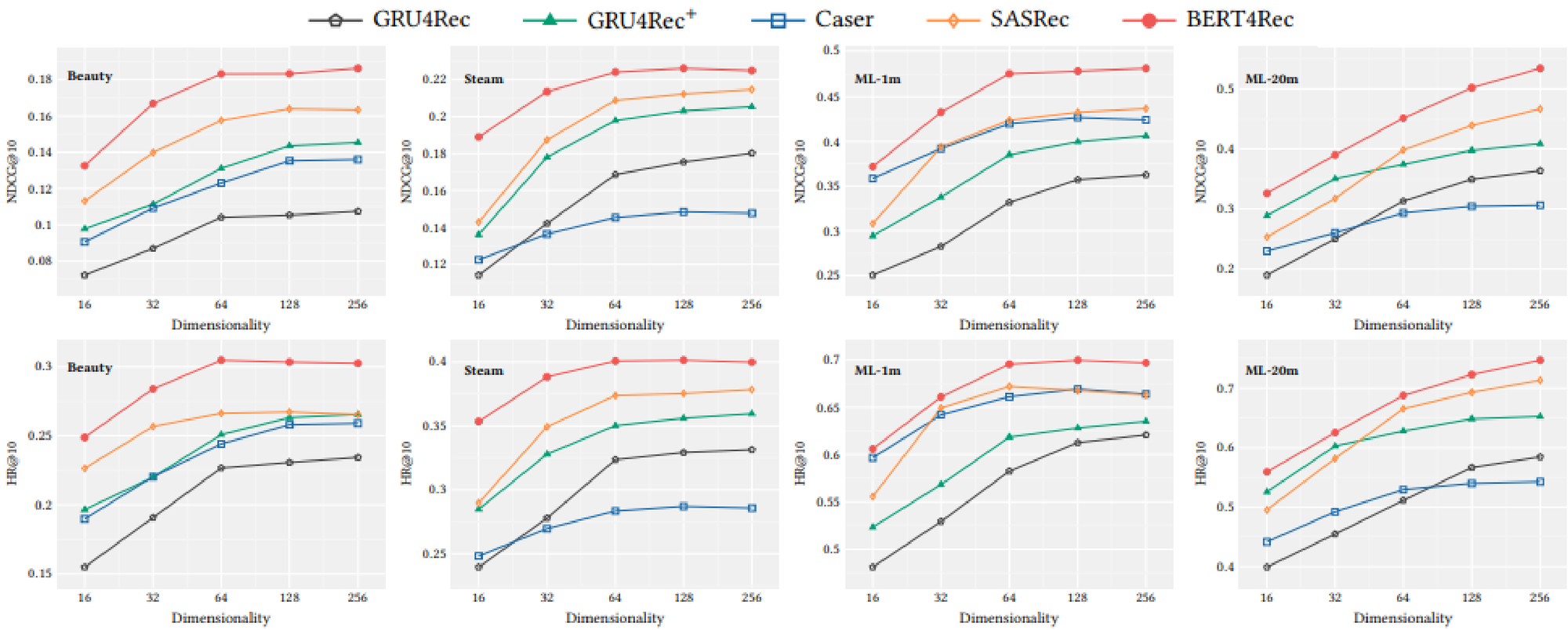


Figure 3: Effect of the hidden dimensionality d on HR@10 and NDCG@10 for neural sequential models.

- 보통 64, 128쯤이 max다.
- 높은 d(차원수)는 항상 좋은 성능을 내는데 필요한 건 아니다.
  Sparse 데이터인 Beauty랑 Steam에서 알 수 있다.
- Dense한 데이터일 수록 d는 커져야 한다.
- BERT4Rec는 항상 성능 좋다.

#4.6 Impact of mask proportion p

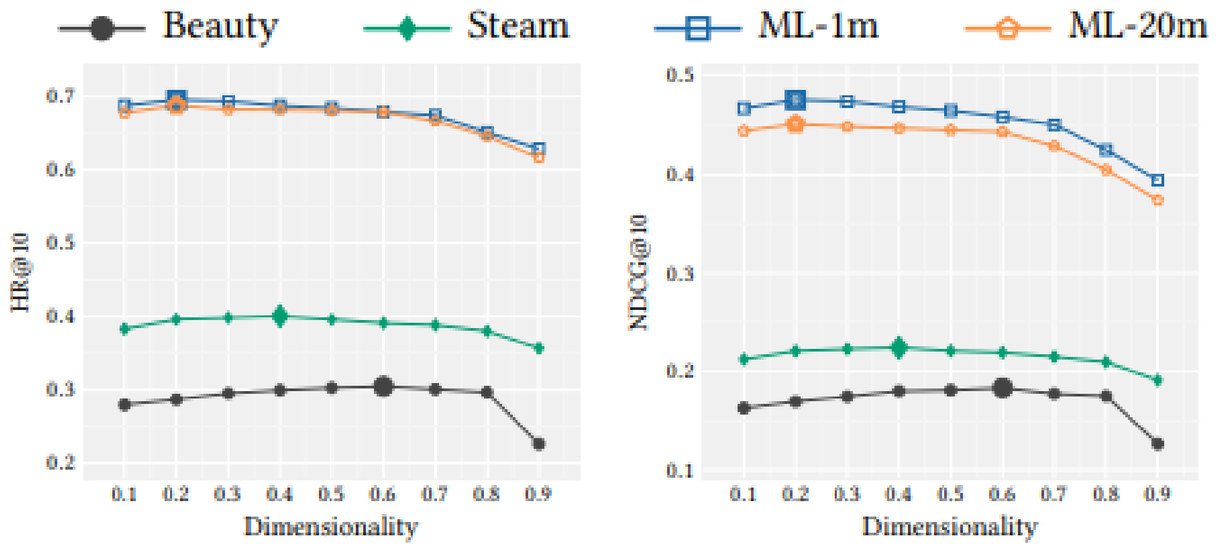


Figure 4: Performance with different mask proportion ρ on $d = 64$. Bold symbols denote the best scores in each line.

그래프 가로축 Dimension 아니고 Proportion인 듯

- p는 너무 작아도 너무 커도 안된다.
- p는 데이터의 length 에 따라 조절해줘야 한다.
  length가 작은 steam이나 beauty는 0.4, 0.6일 때 max. 나머지는 0.2일 때 max

#4.7 Impact of maximum sequence length N

Table 4: Performance with different maximum length N.

		10	20	30	40	50
Beauty	#samples/s	5504	3256	2284	1776	1441
	HR@10	0.3006	0.3061	0.3057	0.3054	0.3047
	NDCG@10	0.1826	0.1875	0.1837	0.1833	0.1832
		10	50	100	200	400
ML-1m	#samples/s	14255	8890	5711	2918	1213
	HR@10	0.6788	0.6854	0.6947	0.6955	0.6898
	NDCG@10	0.4631	0.4743	0.4758	0.4759	0.4715

- 이것도 데이터에 따라 max 다름. Beauty는 20일 때 max. ML은 200일 때 max
- 그러나 BERT4Rec모델은 전반적으로 안정적으로 잘 작동

#4.8 Ablation Study

Architecture	Dataset			
	Beauty	Steam	ML-1m	ML-20m
$L = 2, h = 2$	0.1832	0.2241	0.4759	0.4513
w/o PE	0.1741	0.2060	0.2155↓	0.2867↓
w/o PFFN	0.1803	0.2137	0.4544	0.4296
w/o LN	0.1642↓	0.2058	0.4334	0.4186
w/o RC	0.1619↓	0.2193	0.4643	0.4483
w/o Dropout	0.1658	0.2185	0.4553	0.4471
1 layer ($L = 1$)	0.1782	0.2122	0.4412	0.4238
3 layers ($L = 3$)	**0.1859**	**0.2262**	**0.4864**	**0.4661**
4 layers ($L = 4$)	**0.1834**	**0.2279**	**0.4898**	**0.4732**
1 head ($h = 1$)	**0.1853**	0.2187	0.4568	0.4402
4 heads ($h = 4$)	0.1830	**0.2245**	**0.4770**	**0.4520**
8 heads ($h = 8$)	0.1823	**0.2248**	0.4743	**0.4550**

컴포넌트를 하나씩 없애면서 실험해 봄

- L=2, h=2 행이 모든 컴포넌트가 다 있는, 디폴트 버전
- PE는 long한 데이터에서 아주 성 떨어짐
- LN, RC, Droupout은 오버피팅 방지용이라서, 확실히 작은 데이터인 Beauty 에서 성능 떨어짐
- L(레이어 개수) : 깊을수록 성능 좋음
- H는 long한 데이터일수록 헤드 많은 게 좋고, short 데이터일수록 헤드 적은 게 좋음

#05 Conclusions과 논문관련사항

EWHA EURON

#05 Conclusion과 논문관련 기타사항

- Conclusion
  - BERT를 활용한 양방향 추천 모델을 고안
  - Cloze를 도입하여 성능 향상

#05 Conclusion과 논문관련 기타사항

- 재현성 논란

[2022'Recsys] A Systematic Review and Replicability Study of BERT4Rec for Sequential Recommendation

우리가 읽은 논문은 어휘의 순서가 중요한 NLP 분야에서 BERT가 뜨자, 고객 행동 순서를 모형화할 때도 BERT를 응용해 보자~는 취지 인데, 위 논문은 GPT를 이용한 SASRec이 더 좋을 때도 있다고 주장

Table 1: Results of BERT4Rec vs SASRec comparisons in the peer-reviewed publications. Bold denotes model with more wins on a dataset. Asterisk (*) denotes datasets used in the original BERT4Rec paper [52]. Only the datasets with appearing in at least 5 papers are presented (8 datasets out of 46), however the comparison results on all other 38 datasets are included in the "Total" numbers.

dataset	total	BERT4Rec wins	SASRec wins	Ties	BERT4Rec wins papers	SASRec wins papers	Ties papers
Beauty* [16]	19	**12 (63%)**	5 (26%)	2 (11%)	[52], [40], [68], [38], [29], [7], [64], [55], [62], [18], [60], [53]	[69], [32], [65], [58], [47]	[3], [9]
ML-1M* [14]	18	**13 (72%)**	3 (17%)	2 (11%)	[52], [40], [68], [29], [7], [23], [55], [18], [60], [51], [42], [46], [28]	[12], [62], [47]	[64], [9]
Yelp [2]	10	**6 (60%)**	4 (40%)	0 (0%)	[69], [1], [3], [58], [47], [42]	[12], [32], [65], [33]	
Steam* [43]	8	**7 (88%)**	1 (12%)	0 (0%)	[52], [40], [68], [29], [64], [9], [60]	[62]	
ML-20M* [14]	8	**7 (88%)**	0 (0%)	1 (12%)	[52], [40], [66], [29], [7], [60], [46]		[9]
Sports [16]	6	1 (17%)	**4 (67%)**	1 (17%)	[28]	[69], [32], [65], [47]	[3]
LastFM [5]	6	**4 (67%)**	2 (33%)	0 (0%)	[69], [23], [55], [28]	[1], [65]	
Toys [16]	5	0 (0%)	**5 (100%)**	0 (0%)		[69], [13], [32], [65], [3]	
Total	134	**86 (64%)**	32 (23 %)	16 (12 %)			

BERT4REC과 SASRec을 다룬 논문들을 비교함.
논문이 몇 개인지, 성능이 어떤 게 좋은지 등
결론 : **항상 BERT4Rec이 좋은 건 아니다.**

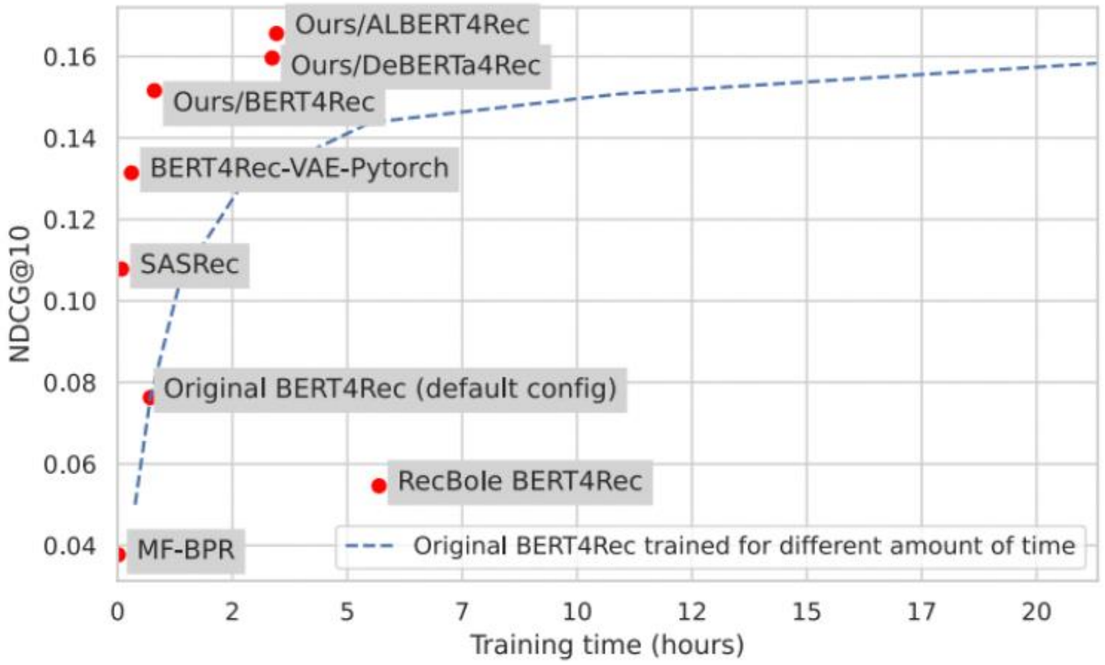
출처 : https://mytype9591.tistory.com/6

#05 Conclusion과 논문관련 기타사항

- 재현성 논란

[2022'Recsys] A Systematic Review and Replicability Study of BERT4Rec for Sequential Recommendation

우리가 읽은 논문은 어휘의 순서가 중요한 NLP 분야에서 BERT가 뜨자, 고객 행동 순서를 모형화할 때도 BERT를 응용해 보자~는 취지 인데, 위 논문은 GPT를 이용한 SASRec이 더 좋을 때도 있다고 주장

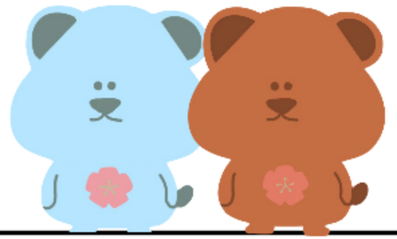


SASRec이 훨씬 빨리 좋은 성능에 도달하는 것을 알 수 있음

오늘 읽은 논문에서 언급된 속도보다 실제로는 30배 더 걸린다고 언급

출처 : https://mytype9591.tistory.com/6

THANK YOU



EWHA EURON