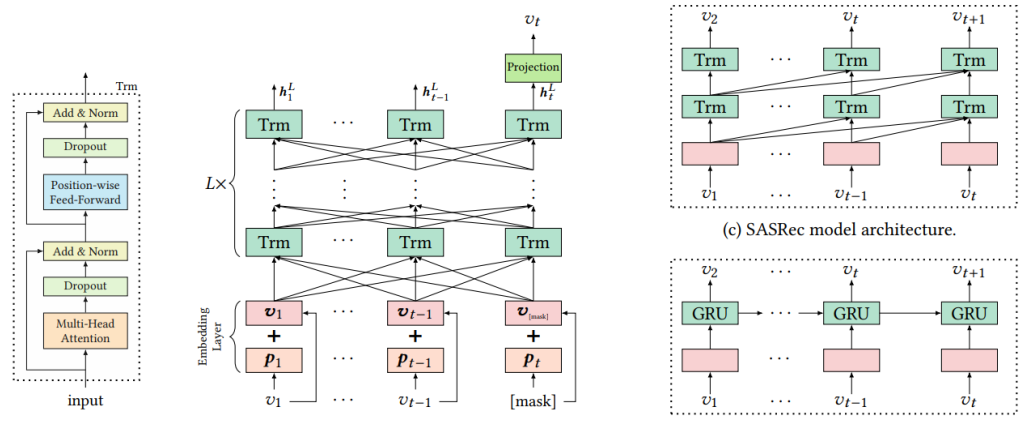
💚

[4주차] BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer

0. Abstract

- bidirectional model: 사용자의 행동 시퀀스를 모델링하기 위해 양방향 정보를 활용
- Cloze objective: 시퀀스에서 무작위로 마스킹된 항목을 예측하여 양방향 모델을 효율적으로 훈련
- BERT4Rec의 장점
    - 양방향 정보 활용: 각 항목이 사용자의 과거 행동에서 양쪽 방향의 정보를 융합할 수 있음
    - 향상된 추천 성능: 네 개의 벤치마크 데이터셋에서 최신 sequential model들을 일관되게 능가함

1. Introduction



(a) Transformer Layer. (b) BERT4Rec model architecture. (c) SASRec model architecture. (d) RNN based sequential recommendation methods.

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.

- 효과적인 추천 시스템의 핵심: 사용자의 관심사를 정확하게 파악하는 것
  - 실제 환경에서 사용자의 현재 관심사는 그들의 과거 행동에 의해 영향을 받으며, 본질적으로 동적이고 진화하는 특성을 가짐
  - 사용자가 닌텐도 스위치를 구매한 직후에는 엑세서리를 구매할 수 있지만, 일반적인 상황에서는 콘솔 엑세서리를 구매하지 않을 수 있음
- 사용자 행동의 순차적 동태성을 모델링을 하기 위해 **과거 상호작용을 기반으로 순차적 추천**을 하는 다양한 방법이 제안됨
  - 목표: 사용자의 과거 상호작용을 바탕으로 사용자가 다음에 상호작용할 가능성이 있는 항목을 예측하는 것
  - 최근에는 순차적 추천을 위해 RNN과 같은 순환 신경망을 사용하는 연구가 활발히 이루어짐
  - 기존 작업의 기본 패러다임: 사용자의 과거 상호작용을 왼쪽에서 오른쪽으로 sequential model을 사용하여 벡터(사용자의 선호도 표현)로 인코딩하고, 이 숨겨진 표현을 기반으로 추천하는 것
- 한계: 왼쪽에서 오른쪽으로의 **단방향 모델**은 사용자 행동 시퀀스에 대한 최적의 표현을 학습하기에 충분하지 않음
  - 단방향 모델이 과거 항목에서만 정보를 인코딩할 수 있기 때문에, 숨겨진 표현을 제한하게 됨
  - 단방향 모델은 본래 자연스러운 순서가 있는 순차적 데이터(텍스트 및 시계열)를 위해 도입되었으며, 실제 사용자 행동에서도 엄격한 순서를 가정함

➡️ 해결책: 사용자의 과거 행동 시퀀스의 표현을 학습하기 위해 **양방향 모델** 사용

    - 텍스트 이해에서 BERT의 성공에 영감을 받아, 순차적 추천에 깊은 양방향 자기 주의 모델을 적용
    - 표현력 측면에서, 양방향 모델이 시퀀스 표현 학습에 양쪽 컨텍스트를 통합하는 것이 유익하다는 것을 보여줌
    - 엄격한 순서를 가정하는 면에서, 양방향 모델의 모든 항목이 왼쪽과 오른쪽 양쪽의 컨텍스트를 활용할 수 있기 때문에 사용자 행동 시퀀스 모델링에 단방향보다 더 적합함

- 또 다른 문제: 순차적 추천을 위해 양방향 모델을 훈련하는 것은 직관적이지 않음
    - 원래 순차적 추천 모델: 입력 시퀀스의 각 위치에 대해 다음 항목을 예측함으로써 왼쪽에서 오른쪽으로 훈련됨
    - 양방향 모델에서 왼쪽과 오른쪽 컨텍스트를 함께 조건으로 하는 것은 정보 유출을 초래할 가능성이 있음
    - 네트워크가 유용한 것을 학습하지 못하게 할 수 있음

➡️ 해결책: **Cloze 작업**을 도입

- 단방향 모델의 목표(다음 항목을 순차적으로 예측)를 대체
- 입력 시퀀스에서 일부 항목을 무작위로 마스킹(특수 토큰 [mask]로 대체)하고, 그 주변 컨텍스트를 기반으로 마스킹된 항목의 ID를 예측함으로써 정보 유출을 피하고 양방향 표현 모델을 학습할 수 있음
- 더 많은 샘플을 생성하여 더 강력한 모델을 여러 에포크에 걸쳐 훈련
- Cloze의 단점
    - 최종 작업(순차적 추천)과 일치하지 않음
    - 이를 해결하기 위해, 테스트 중에는 입력 시퀀스의 끝에 특수 토큰 [mask]를 추가하여 예측해야 할 항목을 나타냄
    - 최종 숨겨진 벡터를 기반으로 추천을 함
- 네 개의 데이터셋에서 실시한 실험 >> Cloze 작업을 도입한 새 모델이 다양한 최신 기준선을 일관되게 능가함을 보여줌

2. Related Work

2.1 General Recommendation

- 일반 추천 시스템에 대한 초기 연구: Collaborative Filtering(CF)

- CF를 사용하여 사용자의 선호도를 이전 상호작용 이력을 바탕으로 모델링
- 행렬 분해: CF 중 가장 인기 있는 부분
  - 사용자와 항목을 공유 벡터 공간에 투영하고 그들 벡터 사이의 내적을 통해 사용자의 항목에 대한 선호도를 추정

👉 **행렬 분해**

<행렬 분해의 기본 원리>

- 벡터 공간 투영: 사용자와 항목을 각각 벡터로 표현하고 이 벡터들을 같은 차원의 공간에 매핑, 이 공간을 '**잠재 특징 공간(Latent Feature Space)**'이라고 하며 사용자 벡터와 항목 벡터는 이 공간에서의 위치에 따라 서로의 관계를 나타냄

- 선호도 추정: 사용자 벡터와 항목 벡터 사이의 내적을 계산하여 사용자의 항목에 대한 선호도를 추정, 내적 값이 클수록 사용자가 해당 항목을 더 선호한다고 해석할 수 있음

<행렬 분해의 과정>

- 사용자-항목 행렬 생성: 사용자와 항목 간의 상호작용(예: 평점, 구매 여부)을 나타내는 행렬 생성, 이 행렬은 대부분의 값이 비어 있는 '희소 행렬(Sparse Matrix)'

- 행렬 분해: 이 행렬을 두 개의 낮은 차원의 행렬(사용자 행렬과 항목 행렬)로 분해, 이 과정에서 잠재 특징을 포착함

- 선호도 예측: 분해된 행렬을 다시 곱하여 누락된 값(사용자의 항목에 대한 선호도)을 예측

<행렬 분해의 장점>

- 정확도: 잠재적인 특징을 통해 사용자의 선호도를 보다 정확하게 예측 가능

- 유연성: 다양한 유형의 데이터와 상호작용을 모델링할 수 있음

- 항목 기반 이웃 방법
  - 사전에 계산된 항목 간 유사도 행렬을 사용

        - 사용자의 상호작용 이력에 있는 항목들과의 유사성을 측정함으로써 항목에 대한 사용자의 선호도를 추정
- 혁신적인 딥러닝 추천 시스템
    - 초기: 넷플릭스의 협업 필터링을 위한 2층 제한 볼츠만 머신(RBM)
    - 이미지, 텍스트, 음향 등과 같은 보조 정보에서 학습된 분산 항목 표현을 CF 모델에 통합하여 추천 성능을 향상시키려고 함
    - 또 다른 연구 방향: 전통적인 행렬 분해를 대체
        - 신경 협업 필터링: 내적 대신 다층 퍼셉트론(MLP)을 사용하여 사용자 선호도 추정
        - AutoRec, CDAE: 오토인코더 프레임워크를 통해 평점 예측

2.2 Sequential Recommendation

- 순차적 추천 시스템
    - 사용자의 행동 순서를 고려하여 추천 제공
    - 초기 연구: 사용자의 이전 상호작용에서 순차적 패턴을 포착하기 위해 Markov Chain(MCs)을 사용
        - 추천 생성을 순차적 최적화 문제로 정식화하고 이를 해결하기 위해 Markov Decision Processes(MDPs)을 사용
    - 이후: MCs와 MF의 장점을 결합하여 순차적 행동과 일반적 관심사를 모델링하기 위해 **개인화된 Markov Chain을 분해**하는 방법(Factorizing Personalized Markov Chains, FPMC)을 제안함
    - 첫 번째 순서의 MCs 외에도 고차 MCs가 더 많은 이전 항목을 고려하기 위해 사용됨
    - 최근에는 **순환 신경망(RNN)**과 그 변형인 **GRU 및 LSTM**이 사용자 행동 시퀀스를 모델링하기 위해 자주 사용됨
        - 기본 아이디어: 다양한 순환 아키텍처와 손실 함수를 사용하여 사용자의 이전 기록을 벡터(예측을 위해 사용되는 사용자의 선호도 표현)로 인코딩하는 것
            - 순위 손실을 가진 세션 기반 GRU
            - 동적 순환 바스켓 모델(DREAM)
            - 사용자 기반 GRU
            - 주의 기반 GRU(NARM)

- 새로운 손실 함수(BPR-max 및 TOP1-max)
- 개선된 샘플링 전략을 가진 GRU4Rec의 개선 버전

📈 **순환 신경망 RNN과 그 변형**

<순환 신경망 RNN>
➡️ RNN은 시퀀스 데이터(예: 문장에서의 단어 순서, 시간에 따른 주식 가격 변동 등)를 처리하기 위해 설계된 신경망으로, 이전의 정보를 기억하면서 현재의 입력과 결합하여 출력을 생성함
➡️ RNN은
'메모리'를 가지고 있어 **과거의 입력 정보를 현재의 결정에 반영**할 수 있으며, 이는 시퀀스 내의 정보가 서로 연관되어 있음을 인식하고 활용할 수 있게 해줌

<Gated Recurrent Unit(GRU)>
➡️ GRU는 RNN의 한 변형으로, 더 복잡한 시퀀스를 효과적으로 처리할 수 있도록 설계되었으며 LSTM에 영감을 받아 만들어졌지만 구조가 더 간단함
➡️ GRU는
'게이트'라는 메커니즘을 사용하여 정보의 흐름을 조절하는데, 이를 통해 **어떤 정보를 기억하고 어떤 정보를 무시할지 결정**하며 이는 **장기 의존성 문제**를 해결하는 데 도움을 줌

<Long Short-Term Memory(LSTM)>
➡️ LSTM도 RNN의 한 변형으로, 특히 장기 의존성 문제를 해결하기 위해 고안됨. LSTM은 GRU보다 먼저 개발되었으며
복잡한 시퀀스 데이터를 처리하는 데 매우 효과적임
➡️ LSTM은 GRU와 유사하게
'게이트' 메커니즘을 사용하는데, 이를 통해 네트워크는 필요한 정보를 장기간 동안 유지할 수 있음

- 순환 신경망 외에도 다양한 딥러닝 모델들이 순차적 추천을 위해 도입됨
  - 컨볼루셔널 시퀀스 모델(Caser): 수평 및 수직 컨볼루셔널 필터를 사용하여 순차적 패턴을 학습
  - 메모리 네트워크: 순차적 추천을 개선

    - STAMP: 주의 기반 MLP 네트워크를 사용하여 사용자의 일반적인 관심사와 현재 관심사를 모두 포착

2.3 Attention Mechanism

- **자기주의 매커니즘**(Attention Mechanism)의 기본
    - 순차적 데이터 모델링에서 **중요한 부분에 집중**할 수 있게 해주는 기술
    - 모델이 입력 시퀀스의 다양한 부분에 대해 다른 가중치를 동적으로 할당함으로써, 고정된 크기의 내부 메모리의 한계를 극복할 수 있게 함
- **Transformer**와 **BERT**
    - Transformer: 입력 시퀀스에서 출력 시퀀스를 생성하기 위해 설계된 모델로, 자기주의 메커니즘에 전적으로 의존함
    - BERT: 고품질의 텍스트 시퀀스 모델링을 위해 설계된 모델로, Transformer 아키텍처를 기반으로 함
- **SASRec**: 순차적 추천을 위한 Transformer 디코더
    - SASRec는 **사용자의 순차적 행동을 포착**하고 여러 공개 데이터셋에서 최고의 결과를 달성하기 위해 설계된 **두 계층 Transformer 디코더**
    - 순차적 추천 시스템에서 중요한 역할을 함

3. BERT4Rec

3.1 Problem Statement

- 순차적 추천 문제 정의: 사용자와 아이템의 상호작용을 통해 사용자가 다음에 어떤 아이템과 상호작용할지를 예측하는 것이 목표
    - 사용자 집합 (U={u_1, u_2, ..., u_{|U|}})과 아이템 집합 (V={v_1, v_2, ..., v_{|V|}})을 정의
    - 각 사용자 (u)의 상호작용 시퀀스를 시간 순서대로 나열한 (S_u=[v^{(u)}_1, ..., v^{(u)}t, ..., v^{(u)}{n_u}])로 표현

$$S_u = [v_1^{(u)}, \ldots, v_t^{(u)}, \ldots, v_{n_u}^{(u)}]$$

        - (v^{(u)}_t): 사용자 (u)가 시간 단계 (t)에서 상호작용한 아이템
        - (n_u): 사용자 (u)의 상호작용 시퀀스 길이

- 상호작용 이력 (S_u)를 바탕으로 사용자 (u)가 다음 시간 단계 (n_u + 1)에서 상호작용할 아이템을 예측해야 함
- 시간 단계 (n_u+1)에서 사용자 (u)에 대한 모든 가능한 아이템에 대한 확률을 모델링하는 것으로 공식화될 수 있음:

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

3.2 Model Architecture

- BERT4Rec 모델 아키텍처
  - 순차적 추천 시스템을 위해 개발된 새로운 모델
  - Transformer의 양방향 인코더 표현
  - 자기 주의 계층인 transforemr 계층을 기반으로 하여 구축
  - 여러 개의 양방향 Transformer 계층을 쌓아 올린 구조를 가짐
  - 각 계층에서는 이전 계층의 모든 위치에서 정보를 교환하며 각 위치의 표현을 반복적으로 수정
- 모델의 특징
  - 양방향 Transformer 계층: 여러 개의 **양방향 Transformer 계층**을 사용함으로써 사용자의 행동 시퀀스 전체에서 정보를 교환하고, 각 아이템의 표현을 반복적으로 수정함
  - 자기 주의 매커니즘: 어떠한 거리에서도 아이템 간의 의존성을 직접 포착할 수 있으며, 이는 **전역 수용 영역**(global receptive field)을 결과로 함(**CNN** 기반 방법들이 가지는 **제한된 수용 영역과 대비**됨)
  - 병렬 처리의 용이성: 자기 주의 메커니즘은 병렬 처리가 용이하며, RNN 기반 방법들과 비교했을 때 계산 효율성이 높음
- 모델의 차별점: **양방향 자기 주의**를 통한 모델링
  - SASRec 및 RNN 기반 방법들이 사용하는 왼쪽에서 오른쪽으로의 단방향 아키텍처와 대비됨
  - 이를 통해 사용자의 행동 시퀀스에 대한 **더 강력한 표현**을 얻어 추천 성능을 향상시킬 수 있음

반복적 수정 과정

<자기 주의 메커니즘과 Transformer 계층>
- BERT4Rec 모델은 여러 개의 Transformer 계층을 사용
- 각 Transformer 계층은
입력 시퀀스의 각 아이템에 대한 정보를 처리하고 이 정보를 다음 계층으로 전달함
- 자기 주의 메커니즘을 통해 모델은 시퀀스 내의 모든 아이템 간의 관계를 평가하고, 이를 바탕으로 각 아이템의 새로운 표현을 생성함

<첫 번째 계층>
- 입력 시퀀스의
각 아이템은 초기 표현(임베딩 등)을 가지고 있음
- 첫 번째 Transformer 계층은 이 초기 표현을 바탕으로 각 아이템의
새로운 표현을 계산
-
자기 주의 메커니즘을 통해 시퀀스 내의 다른 아이템들과의 관계를 고려

<이후 계층들>
- 첫 번째 계층에서 생성된 새로운 표현은 다음 계층의 입력으로 사용됨
- 각 계층은 이전 계층에서 계산된 표현을 바탕으로
다시 자기주의 메커니즘을 통해 각 아이템의 표현을 수정
- 모델에 설정된 계층 수만큼 반복

3.3 Transformer Layer

- Transformer 계층(Trm)은 두 개의 서브 계층으로 구서됨
  - 멀티 헤드 자기 주의(Multi-Head Self-Attention) 서브 계층
    - 입력 시퀀스 내의 각 아이템이 시퀀스 내 다른 아이템들과 어떻게 관련되는지를 학습
    - 모델은 이를 통해 각 아이템의 문맥을 더 잘 이해할 수 있음
  - 위치별 전방 피드포워드 네트워크(Position-wise Feed-Forward Network)
    - 각 위치에서의 숨겨진 표현을 독립적으로 변환

        - 모델이 각 아이템의 표현을 더욱 세밀하게 조정할 수 있음
- 반복적 계산 과정
    - 입력 시퀀스의 각 아이템에 대해, BERT4Rec 모델은 두 서브 계층을 포함하는 Transformer 계층을 통해 숨겨진 표현을 반복적으로 계산하고 수정함
    - 이 과정은 모델이 더 깊은 계층으로 진행됨에 따라 각 아이템의 표현이 점점 더 풍부해지게 만들어줌
    - 모델은 사용자의 행동 시퀀스를 더 정확하게 반영하는 아이템의 표현을 생성할 수 있게 됨

➡️ **모델은 사용자의 행동 시퀀스를 더 정확하게 반영하는 아이템의 표현을 생성할 수 있게 됨**

Multi-Head Self-Attention

- 동작 원리
    - 단일 주의 함수를 수행하는 대신, 멀티 헤드 자기 주의를 채택
    - (H^l)을 다른 학습 가능한 선형 투영을 사용하여 (h)개의 부공간으로 선형적으로 투영한 다음, (h)개의 주의 함수를 병렬로 적용하여 출력 표현을 생성
        - 수식: (MH(H^l) = [head_1; head_2; ... ; head_h]W^O)
        - 각 헤드: (head_i = Attention((H^l)(W_i^Q), (H^l)(W_i^K), (H^l)(W_i^V)))
        - 투영 행렬: 각 헤드의 (W^Q_i, W^K_i, W^V_i ∈ R^{d*d/h}), 그리고 (W^O_i ∈ R^{d*d})는 학습 가능한 매개변수

📏 **투영 행렬**
<투영 행렬의 기본 개념>

- 벡터 투영: 하나의 벡터를 다른 벡터 위에 "옮겨 놓는" 과정. 예를 들어 햇빛에 의해 생성된 그림자가 있을 때, 여기서 우리 몸이 하나의 벡터이고 땅이 다른 벡터라면, 그림자는 땅 위에 투영된 우리 몸의 벡터로 볼 수 있음

- 투영 행렬의 정의: 투영 행렬 (P)는 **(R^n)에서 부분 공간 (W)로의 벡터 공간 투영을 나타내는 nxn정사각 행렬**로, (P)의 열은 (R^n)에서 (W)로의 투영을 나타냄

<투영 행렬의 응용>
예를 들어 응답 값(종속 변수 값)의 벡터를 적합된 값(또는 예측된 값)의 벡터로 매핑하는 데 사용되며, 이는
각 응답 값이 각 적합된 값에 미치는 영향을 설명함. 투영 행렬의 대각 요소는 레버리지라고 하며, 각 응답 값이 해당 관측치의 적합된 값에 미치는 영향을 설명함

- 쿼리, 키, 값의 역할
  - 같은 행렬(H^l)에서 다른 학습된 투영 행렬을 사용하여 투영됨
    - 주의 함수(Attention): (Attention(Q, K, V) = softmax((QK^T)/(sqrt(d/h)))V
    - (d/h)는 주의 분포를 더 부드럽게 만들어 극단적으로 작은 기울기를 피하기 위해 도입된 온도(temperature)
- 멀티 헤드 자기주의의 중요성
  - 모델이 다양한 표현 공간과 위치에서 정보를 동시에 처리할 수 있게 함
  - 시퀀스 내 복잡한 의존성을 더 잘 포착

Position-wise Feed-Forward Network

- 위치별 피드포워드 네트워크 구조
  - 기본 구조: 두 개의 아핀 변환으로 구성되며, 이 사이에 가우시안 오류 선형 유닛 활성화 함수가 존재

    - 수식: (PFFN(H^l) = [FFN((h_1)^l)^T; ... ;FFN((h_t)^l)^T]^T)
    - FFN 함수: (FFN(x) = GELU(xW^{(1)} + b^{(1)})W^{(2)} + b^{(2)})
    - GELU 활성화 함수: (GELU(x) = xΦ(x)), 여기서 Φ(x)는 표준 가우시안 분포의 누적 분포 함수
- 핵심 매개변수
    - 투영 행렬과 바이어스: (W^{(1)} ∈ R^{d*4d}), (W^{(2)} ∈ R^{4d*d}), (b^{(1)} ∈ R^{4d}), (b^{(2)} ∈ R^{d}는 학습 가능한 매개변수이며, 모든 위치에서 공유됨
    - 레이어별 차이: 실제로 이 매개변수들은 레이어마다 다름
- GELU 활성화 함수
    - OpenAI GPT와 BERT에서 사용된 것처럼 표준 ReLU 활성화 함수보다 더 부드러운 GELU 활성화 함수를 사용
    - GELU는 모델에 비선형성을 추가하고 다양한 차원 간의 상호작용을 가능하게 하여, BERT4Rec 모델의 성능을 향상시킴

Stacking Transformer Layer

- BERT4Rec에서 Transformer 층의 정제 과정
    - H^l = Trm(H^(l-1), ∀i∈[1,...,L]
        - l번째 Transformer 층의 출력 (H^l)을 나타냄
        - (Trm)은 Transformer 층의 연산을 의미하며, (H^{(l-1)})은 이전 층의 출력
        - 각 층이 이전 층의 출력을 입력으로 받아 처리함을 나타냄
        - (L)은 Transformer 층의 총 개수
    - Trm(H^(l-1) = LN(A^(l-1)+Dropout(PFFN(A^(l-1)))
        - (l)번째 Transformer 층의 연산 과정
        - (A^{(l-1)})은 이전 층의 자기 주의(Attention) 연산 결과
        - (PFFN)은 Position-wise Feed-Forward Network를 의미, 각 위치의 정보를 독립적으로 처리하는 밀집층
        - (Dropout)은 과적합을 방지하기 위해 일부 뉴런을 무작위로 비활성화하는 기법
        - (LN)은 Layer Normalization으로, 층의 출력을 정규화하여 학습을 안정화하고 가속화함

    - A^(l-1)=LN(H^(l-1)+Dropout(MH(H^(l-1)))
        - 자기 주의 연산의 결과를 계산하는 과정
        - (MH)은 Multi-Head Attention을 의미하며, 입력 시퀀스 내의 각 아이템 간의 관계를 모델링함
        - (Dropout)과 (LN)을 적용하여 모델의 일반화 능력을 향상시키고 학습을 안정화함
- 핵심 구성 요소
    - 잔차 연결(Residual Connection): 각 서브층의 입력과 출력을 더함으로써, 깊은 네트워크에서도 효과적인 학습이 가능하게 함
    - 층 정규화(Layer Normalization): 각 층의 출력을 정규화하여 학습 과정을 안정화 및 가속화
    - 드롭아웃: 과적합 방지를 위해 서브층의 출력에 무작위로 뉴런을 제거
- 작동 원리
    - 자기 주의 층: 입력 시퀀스 내의 아이템 간의 상호작용을 모델링
    - 포지션 와이즈 피드포워드 네트워크: 각 위치에서 독립적으로 동작하는 밀집층을 통해 추가적인 변환 제공
    - 잔차 연결과 층 정규화: 각 서브층의 출력에 잔차 연결을 적용한 후 층 정규화를 수행하여 모델의 깊이에 따른 학습 문제를 완화

3.4 Embedding Layer

- 위치 임베딩의 필요성
    - Transformer 구조는 순환(recurrence)이나 합성곱(convolution) 모듈을 사용하지 않음 ➡️ **입력 시퀀스의 정보를 직접적으로 인식할 수 없음**
    - 시퀀스 내 아이템의 **순서 정보를 모델에 주입**하기 위해 위치 임베딩이 필요

Transformer 구조와 입력 시퀀스의 순서 인식

Transformer 구조는 순환(recurrence)이나 합성곱(convolution) 모듈을 사용하지 않고, 대신 자기 주의(self-attention) 메커니즘을 기반으로 함. 이 구조는 입력 시퀀스의 순서 정보를 직접적으로 인식하지 못하는 특징이 있음

<자기주의 메커니즘과 순서 인식>
- 자기 주의 메커니즘
: Transformer는 입력 데이터의 각 부분의 중요도를 다르게 가중치를 두는 자기 주의 메커니즘을 사용함. 이 메커니즘은 모델이 입력의 가장 관련성 높은 부분에 집중할 수 있도록

- 순서 정보의 부재: Transformer 구조는 입력 시퀀스 전체를 한 번에 처리함. 이는 순환 신경망(RNN)과 달리 입력 데이터를 순차적으로 한 단어씩 처리하지 않음. 따라서, Transformer는 기본적으로 입력 시퀀스의 순서 정보를 직접적으로 인식하지 못합니다

+) CNN과 순서 정보의 관계: 합성곱 신경망은 입력 데이터의 순서 정보를 '직접적으로' 인식하는 것이 아니라 데이터 내의 공간적 패턴을 인식하여 작동함. 예를 들어 이미지 처리에서 CNN은 이미지의 픽셀 배열 순서보다는 **픽셀 간의 공간적 관계**를 중요시함

- 위치 임베딩의 도입
  - BERT4Rec에서는 **각 아이템 임베딩에 위치 임베딩을 더함**으로써 입력 시퀀스의 순서 정보를 모델에 제공함
  - 주어진 아이템 (v_i)에 대해, 그 입력 표현 ((h_i)^0)은 해당 아이템의 임베딩과 위치 임베딩을 합하여 구성

$$h_i^0 = v_i + p_i$$

  - (v_i ∈ E)는 d차원의 아이템 임베딩이고, (p_i ∈ P)는 위치 인덱스 i에 대한 d차원의 위치 임베딩
- 학습 가능한 위치 임베딩

    - BERT4Rec에서는 고정된 사인파(sinusoid) 임베딩 대신 **학습 가능한 위치 임베딩**을 사용 ➡️ **모델의 성능을 향상시키기 위함**
    - 위치 임베딩 행렬 P ∈ R^(N×d)을 통해 모델은 입력 시퀀스의 어느 부분을 처리하고 있는지 식별할 수 있음
- 최대 시퀀스 길이의 제한
    - 위치 임베딩은 모델이 처리할 수 있는 최대 시퀀스 길이 (N)에 제한을 둠
    - 입력 시퀀스의 길이 (t)가 (N)보다 클 경우, 마지막 (N)개의 아이템 ([v_{t-N+1}, ..., v_t])으로 시퀀스를 잘라내어 사용함

3.5 Output Layer

- 마스킹과 예측
    - 입력 시퀀스에서 아이템 (v_t)를 마스킹(가리기)하고, 이를 예측하기 위해 마지막 계층 (L)의 출력 (h_t^L)을 사용
    - BERT4Rec 모델이 어떻게 시퀀스 내의 아이템을 효과적으로 예측하는지 보여줌
- 피드포워드 네트워크와 GELU 활성화 함수
    - 예측을 위해 (h_t^L)에 두 계층 피드포워드 네트워크를 적용하고, 이 사이에 GELU(Gaussian Error Linear Unit) 활성화 함수를 사용
    - 최종 출력 분포를 생성하기 위해 사용
- 소프트맥스와 출력 분포
    - 최종적으로 소프트맥스 함수를 사용하여 타겟 아이템에 대한 출력 분포 (P(v))를 생성
    - 모델이 각 아이템을 예측할 확률을 나타냄

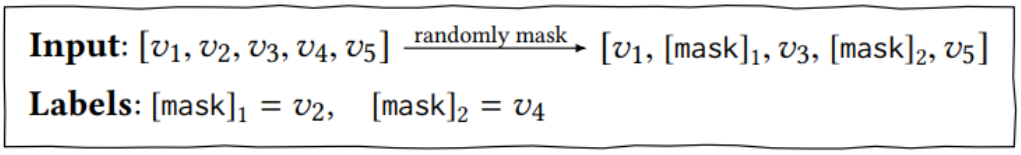
$$P(v) = \text{softmax}\left(\text{GELU}(\boldsymbol{h}_t^L \boldsymbol{W}^P + \boldsymbol{b}^P)\boldsymbol{E}^\top + \boldsymbol{b}^O\right)$$

        - W^P: 학습 가능한 투영 행렬
        - b^P, b^O: 편향 항
        - E: 아이템 집합 (V)에 대한 임베딩 행렬
        - 입력과 출력 계층에서 아이템 임베딩 행렬을 공유 ➡️ **오버피팅을 완화하고 모델 크기를 줄임**

3.6 Model Learning

Train

- 기존의 순차적 추천 모델: 단방향 순차적 추천 모델
  - 입력 시퀀스의 각 위치에서 다음 아이템을 예측함으로써 모델을 학습
  - 입력 시퀀스 ([v1, ..., vt])의 목표는 이동된 버전 ([v2, ..., vt+1])
- BERT4Rec의 접근 방식
  - 양방향 모델의 문제점: 양방향 모델에서는 왼쪽과 오른쪽 컨텍스트를 함께 고려하게 되는데, 이로 인해 **각 아이템의 최종 출력 표현이 타겟 아이템의 정보를 포함**하게 되어 미래를 예측하는 것이 너무 쉬워지고 네트워크가 유용한 것을 학습하지 못함
  - Cloze 작업을 통한 해결책
    - BERT4Rec은 효율적인 모델 학습을 위해 Cloze 작업(또는 "마스크된 언어 모델")을 새로운 목표로 적용함
    - Cloze 작업: **언어의 일부분에 단어를 제거**하고, 참가자가 빠진 단어를 채우는 테스트(빈칸 채우기), 모델은 더 깊은 언어 이해와 컨텍스트 분석을 통해 마스크된 아이템을 예측해야 함



    - 학습 단계마다 입력 시퀀스의 아이템을 무작위로 마스킹(특수 토큰 "[mask]"로 대체)하고, 왼쪽과 오른쪽 컨텍스트만을 기반으로 마스크된 아이템의 원래 ID를 예측함
- BERT4Rec의 장점: **더 많은 샘플 생성**
  - Cloze 작업을 통해 BERT4Rec은 기존의 순차적 추천 모델이 생성하는 n개의 유니크 샘플에 비해, 무작위로 k개의 아이템을 마스킹할 경우 (nCk)개의 샘플을 여러 에폭에 걸쳐 얻을 수 있음
  - 이를 통해 이전의 한계를 보완한 양방향 표현 모델을 학습할 수 있음

Test

- 훈련과 최종 추천 작업 간의 불일치 해결

    - 특수 토큰 [mask]의 추가: 사용자의 행동 시퀀스 끝에 특수 토큰 "[mask]"를 추가하고, 이 토큰의 최종 숨겨진 표현을 기반으로 다음 아이템을 예측함. 이 방법은 순차적 추천 작업과 더 잘 일치하도록 설계됨
    - 마지막 아이템 마스킹: 훈련 중 입력 시퀀스의 마지막 아이템만을 마스킹하여 샘플을 생성함. 이는 순차적 추천에 대한 미세 조정과 같은 역할을 하며 추천 성능을 더욱 향상시킬 수 있음
- 장점
    - BERT4Rec 모델이 순차적 추천 작업에 더 잘 맞도록 하여, 사용자의 미래 행동을 예측하는 데 있어 더 정확한 추천을 제공할 수 있도록 함
    - 특히 마지막 아이템만을 마스킹하는 방법은 모델이 실제 추천 시스템에서의 작업과 더 유사한 상황에서 학습하게 하여, 실제 환경에서의 성능을 개선하는 데 도움이 됨

3.7 Discussion

SASRec

- **SASRec**: SASRec은 BERT4Rec의 단방향 버전으로 볼 수 있으며, 단일 헤드 어텐션과 인과적 어텐션 마스크를 사용함. SASRec은 **시퀀스의 각 위치에 대해 다음 아이템을 예측**하는 반면, BERT4Rec은 **Cloze 목표를 사용하여 시퀀스에서 마스크된 아이템을 예측**함. 이 두 모델의 다른 아키텍처는 다른 훈련 방법으로 이어짐

CBOW & SG

- **CBOW**: CBOW는 **주변 단어 벡터의 평균**을 사용하여 타겟 단어를 예측함. BERT4Rec을 단일 셀프 어텐션 레이어로 사용하면서 아이템 임베딩을 공유하지 않으며 위치 임베딩을 제거하고 ,중앙 아이템만 마스킹한다면 CBOW의 단순화된 경우로 볼 수 있음.
- **SG(Skip-Gram)**: SG는 CBOW와 유사하게 **모든 아이템을 마스킹하고 오직 하나만 예측하는 방식**으로, BERT4Rec의 단순화된 경우로 볼 수 있음. 결과적으로 Cloze 작업은 CBOW와 SG의 목표를 일반화한 형태로 볼 수 있음.

4. Experiments

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%

- 사용된 데이터셋
  - Amazon Beauty: McAuley 등에 의해 Amazon.com에서 크롤링된 제품 리뷰 데이터셋 시리즈 중 하나. Amazon의 최상위 제품 카테고리에 따라 별도로 분리되었으며, 이 연구에서는 "Beauty" 카테고리를 사용함
  - Steam: Kang과 McAuley에 의해 수집된 데이터셋으로, 대규모 온라인 비디오 게임 배포 플랫폼인 Steam에서 가져온 것
  - MovieLens: 추천 알고리즘 평가를 위한 인기 있는 벤치마크 데이터셋으로, 이 연구에서는 두 가지 잘 알려진 버전인 MovieLens 1m (ML-1m)과 MovieLens 20m (ML-20m)을 사용함
- 데이터 전처리 과정
  - 모든 데이터셋에 대해, 모든 숫자 평점 또는 리뷰의 존재를 사용자가 아이템과 상호작용했다는 암시적 피드백 1로 변환
  - 사용자별로 상호작용 기록을 그룹화하고 이러한 상호작용 기록을 타임스탬프에 따라 정렬하여 각 사용자별 상호작용 시퀀스를 구축함
  - 데이터셋의 품질을 보장하기 위해, 적어도 다섯 개의 피드백이 있는 사용자만을 유지함

4.2 Task Settings & Evaluation Metrics

- 평가 방법
  - **Leave-one-out 평가**: 이 방법은 널리 사용되며, 다음 아이템 추천(next item recommendation) 작업에 적용됨. 각 사용자에 대해 행동 시퀀스의 마지막 아이템을 테스트 데이터로 보류하고, 마지막 바로 전 아이템을 검증 세트로 취급하며 나머지 아이템들은 훈련에 사용됨

    - **평가를 위한 샘플링**: ground truth 아이템(실제 아이템)을 테스트 세트에서 사용자가 상호작용하지 않은 100개의 무작위 샘플링된 부정적 아이템과 짝지어 평가함. 이러한 100개의 부정적 아이템은 그들의 인기도에 따라 샘플링되어 신뢰성 있고 대표적인 샘플링을 보장하며 따라서, 작업은 이 부정적 아이템들과 ground truth 아이템을 각 사용자에 대해 순위를 매기는 것이 됨

- 평가 지표
    - Hit Ratio (HR), Normalized Discounted Cumulative Gain (NDCG), Mean Reciprocal Rank (MRR)을 포함한 다양한 평가 지표를 사용하여 모든 모델의 순위 목록을 평가함
    - 각 사용자에 대해 하나의 ground truth 아이템만 있기 때문에, HR@k는 Recall@k와 동일하며 Precision@k와 비례함. MRR은 Mean Average Precision (MAP)과 동일함
    - 이 연구에서는 k = 1, 5, 10에 대한 HR과 NDCG를 보고하며, 이러한 모든 지표에 대해 값이 높을수록 성능이 더 좋다는 것을 의미함.

4.3 Baselines & Implementation Details

- 비교 대상 기준 모델
    - POP: 가장 간단한 기준 모델로, 아이템의 인기도(상호작용 횟수에 따라)에 따라 아이템을 순위 매김
    - BPR-MF: 쌍(pairwise) 순위 손실을 사용하여 암시적 피드백을 가진 행렬 분해를 최적화함
    - NCF: 행렬 분해에서 내적 대신 MLP(Multi-Layer Perceptron)를 사용하여 사용자-아이템 상호작용을 모델링함
    - FPMC: MF(행렬 분해)와 1차 MC(Markov Chains)를 결합하여 사용자의 일반적인 취향과 순차적 행동을 포착함
    - GRU4Rec: 세션 기반 추천을 위해 순위 기반 손실을 가진 GRU(Gated Recurrent Unit)를 사용함
    - GRU4Rec+: 새로운 손실 함수와 샘플링 전략을 가진 GRU4Rec의 개선
    - Caser: 순차적 추천을 위해 수평 및 수직 방향으로 CNN(Convolutional Neural Network)을 사용함
    - SASRec: 사용자의 순차적 행동을 포착하기 위해 왼쪽에서 오른쪽으로의 Transformer 언어 모델을 사용하며, 순차적 추천에서 최고의 성능을 달성함
- 구현 세부 사항

- 모델 구현: NCF, GRU4Rec, GRU4Rec+, Caser, SASRec에 대해서는 해당 저자들이 제공한 코드를 사용, BPR-MF와 FPMC는 TensorFlow를 사용하여 구현함
- 하이퍼파라미터: 모든 모델에 대해 공통적으로 고려된 하이퍼파라미터는 **숨겨진 차원 크기(d), ℓ2 정규화, 드롭아웃 비율**임. 기타 하이퍼파라미터(예: Caser의 Markov 순서)와 초기화 전략은 해당 방법의 저자들의 제안을 따르거나 검증 세트에서 조정됨
- BERT4Rec 구현: TensorFlow를 사용하여 구현하며, 모든 파라미터는 절단 정규 분포를 사용하여 초기화됨. Adam 최적화 알고리즘을 사용하여 모델을 훈련시키며, 학습률, β1, β2, ℓ2 가중치 감소, 학습률의 선형 감소 등을 설정함
- 공정한 비교를 위한 설정: **레이어 수(L), 헤드 수(h), 최대 시퀀스 길이(N)** 등을 설정함. 헤드 설정에 대해서는 각 헤드의 차원을 경험적으로 설정했으며 **마스크 비율(ρ)**은 검증 세트를 사용하여 조정됨
- 모델 훈련: 모든 모델은 사전 훈련 없이 처음부터 단일 NVIDIA GeForce GTX 1080 Ti GPU에서 배치 크기 256으로 훈련됨

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%

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

- 주요 관찰 사항
  - 비개인화된 POP 방법은 모든 데이터셋에서 가장 나쁜 성능을 보여주는데, 이는 사용자의 개인화된 선호도를 과거 기록을 사용하여 모델링하지 않기 때문임
  - **순차적 방법** (예: FPMC, GRU4Rec+)은 모든 데이터셋에서 **비순차적 방법** (예: BPR-MF, NCF)보다 일관되게 더 나은 성능을 보여줌. FPMC의 주요 개선 사항은 사용자의 과거 기록을 순차적으로 모델링하는 것으로, 이는 **순차적 정보를 고려하는 것이 추천 시스템의 성능을 향상**시키는 데 유익하다는 것을 확인시켜 줌
  - **Caser**는 특히 밀집 데이터셋인 ML-1m에서 FPMC보다 모든 데이터셋에서 더 나은 성능을 보여주며 이는 **고차 MC(Markov Chains)가 순차적 추천에 유익**하다는 것을 시사함. 그러나 고차 MC는 일반적으로 작은 순서 L을 사용하며, 순서 L과 **잘 확장되지 않음**. 이로 인해 Caser는 특히 희소 데이터셋에서 GRU4Rec+ 및 SASRec보다 성능이 떨어짐
  - **SASRec**은 GRU4Rec 및 GRU4Rec+보다 뚜렷하게 더 나은 성능을 보여줌. 이는 **자기 주의 메커니즘이 순차적 추천을 위한 더 강력한 도구**임을 시사함
  - 결과에 따르면 BERT4Rec은 모든 평가 지표에서 네 데이터셋 모두에서 모든 방법 중 가장 우수한 성능을 보임. 평균적으로 가장 강력한 기준 모델에 비해 HR@10에서 7.24%, NDCG@10에서 11.03%, MRR에서 11.46%의 개선을 보였음
- Question1(이득의 원인): **이득은 양방향 자기 주의 모델에서 오는 것인지, 아니면 Cloze 목표에서 오는 것인지?**
  - Cloze 작업을 한 번에 하나의 항목만 마스킹하도록 제한함으로써 이 두 요소의 효과를 분리하고자 함
  - 이 방법으로 BERT4Rec(1개의 마스크 사용)과 SASRec의 주요 차이점은 BERT4Rec이 왼쪽과 오른쪽 컨텍스트 모두에 조건을 부여하여 대상 항목을 예측한다는 것
  - Beauty와 ML-1m 데이터셋에서 d=256으로 결과를 보고하고, BERT4Rec(1개의 마스크 사용)은 모든 지표에서 SASRec을 크게 능가함

- 이는 순차적 추천을 위한 양방향 표현의 중요성을 보여주며 또한 Cloze 목표도 성능을 향상시킨다는 것을 나타냄

- Question2(양방향 모델의 우수성): **양방향 모델이 단방향 모델보다 왜, 어떻게 더 우수한가?**

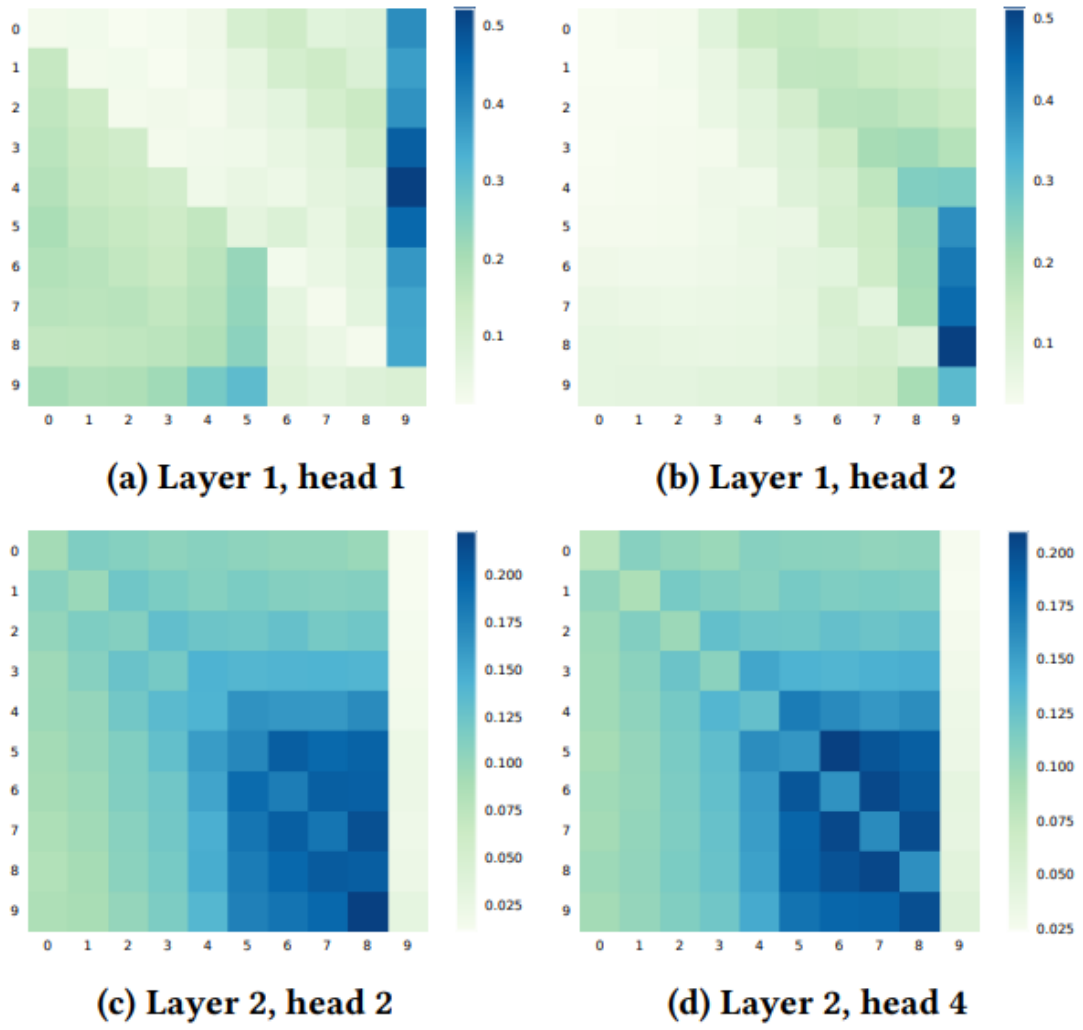


Figure 2: Heat-maps of average attention weights on Beauty, the last position "9" denotes "[mask]" (best viewed in color).

- Beauty 데이터셋에서 테스트 중 마지막 10개 항목의 평균 주의 가중치를 히트맵으로 시각화하여 의미 있는 패턴을 밝히고자 함. 양방향 모델은 단방향 모델과 달리 항목이 양쪽 모두에 주의를 기울이는 경향이 있음을 나타내며, 이는 양방향이 사용자 행동 시퀀스 모델링에 필수적이고 유익하다는 것을 시사함

4.5 Impact of Hidden Dimensionality

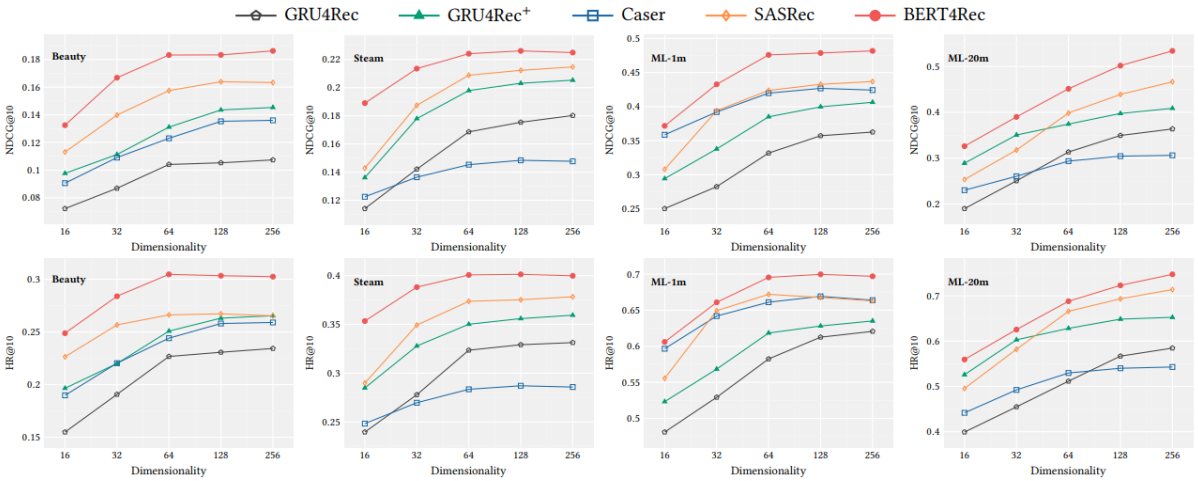


Figure 3: Effect of the hidden dimensionality d on HR@10 and NDCG@10 for neural sequential models.

- 모델 성능의 수렴
  - 각 모델의 성능은 차원성이 증가함에 따라 수렴하는 경향을 보임
  - 더 큰 숨겨진 차원성이 반드시 더 나은 모델 성능을 의미하지는 않으며, 특히 Beauty와 Steam과 같은 희소 데이터셋에서는 그렇지 않음
  - 이는 과적합(overfitting)에 의해 발생할 가능성이 있음
- Caser의 불안정성: Caser는 네 데이터셋에서 불안정한 성능을 보여, 그 유용성이 제한될 수 있음
- 자기 주의 기반 방법의 우수성: 자기 주의 기반 방법(즉, SASRec과 BERT4Rec)은 모든 데이터셋에서 우수한 성능을 달성함
- 상대적으로 작은 숨겨진 차원성에서도 우수한 성능
  - 논문의 모델은 상대적으로 작은 숨겨진 차원성에서도 모든 데이터셋에서 다른 모든 기준 모델을 일관되게 능가함
  - 해당 모델이 d≥64에서 만족스러운 성능을 달성함을 고려하여, 이후 분석에서는 d=64의 결과만을 다룸

4.6 Impact of Mask Proportion ρ

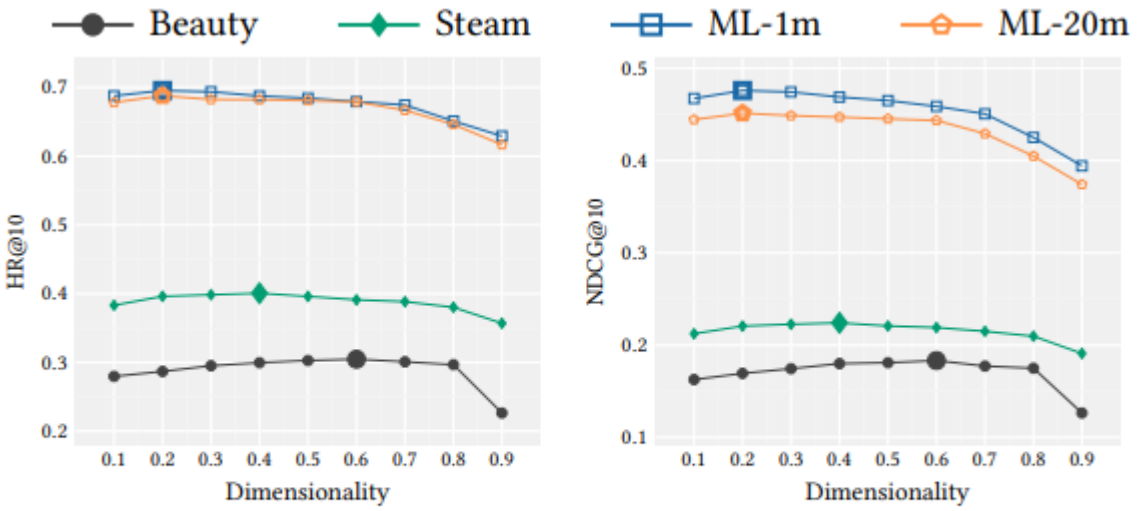


Figure 4: Performance with different mask proportion ρ on $d = 64$. Bold symbols denote the best scores in each line.

- 마스크 비율 ρ의 변화에 따른 결과
  - Figure4는 마스크 비율 ρ를 0.1에서 0.9까지 변화시키며 얻은 결과를 보여줌
  - 모든 데이터셋에서 ρ > 0.6의 결과를 고려할 때, 일반적인 패턴이 나타나며, ρ가 증가함에 따라 성능이 감소함
  - 첫 두 열의 결과에서 쉽게 볼 수 있듯이, 모든 데이터셋에서 ρ = 0.2가 ρ = 0.1보다 더 나은 성능을 보임 ➡️ **이 결과는 위에서 주장한 내용을 검증함**
- 최적의 마스크 비율 ρ
  - 최적의 마스크 비율 ρ는 데이터셋의 시퀀스 길이에 크게 의존적임
  - **짧은 시퀀스 길이**를 가진 데이터셋(예: Beauty와 Steam)의 경우, 최고 성능은 **ρ=0.6(Beauty) 및 ρ=0.4(Steam)**에서 달성됨
  - **긴 시퀀스 길이**를 가진 데이터셋(예: ML-1m 및 ML-20m)은 상대적으로 작은 **ρ=0.2**를 선호함
  - 짧은 시퀀스 데이터셋에 비해 긴 시퀀스 데이터셋에서 큰 ρ값은 예측해야 할 항목이 훨씬 더 많음을 의미하기 때문
  - 예를 들어, ML-1m과 Beauty를 예로 들면, ρ=0.6은 ML-1m에서 평균적으로 시퀀스 당 98=⌊163.5×0.6⌋ 항목을 예측해야 함을 의미하며, Beauty의 경우는 단지 5=⌊8.8×0.6⌋ 항목만 예측하면 됨. 여기서 전자는 모델 훈련이 매우 어려움

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

- BERT4Rec의 확장성 문제
  - BERT4Rec의 계산 복잡도는 레이어 당 O(n^2d)로, 길이 n에 대해 이차적임
  - Table 4의 결과는 자기 주의 레이어를 GPU를 사용하여 효과적으로 병렬 처리할 수 있음을 보여줌
- 최대 시퀀스 길이 N
  - 최대 시퀀스 길이 N이 추천 시스템의 성능과 효율성에 중요한 영향을 미치며, 특히 데이터셋의 평균 시퀀스 길이에 따라 최적의 N 값이 달라질 수 있음
  - 너무 큰 N은 모델에 불필요한 노이즈를 도입할 수 있으며, 따라서 적절한 최대 길이를 선택하는 것이 중요함
  - BERT4Rec 모델은 계산 복잡도에도 불구하고 GPU를 활용한 병렬 처리를 통해 효율적으로 작동할 수 있음

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
$L = 2, h = 2$	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**

- PE(임베딩 위치)
  - 위치 임베딩을 제거하면, 특히 긴 시퀀스 데이터셋(예: ML-1m, ML-20m)에서 BERT4Rec의 성능이 크게 감소함
  - 위치 임베딩 없이 각 항목 vi에 대한 숨겨진 표현 HL_i는 **오직 항목 임베딩에만 의존**하게 되며, 이로 인해 같은 숨겨진 표현을 사용해 다른 타겟 항목을 예측하게 되고 이는 모델을 부적절하게 만듦
  - 이 문제는 더 많은 마스크된 항목을 예측해야 하는 긴 시퀀스 데이터셋에서 더욱 심각한 것으로 나타남
- PFFN(위치별 피드포워드 네트워크)
  - 긴 시퀀스 데이터셋(예: ML-20m)은 PFFN에서 더 큰 이득을 얻음
  - PFFN의 목적 중 하나가 많은 헤드로부터의 정보를 통합하는 것이며, 이는 긴 시퀀스 데이터셋에서 선호되는 것으로 논의되었기 때문

- LN, RC, 드롭아웃
  - 이러한 구성 요소들은 주로 과적합을 완화하기 위해 도입되었음
  - 명백히 이 요소들은 Beauty와 같은 작은 데이터셋에서 더 효과적으로 작용
  - 큰 데이터셋에서의 효과를 검증하기 위해 ML-20m에서 L=4로 실험을 수행했고, 결과는 RC 없이 NDCG@10이 약 10% 감소함을 보여줌
- 레이어 수 L
  - 변환기(Transformer) 레이어를 쌓는 것은 특히 큰 데이터셋(예: ML-20m)에서 성능을 향상시킬 수 있음을 보여줌
  - 깊은 자기 주의 아키텍처를 통해 더 복잡한 항목 전환 패턴을 학습하는 것이 유용함을 검증함
  - Beauty에서 L = 4로 감소하는 것은 주로 과적합인 것으로 나타남
- 헤드 수 h
  - 긴 시퀀스 데이터셋(예: ML-20m)은 더 큰 h에서 이득을 얻는 반면, 짧은 시퀀스 데이터셋(예: Beauty)은 더 작은 h를 선호
  - 멀티 헤드 자기 주의를 사용하여 장거리 의존성을 포착하는 데 큰 h가 필수적이라는 경험적 결과와 일치

5. Conclusion And Future Work

- 향후 연구 방향
  - **아이템 특성 통합**: 단순히 아이템 ID를 모델링하는 대신 제품의 카테고리와 가격, 영화의 캐스트와 같은 풍부한 아이템 특성을 BERT4Rec에 통합하는 것이 유망한 것으로 나타남
  - **사용자 구성 요소 도입**: 사용자가 여러 세션을 가지고 있을 때 명시적인 사용자 모델링을 위해 모델에 사용자 구성 요소를 도입하는 것도 흥미로운 미래 연구 방향인 것으로 보임

논문에 대한 의견 및 의문점(꼭지)

➡️ BERT4Rec 모델이 실제 추천 시스템에 어떻게 적용될 수 있는지, 그리고 이를 통해 얻을 수 있는 실질적인 이점은 무엇인지 궁금함. 특히, 다양한 산업 분야에서의 적용 가능성을 탐색해보고 싶음. 학기 초에 유튜브 추천 시스템을 다루는 논문에 대해서도 세션 때 공부했었는데, 해당 논문의 기술과 비교했을 때 어떤 이점이 있을지 알아보고 싶음