💚

BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer

BERT4Rec
- 양방향(bi-directional) self-attention을 사용하여 사용자의 과거 행동 시퀀스를 모델링하는 순차 추천 모델
- 기존의 단방향 모델의 한계를 극복하기 위해 제안되었으며, Cloze objective를 사용하여 효율적으로 양방향 모델을 훈련시킴
- 각 항목이 좌우 컨텍스트로부터 정보를 결합하여 추천을 위한 표현 모델을 학습
- 실험 결과, BERT4Rec은 다양한 최첨단 순차 모델에 비해 우수한 성능을 보임

📌 Keywords

▼ Sequential Recommendation

- 사용자가 과거에 상호 작용한 항목의 순서를 고려하여 미래에 상호 작용할 가능성이 높은 항목을 추천하는 추천 시스템의 한 유형
    - 이는 사용자의 관심과 취향을 더 정확하게 파악하고 개인화된 추천을 제공
- 주로 시간적 순서와 항목 간의 상호 작용을 고려하는 모델을 사용하여 구현
    - 실시간으로 업데이트되는 동적 시스템으로도 구현될 수 있음

▼ Bidirectional Sequential Model

- 입력 시퀀스를 양방향으로 처리하여 시간적 의존성을 파악하는 모델
    - 순방향과 역방향으로 모두 처리하여 현재 시점에서의 입력에 대한 좀 더 포괄적인 context를 얻음

- 주로 순환신경망(RNN)이나 변형된 트랜스포머와 같은 모델에서 활용
    - 모델이 과거와 미래의 context를 모두 고려하여 더욱 풍부한 정보를 활용할 수 있도록 함

▼ Cloze

- 주어진 텍스트에서 일부 단어나 문장이 빠진 상태로 주어지고, 이를 채워넣는 문제 유형을 가리킴

ex) "나는 매일 아침 무엇을 먹어야 할지 고민합니다."

→ "무엇을" 부분이 빠져 있는 상태로 문제가 출제될 수 있음

→ 학습자는 빈칸을 채우는 작업을 수행

- 텍스트 이해와 문맥 파악 능력을 검사하는데 유용하며, 언어 학습에서도 많이 활용됨

1. Introduction

- 효과적인 추천 시스템은 사용자의 관심사를 정확히 파악해야 함
    - 사용자의 동적인 행동을 고려한 모델링이 필요
- 이전 방법은 주로 단방향 모델을 사용하여 숨겨진 표현을 학습하고 이를 기반으로 추천을 수행하였음
    - 이러한 방법은 제한적이며 순차 데이터에 대한 최적의 표현을 학습하는 데 한계가 있음
- 이를 극복하기 위해 양방향 self-attention을 사용하여 사용자의 행동 시퀀스를 모델링하는 모델을 제안
    - Cloze 작업을 통해 훈련 ⇒ 각 항목의 표현이 좌우 컨텍스트를 모두 활용할 수 있도록 함
- 실험 결과, 우리의 모델(**BERT4Rec**)은 다양한 최첨단 기법보다 우수한 성능을 보여줌

2. Related Work

2-1. Generated Recommendation

- 추천 시스템은 일반적으로 사용자의 상호작용 기록을 기반으로 사용자의 선호도를 모델링하기 위해 협업 필터링(CF)을 사용하였음

    - 행렬 분해(MF)를 가장 많이 활용하였음

      → 사용자와 항목을 공유 벡터 공간으로 투영하고 그들의 벡터 간의 내적을 통해 사용자의 항목에 대한 선호도를 추정

    - 또 다른 방법으로 항목 기반 이웃 방법이 있음

      → 미리 계산된 항목 간 유사도 행렬을 사용하여 사용자의 항목에 대한 선호도를 추정

- 최근에는 **딥러닝**이 추천 시스템을 혁명적으로 변화시키고 있음
    - Netflix Prize: 두 층 제한 볼츠만 머신(RBM)

      → 보조 정보인 텍스트, 이미지 및 음향 기능에서 학습된 분산 항목 표현을 CF 모델에 통합하여 추천 성능을 향상시키는 것을 목표로 함

    - 또 다른 방법으로 Neural Collaborative Filtering(NCF)를 제안

      → 내적 대신 Multi-Layer Perceptions(MLP)를 사용하여 사용자 선호도를 추정하고, AutoRec 및 CDAE는 오토인코더 프레임워크를 사용하여 사용자의 평가를 예측

      → 전통적인 행렬 분해를 대체

2-2. Sequential Recommendation

- 그러나, 위에서 언급한 방법들은 모두 사용자의 행동 순서를 무시하기에 **순차적 추천**이 이루어지지는 않음
- 초기 연구 → 주로 마르코프 체인을 사용하여 순차적 패턴을 캡처하였음
- 최근에는 RNN 및 그 변형인 GRU 및 LSTM이 사용자의 행동 시퀀스를 모델링하는 데 더 많이 사용되며, 또한 다양한 딥러닝 모델이 순차적 추천에 도입되었음

⇒ 사용자의 일반적인 관심사와 현재 관심사를 모두 캡처하기 위해 다양한 아키텍처와 손실 함수를 활용

2-3. Attention Mechanism

- 최근 몇 년간 어텐션 메커니즘이 추천 시스템에 도입되어 성능을 향상시키고 해석 가능성을 높이는 시도가 있었음
    - 어텐션 메커니즘을 기존 모델에 추가적인 구성 요소로 적용하는 것이 일반적이었으나, `Transformer`와 `BERT`와 같은 모델은 어텐션 메커니즘을 중심으로 구축되어 텍스트 시퀀스 모델링에서 성과를 거두었음

- 특히, 순차 데이터를 모델링하기 위해 어텐션 기반의 네트워크가 더 많이 사용되고 있음

- 최근에는 순차적 추천을 위해 **양방향 모델**을 사용하여 사용자의 행동 시퀀스를 인코딩하는 연구가 진행되고 있음

3. BERT4REC

3-1. Problem Statement

- 사용자 집단: $u = \{u_1, u_2, \cdots, u_{|u|}\}$
- 아이템 집단: $v = \{v_1, v_2, \cdots, v_{|v|}\}$
- 순차적 상호작용 기록: $S_u = [v_1^{(u)}, \cdots, v_t^{(u)}, \cdots, v_{n_u}^{(u)}]$

⇒ **목표)** 시간 단계 $n_u + 1$에서 사용자 u가 상호 작용할 아이템을 예측

- 확률 표현: $p(v_{n_u+1}^{(u)} = v | S_u)$

3-2. Model Architecture

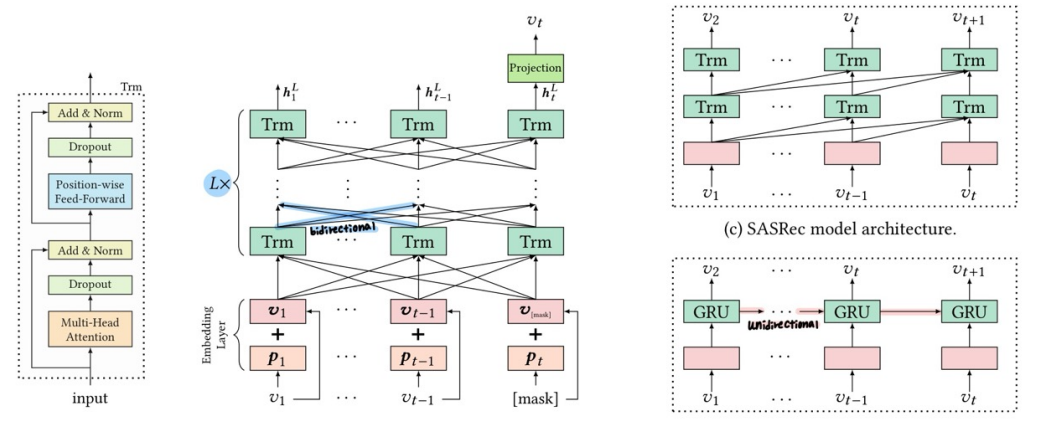


Figure 1: Differences in sequential recommendation model architectures. BERT4Rec learns a bidirectional model via Cloze task, while SASRec and RNN based methods are all left-to-right unidirectional model which predict next item sequentially.

- BERT4Rec은 **순차적** 추천을 위한 새로운 모델로, 양방향 Transformer 레이어를 사용
  - 각 레이어에서 모든 위치 간의 정보를 병렬로 교환하여 순차적 행동의 표현을 반복적으로 수정

    ⇒ 어떤 거리에서든 종속성을 캡처할 수 있으며, RNN과 같은 기존 방법보다 전역 수용 영역을 가짐

- 양방향 셀프 어텐션을 사용하여 사용자의 행동 시퀀스를 모델링

  → 추천 성능을 향상시킬 수 있음

3-3. Transformer Layer

- 길이가 t인 입력 시퀀스가 주어지면 각 위치 i에서 각 레이어 l에 대한 은닉 표현인 h_i^l을 반복적으로 계산
  - 이를 위해 Transformer Layer를 적용
  - 모든 위치에서 **동시적**으로 attention 함수를 계산함
- Transformer 레이어 Trm은 두 개의 서브 레이어(= Multi-Head Self-Attention 서브 레이어, Position-wise Feed-Forward Network)를 포함함

Multi-Head Self-Attention

- 시퀀스에서의 거리와 관계없이 표현 쌍 간의 의존성을 캡처할 수 있도록 함
- 먼저 H^l을 서로 다른 학습 가능한 linear projection으로 h개의 하위 공간으로 linear projection하고, 그런 다음 h개의 어텐션 함수를 병렬로 적용하여 출력 표현을 생성한 다음 이를 연결하고 다시 투영함

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

$$\mathrm{head}_i = \mathtt{Attention}\left(H^l W_i^Q, H^l W_i^K, H^l W_i^V\right)$$

  - head에 있는 각각의 projection matrix($W_i^Q, W_i^K, W_i^V, W_i^O$) 는 학습 가능함

    cf) parameter들이 layer 간에 공유되지는 않음!
  - Attention fn → 정규화된 dot-product attention임

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

query, key, value, temperature

Position-wise Feed-Forward Network

- self-attention 층은 linear projection에 집중되어 있음

- 비선형성과 다른 차원 간의 상호작용을 고려하기 위해 Position-wise Feed-Forward Network를 도입

$$\begin{aligned} \mathrm{PFFN}(\boldsymbol{H}^l) &= \left[\mathrm{FFN}(\boldsymbol{h}_1^l)^\top; \ldots; \mathrm{FFN}(\boldsymbol{h}_t^l)^\top\right]^\top \\ \mathrm{FFN}(\boldsymbol{x}) &= \mathrm{GELU}\left(\boldsymbol{x}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)}\right)\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} \\ \mathrm{GELU}(x) &= x\Phi(x) \end{aligned} \tag{3}$$

GELU: ReLU에 비해 smooth한 결과 / $\Phi(x)$: cdf of standard gaussian dist

where $\Phi(x)$ is the cumulative distribution function of the standard gaussian distribution, $\boldsymbol{W}^{(1)} \in \mathbb{R}^{d \times 4d}$, $\boldsymbol{W}^{(2)} \in \mathbb{R}^{4d \times d}$, $\boldsymbol{b}^{(1)} \in \mathbb{R}^{4d}$ and $\boldsymbol{b}^{(2)} \in \mathbb{R}^{d}$ are learnable parameters and shared across all positions. We omit the layer subscript l for convenience. In fact, these

Stacking Transformer Layer

$$\begin{aligned} \boldsymbol{H}^l &= \mathrm{Trm}\left(\boldsymbol{H}^{l-1}\right), \quad \forall i \in [1, \ldots, L] \\ \mathrm{Trm}(\boldsymbol{H}^{l-1}) &= \mathrm{LN}\left(\boldsymbol{A}^{l-1} + \mathrm{Dropout}\left(\mathrm{PFFN}(\boldsymbol{A}^{l-1})\right)\right) \\ \boldsymbol{A}^{l-1} &= \mathrm{LN}\left(\boldsymbol{H}^{l-1} + \mathrm{Dropout}\left(\mathrm{MH}(\boldsymbol{H}^{l-1})\right)\right) \end{aligned}$$

- 더 복잡한 항목 전이 패턴을 학습하기 위해 셀프 어텐션 레이어를 여러 겹으로 쌓음
  - 이를 위해 잔여 연결과 레이어 정규화를 사용
- 또한, 각 서브레이어의 출력에는 드롭아웃을 적용하고, 이후에 정규화를 수행

⇒ 네트워크 훈련 안정화/가속화

3-4. Embedding Layer

- Transformer 레이어 Trm에는 순환(recurrent) 또는 합성곱(convolutional) 모듈이 없기에 입력 순서를 인식할 수 없음
  - 입력의 순차적 정보를 활용하기 위해 Transformer 레이어 스택의 맨 아래에 **위치 임베딩**을 추가하였음
  - 주어진 항목 v_i에 대한 입력 표현 h_i^0은 해당 항목(v_i)과 위치 임베딩(p_i)을 더하여 구성 ⇒ $h_i^0 = v_i + p_i$

    - 위치 임베딩에 대한 학습이 가능하도록(→ learnable) 구성
- 입력 길이는 N으로 제한됨
  ⇒ 길이가 넘어가면 맨 뒤의 N개를 절사

3-5. Output Layer

- 이전 레이어(= embedding layer)의 모든 위치 간에 정보를 계층적으로 교환한 후, 입력 시퀀스의 모든 항목에 대한 최종 출력 H^L을 얻음
    - 시간 단계 t에서 항목 v_t를 마스크하고, 이후 h_t^L을 기반으로 마스크된 항목 v_t를 예측
- 구체적으로, 두 개의 레이어로 구성된 feed-forward 네트워크를 적용하고, 중간에 `GELU` 활성화 함수를 사용하여 목표 항목에 대한 출력 분포를 생성
    - $P(v) = softmax(GELU(h_t^L W^P + b^P)E^T + b^O)$
        - W^P: learnable projection mtx
        - b^p, b^O: bias term
        - E: embedding mtx for item set V
    - input/output layer에서 item embedding matrix를 **공유**

      ⇒ overfitting 방지, 모델 사이즈 감소

3-6. Model Learning

Training

- 기존의 단방향(unidirectional) 순차 추천 모델은 입력 시퀀스의 각 위치에서 다음 항목을 예측하여 모델을 훈련
    - 입력 시퀀스 $[v_1, ..., v_t]$의 목표는 이동된 버전 $[v_2, ..., v_{t+1}]$임
- 양방향 모델에서 좌우 컨텍스트에 동시에 조건을 걸면 각 항목의 최종 출력 표현에는 대상 항목의 정보가 포함될 수 있음

  ⇒ 네트워크가 유용한 정보를 학습하지 못함
  (미래 예측이 너무 쉬워짐)
    - **해결책>** 원래 길이 t의 동작 시퀀스에서 t - 1개의 샘플(다음 항목과 함께 하위 시퀀스인 $([v_1], v_2)$ 및 $([v_1, v_2], v_3)$와 같은)을 생성하고 각각의 기록된 하위 시퀀스를 양방향 모델로 인코딩하여 대상 항목을 예측

→ 많은 시간/리소스 요구

⇒ 위의 문제들을 모두 종합한 후 **Cloze 작업**을 제안

- **Cloze 작업**
  - 일부 단어가 제거된 언어의 일부로 구성된 테스트로, 참가자에게 빈칸을 채우도록 요구
  - 각 훈련 단계마다 입력 시퀀스에서 모든 항목의 p 비율을 무작위로 마스크 한 후 좌우 context를 기반으로 마스킹된 항목을 예측하도록 함

$$\textbf{Input}: [v_1, v_2, v_3, v_4, v_5] \xrightarrow{\text{randomly mask}} [v_1, [\text{mask}]_1, v_3, [\text{mask}]_2, v_5]$$
$$\textbf{Labels}: [\text{mask}]_1 = v_2, \quad [\text{mask}]_2 = v_4$$

  - loss 정의

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

user behavior history / neg log-likelihood / probability / pred / actual / masked input

---

  - 모델 훈련을 위한 추가적인 샘플을 생성할 수 있음
    - 길이가 n인 시퀀스를 가정하면 여러 epoch을 통해 $\binom{n}{k}$의 샘플을 생성할 수 있음(ramdomly masked 가정 시)

Test

- Cloze 작업은 현재 마스크된 항목을 예측하고, 순차적 추천은 미래를 예측 ⇒ 이들 간의 불일치 발생
  - 이를 해결하기 위해, 특별한 토큰 “[mask]”을 사용자의 행동 시퀀스 끝에 추가하여 순차적 추천 task에 더 잘 부합하도록 함
- 또한, 훈련 중에는 입력 시퀀스의 마지막 항목만을 마스킹하는 샘플을 생성하여 순차적 추천을 위한 fine-tuning을 수행

3-7. Discussion

- SASRec
- CBOW & Skip-Gram

- BERT

4. Experiments

4-1. Datasets

- 4가지 데이터셋에 대해 평가
  - Amazon Beauty
  - Steam
  - MovieLens

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%

데이터셋 통계

- 일반적인 데이터 전처리 과정을 채택
  - 모든 데이터셋에서 모든 숫자 등급이나 리뷰의 존재를 암시적 피드백(즉, 사용자가 항목과 상호작용함)으로 변환
  - 이후 사용자별로 상호작용 레코드를 그룹화하고 타임스탬프에 따라 상호작용 레코드를 정렬하여 각 사용자에 대한 상호작용 시퀀스를 구축
  - 데이터셋의 품질을 보장하기 위해, 최소 다섯 개의 피드백을 갖는 사용자만 유지

4-2. Task Settings & Evaluation Metrics

- 널리 사용되는 방법인 **leave-one-out**(= 다음 항목 추천) 평가 방법을 채택
  - 각 사용자에 대해 행동 시퀀스의 마지막 항목을 테스트 데이터로 유지하고, 마지막 이전 항목을 검증 세트로 취급하고, 나머지 항목을 훈련에 사용
- **평가 지표**
  - 히트 비율(Hit Ratio, HR)

- 정규화된 할인 누적 이익(Normalized Discounted Cumulative Gain, NDCG)
- 평균 역수 순위(Mean Reciprocal Rank, MRR) 등

⇒ 모든 metric에서 값이 높을수록 우수한 성능

4-3. Baselines & Implementation Details

- 다음과 같은 **베이스라인**들과 비교
  - POP: 상호작용 수에 따라 항목을 인기순으로 순위 매김하는 가장 간단한 베이스라인
  - BPR-MF: 쌍대 랭킹 손실을 사용하여 암묵적 피드백으로 행렬 인수분해를 최적화
  - NCF: 행렬 인수분해의 내적 대신 MLP로 사용자-항목 상호작용을 모델링
  - FPMC: MF와 1차 MC를 결합하여 사용자의 일반적인 취향과 순차적 행동을 모델링
  - GRU4Rec: 세션 기반 추천을 위해 랭킹 기반 손실을 사용하는 GRU를 사용하여 사용자 시퀀스를 모델링
  - GRU4Rec+: 새로운 클래스의 손실 함수와 샘플링 전략으로 GRU4Rec를 개선한 버전
  - Caser: 고차원 MC를 순차적 추천을 위해 가로 및 세로 방향으로 CNN을 사용하여 모델링
  - SASRec: 사용자의 순차적 행동을 캡처하기 위해 왼쪽에서 오른쪽으로 Transformer 언어 모델을 사용하며, 순차 추천에서 SOTA 달성
- 모든 모델은 TensorFlow를 사용하여 구현되었고, 일관된 하이퍼파라미터 범위에서 훈련되었음
- Adam 최적화기와 특정 학습률, $\ell 2$ 가중치 감소 등의 설정으로 학습되었음

4-4. Overall Performance Comparison

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	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%

- 개인화되지 않은 POP 방법은 모든 데이터셋에서 가장 나쁜 성능

  (사용자의 개인화된 선호도를 고려하지 않기 때문)

- 모든 베이스라인 방법 중에서 순차적 방법(→ FPMC 및 GRU4Rec+)이 비순차적 방법(→ BPR-MF 및 NCF)보다 모든 데이터셋에서 일관되게 성능이 우수
- 순차적 추천 베이스라인 중에서 Caser는 특히 밀도가 높은 데이터셋 ML-1m에서 FPMC를 능가
  - 그러나 고차 Markov 체인은 주로 아주 작은 순서 L을 사용하므로 L에 대한 확장성이 좋지 않음

    ⇒ 희소한 데이터셋에서 특히 GRU4Rec+ 및 SASRec보다 성능이 떨어짐
- 또한, SASRec는 순차적 추천에 대해 self attention 메커니즘이 더 강력한 도구임을 나타내는 것과 같이 GRU4Rec 및 GRU4Rec+보다 뚜렷하게 우수한 성능을 보임

⇒ BERT4Rec이 모든 메트릭에서 네 개의 데이터셋 중 가장 우수한 성능을 보임

Q1. 이러한 성과는 bi-directional self-attention 모델에서 오는 것인가 아니면 Cloze 목표에서 오는 것인가?

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

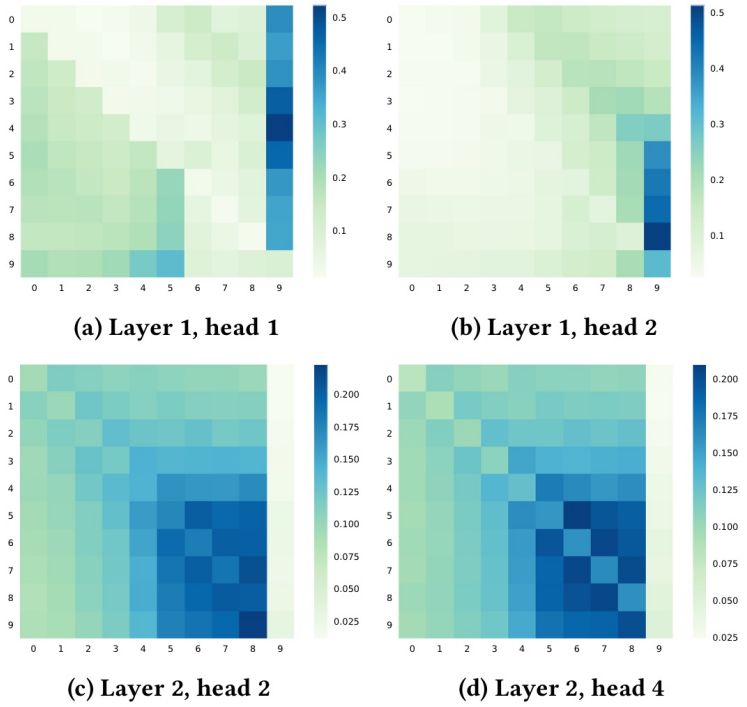
- 이를 확인하기 위해 Cloze 작업에서 한 번에 하나의 항목만 마스킹하도록 제약을 가함
- BERT4Rec(1 mask)가 모든 메트릭에서 SASRec보다 크게 우수하였음

  ⇒ 순차적 추천에 양방향 표현이 중요하다는 것을 입증

  ⇒ Cloze 목표 또한 성능 향상에 기여

Q2. 왜 양방향 모델이 단방향 모델보다 우수한가? 그리고 어떻게 더 나은 성능을 내는 걸까?

- Beauty의 테스트 중 마지막 10개 항목의 평균 어텐션 가중치를 시각화하여 의미 있는 패턴을 밝히고자 함



(a) Layer 1, head 1 (b) Layer 1, head 2

(c) Layer 2, head 2 (d) Layer 2, head 4

Figure 2: Heat-maps of average attention weights on Beauty, the last position "9" denotes "[mask]" (best viewed in color).

(a) 어텐션은 서로 다른 헤드에서 차이

- 예를 들어, 레이어 1에서 헤드 1은 왼쪽의 항목에 관심을 가지고, 헤드 2는 오른쪽의 항목에 관심을 가짐

(b) 어텐션은 서로 다른 레이어에서 차이

- 분명히, 레이어 2의 어텐션은 보다 최근 항목에 집중하는 경향이 있음

  ⇒ 레이어 2가 직접적으로 출력 레이어에 연결되어 있으며 최근 항목이 미래를 예측하는 데 더 중요한 역할을 수행

(c) 단방향 모델이 왼쪽의 항목에만 관심을 가지는 것과는 달리, BERT4Rec의 항목들은 양쪽 항목에 관심을 가짐

⇒ 양방향 모델이 사용자의 동작 시퀀스 모델링에 필수적이고 유익하다는 것을 시사

4-5. Impact of Hidden Dimensionality d

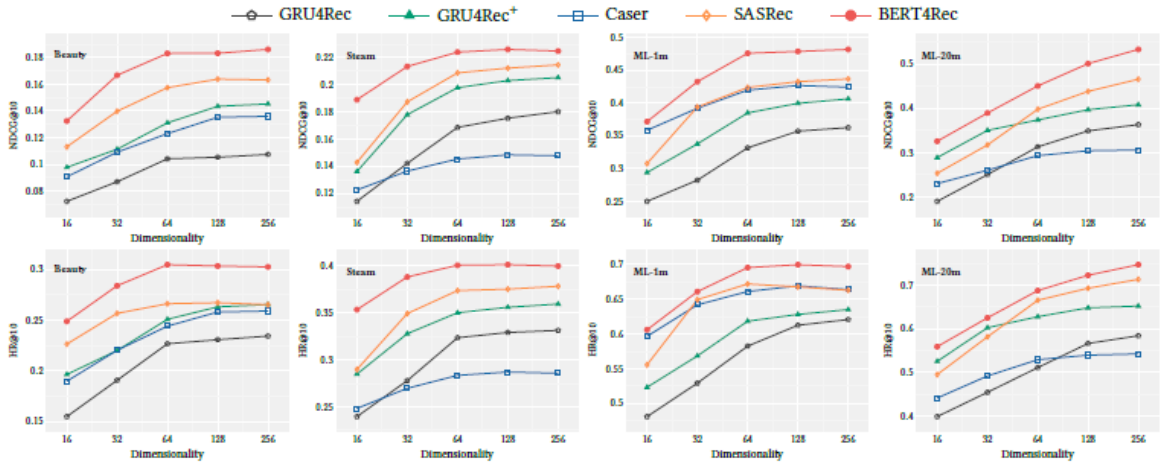


Figure 3: Effect of the hidden dimensionality d on HR@10 and NDCG@10 for neural sequential models.

- 차원이 증가함에 따라 모델 성능이 수렴하는 경향을 보임
  - 일정 수준 이상을 벗어나면 더 이상 증가 x
  - Beauty 및 Steam과 같은 희소한 데이터셋에서는 큰 은닉 차원이 더 나은 모델 성능으로 이어지지는 않음
- **BERT4Rec**의 경우 상대적으로 작은 은닉 차원으로도 괜찮은 성능을 달성

4-6. Impact of Mask Proportion ρ

- 마스크 비율(ρ)은 손실 함수에 직접적으로 영향을 미침
  - 모델 훈련에 있어서 중요한 요소

- 너무 작으면 강력한 모델로 학습시키기에 어렵고, 너무 큰 경우 훈련 자체에 어려움이 생길 수 있음

⇒ 다양한 마스크 비율에 따른 추천 성능의 변화 양상 관찰

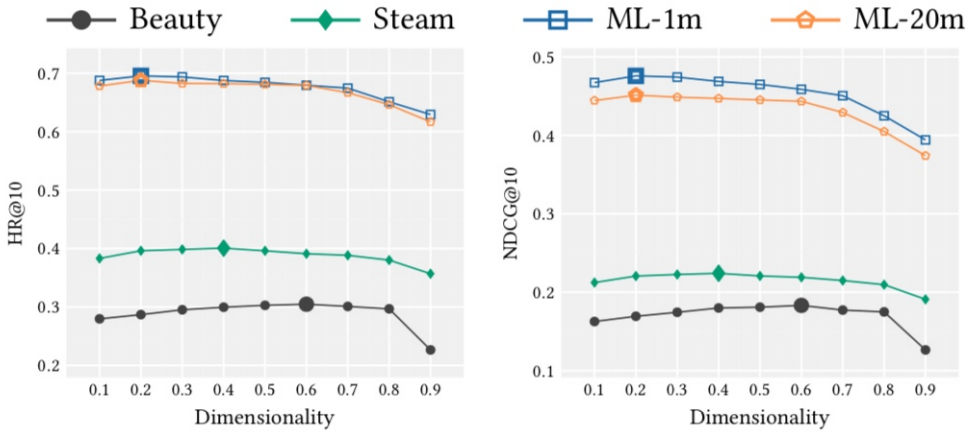


Figure 4: Performance with different mask proportion ρ on $d = 64$. Bold symbols denote the best scores in each line.

- 짧은 시퀀스 데이터셋에서는 ρ=0.6 또는 0.4가 좋은 성능을 보임
- 반면 긴 시퀀스 데이터셋에서는 ρ=0.2가 선호됨

  ⇒ 시퀀스 길이에 따라 최적의 마스크 비율이 달라짐

4-7. Impact on Maximum Sequence Length N

- 최대 시퀀스 길이(N)가 모델의 추천 성능과 효율에 미치는 영향을 조사

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

- 적절한 최대 길이(N)가 데이터셋의 평균 시퀀스 길이에 크게 의존함

    - 짧은 시퀀스 데이터셋에서는 최근 항목이 사용자의 행동에 더 영향을 주고, 긴 시퀀스 데이터셋에서는 최근 항목보다는 좀 이전의 최근 항목이 영향을 미치기 때문
- N이 커지는 경우 추가 정보와 더불어 많은 잡음을 동시에 가져오기 때문에 일관되게 더 큰 N에서 이득을 보지는 않음
    - 그러나 BERT4Rec은 길이 N이 더 길어짐에 따라 매우 안정적으로 수행됨

      ⇒ BERT4Rec이 잡음이 많은 과거 기록에서 정보가 있는 항목에 주의를 기울일 수 있다는 것을 시사
- BERT4Rec의 확장성
  → GPU를 사용하여 self-attention 계층을 효과적으로 병렬화할 수 있을 것으로 기대됨

4-8. Ablation Study

Table 5: Ablation analysis (NDCG@10) on four datasets. Bold score indicates performance better than the default version, while ↓ indicates performance drop more than 10%.

Architecture	Dataset			
	Beauty	Steam	ML-1m	ML-20m
$L = 2, h = 2$ 기본 셋팅	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**

- BERT4Rec 모델의 주요 구성 요소를 효과적으로 이해하기 위해 위치 임베딩(PE), 위치별 피드포워드 네트워크(PFFN), 레이어 정규화(LN), 잔여 연결(RC), 드롭아웃, 자기-

주의 레이어 수(L) 및 멀티 헤드 어텐션의 헤드 수(h) 등을 포함한 다양한 요소에 대한 **제거 실험**을 실시

- 위치 임베딩을 제거하면 BERT4Rec의 성능이 긴 시퀀스 데이터셋에서 급격하게 감소하는 것으로 나타났음
- 레이어 수를 쌓음으로써 성능이 향상되는 것이 확인되었으며, 헤드 수가 증가함에 따라 긴 시퀀스 데이터셋에서 더 많은 이득을 얻을 수 있음을 발견했음

5. Conclusion and Future Work

- 순차적 추천을 위한 deep bi-directional sequence 모델인 **BERT4Rec**를 소개
- 모델 훈련을 위해, 좌우 컨텍스트를 **모두** 활용하여 마스크된 항목을 예측하는 Cloze 작업을 소개

  ⇒ 네 가지 실제 데이터셋에 대한 방대한 실험 결과 최신 벤치마크 모델보다 우수한 성능을 보임을 확인

Discussion

- BERT4Rec에 아이템 특성(ex. 제품의 카테고리 및 가격, 영화의 출연진)을 포함시키는 것
- 사용자가 여러 세션을 가지고 있는 경우 명시적인 사용자 모델링을 위해 모델에 사용자 구성 요소를 도입하는 것