

BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer

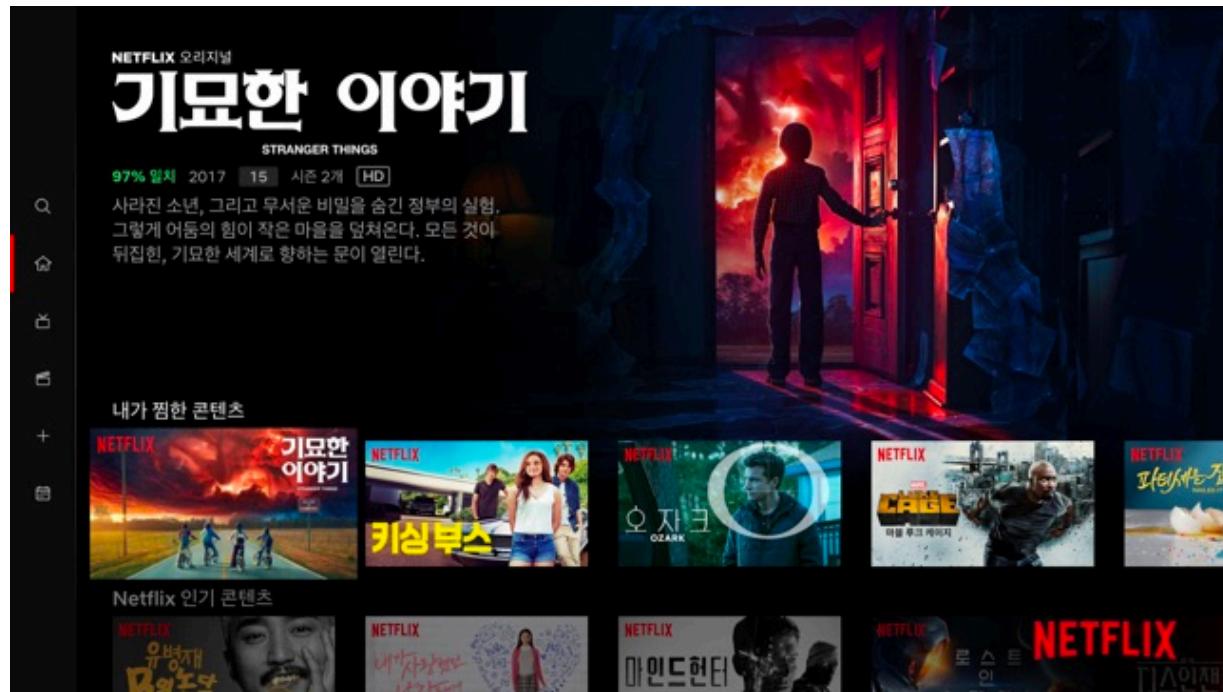
ICIK 2019

Recommend system

01. Recommend System

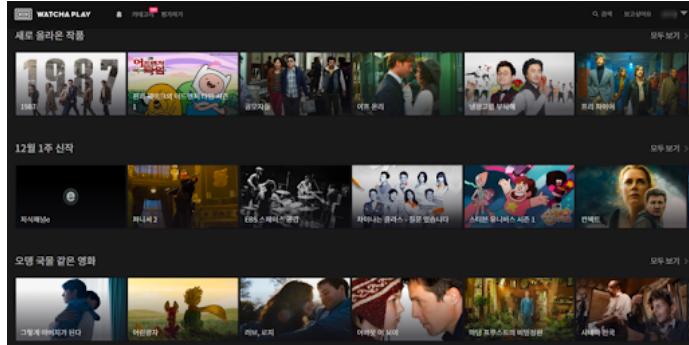
(1) Recommend system이란

추천시스템 이란? 유저가 좋아할 만한 아이템을 추천하는 것

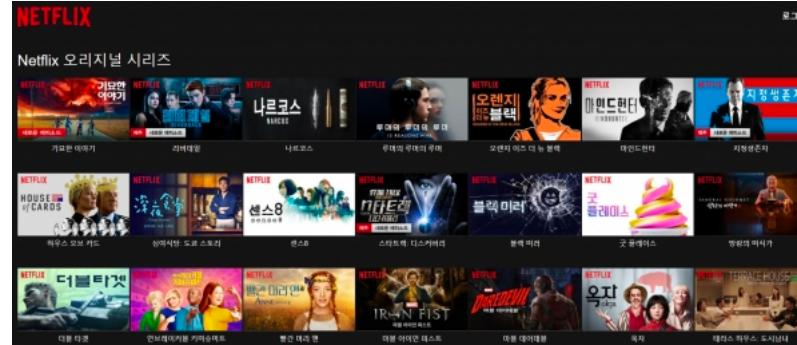


01. Recommend System

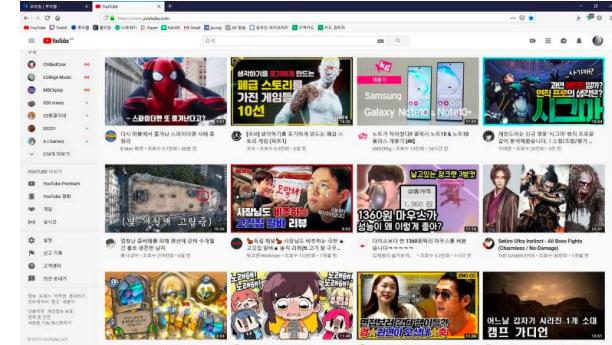
(2) Recommend system을 사용하는 예시



왓챠



넷플릭스



유튜브

아마존 : 상품 판매의 1/3이상이 추천에 의해 발생

Google News : 1/3이상의 조회가 추천에 의해 발생

넷플릭스 : 시청 상품의 ¾이상이 추천에 의해 발생.

Netflix Prize 대회에서 10%이상 성능 올리는 팀에게 1,000,000달러(약 12억).

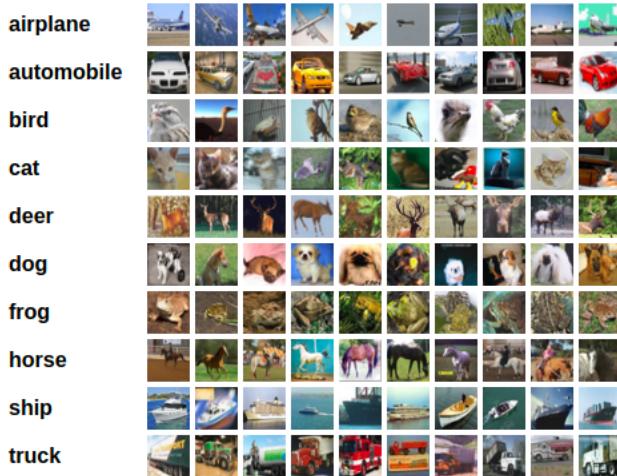
SVD(Singular Value Decomposition), ensemble의 중요성 등이 부각

페이스북 : 친구추천. 친구추천은 서로의 교류를 증대 시킬 목적을 가지고 있기 때문에 Link Prediction이라는 것에 중점을 두고 있다.

Related Works

02. Related Works

(1) 추천 시스템의 시작 : Interaction matrix



- [POS] 오늘 점심 맛있었다
- [NEG] 아 출근하기 싫어
- [POS] 학습을 돌렸는데 좋은 성능이 나왔다
- [POS] 큐 카페의 바닐라 라떼가 제일 좋아!
- [NEG] 비 작작 좀 와라—
- [POS] 이번주 보고는 메일 보고입니다~~ 아싸
- [NEG] 뭘 잘못했는데?
- [POS] 다만 악에서 구하소서 짱谮쩔谮존잼!

추천시스템의 데이터는 어떻게 생성하는가?

02. Related Works

(1) 추천 시스템의 시작 : Interaction matrix

	Harry Potter	The Triplets of Belleville	Shrek	The Dark Knight Rises	Memento
1	✓		✓	✓	
2		✓			✓
3	✓	✓	✓		
4				✓	✓

어떤 유저가 어떤 아이템에 대해 어떤 피드백을 남겼는가

: 어떠한 유저가 어떠한 아이템을 선택했으니, 그 다음엔 어떠한 아이템을 선호할 것이다

02. Related Works

(1) 추천 시스템의 시작 : Interaction matrix



봤다 안봤다 / 0-5점 ratings / click 유무

어떤 유저가 어떤 아이템에 대해 어떤 피드백을 남겼는가

: 어떠한 유저가 어떠한 아이템을 선택했으니, 그 다음엔 어떠한 아이템을 선호할 것이다

02. Related Works

(1) 추천 시스템의 시작 : Interaction matrix



저차원 feature -> 유사한 유저가 소비한 아이템을 추천

02. Related Works

(2) Matrix factorization의 등장



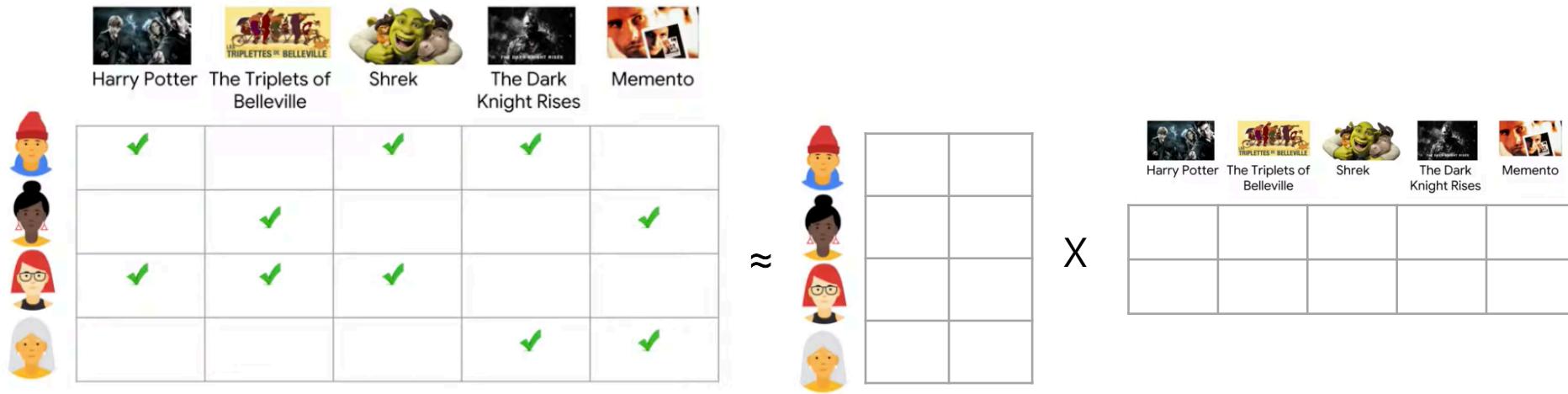
보통은 user, item에 대한 수가 너무 큼

sparse한 유저-아이템 매트리스를 저차원의 유저-아이템 매트리스로 분해

-> 딥러닝에 적용

02. Related Works

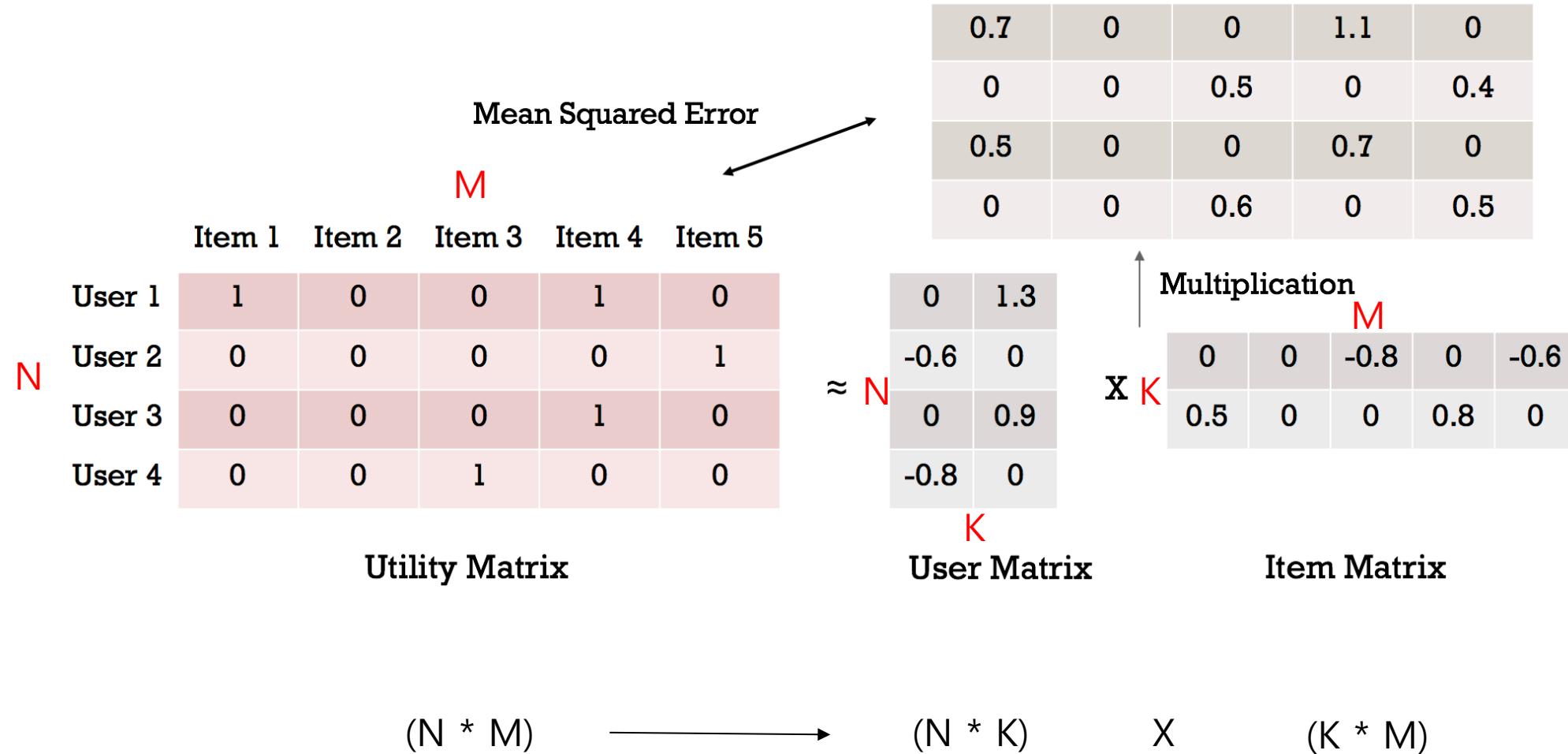
(3) NCF(Neural collaborate filtering)



User-item feedback을 바탕으로 구축한 user-item을 곱으로 표현

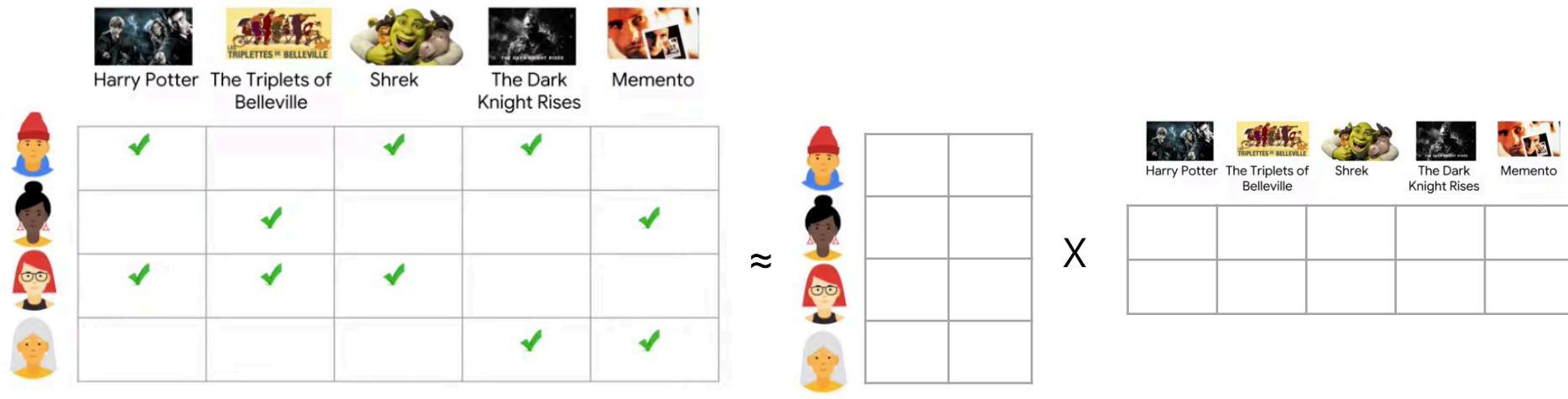
02. Related Works

(3) NCF(Neural collaborate filtering)



02. Related Works

(3) NCF(Neural collaborate filtering)



이를 바탕으로 특정 유저, 아이템에 대한 K차원의 **dense한 feature**를 뽑을 수 있다

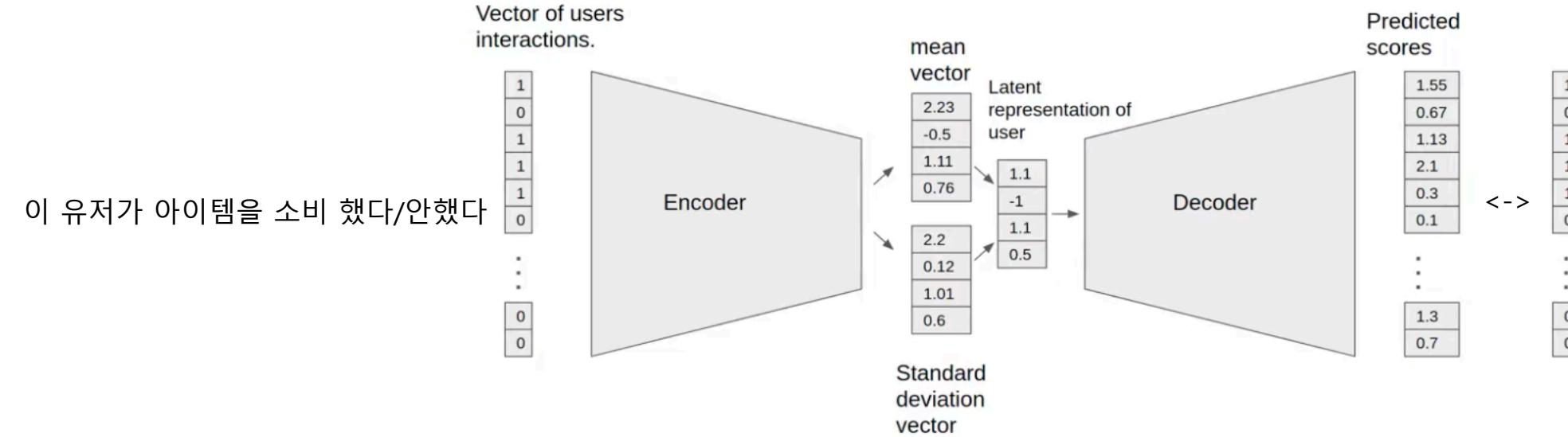
02. Related Works

(4) 오토인코더

유저가 전체 아이템에 대해 어떠한 피드백을 줬는지를 다시 복원

인코더, 디코더를 통해 다시 복원하면서 loss를 통해 중간에 dense한 피쳐를 학습

Movie lense라는 Dataset에서 가장 좋은 성능이 이 VAE 방식



02. Related Works

(5) 지난 모델들의 한계

유저가 아이템을 어떠한 순서로 소비했는지
최근에 소비한 아이템은 무엇인지를 알 수 없다

Case1)

지난 11개월 간의 소영소영 : 아무튼 행복한 나머지 90편의 로맨스 영화를 봄

최근 1개월 간의 소영소영 : 연구 스트레스를 받은 나머지 공포영화 10편을 봄

-> 다음으로 추천할 영화는?

Case2)

황금 연휴, 마블 시리즈 10편을 달리기로 한 소영소영

현재까지 3편까지 봄

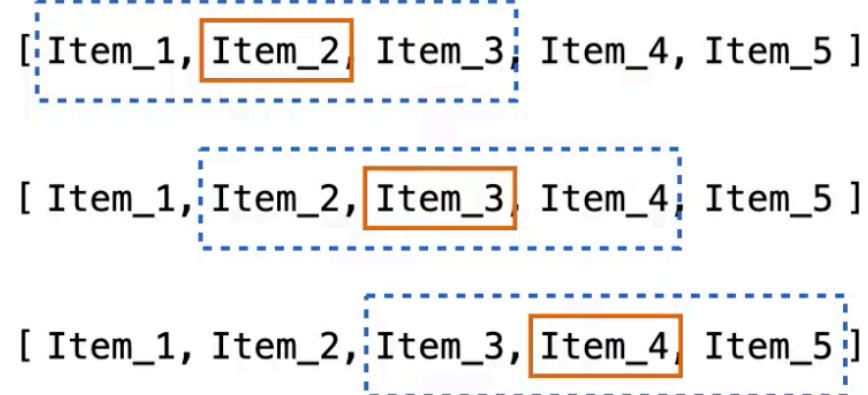
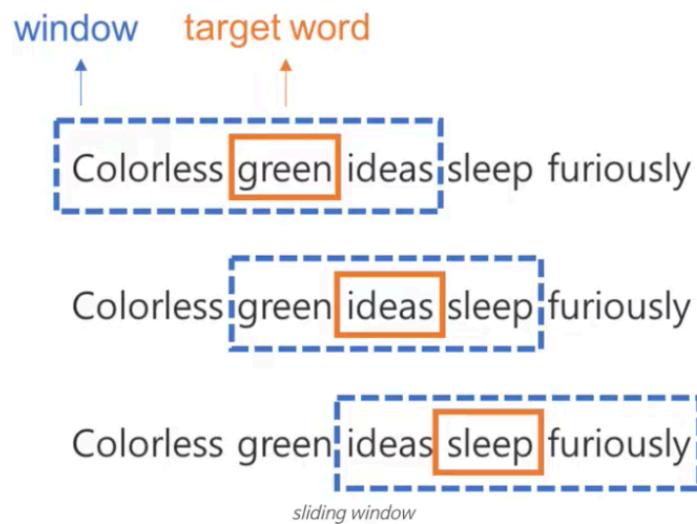
-> 눈치껏 다음으로 추천할 영화는?

Sequential Recommendation

03. Sequential Recommendation

(1) Idea : NLP처럼 생각해볼까!

Word2vec : 하나의 token으로 이루어진 sequence가 있을 때 token을 학습
token 대신 item의 id로 쓰면 -> item의 feature vector를 학습

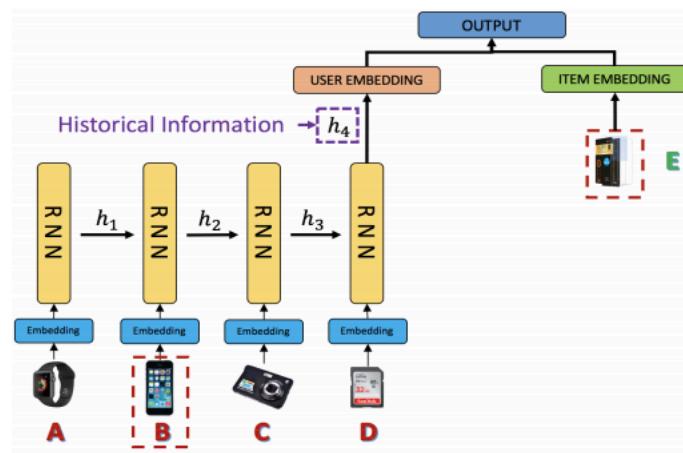


03. Sequential Recommendation

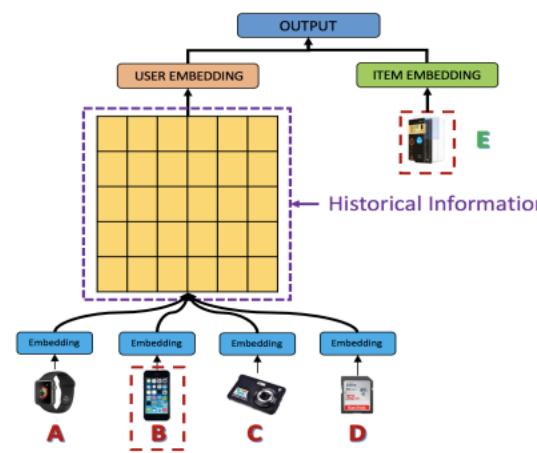
(1) Idea : NLP처럼 생각해볼까!

RNN방식 적용

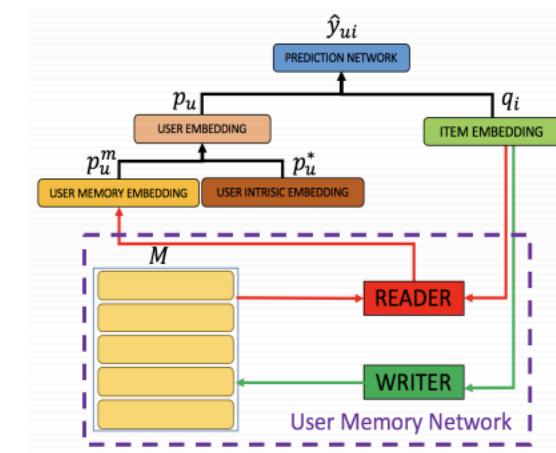
- 유저의 history를 보고 ABCD 봤을 때 recommendation model 과 같이 다음 아이템이 뭔지 찾는 방식
- Token 대신 아이템 집어넣고 문장 대신 유저의 Action 을 집어넣음



(a) Sequential recommendation based on RNN



(b) General idea to adopt memory for recommendation



(c) The general framework to implement our idea

BERT4Rec

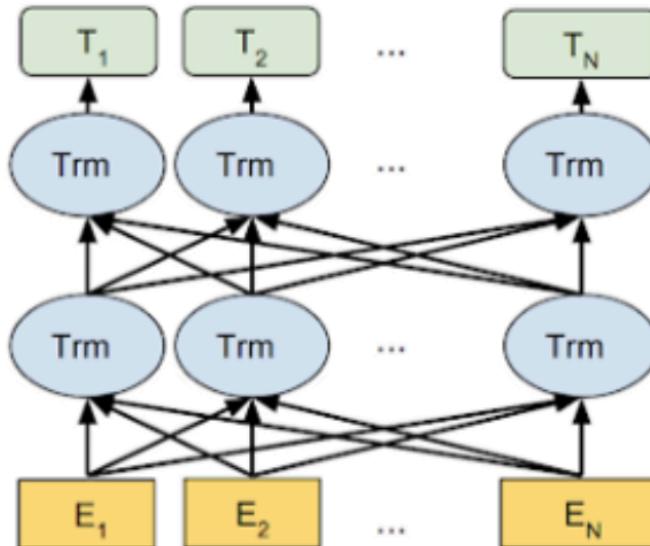
04. BERT4Rec

(1) BERT

BERT

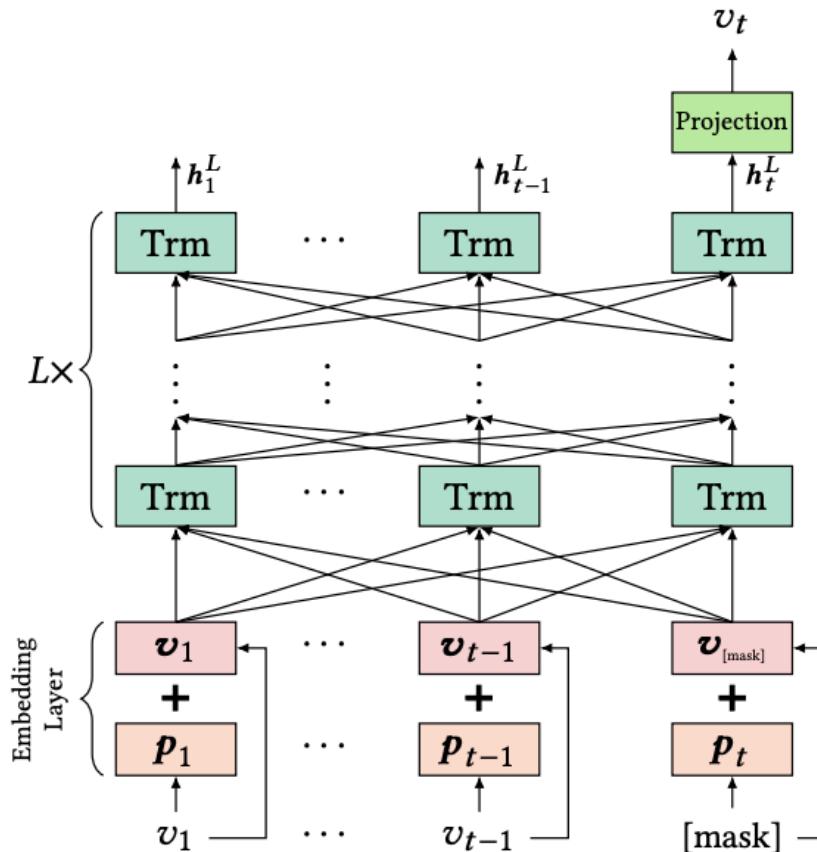
맨 마지막 단어를 예측하는게 아닌 Masking한 단어를 학습하는 구조

전체 Sequence의 표현을 학습하기엔 좋지만 sequential recommendation처럼 **다음 아이템 예측**엔 적합하지 않다



04. BERT4Rec

(2) BERT4Rec



(b) BERT4Rec model architecture.

- BERT의 Next sentence loss, segment embedding 사용하지 않음
- Masked 단어에서 Loss 계산
- End to end 형식으로 변경

Experiments

05. Experiments

(1) Dataset

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%

Density?

유저아이템의 수에 비해 interaction이 얼마나 있는가
작을수록 유저의 아이템은 많은데 interaction이 적다는 것

05. Experiments

(2) Metric

HR@k

내가 추천한 상위 k개 아이템 중 next item이 있는 비율

NDCG@k

내가 추천한 아이템들의 상위 점수/k개 아이템의 이상적 상위 점수

MRR

GT는 몇 순위에 있는가 (위치 별로 점수 획득)

05. Experiments

(3) Baseline

POP

: 전체 아이템을 인기 탑K개를 추천 / 예상하는 것보단 성능 좋음

BPR-MF

: (분해한거) positive, negative feedback를 pairwise ranking loss 사용

NCF

: MF기반에 dot product 대신 NLP로 학습

FPMC

: 마르코프 체인 적용

GRU4Rec

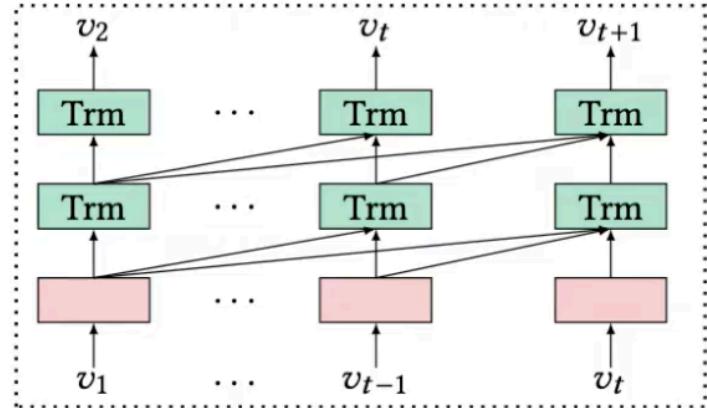
: GRU에 넣고 학습

Caser

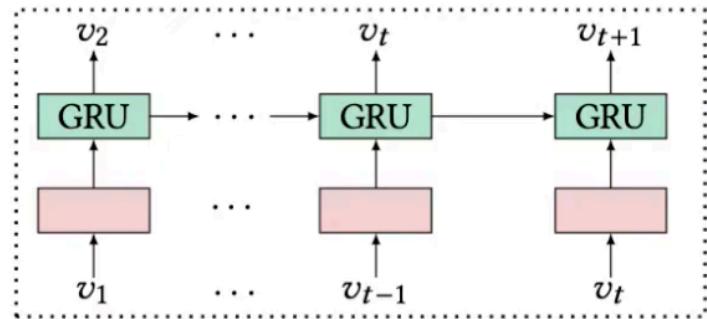
: CNN Base

SASRec (ICDM2018)

: self attention을 사용.



(c) SASRec model architecture.



(d) RNN based sequential recommendation methods.

05. Experiments

(4) Results

Table 2: Performance comparison of different methods on next-item prediction. **Bold scores** are the best in each row, while underlined scores are the second best. Improvements over baselines are statistically significant with $p < 0.01$.

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

기존 모델보다 좋은 성능을 보임 도메인, sparsity랑 상관없이 SOTA

05. Experiments

(4) Results

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

성능 향상의 원인이 self-attention 인지 masked learning 때문인가?

SASRec	item1	item2	item3	item4	item5	item6	pred
BERT4Rec(1)	item1	item2	mask	item4	item5	item6	item7
BERT4Rec	item1	mask	item3	item4	mask	item6	item7

05. Experiments

(4) Results

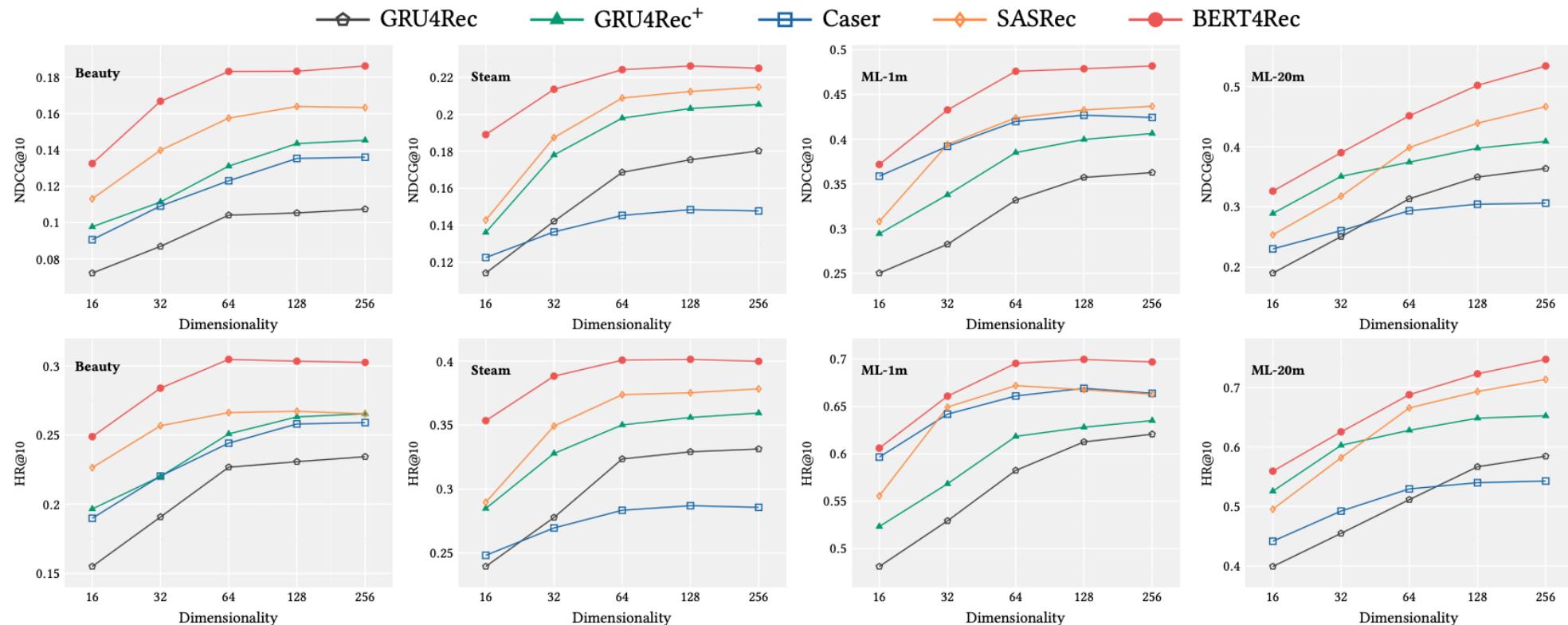


Figure 3: Effect of the hidden dimensionality d on HR@10 and NDCG@10 for neural sequential models.

트랜스포머에서 **hidden dimension**에 따라 얼마나 달라지는가 : 128, 64D가 제일 굿

05. Experiments

(4) Results

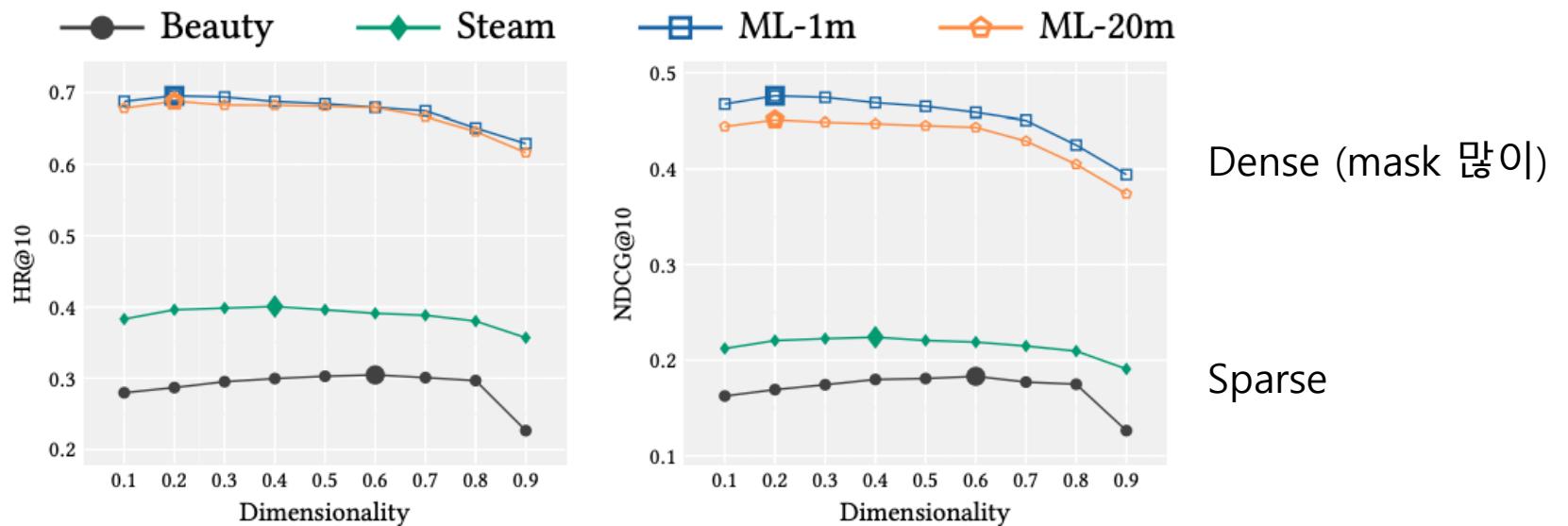


Figure 4: Performance with different mask proportion ρ on $d = 64$. Bold symbols denote the best scores in each line.

Dataset0 | 적당히 dense할수록 성능 올라감
그러나 너무 많이 dense하면 성능 저하

05. Experiments

(4) Results

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

Next item prediction에서 최신 아이템을 많이 받을수록 (N클수록) 성능 향상
데이터셋마다 다른 이유 : 데이터셋 자체의 avg length 차이

Conclusion & future work

06. Conclusion & Future work

(1) Restriction

이 논문에서 사용하는 데이터셋은 대부분 아이템, 유저가 정해져있다
그러나 실제 추천시스템에선 아이템, 유저의 수가 정해져있지 않다

Domain에 따른 item의 lifespan이 다르다

- 뉴스 : 하루 전의 것을 추천하지 않음 / 시간 별로 추천뉴스 달라짐

06. Conclusion & Future work

(2) Inference

실용적인 면에서 달라지는게 너무 많다 : 성능이 낮아도 **시간 빠른게 더 최고**