BERT4Rec

김태공 | 11.12.2022

tada

Contents

1. Introduction

BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang
Alibaba Group, Beijing, China
{ofey.sf,yanhan.lj,joshuawu.wujian,changhua.pch,hc.lx,santong.oww,jiangpeng.jp}@alibaba-inc.com

ABSTRACT

Modeling users' dynamic preferences from their historical behaviors is challenging and crucial for recommendation systems. Previous methods employ sequential neural networks to encode users' historical interactions from left to right into hidden representations for making recommendations. Despite their effectiveness, we argue that such left-to-right unidirectional models are sub-optimal due to the limitations including: ***a)*** unidirectional architectures restrict the power of hidden representation in users' behavior sequences; ***b)*** they often assume a rigidly ordered sequence which is not always practical. To address these limitations, we proposed a sequential recommendation model called **BERT4Rec**, which employs the deep bidirectional self-attention to model user behavior sequences. To

users' current interests are intrinsically dynamic and evolving, influenced by their historical behaviors. For example, one may purchase accessories (*e.g.*, Joy-Con controllers) soon after buying a Nintendo Switch, though she/he will not buy console accessories under normal circumstances.

To model such sequential dynamics in user behaviors, various methods have been proposed to make *sequential recommendations* based on users' historical interactions [15, 22, 40]. They aim to predict the successive item(s) that a user is likely to interact with given her/his past interactions. Recently, a surge of works employ sequential neural networks, *e.g.*, Recurrent Neural Network (RNN), for sequential recommendation and obtain promising results [7, 14, 15, 56, 58]. The basic paradigm of previous work is to encode

2019 CIKM Accept Paper

1. Introduction

tada

Why use BERT!?!?!

BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang
Alibaba Group, Beijing, China
{ofey.sf,yanhan.lj,joshuawu.wujian,changhua.pch,hc.lx,santong.oww,jiangpeng.jp}@alibaba-inc.com

21 Aug 2019

ABSTRACT

Modeling users' dynamic preferences from their historical behaviors is challenging and crucial for recommendation systems. Previous methods employ sequential neural networks to encode users' historical interactions from left to right into hidden representations for making recommendations. Despite their effectiveness, we argue that such left-to-right unidirectional models are sub-optimal due to the limitations including: ***a)*** unidirectional architectures restrict the power of hidden representation in users' behavior sequences; ***b)*** they often assume a rigidly ordered sequence which is not always practical. To address these limitations, we proposed a sequential recommendation model called **BERT4Rec**, which employs the deep bidirectional self-attention to model user behavior sequences. To

users' current interests are intrinsically dynamic and evolving, influenced by their historical behaviors. For example, one may purchase accessories (*e.g.*, Joy-Con controllers) soon after buying a Nintendo Switch, though she/he will not buy console accessories under normal circumstances.

To model such sequential dynamics in user behaviors, various methods have been proposed to make *sequential recommendations* based on users' historical interactions [15, 22, 40]. They aim to predict the successive item(s) that a user is likely to interact with given her/his past interactions. Recently, a surge of works employ sequential neural networks, *e.g.*, Recurrent Neural Network (RNN), for sequential recommendation and obtain promising results [7, 14, 15, 56, 58]. The basic paradigm of previous work is to encode

2019 CIKM Accept Paper

2. History

GRU4Rec(GRU) → **SASRec(self-attention) → BERT4Rec(BERT)**

2. History

GRU4Rec

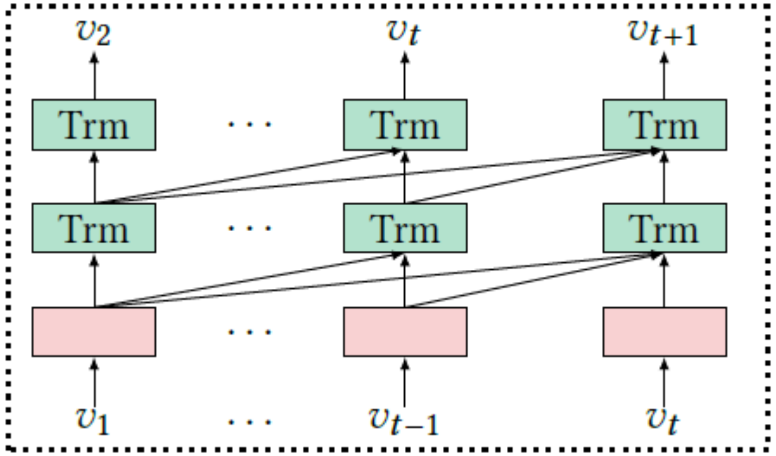
GRU4Rec(GRU) → SASRec(self-attention) → BERT4Rec(BERT)

- sequential calculation
  - long learning time
- RNN Problem
  - Long-Term Dependency

2. History

SASRec

GRU4Rec(GRU) → SASRec(self-attention) → BERT4Rec(BERT)



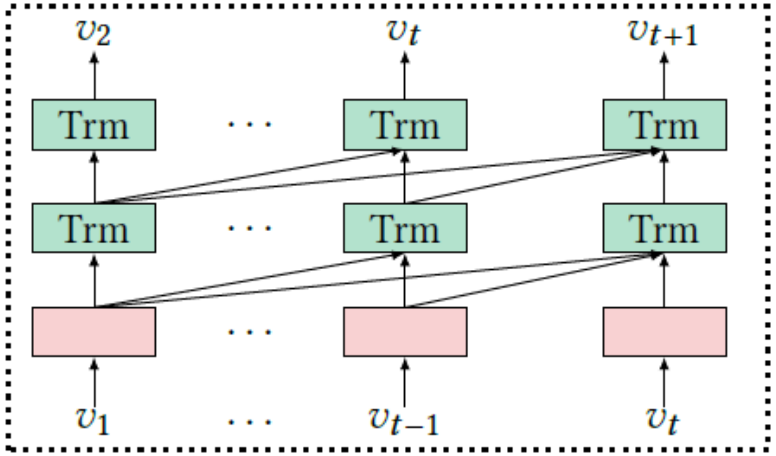
(c) SASRec model architecture.

- Transformer's self-attention usage model
  - Masked self-attention
- Predict the present with past information (Problem)

2. History

SASRec

GRU4Rec(GRU) → $\boxed{\text{SASRec(self-attention)}}$ → BERT4Rec(BERT)



(c) SASRec model architecture.

- Transformer's self-attention usage model
  - Masked self-attention
- Predict the present with past information (Problem)
  - **Why is this a disadvantage?**

2. History

SASRec

GRU4Rec(GRU) → **SASRec(self-attention)** → BERT4Rec(BERT)

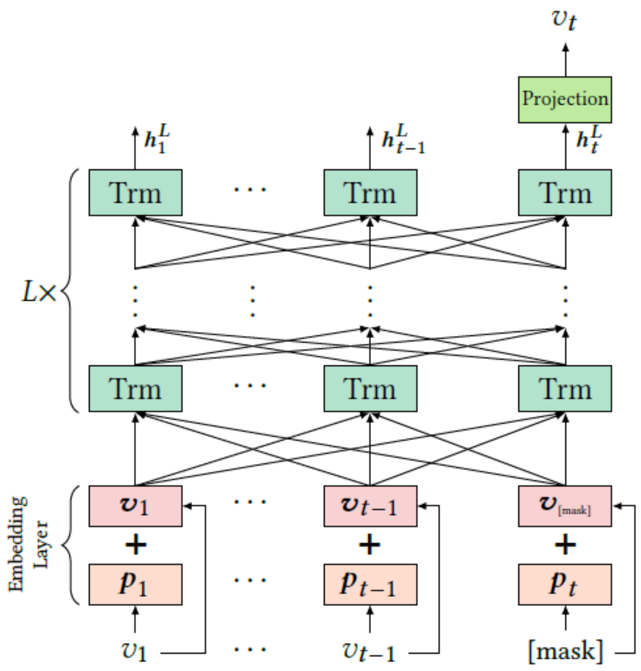
Q. Is the order of the items important when recommending them?

A. Nope!

Because Patterns are more important than order.

When User K was buying coffee and donuts

GRU4Rec(GRU) → **SASRec(self-attention)** → **BERT4Rec(BERT)**



(b) BERT4Rec model architecture.

- Use the BERT
  - Bidirectional Learning
  - Position Embedding (Time, Order Information)
  - Learning Overall Behavior Patterns

2. History

Next Model????

tada

GRU4Rec(GRU) → SASRec(self-attention) → BERT4Rec(BERT)

NEXT MODEL???

GPT ?!?!?!

3. Background

BERT

- Bidirectional learning model using Encoder in Transformer
- Embedding Layer
  - Position Embedding + Segment Embedding + Token Embedding
  - So, **Word Order + Doc Category + Word Information**
- Training
  - MLM (Masked Language Model)
  - NSP (Next Sentence Prediction)
- Training Phase
  - Pre-training
  - Fine-tuning

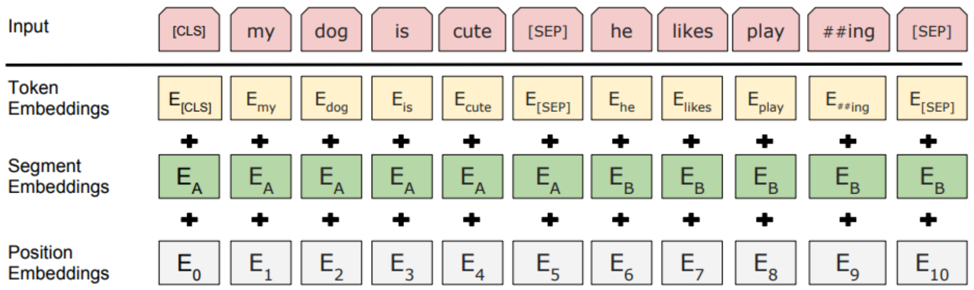
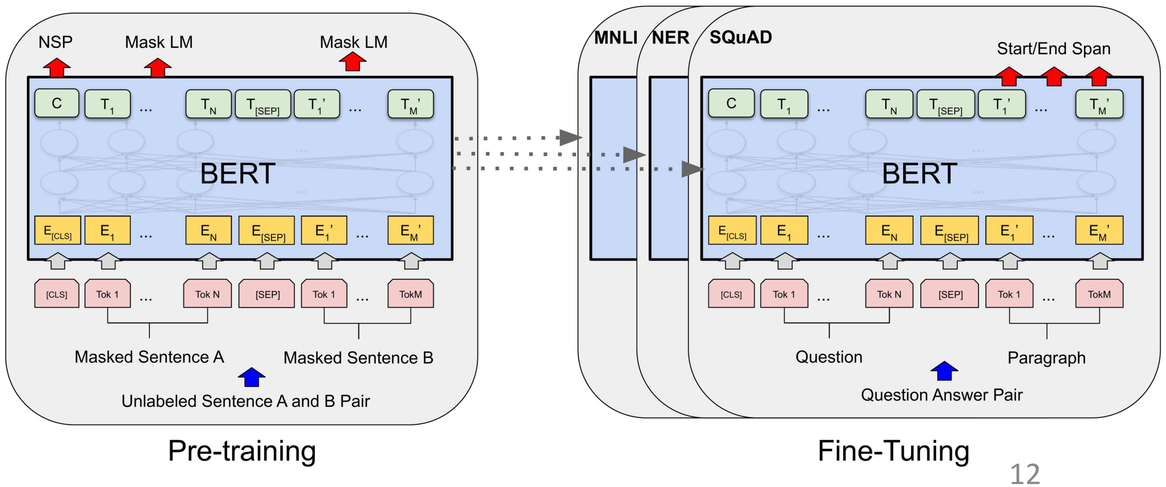


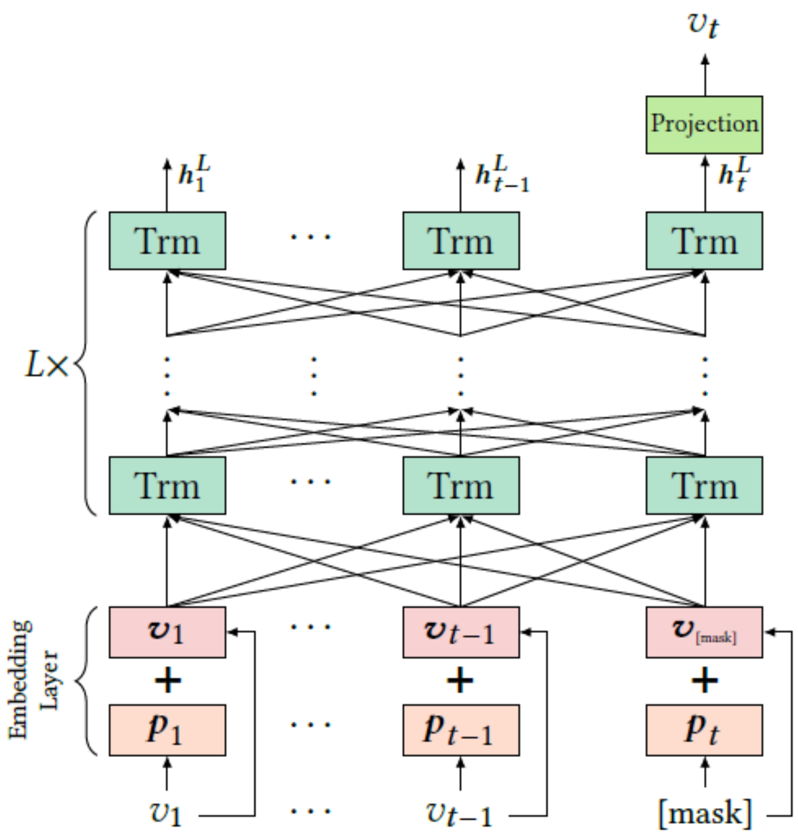
Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.



4. BERT4Rec

Purpose

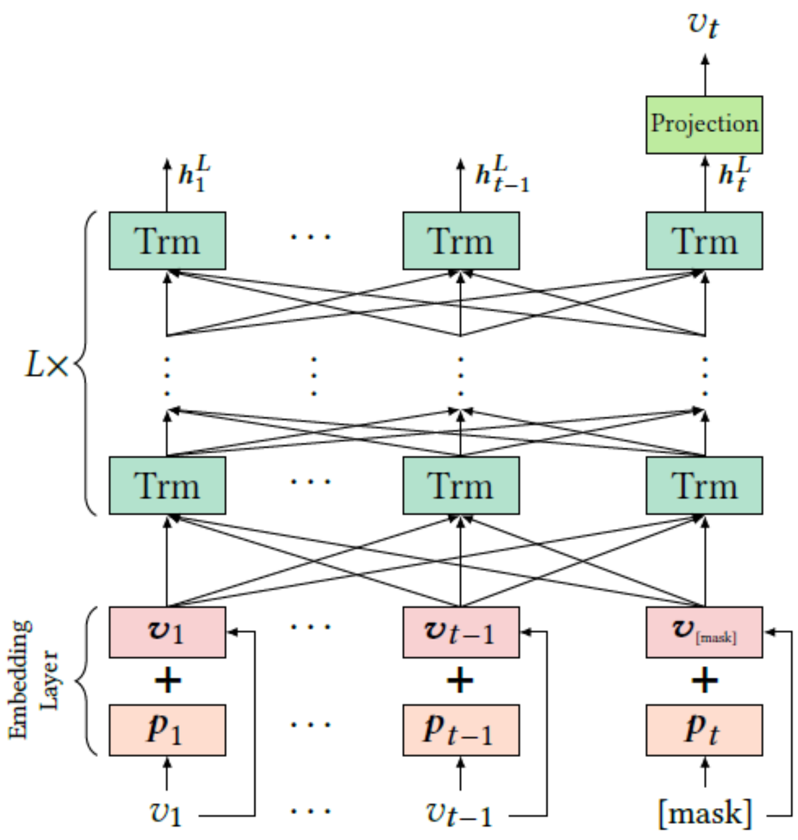
- Notation
  - u, v: user, item
  - $S_u = [v_1^u, \dots, v_t^u, \dots, v_{n_u}^u]$, $\quad user \ \ u \in U$ (interaction sequence)
  - $v_t^u \in V$: user u items interacted with in time t
  - n_u: user u interaction sequence length
  - **Purpose**
    - $p(v_{n_u+1}^u = v | S_u)$
    - **Probability that the user u selects a specific item v at $n_u + 1$ time**



4. BERT4Rec

Architecture

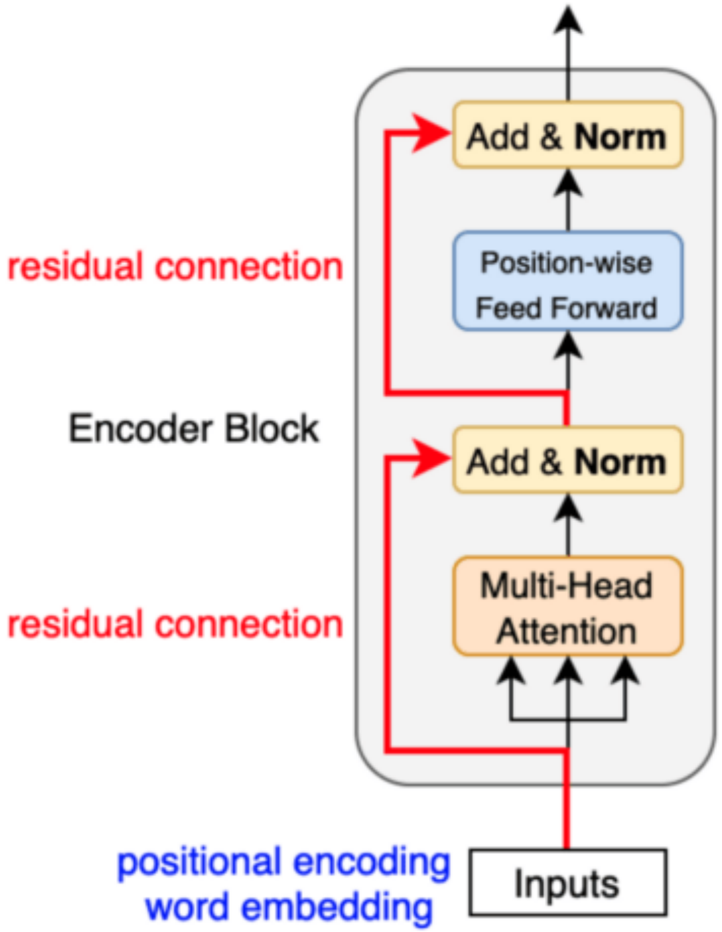
- Architecture
    - Stacked L bidirectional Transformer layer
    - Information from all positions in the previous layer is exchanged in parallel
    - And Modify the representation for all positions and proceed with the learning
    - The SA mechanism allows direct detection of dependencies without location/distance constraints.



4. BERT4Rec

Architecture

- Embedding Layer
  - Positional embedding: weight Learning (≠ transformer)
  - If len(seq) > n : Learn/Inference only for the last N
- Transformer layer
  - Multi-head self-attention
  - Position-wise feed-forward network
- Residual connection
  - Preserving information in a deep network
- Position-wise FFN
  - Captures interactions and improves nonlinearity
  - Activation function: GELU



- MLM (Masked Language Model)
  - Random masking in p% of the Input sequence
  - inference the original ID of the masked item through the surrounding context

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

$$\textbf{Labels}: [\mathtt{mask}]_1 = v_2, \quad [\mathtt{mask}]_2 = v_4$$

- Object function
  - Negative log-likelihood with the probability of getting the correct answer in the masking sample

$$\mathcal{L} = \frac{1}{|S_u^m|} \Sigma_{v_m \in S_u^m} - logP(v_m = v_m^* | \grave{S}_u)$$

4. BERT4Rec

Learning

- Test
  - Add a <Mask> special token at the end of the sequence
  - Model prediction

5. Experiments and Conclusion

tada

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%

- Density: Ratio of the number of items to the sequence Avg.length -> Sampling probability in the item list

5. Experiments and Conclusion

tada

Not Deep learning | GRU | CNN

	Datasets	Metric	POP	BPR-MF	NCF	FPMC	GRU4Rec	GRU4Rec+	Caser	SASRec	BERT4Rec	Improv.
Sparse	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%
Dense	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%

5. Experiments and Conclusion

- Purpose

$$p\left(v_{n_{\boldsymbol{u}}+\mathbf{1}}^{u} = v \middle| S_u\right)$$

- SASRec: Predict the present with past information
- BERT4Rec: Learning Overall Behavior Patterns with Bidirectional Learning
- SASRec Performance << BERT4Rec

6. Reference

- Paper
  - https://arxiv.org/abs/1904.06690
- Code
  - Original version: https://github.com/FeiSun/BERT4Rec
  - Pytorch version: https://github.com/jaywonchung/BERT4Rec-VAE-Pytorch
  - https://github.com/CVxTz/recommender_transformer
- Youtube
  - https://www.youtube.com/watch?v=PKYVHGrSO2U
  - https://www.youtube.com/watch?v=d2IaWtBbJjg
- Blog
  - https://greeksharifa.github.io/nlp(natural%20language%20processing)%20/%20rnns/2019/08/23/BERT-Pre-training-of-Deep-Bidirectional-Transformers-for-Language-Understanding/
  - https://greeksharifa.github.io/machine_learning/2021/12/12/Bert4Rec/
  - https://towardsdatascience.com/build-your-own-movie-recommender-system-using-bert4rec-92e4e34938c5