BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer

2019년 ACM에서 발표, Sequential Recommendation에서도 좋은 성능을 보임

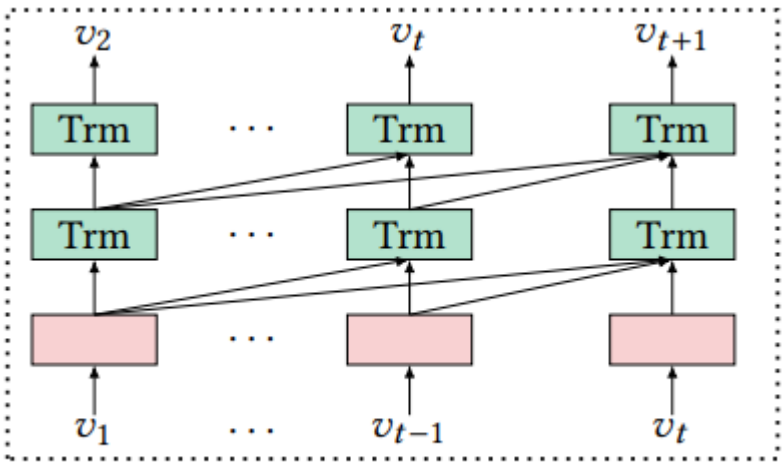
사용자 구매 패턴을 양방향으로 학습하여 이전의 단방향 추천모델인 SASRec, GRU4Rec 보다 좋은 성능

Sequential Recommendation System

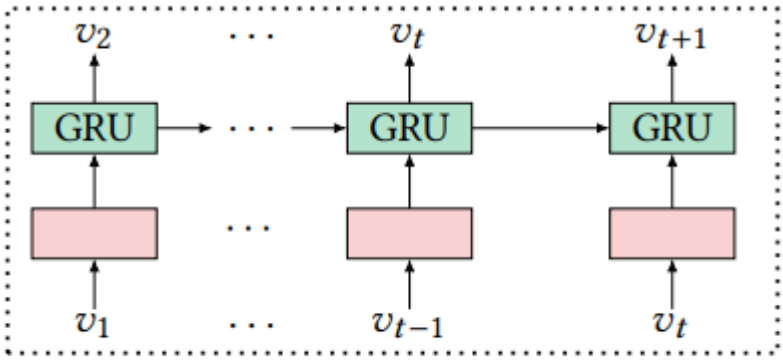
Sequential Recommendation System은 유저의 과거 행동 패턴을 학습하여 이후에 유저가 구매할 앙이템을 추천하는 모델로, NLP task와 많은 연관을 가짐

입력된 단어를 바탕으로 다음 단어를 예측하는 NLP task가 Sequential Recommendation System과 유사하기 때문에 NLP 분야의 모델이 Sequential Recommendation System에서도 사용

Introduction



(c) SASRec model architecture.

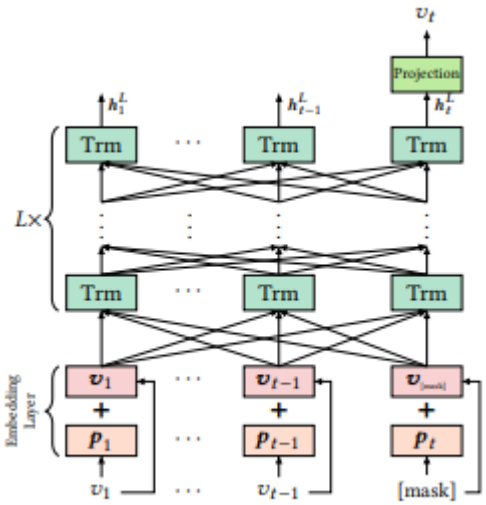


(d) RNN based sequential recommendation methods.

SASRec, GRU4Rec등 단방향 추천모델의 경우 유저의 이전 행동패턴만을 고려하기 때문에 과거에 구매했던 아이템의 정보만으로 학습

유저의 행동패턴을 단방향으로 학습하는 것이 아닌 양방향으로 학습하는 BERT4Rec을 제안

BERT4Rec Model



(b) BERT4Rec model architecture.

BERT4Rec 모델은 Embedding Layer, Transformer Layer, Output Layer로 구성

Cloze task

모델을 양방향으로 학습하기 위해 BERT의 학습 방법처럼 유저의 행동 시퀀스에 대해 [Mask] 토큰을 사용하여 앞 뒤 정보로부터 [Mask]의 정보를 파악

모델이 양방향으로 학습하면서 target 아이템의 정보를 직접 보게 되는 information leakage가 발생할 수 있기 때문에 [Mask]를 사용

기존의 단뱡향 모델은 n 크기 만큼의 시퀀스 길이가 있다고 가정할 때, 모델 학습 시 시퀀스 마다 마지막 아이템을 맞추는 방식으로 n개만큼 학습 샘플을 구할 수 있음

BERT4Rec 모델은 랜덤하게 마스킹을 처리하는 k개 역시 학습 샘플로 설정하여 기존 모델보다 많은 $\binom{n}{k}$개의 샘플을 학습에 사용할 수 있음

이는 모델이 더욱 좋은 추천 성능을 나타내는 역할

[Embedding Layer]

아이템의 정보와 아이템의 위치정보를 더해 [Mask]와 함께 Transformer 모델의 입력으로 들어감, 유저의 시퀀스 길이가 전체 시퀀스 길이인 하이퍼파라미터 N보다 크면 잘라내고, N보다 작으면 제로패딩을 진행

[Transformer Layer]

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

기존 Transformer와 동일하게 MultiHead Attention, Point-Wise Feed Forward를 사용하고, 레이어의 수 L만큼 반복연산을 수행

[Output Layer]

$$
P(v) = \mathrm{softmax}\big(\mathtt{GELU}(\boldsymbol{h}_t^L \boldsymbol{W}^P + \boldsymbol{b}^P)\boldsymbol{E}^\top + \boldsymbol{b}^O\big)
$$

Transformer 모델로부터 받은 Final Output으로부터 softmax를 거쳐 [Mask] 토큰의 확률값을 구함

아이템에 대한 Embedding Matrix는 공유된 Embedding Matrix를 사용하여 모델의 Overfitting과 size를 줄임

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

모델의 입력에 유저의 Sequence 중 p의 확률만큼 [Mask]를 수행하여 들어가게 되고, 출력으로는 [Mask]된 아이템의 확률값이 나옴

Mask의 비율 p의 경우 데이터셋마다 다르지만, 너무 큰 값으로 설정할 경우 성능이 좋지 않음

모델의 Loss의 경우 Negative-Log-Likelihood를 사용하여 [Mask]가 반영된 유저의 행동 Sequence가 주어졌을 때, [Mask] 아이템과 실제 [Mask] 아이템을 비교하여 낮은 확률을 가질 수록 weight를 더 많이 업데이트하는 방식으로 학습을 진행

실제 테스트 단계에서는 유저의 행동 Sequence 마지막 부분에 [Mask]를 사용하여 이후 행동에 추천할 아이템의 확률값을 계산, 학습 과정에서 해당 모델을 통해 성능이 증가

Experiments

4가지 데이터 셋을 통해 실험을 진행

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%

전체적인 모델의 성능 비교 결과를 보게 되면 저자가 제안한 BERT4Rec이 모든 데이터셋에서 좋은 성능

SASRec 모델의 경우에도 제안 모델보다는 낮은 성능을 보이지만 이전의 Sequential 추천 모델에 비해 좋은 성능

단방향으로 모델을 학습하는 것보다 양방향으로 유저의 행동 패턴을 학습하는 것이 더 좋은 추천 성능

Conclusion

기존 단방향 추천모델의 한계를 극복하기 위해 유저의 행동 시퀀스에 [Mask]를 반영한 양방향 학습모델인 BERT4Rec 모델을 제안

NLP 분야에서 BERT는 문장들의 representation을 학습하기 위해 pre-training의 목적으로 주로 사용되며 next sentence loss, segment embeddings도 같이 사용

하지만 BERT4Rec은 유저의 행동 패턴만을 바탕으로 시퀀셜 추천을 위한 end-to-end 방식의 추천 모델로 차별점, Sequential recommendation system에서 좋은 성능