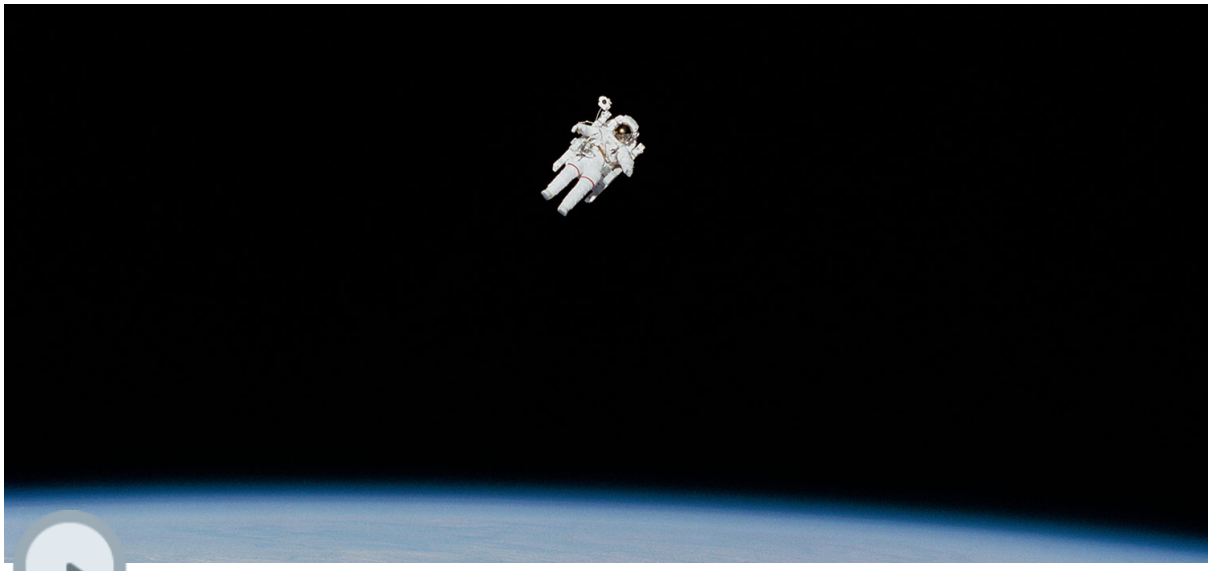


Bert4Rec : Sequential Recommendation with Bidirectional Encoder Representations from Transformers

Abstract

추천 시스템에서 historical behavior 로 부터 계속 변화하는 유저의 선호도 (preferences)를 모델링 하기는 쉽지 않다. 이전에는 추천을 하기 위해서 왼쪽에서 오른쪽으로 hidden representation으로 유저의 historical interactions을 인코딩하는 시퀀스 형태의 네트워크를 사용 했었다. 근데 한쪽 방향으로만 처리 하는 구조는 당연히 제한적이다. 그래서 User Bahavior Sequence의 모델링에 Bi-Directional Self-Attention 네트워크를 사용한 Bert4Rec을 소개한다. Cloze Objective를 Sequential 추천에 채택해서 랜덤 마스킹 된 아이템들을 좌우에 위치한 Context를 사용해서 예측한다. 네 가지의 벤치 마크 데이터 셋을 사용해서 SOTA인 Sequential 모델들에 비해 월등한 성능을 보임

Keywords

Sequential, Bi-Directional, Cloze

1. Introduction

계속해서 변화하는(Dynamic) Sequential 한 유저 행동을 모델링 하려고, 다양한 Sequential Recommendations가 제시 되었다. 그것들의 목표는 주어진 유저의 상호 작용(Interaction)으로 부터 좋아할 만한 아이템을 예측하는 것이다. RNN은 좋은 결과를 냈다. 이전 패러다임은 left-to-right sequential model을 사용해서 유저의 상호 작용 이력을 인코딩하고 이 hidden representation을 사용해서 추천을 하는 것 이었다.

left-to-right sequential model은 한계가 있다. 한쪽 방향으로 만 처리하기 때문에 이전 아이템들에 대해서만 정보가 인코딩 된다. 사실 유저의 interaction 이력 내에서 상품들의 선택은 엄격한 순서가 있다고 말할 수 없다. 다양한 외부적인 요소들을 고려하지는 않았기 때문이다. 그래서 양방향 모두를 고려 해야 한다.

텍스트 분야에서 Bert에 영감을 받아서 Deep Bi-Directional Self-Attention 모델을 사용한다. 모델을 학습하기에 데이터가 부족할 수 있는데, Cloze Task를 도입해 기존의 다음 아이템을 예측하는 것 대신에 랜덤 하게 표현한 어느 정도의 아이템 (기존의 상품들을 [mask]로 바꾼다)을 입력 시퀀스로 사용하고, 주변의 Context 아이템들을 이용해서 마스킹 된 아이템을 예측한다. 많은 학습 데이터 샘플을 만들 수 있다는 장점이 있지만, Cloze Task의 단점으로는 마지막에 우리가 해야할 추천과 동일한 Task가 아니라는 점이다. 그래서 테스트를 할 때, 입력 시퀀스의 마지막에 [mask]를 넣어서 예측해야 할 아이템을 표시한다.

모델에 네 가지의 데이터 셋을 적용해서 다양한 SOTA 베이스 라인에 비해 월등한 성능을 낸다는 것을 보인다. 논문의 기여는 다음과 같다.

- Deep Bidirectional Sequential Model과 Cloze를 처음 제안했다
- 네 가지의 데이터 셋으로 성능을 입증했다
- 제안된 모델에서 ablathon study를 수행했다

2. Related work

General Recommendation

Collaborate Filtering , Matrix Factorization, Item-base neighborhood,Restricted Boltzmann Machine, Neural Collaborative Filtering, AutoRec and CDAE

Sequential Recommendation

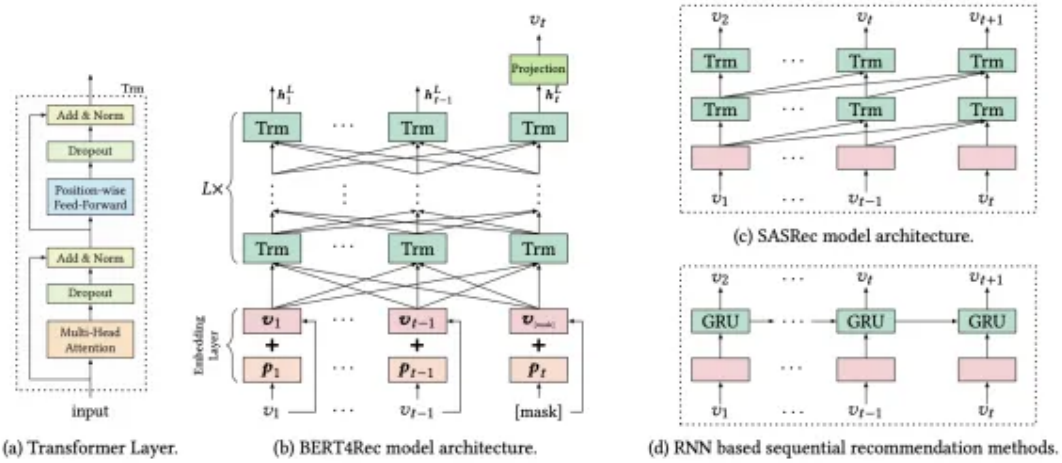
Markov Chains, Markov Decision Processes, Factorizing Personalized Markov Chains. RNN, GRU, LSTM, Session Based GRU with ranking loss(GRU4REC),

Dynamic Recurrent Basket Model(DREAM), user-based GRU, attention-based GRU(NARM), Convolutional Sequence Model to learn sequential patterns using both horizontal and vertical convolutional filters. Memory Network to improve sequential recommendations

Attention Mechanism

SASRec

3. Model Architecture



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

Figure 2: 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 predicts next item sequentially.

3.1 Problem Statement

u는 유저 집합, v는 아이템 집합이다. $S_u = [u_1^{(u)}, ..., u_t^{(u)}, ..., u_{n_u}^{(u)}]$ 는 유저 u의 시간 순서의 상호 작용 시퀀스다. n_u는 시퀀스의 길이이고 n_{u+1}의 타임 스탬프의 아이템을 예측해야한다. 유저 u에게 모든 가능한 아이템들에 대한 가능성을 모델링한 것을 공시으로 만들면 아래와 같다.

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

3.2 Model Architecture

L개의 Transformer Layer를 가진다. 이전 Layer의 모든 위치에서 정보를 교환하면서 트랜스포머 Layer와 병렬로 각 Layer에서 반복적으로 표현을 수정한다.

3.3 Transformer Layer

주어진 길이 t의 입력 시퀀스에 대해 반복적으로 h_i^l를 각 Layer에서 계산한다. 모든 위치에서의 각 i에 Transformer를 적용한다. $h_i^l \in R^d$를 쌓아서 $H^l \in R^{t \times d}$ 매트릭스를 만든다. 실제로는 모든 포지션에서 동시에 attention 함수를 계산한다. Transformer Layer는 Multi-Head Self-Attention과 Position-Wise Feed-Forward Network의 Sub-Layer 둘로 구성된다

Multi-Head Self-Attention

Attention 메커니즘은 이제 다양한 Task들에서 시퀀스 모델링의 필수적인 파트가 되었다. representation 쌍들 간의 의존성을 시퀀스 내에서의 거리를 고려하지 않고 포착하게 함으로써. 선행 연구에서 다른 위치 들에서 서로 다른 representation의 부속 공간 들로부터 정보를 함께 만드는 데에 강점이 있다. 그래서 single attention 함수가 아닌 multi-head self-attention을 사용한다. multi-head attention은 우선 선형적으로 H^l을 h 라는 하위(부속) 공간 들로 투영한다. 서로 다르게 다양하게 그리고 학습 가능한 선형적 투영으로 (Linear Projections). 그 다음 h attention 함수들을 병렬적으로 적용해서 결합되고 한번 더 투영한 output representation을 만든다.

$$MH(H^l) = [head_1, head_2; ...; head_h]W^O$$

$$head_i = Attention(H^l W_i^Q, H^l W_i^K, H^l W_i^V)$$

각 head의 projection matrices는 $W_i^Q \in R^{d \times d/h}$, $W_i^K \in R^{d \times d/h}$, $W_i^V \in R^{d \times d/h}$ 그리고 $W_i^O \in R^{d \times d/h}$ 들은 학습 가능한 파라미터 들이다. 여기서는 layer subscript l 은 생략한다. 사실 projection matrixes는 layer들 간에 공유는 되지 않는다. Attention 함수는 Scaled Dot Product Attention을 사용한다.

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

quert Q, key K, value V 들은 매트릭스 H^l에서 투영되지만 서로 다른 학습된 투영 매트릭스로 부터 투영된다. $temperature\sqrt{d/h}$는 더 부드러운 attention 분포를 사용해서 극도로 작은 gradients를 없애기 위해 사용한다.

Position-wise Feed-Forward Network

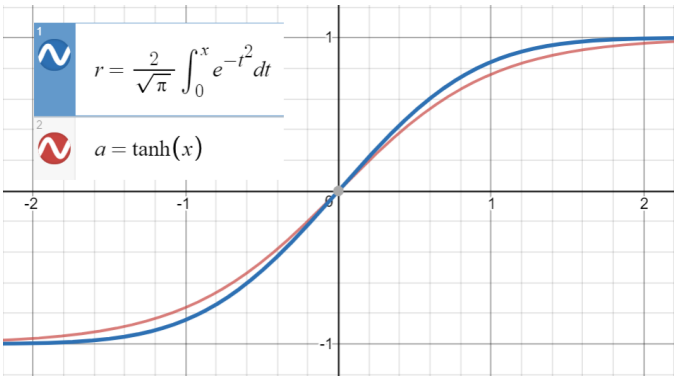
self-attention sub-layer는 linear-projection에 기반 한다. 모델에 non-linearity와 서로 다른 차원들 간에 상호작용을 부여하기 위해, self-attention sub-layer에 Position-wise Feed-Forward Network를 output으로 d각 포지션 별로 분리해서 동일하게 적용한다. 이건 Gaussian Error Linear Unit (GELU)와 함께 두 개의 affine transformation로 구성된다.

$$PFFN(H^l) = [FFN(h_1^l)^T; ...; FFN(h_t^l)^T]^T$$

$$FFN(x) = GELU(xW^{(1)} + b^{(1)})W^{(2)} + b^{(2)}$$

$$GELU(x) = x\Phi(x)$$

$\Phi(x)$는 표준 gaussian 분포의 누적 분포 함수고, $W^{(1)} \in R^{d \times 4d}, W^{(2)} \in R^{4d \times d}$ 그리고 $b^{(2)} \in R^d$는 학습 가능한 parameter 들이고 모든 위치에 공유된다. layer subscript l은 편의를 위해 생략한다. 사실 이러한 parameter 들은 layer에 따라서 달라진다. OpenAI GPT와 Bert를 따라서 Relu 대신에 Gelu를 활성화 함수로 사용한다.



Stacking Transformer Layer

self-attention 메커니즘을 사용함으로 전체 유저 행동 시퀀스에 대해서 item-item 상호작용을 쉽게 잡아 낼 수 있다. 그럼에도 불구하고 self-attention으로 더 복잡한 아이템 전환을 더 잘 학습할 수 있다. 그러나 네트워크는 깊어질 수록 학습이 어려워 진다. 그래서 residual connection을 sub-layer 들에 적용하고, layer normalization을 적용한다. 게다가 normalize 되기 전에 drop-out을 적용한다. 그래서 sub-layer의 output은 LN(x + Dropout(sublayer(x)))가 되고 Layer Normalization으로 모든 입력 들을 모든 은닉 유닛에 대해 normalize 해서 네트워크 학습을 안정화하고 빠르게 한다. 요약 하자면, BERT4REC의 각 layer 들의 hidden representations들을 다음과 같이 정의한다.

$$H^l = Trm(H^{l-1}), \forall i \in [1, ..., L]$$

$$Trm(H^{l-1}) = LN(A^{l-1} + Dropout(PFFN(A^{l-1})))$$

$$A^{l-1} = LN(H^{l-1} + Dropout * MH(H^{l-1}))$$

3.4 Embedding Layer

recurrence 나 convolution 모듈 없이, Transformer 레이어 Trm은 입력 시퀀스의 순서를 인식하지 못한다. 입력의 순서 정보를 사용하기 위해, 트랜스포머 레이어 스택 들의 바닥에 위치하는 입력 아이템 Embedding에 Positional Embeddings 를 삽입 해야 한다. 주어진 아이템 v_i에 대해 입력 representation h_i^0 은 해당하는 아이템과 positional embedding에 더해서 생성 된다.

$h_i^0 = v_i + p_i$

$v_i \in E$는 각 아이템 v_i의 d 차원의 embedding. $p_i \in P$ 는 Positional Index i에 대한 d 차원의 positional embedding, 학습 가능한 positional embedding이 성능이 더 좋아서 고정된 sigmoid embedding 대신에 사용한다. 근데 이렇게 하는 것이 모델이 다룰 수 있는 최대 시퀀스 길이 N에 제한을 준다. 그래서 t > N 이면, 입력 순서 $[v_1, ..., v_t]$를 마지막 N 아이템들 $[v_{t-N+1}^u, ..., v_t]$로 잘라 낸다.

3.5 Output Layer

레이어 L 다음에 이전 레이어 내의 모든 position들에 대해서 계층적으로 정보가 교체 되어서, 입력 시퀀스의 모든 아이템 들의 마지막 출력 H^L을 가지게 된다. 시간 t에서의 아이템을 v_t로 마스킹 하면, h_t^L에 기반 해서 마스킹 된 아이템 v_t를 예측한다. 타겟 아이템들에 대해 출력 분포를 생성하기 위해서 구체적으로는, GELU 활성화 함수를 사용한 두 feed-forward network 계층을 적용한다.

$P(v) = softmax(GELU(h_t^L W^P + b^P)E^T + b^O)$

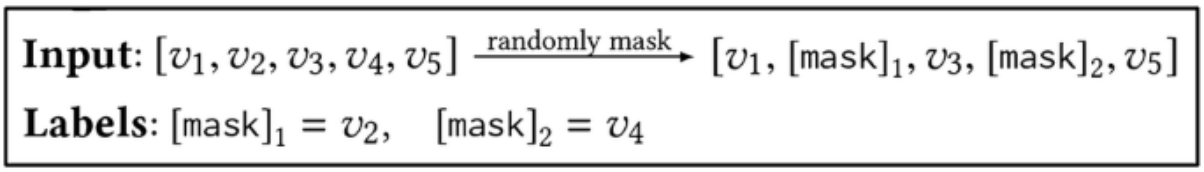
W^P는 학습 가능한 투영 매트릭스 고, b^P, 그리고 b^O는 bias 다. $E \in R^{|V| \times d}$는 아이템 셋 V 의 Embedding 매트릭스 다. 입력과 출력 Layer와 아이템 Embedding을 공유하고 오버 피팅을 방지하고 모델 사이즈를 줄인다.

3.6 Model Learning

Training.

보통의 단 방향 sequential 추천 모델들은 학습을 입력 시퀀스 내의 각 위치에서의 다음 아이템을 예측한다. 구체적으로는, 입력 시퀀스의 target은 한번 shift된 버전이다. 그러나 bi-directional 모델에서 왼쪽과 오른쪽 Context가 함께 공유된 각 아이템의 마지막 출력 표현에는 대상 항목의 정보가 포함될 수 있다. 이것이 미래를 예측하는 것을 당연하게 하고 네트워크가 다른 유용한 것을 배우지 못하게 된다. 이 이슈의 간단한 해결책은 t-1 개의 샘플 들을 만드는 것이다. 원래 길이가 t인 동작 시퀀스에서 $([v_1], v_2)$ 와 $([v_1, v_2], v_3)$와 같은 다음 항목으로 서브-시퀀스로 만든 다음 historical 서브-시퀀스를 bi-directional 모델로 인코딩해서 타겟 아이템을 예측한다. 하지만 이 접근 방법은 오래 걸리고 resource도 많이 잡는다. 시퀀스 내의 각 위치를 위해 새로운 샘플을 만들어야 하고 분산 해서 예측 해야 하기 때문이다.

효율적으로 제안된 모델을 학습하기 위해, Cloze Task를 적용했다. (Masked Language Model). 각 학습 과정에서, 입력 시퀀스 내에서 모든 아이템 중 p 비중을 임의로 마스킹 한다.("[mask]") 그리고 왼쪽과 오른쪽 context를 사용해서 기존 id를 예측한다.



"[mask]"에 해당하는 마지막 히든 벡터 들은 아이템 셋에 대해 출력 softmax로 공급 된다. 결론적으로 각 마스킹 된 아이템 $S_u^{'}$에 대한 손실을 마스크 된 대상의 Negative Log-Likelihood로 정의한다.

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

$S_u^{'}$는 유저 행동 히스토리 S_u, S_u^m의 마스크 버전이다. v_m^*은 마스킹 된 아이템 v_m의 진짜 아이템이다. Cloze 태스크는 모델 학습을 하기 위해 더 많은 샘플 들을 만들 수 있다.

Test

"[mask]"의 특별한 토큰을 유저의 행동 시퀀스의 마지막에 삽입한다. 이 토큰의 마지막 hidden representation에 기반 해서 다음 아이템을 예측한다.