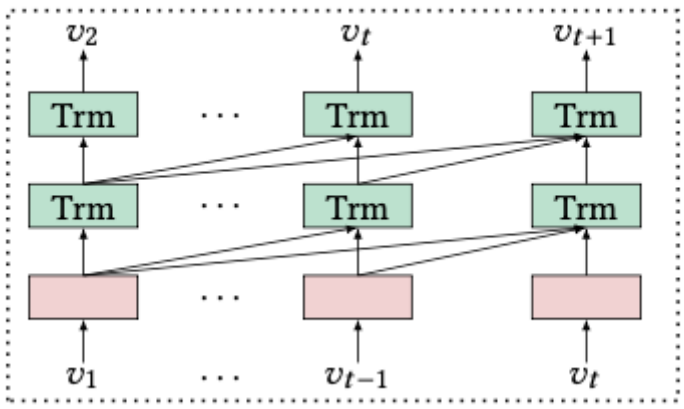
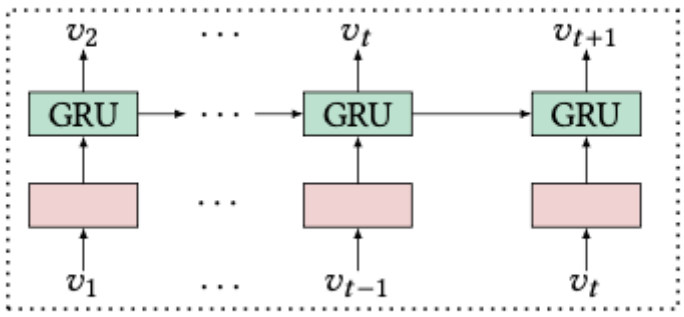
BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer

Introduction

- Sequential Recommendation System은 유저의 과거 행동 패턴을 학습하여 이후에 유저가 구매할 아이템을 추천하는 모델
- 기존의 Sequential Recommendation System (SASRec, GRU4Rec)은 단방향 모델 (left-to-right unidirectional model)로 과거에 구매한 아이템 정보 만을 학습하여 유저의 이전 행동패턴 만을 모델이 고려하기 때문에 성능에 한계가 생김
    - 표현력 한계
    - 순서가 무의미한 경우도 순서를 정함
- 본 논문은 BERT의 Masked Language Model을 활용하여 다음 아이템을 예측하는 **BERT4Rec**을 제안



(c) SASRec model architecture.



(d) RNN based sequential recommendation methods.

기존 단방향(unidirectional) 추천 시스템 모델

BERT4Rec

Problem Statement

- 유저 벡터

$$\mathcal{U}=\{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$$

- 아이템 벡터

$$\mathcal{V}=\{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$$

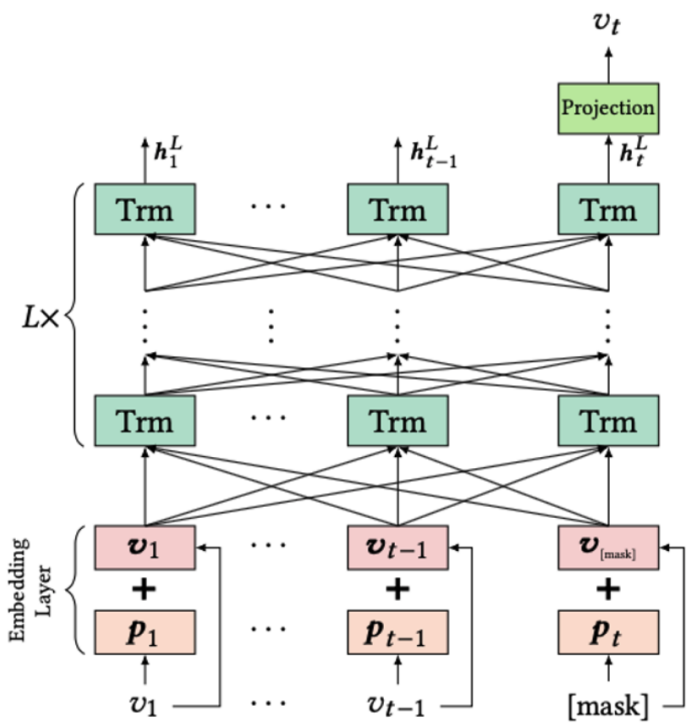
- Interaction sequence: 특정 유저 u가 t 시점에 상호작용하는 아이템 시퀀스

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

- 이전 n_u 시점까지의 정보를 통해 n_u+1 시점에서 유저가 어떤 아이템과 상호작용할지 예측하는 문제

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

Model Architecture



- BERT4Rec은 *L*개의 양방향 Transformer layers로 이루어짐
- 병렬적으로 이전 layer에 존재하는 모든 위치 정보들을 상호 교환하여 각 위치의 representation을 반복 수정함으로써 학습을 진행
- 한쪽으로만 정보를 전달하는 기존의 RNN 기반의 방법과 달리 Self-Atttention을 활용하여 거리가 멀어져도 의존도를 반영할 수 있음

Transformer Layer

- Transformer layer는 multi-head self-attention과 position-wise feed-forward network로 구성
- **Mult-Head Self Attention**

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

$$\mathrm{head}_i=\mathrm{Attention}(H^lW_i^Q,H^lW_i^K,H^lW_i^V)$$

  - Scaled Dot-Product Attention 수식

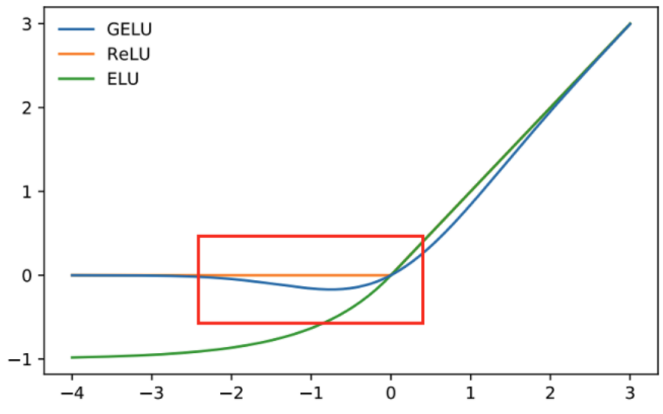
$$\mathrm{Attention}(Q,K,V)=\mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d/h}}\right)V$$

- **Position-Wise Feed Forward Network**

$$\mathrm{PFFN}(H^l)=\left[\mathrm{FFN}(h_1^l)^\top;\ldots;\mathrm{FFN}(h_t^l)^\top\right]^\top$$
$$\mathrm{FFN}(x)=\mathrm{GELU}(xW^{(1)}+b^{(1)})W^{(2)}+b^{(2)}$$
$$\mathrm{GELU}(x)=x\Phi(x)$$

  - Multi-Head Self-Attention을 통해 나온 결과 값은 linear projection 이기 때문에 FC Layer에 GELU (Gaussian Error Linear Unit) 함수를 활성화 함수로 사용하여 비선형성을 줌

- $\phi(x)$는 표준 정규 분포의 누적 분포 함수이며 학습가능한 파라미터로 모든 위치에서 사용됨
- GELU는 BERT 모델에 사용되는 활성화 함수로 음수에서 미분이 가능함



- 여러 Transformer layer를 쌓아 너무 깊어지면 학습하기 어려우므로 residual connection과 layer normalization, dropout을 함께 사용

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

Embedding Layer

- Transformer에는 위치 정보가 포함되지 않기 때문에 Positional Embedding Layer를 추가 함

$$\boldsymbol{h}_i^0 = \boldsymbol{v}_i + \boldsymbol{p}_i$$

- 모델이 처리할 수 있는 최대 길이를 제한하여 길이 N을 넘는 sequence는 마지막 N개에 대해서만 학습을 진행

Output Layer

- L개의 layer를 통과하고 나면 다음과 같은 결과물을 얻을 수 있음

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

Model Learning

- 모델을 학습할 때는 BERT처럼 [MASK]를 씌우고 이를 맞히게 하는 Cloze Task (Masked Language Model)를 진행하며, [MASK]가 여러 개인 경우네는 순차적으로 진행함

$$\textbf{Input}: [v_1,\ v_2,\ v_3,\ v_4,\ v_5] \xrightarrow{\text{random mask}} [v_1, \mathbf{mask}_1, v_3, \mathbf{mask}_2, v_5]$$

$$\textbf{Labels}: \mathbf{mask}_1 = v_2, \mathbf{mask}_2 = v_4$$

- Negative Log Likelihood를 이용하여 label과 예측값 간의 loss를 정의

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

- S'_u는 유저 S_u에 maksed를 한 것이고 v^*_m은 masked된 item으로 v_m의 target item 임
- Cloze task의 또 하나의 장점은 하나의 sequence에서 여러 학습 샘플을 뽑을 수 있음
- 실제로 테스트 단계에서는 마지막 토큰만 [MASK]처리해서 추론함
  - 학습 단계와 테스트 단계에서의 Task의 목적이 다름
  - Test task 와 학습 일치를 위해, Masked pre-training 후 마지막 아이템을 Mask 처리 후, 예측 추가로 학습

Discussion

- SASRec 비교
  - SASRec은 BERT4Rec의 단반향 버전 (left-to-right unidirectional version)으로 sequence의 각 위치에 대해 다음 아이템을 예측하는 반면 BERT4Rec는 mask된 항목을 예측
- CBOW & Skip-Gram 비교
  - 단순한 버전의 BERT4Rec로 볼 수 있음
    - 하나의 self-attention layer (attention weight uniform)
    - unshared item embedding
    - position embedding 사용 안함, mask는 sequence중 한 개만 적용
  - CBOW & Skip-Gram은 sequence의 representation을 학습 하는 것이 아닌 단어 (아이템)의 representation를 잘 학습하는 것이 목적이기 때문에 BERT4Rec와 차이가 있음
- BERT 비교
  - Bert는 NLP를 위한 pre-trained 모델이라 end-to-end 모델이 아닌 반면, Bert4Rec은 도메인 별 데이터셋이 상이하므로 end-to-end 모델
  - Bert4Rec은 유저의 sequence를 하나로 보기 때문에 next sentence loss와 segment embedding 사용 안 함

Experiments

Datasets

- 4개의 데이터 셋 (Beauty, Steam, ML-1m, ML-20m)으로 실험을 진행
- Density는 유저 아이템의 수에 비해 interaction이 얼마나 있는가
  - Sparse : 유저의 아이템은 많은데 interaction이 적음 (Beauty, Steam)
  - Dense : 유저의 아이템과 interaction이 둘 다 많음 (ML-1m, ML-20m)

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%

데이터 통계량

Evaluation

- **HR@k** : 내가 추천한 상위 k개 아이템 중 다음 아이템이 있는 비율
- **NDCG@k** : Ground Truth의 순위가 Top K개를 추천했을 때 몇 위에 존재하는지 확인
  - 내가 추천한 아이템들의 순위 점수 / k개 아이템의 이상적 순위 점수
- **MRR** : Ground Truth가 몇 순위에 있는지 위치 별로 점수 획득

Baseline 비교

- BERT4Rec이 모든 데이터셋에서 좋은 성능을 보이고 있음
- sequential recommendation 방법(FPMC, GRU4Rec)은 non-sequential(BPR-MF와 NCF)를 능가함
  - POP, BPR-MF, NCF, FPMC (not deep learning model), Caser (CNN model)
- POP, BPR-MF, NCF, FPMC는 딥러닝 모델이 아니지만 다른 모델들과 큰 성능 차이는 없는 것으로 나타남

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%

모델 성능 비교

성능 향상의 원인 : **Self Attention vs Masked Learning ?**

- 양방향을 사용할 때와, Mask 수를 조절했을 때의 성능을 비교함
- 양방향을 통해 성능이 유의미하게 향상되었으며, 적당한 수의 Mask는 성능 향상에 도움이 됨

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

- 단방향으로 모델을 학습하는 것보다 양방향으로 유저의 행동 패턴을 학습하는 것이 더 좋은 추천 성능을 나타냄
- 이외에도 하이퍼파라미터 (Hidden Dimension, Mask Proportion, Maximum Sequence Length) 변화에 따른 성능 변화, Ablation study에 대해 실험 진행

참고 자료

https://arxiv.org/abs/1904.06690

https://www.youtube.com/watch?v=PKYVHGrSO2U

https://greeksharifa.github.io/machine_learning/2021/12/12/Bert4Rec/

https://hyunlee103.tistory.com/115

https://dhgudxor.tistory.com/9