Self-Attentive Sequential Recommendation

https://arxiv.org/pdf/1808.09781

0. Introduction

- 순차적 사용자 행동 데이터를 활용한 추천 문제 해결을 목표로 함
- 기존 RNN, CNN 기반 모델은 장기 의존성 학습과 병렬 처리에 한계 존재
- 긴 시퀀스에서 중요한 행동만 선택적으로 반영하기 어려움
- Self-Attention 기반 구조로 효율성과 성능을 동시에 개선하는 것이 핵심 기여

1. Overview

- Transformer의 Self-Attention을 추천 시스템에 적용한 순차 추천 모델 제안
- 사용자 행동 시퀀스에서 중요한 아이템 관계만 선택적으로 학습
- 병렬 계산이 가능하여 학습 속도 및 확장성 개선
- 다양한 길이의 사용자 행동 데이터에 안정적으로 적용 가능

2. Challenges

- 사용자 행동 데이터는 희소하며 노이즈가 많음
- 긴 시퀀스에서 모든 과거 행동이 중요하지 않음
- RNN 계열 모델은 긴 거리 의존성 학습이 어려움
- 실서비스 환경에서는 대규모 사용자 처리 속도가 중요함

3. Method

- 입력은 사용자 행동 시퀀스를 임베딩 벡터로 변환
- Positional embedding을 추가하여 순서 정보 유지

- Self-Attention을 통해 시퀀스 내부 아이템 간 관계 학습
- Attention mask를 사용하여 미래 아이템 정보 차단
- Feed-forward layer와 residual connection으로 표현력 강화
- 마지막 시점 representation을 활용해 다음 아이템 예측 수행

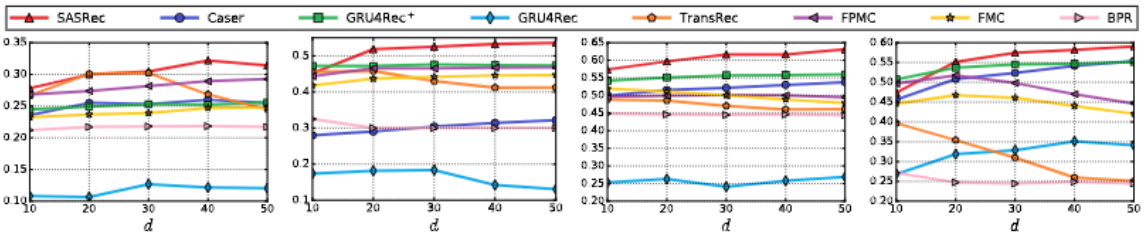
4. Experiments

Dataset	#users	#items	avg. actions /user	avg. actions /item	#actions
Amazon Beauty	52,024	57,289	7.6	6.9	0.4M
Amazon Games	31,013	23,715	9.3	12.1	0.3M
Steam	334,730	13,047	11.0	282.5	3.7M
MovieLens-1M	6,040	3,416	163.5	289.1	1.0M

- Amazon 리뷰 데이터와 MovieLens 데이터셋 사용
- 다양한 시퀀스 길이와 sparsity 환경에서 실험 수행
- 비교 모델은 GRU4Rec, CNN 기반 모델 등 기존 순차 추천 모델
- 평가 지표는 HR@10, NDCG@10 사용
- 데이터 sparsity 수준에 따른 성능 변화도 분석

5. Results

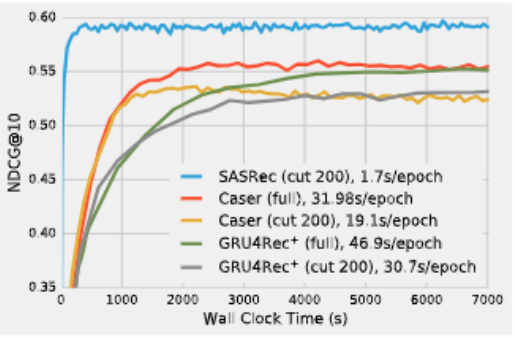
Dataset	Metric	(a) PopRec	(b) BPR	(c) FMC	(d) FPMC	(e) TransRec	(f) GRU4Rec	(g) GRU4Rec$^+$	(h) Caser	(i) SASRec	Improvement vs. (a)-(e)	Improvement vs. (f)-(h)
Beauty	Hit@10	0.4003	0.3775	0.3771	0.4310	0.4607	0.2125	0.3949	0.4264	**0.4854**	5.4%	13.8%
	NDCG@10	0.2277	0.2183	0.2477	0.2891	0.3020	0.1203	0.2556	0.2547	**0.3219**	6.6%	25.9%
Games	Hit@10	0.4724	0.4853	0.6358	0.6802	0.6838	0.2938	0.6599	0.5282	**0.7410**	8.5%	12.3%
	NDCG@10	0.2779	0.2875	0.4456	0.4680	0.4557	0.1837	0.4759	0.3214	**0.5360**	14.5%	12.6%
Steam	Hit@10	0.7172	0.7061	0.7731	0.7710	0.7624	0.4190	0.8018	0.7874	**0.8729**	13.2%	8.9%
	NDCG@10	0.4535	0.4436	0.5193	0.5011	0.4852	0.2691	0.5595	0.5381	**0.6306**	21.4%	12.7%
ML-1M	Hit@10	0.4329	0.5781	0.6986	0.7599	0.6413	0.5581	0.7501	0.7886	**0.8245**	8.5%	4.6%
	NDCG@10	0.2377	0.3287	0.4676	0.5176	0.3969	0.3381	0.5513	0.5538	**0.5905**	14.1%	6.6%



(a) *Beauty* (b) *Games* (c) *Steam* (d) *ML-1M*

Architecture	*Beauty*	*Games*	*Steam*	*ML-1M*
(0) Default	0.3142	0.5360	0.6306	0.5905
(1) Remove PE	**0.3183**	0.5301	0.6036	0.5772
(2) Unshared IE	0.2437↓	0.4266↓	0.4472↓	0.4557↓
(3) Remove RC	0.2591↓	0.4303↓	0.5693	0.5535
(4) Remove Dropout	0.2436↓	0.4375↓	0.5959	0.5801
(5) 0 Block (b=0)	0.2620↓	0.4745↓	0.5588↓	0.4830↓
(6) 1 Block (b=1)	0.3066	**0.5408**	0.6202	0.5653
(7) 3 Blocks (b=3)	0.3078	0.5312	0.6275	**0.5931**
(8) Multi-Head	0.3080	0.5311	0.6272	0.5885

n	10	50	100	200	300	400	500	600
Time(s)	75	101	157	341	613	965	1406	1895
NDCG@10	0.480	0.557	0.571	0.587	0.593	0.594	0.596	0.595



- 대부분의 데이터셋에서 기존 RNN, CNN 모델 대비 성능 향상 확인
- 긴 시퀀스 환경에서 특히 높은 성능 개선 확인
- sparse 데이터에서도 안정적인 추천 품질 유지
- 모델 구조가 단순하여 학습 속도 및 확장성 우수
- Attention layer 수와 시퀀스 길이에 따른 성능 변화 확인

6. Insight

- 추천 시스템에서도 Attention 구조가 효과적임을 입증
- 모든 과거 행동이 아닌 중요한 행동만 선택적으로 반영하는 것이 핵심
- 구조 단순화와 성능 개선을 동시에 달성한 모델
- 이후 BERT4Rec 등 Transformer 기반 추천 모델 발전의 출발점 역할 수행
- 대규모 서비스 추천 시스템에 적용 가능한 실용적 접근 방식 제시