Self-Attentive Sequential Recommendation

Авторы:
Wang-Cheng Kang
Julian McAuley

Докладчик: Челпанов Василий
#recommender_systems: https://t.me/ods_recommender_systems

Вводная часть

Self-attentive sequential recommendation

WC Kang, J McAuley - 2018 IEEE international conference on …, 2018 - ieeexplore.ieee.org

… Inspired by Transformer, we seek to build a new **sequential recommendation** model based upon the self-attention approach, though the problem of **sequential recommendation** is quite …

☆ Save 99 Cite Cited by 1198 Related articles All 7 versions

Introduction

Capturing useful patterns from sequential dynamics **is challenging**, primarily because the dimension of the input space grows exponentially with the number of past actions used as context.

Markov Chains (MCs):

- perform well in high-sparsity settings
- may fail to capture the intricate dynamics of more complex scenarios

Recurrent Neural Networks (RNN):

- expressive
- require large amounts of data

Introduction

Capturing useful patterns from sequential dynamics **is challenging**, primarily because the dimension of the input space grows exponentially with the number of past actions used as context.

Transformer:

- able to draw context from all actions in the past (RNNs)
- able to frame predictions in terms of just a small number of actions (MCs)

Related work

Sequential Recommendation:

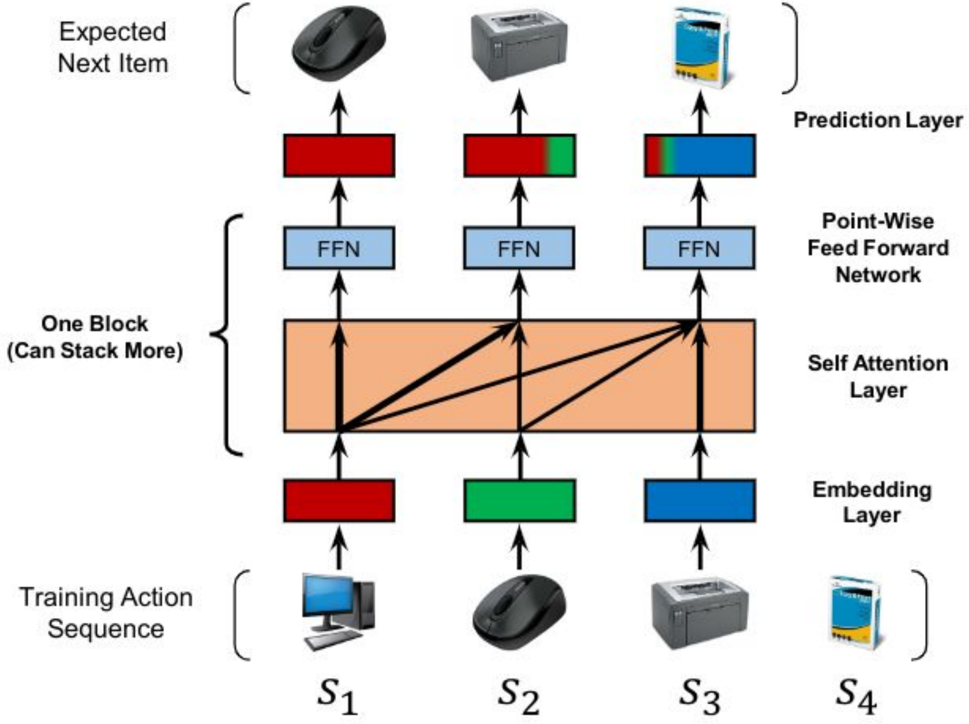
- first-order MCs
- higher-order MCs
- convolutional sequence embedding (Caser)
- RNNs

Attention Mechanisms

Essentially, the idea behind such mechanisms is that sequential outputs (for example) each depend on “relevant” parts of some input that the model should focus on successively.

Methodology

Input file:
user_id, item_id
1,12888
1,49583
1,1
1,4733
…
2 16759
2 1
2 15161
2 30033
…



Methodology

Given sequence:

$$(\mathcal{S}_1^u, \mathcal{S}_2^u, \ldots, \mathcal{S}_{|\mathcal{S}^u|}^u)$$

Model input:

$$(\mathcal{S}_1^u, \mathcal{S}_2^u, \ldots, \mathcal{S}_{|\mathcal{S}^u|-1}^u)$$

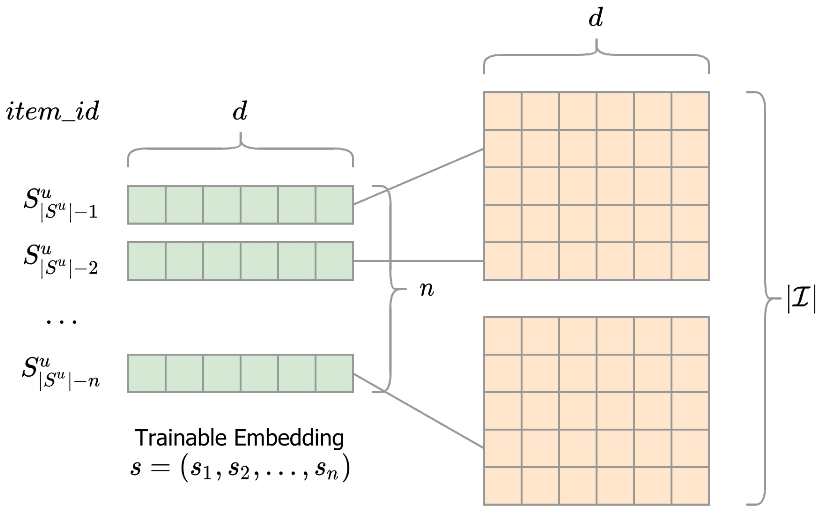
Model output:

$$(\mathcal{S}_2^u, \mathcal{S}_3^u, \ldots, \mathcal{S}_{|\mathcal{S}^u|}^u)$$

Notation	Description
$\mathcal{U}, \mathcal{I}$	user and item set
$\mathcal{S}^u$	historical interaction sequence for a user u: $(\mathcal{S}_1^u, \mathcal{S}_2^u, ..., \mathcal{S}_{|\mathcal{S}^u|}^u)$
$d \in \mathbb{N}$	latent vector dimensionality
$n \in \mathbb{N}$	maximum sequence length
$b \in \mathbb{N}$	number of self-attention blocks
$\mathbf{M} \in \mathbb{R}^{|\mathcal{I}| \times d}$	item embedding matrix
$\mathbf{P} \in \mathbb{R}^{n \times d}$	positional embedding matrix
$\widehat{\mathbf{E}} \in \mathbb{R}^{n \times d}$	input embedding matrix
$\mathbf{S}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th self-attention layer
$\mathbf{F}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th feed-forward network

Methodology

Model input:



Notation	Description
$\mathcal{U}, \mathcal{I}$	user and item set
$\mathcal{S}^u$	historical interaction sequence for a user u: $(\mathcal{S}^u_1, \mathcal{S}^u_2, ..., \mathcal{S}^u_{\lvert\mathcal{S}^u\rvert})$
$d \in \mathbb{N}$	latent vector dimensionality
$n \in \mathbb{N}$	maximum sequence length
$b \in \mathbb{N}$	number of self-attention blocks
$\mathbf{M} \in \mathbb{R}^{\lvert\mathcal{I}\rvert \times d}$	item embedding matrix
$\mathbf{P} \in \mathbb{R}^{n \times d}$	positional embedding matrix
$\widehat{\mathbf{E}} \in \mathbb{R}^{n \times d}$	input embedding matrix
$\mathbf{S}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th self-attention layer
$\mathbf{F}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th feed-forward network

Methodology

Model input:

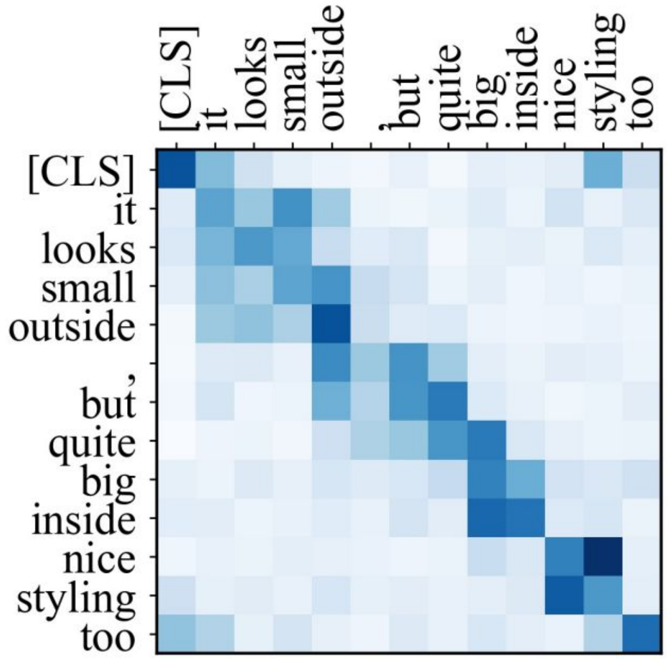
$$\widehat{\mathbf{E}} = \begin{bmatrix} \mathbf{M}_{s_1} + \mathbf{P}_1 \\ \mathbf{M}_{s_2} + \mathbf{P}_2 \\ \dots \\ \mathbf{M}_{s_n} + \mathbf{P}_n \end{bmatrix}$$

Notation	Description
$\mathcal{U}, \mathcal{I}$	user and item set
$\mathcal{S}^u$	historical interaction sequence for a user u: $(\mathcal{S}_1^u, \mathcal{S}_2^u, ..., \mathcal{S}_{\lvert\mathcal{S}^u\rvert}^u)$
$d \in \mathbb{N}$	latent vector dimensionality
$n \in \mathbb{N}$	maximum sequence length
$b \in \mathbb{N}$	number of self-attention blocks
$\mathbf{M} \in \mathbb{R}^{\lvert\mathcal{I}\rvert \times d}$	item embedding matrix
$\mathbf{P} \in \mathbb{R}^{n \times d}$	positional embedding matrix
$\widehat{\mathbf{E}} \in \mathbb{R}^{n \times d}$	input embedding matrix
$\mathbf{S}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th self-attention layer
$\mathbf{F}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th feed-forward network

Methodology

Self-Attention Block:

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

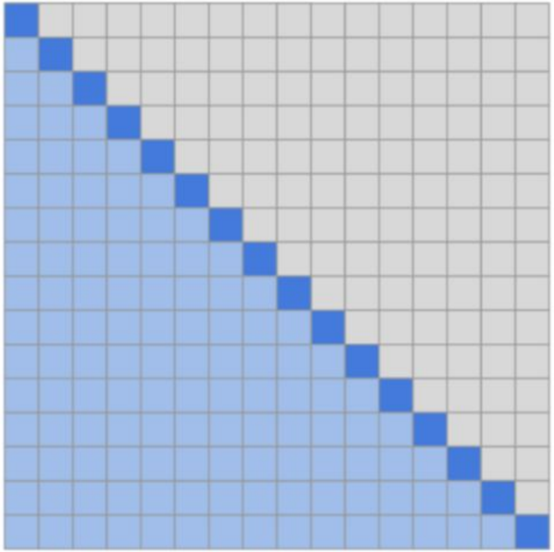


Methodology

Self-Attention Block:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

$$\mathbf{S} = \text{SA}(\widehat{\mathbf{E}}) = \text{Attention}(\widehat{\mathbf{E}}\mathbf{W}^Q, \widehat{\mathbf{E}}\mathbf{W}^K, \widehat{\mathbf{E}}\mathbf{W}^V)$$



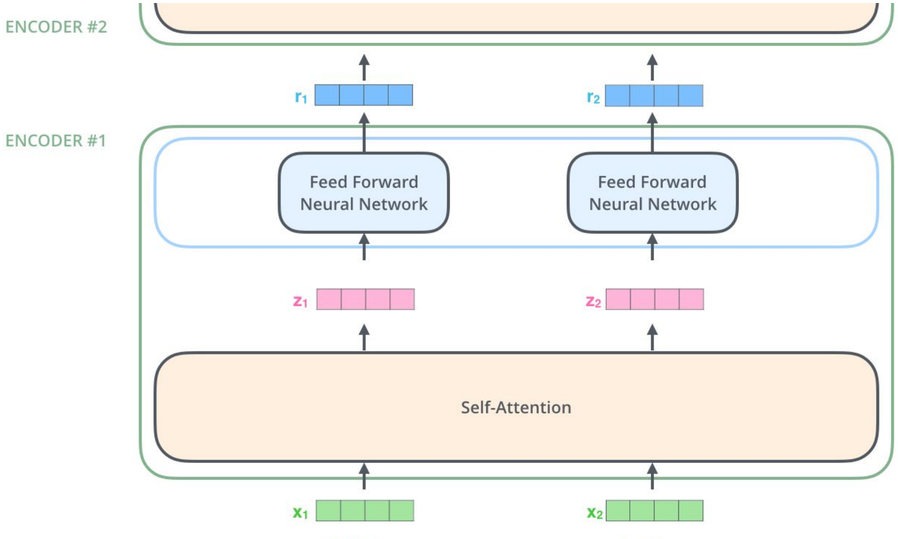
#recommender_systems: https://t.me/ods_recommender_systems

Methodology

Point-Wise Feed-Forward Network:

$$\mathbf{F}_i = \mathrm{FFN}(\mathbf{S}_i) = \mathrm{ReLU}(\mathbf{S}_i\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$$

where $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}$ are $d \times d$ matrices and $\mathbf{b}^{(1)}, \mathbf{b}^{(2)}$ are d-dimensional vectors. Note that there is no interaction between $\mathbf{S}_i$ and $\mathbf{S}_j$ $(i \neq j)$, meaning that we still prevent information leaks (from back to front).



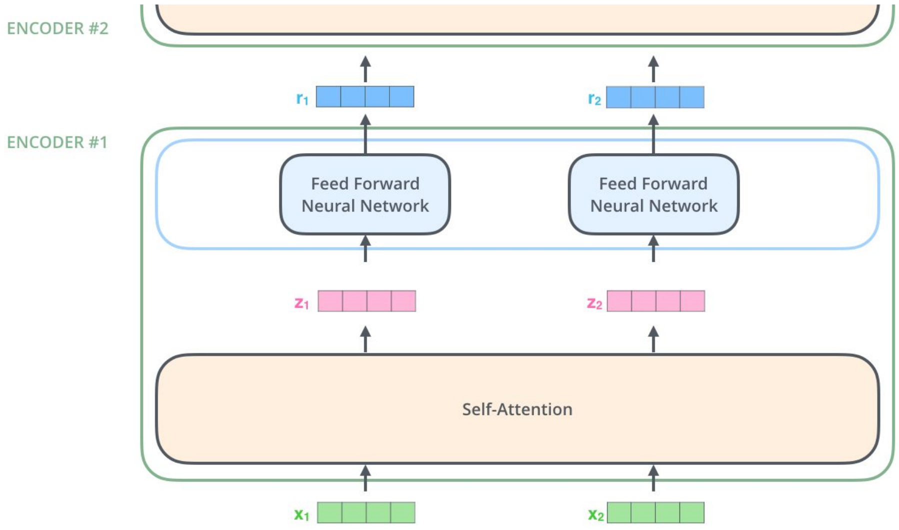
Methodology

Stacking Self-Attention Blocks:

$$\mathbf{S} = \mathrm{SA}(\widehat{\mathbf{E}}) = \mathrm{Attention}(\widehat{\mathbf{E}}\mathbf{W}^Q, \widehat{\mathbf{E}}\mathbf{W}^K, \widehat{\mathbf{E}}\mathbf{W}^V)$$

$$\mathbf{F}_i = \mathrm{FFN}(\mathbf{S}_i) = \mathrm{ReLU}(\mathbf{S}_i\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$$

$$\mathbf{S}^{(b)} = \mathrm{SA}(\mathbf{F}^{(b-1)}),$$
$$\mathbf{F}_i^{(b)} = \mathrm{FFN}(\mathbf{S}_i^{(b)}), \quad \forall i \in \{1, 2, \ldots, n\}$$



Methodology

Prediction Layer:

$$r_{i,t} = \mathbf{F}_t^{(b)} \mathbf{N}_i^T,$$

where $r_{i,t}$ is the relevance of item i being the next item given the first t items (i.e., $s_1, s_2, \ldots, s_t$), and $\mathbf{N} \in \mathbb{R}^{|\mathcal{I}| \times d}$ is an item embedding matrix. Hence, a high interaction score $r_{i,t}$ means a high relevance, and we can generate recommendations by ranking the scores.

Notation	Description
$\mathcal{U}, \mathcal{I}$	user and item set
$\mathcal{S}^u$	historical interaction sequence for a user u: $(\mathcal{S}_1^u, \mathcal{S}_2^u, ..., \mathcal{S}_{\lvert\mathcal{S}^u\rvert}^u)$
$d \in \mathbb{N}$	latent vector dimensionality
$n \in \mathbb{N}$	maximum sequence length
$b \in \mathbb{N}$	number of self-attention blocks
$\mathbf{M} \in \mathbb{R}^{\lvert\mathcal{I}\rvert \times d}$	item embedding matrix
$\mathbf{P} \in \mathbb{R}^{n \times d}$	positional embedding matrix
$\widehat{\mathbf{E}} \in \mathbb{R}^{n \times d}$	input embedding matrix
$\mathbf{S}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th self-attention layer
$\mathbf{F}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th feed-forward network

Methodology

Prediction Layer:

Shared Item Embedding: To reduce the model size and alleviate overfitting, we consider another scheme which only uses a single item embedding $\mathbf{M}$:

$$r_{i,t} = \mathbf{F}_t^{(b)} \mathbf{M}_i^T. \quad (6)$$

However, we can also insert an explicit user embedding at the last layer, for example via addition: $r_{u,i,t} = (\mathbf{U}_u + \mathbf{F}_t^{(b)})\mathbf{M}_i^T$ where $\mathbf{U}$ is a user embedding matrix. However, we empirically find that adding an explicit user embedding doesn't improve performance (presumably because the model already considers all of the user's actions).

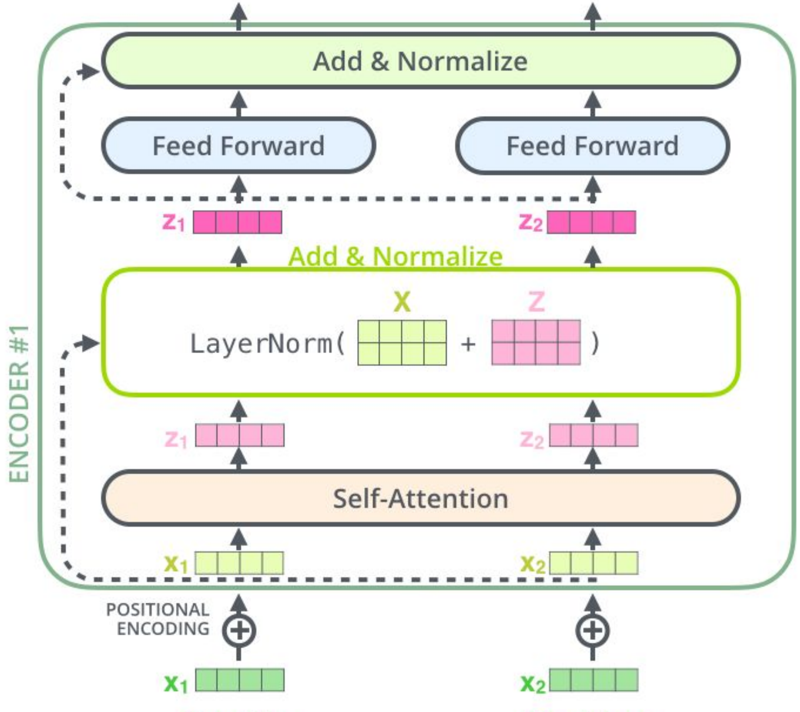
Notation	Description
$\mathcal{U}, \mathcal{I}$	user and item set
$\mathcal{S}^u$	historical interaction sequence for a user u: $(\mathcal{S}_1^u, \mathcal{S}_2^u, ..., \mathcal{S}_{\lvert\mathcal{S}^u\rvert}^u)$
$d \in \mathbb{N}$	latent vector dimensionality
$n \in \mathbb{N}$	maximum sequence length
$b \in \mathbb{N}$	number of self-attention blocks
$\mathbf{M} \in \mathbb{R}^{\lvert\mathcal{I}\rvert \times d}$	item embedding matrix
$\mathbf{P} \in \mathbb{R}^{n \times d}$	positional embedding matrix
$\widehat{\mathbf{E}} \in \mathbb{R}^{n \times d}$	input embedding matrix
$\mathbf{S}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th self-attention layer
$\mathbf{F}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th feed-forward network

Methodology

Some improvements

- Residual Connections
- Layer Normalization

$$\mathrm{LayerNorm}(\mathbf{x}) = \boldsymbol{\alpha} \odot \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \boldsymbol{\beta},$$



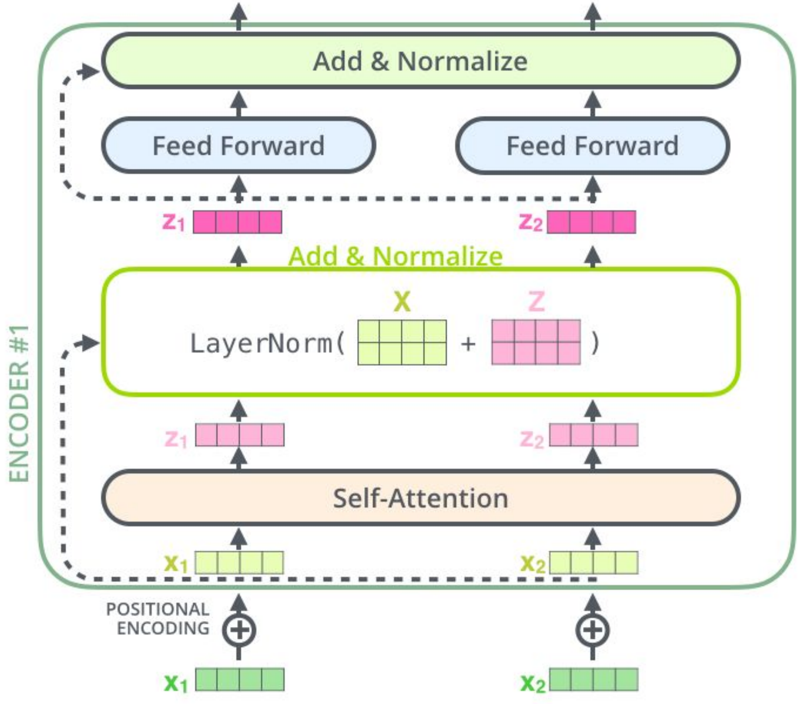
Methodology

Some improvements

- Dropout

$$g(x) = x + \text{Dropout}(g(\text{LayerNorm}(x))),$$

where $g(x)$ represents the self attention layer or the feed-forward network. That is to say, for layer g in each block, we apply layer normalization on the input x before feeding into g, apply dropout on g's output, and add the input x to the final output. We introduce these operations below.



Methodology

Network Training:

Recall that we convert each user sequence (excluding the last action) $(\mathcal{S}_1^u, \mathcal{S}_2^u, \dots, \mathcal{S}_{|\mathcal{S}^u|-1}^u)$ to a fixed length sequence $s = \{s_1, s_2, \dots, s_n\}$ via truncation or padding items. We define o_t as the expected output at time step t:

$$o_t = \begin{cases} \texttt{<pad>} & \text{if } s_t \text{ is a padding item} \\ s_{t+1} & 1 \le t < n \\ \mathcal{S}_{|\mathcal{S}^u|}^u & t = n \end{cases},$$

Methodology

Network Training:

output, and we adopt the binary cross entropy loss as the objective function:

$$-\sum_{\mathcal{S}^u \in \mathcal{S}} \sum_{t \in [1,2,\ldots,n]} \left[\log(\sigma(r_{o_t,t})) + \sum_{j \notin \mathcal{S}^u} \log(1 - \sigma(r_{j,t})) \right].$$

Note that we ignore the terms where $o_t = \texttt{<pad>}$.

The network is optimized by the *Adam* optimizer [41], which is a variant of Stochastic Gradient Descent (SGD) with adaptive moment estimation. In each epoch, we randomly generate one negative item j for each time step in each sequence. More implementation details are described later.

Methodology

Complexity Analysis:

Space Complexity

$$O(|\mathcal{I}|d + nd + d^2)$$

Time Complexity

$$O(n^2 d + nd^2)$$

Handing Long Sequences

Notation	Description
$\mathcal{U}, \mathcal{I}$	user and item set
$\mathcal{S}^u$	historical interaction sequence for a user u: $(\mathcal{S}_1^u, \mathcal{S}_2^u, ..., \mathcal{S}_{\|\mathcal{S}^u\|}^u)$
$d \in \mathbb{N}$	latent vector dimensionality
$n \in \mathbb{N}$	maximum sequence length
$b \in \mathbb{N}$	number of self-attention blocks
$\mathbf{M} \in \mathbb{R}^{\|\mathcal{I}\| \times d}$	item embedding matrix
$\mathbf{P} \in \mathbb{R}^{n \times d}$	positional embedding matrix
$\widehat{\mathbf{E}} \in \mathbb{R}^{n \times d}$	input embedding matrix
$\mathbf{S}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th self-attention layer
$\mathbf{F}^{(b)} \in \mathbb{R}^{n \times d}$	item embeddings after the b-th feed-forward network

Experiments

RQ1: Does SASRec outperform state-of-the-art models, including CNN/RNN based methods?

Dataset	#users	#items	avg. actions /user	avg. actions /item	#actions
Amazon Beauty	52,024	57,289	7.6	6.9	0.4M
Amazon Games	31,013	23,715	9.3	12.1	0.3M
Steam	334,730	13,047	11.0	282.5	3.7M
MovieLens-1M	6,040	3,416	163.5	289.1	1.0M

#recommender_systems: https://t.me/ods_recommender_systems

Experiments

RQ1: Does SASRec outperform state-of-the-art models, including CNN/RNN based methods?

Hit@10 - counts the fraction of times that the ground-truthnext item is among the top 10 items

NDCG@10 - is aposition-aware metric which assigns larger weights on higher positions

$$NDCG@K = \frac{DCG@K}{IDCG@K} = \frac{\sum_{i=1}^{k \text{ (actual order)}} \frac{Gains}{\log_2(i+1)}}{\sum_{i=1}^{k \text{ (ideal order)}} \frac{Gains}{\log_2(i+1)}}$$

Experiments

RQ1: Does SASRec outperform state-of-the-art models, including CNN/RNN based methods?

To avoid heavy computation on all user-item pairs, we followed the strategy in [14], [48]. For each user u, we randomly sample 100 negative items, and rank these items with the ground-truth item. Based on the rankings of these 101 items, Hit@10 and NDCG@10 can be evaluated.

Experiments

RQ1: Does SASRec outperform state-of-the-art models, including CNN/RNN based methods?

Dataset	Metric	(a) PopRec	(b) BPR	(c) FMC	(d) FPMC	(e) TransRec	(f) GRU4Rec	(g) GRU4Rec^{+}	(h) Caser	(i) SASRec	Improvement vs. (a)-(e)	Improvement vs. (f)-(h)
Beauty	Hit@10	0.4003	0.3775	0.3771	0.4310	0.4607	0.2125	0.3949	0.4264	**0.4854**	5.4%	13.8%
	NDCG@10	0.2277	0.2183	0.2477	0.2891	0.3020	0.1203	0.2556	0.2547	**0.3219**	6.6%	25.9%
Games	Hit@10	0.4724	0.4853	0.6358	0.6802	0.6838	0.2938	0.6599	0.5282	**0.7410**	8.5%	12.3%
	NDCG@10	0.2779	0.2875	0.4456	0.4680	0.4557	0.1837	0.4759	0.3214	**0.5360**	14.5%	12.6%
Steam	Hit@10	0.7172	0.7061	0.7731	0.7710	0.7624	0.4190	0.8018	0.7874	**0.8729**	13.2%	8.9%
	NDCG@10	0.4535	0.4436	0.5193	0.5011	0.4852	0.2691	0.5595	0.5381	**0.6306**	21.4%	12.7%
ML-1M	Hit@10	0.4329	0.5781	0.6986	0.7599	0.6413	0.5581	0.7501	0.7886	**0.8245**	8.5%	4.6%
	NDCG@10	0.2377	0.3287	0.4676	0.5176	0.3969	0.3381	0.5513	0.5538	**0.5905**	14.1%	6.6%

Experiments

RQ1: Does SASRec outperform state-of-the-art models, including CNN/RNN based methods?

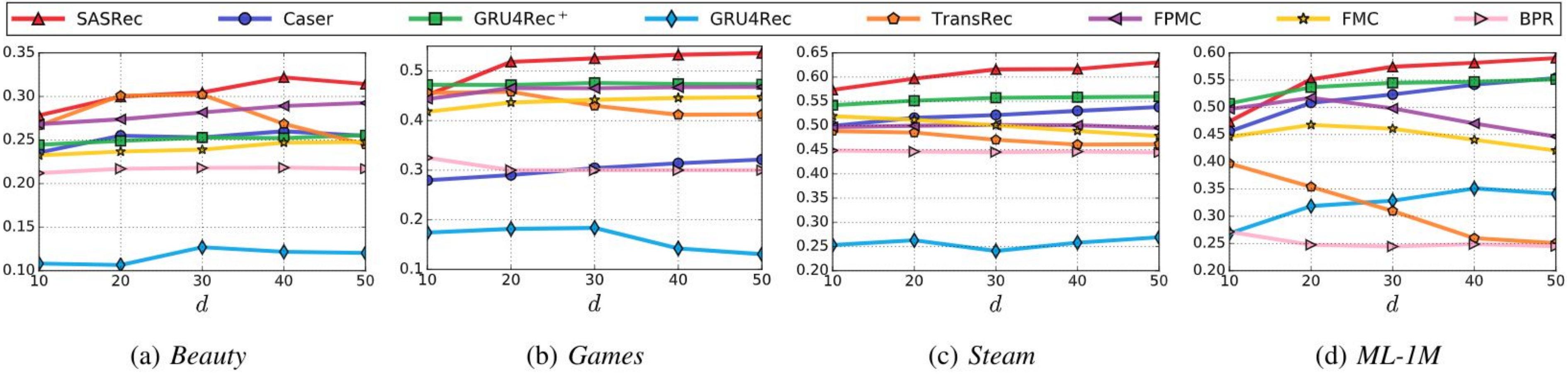


Figure 2: Effect of the latent dimensionality d on ranking performance (NDCG@10).

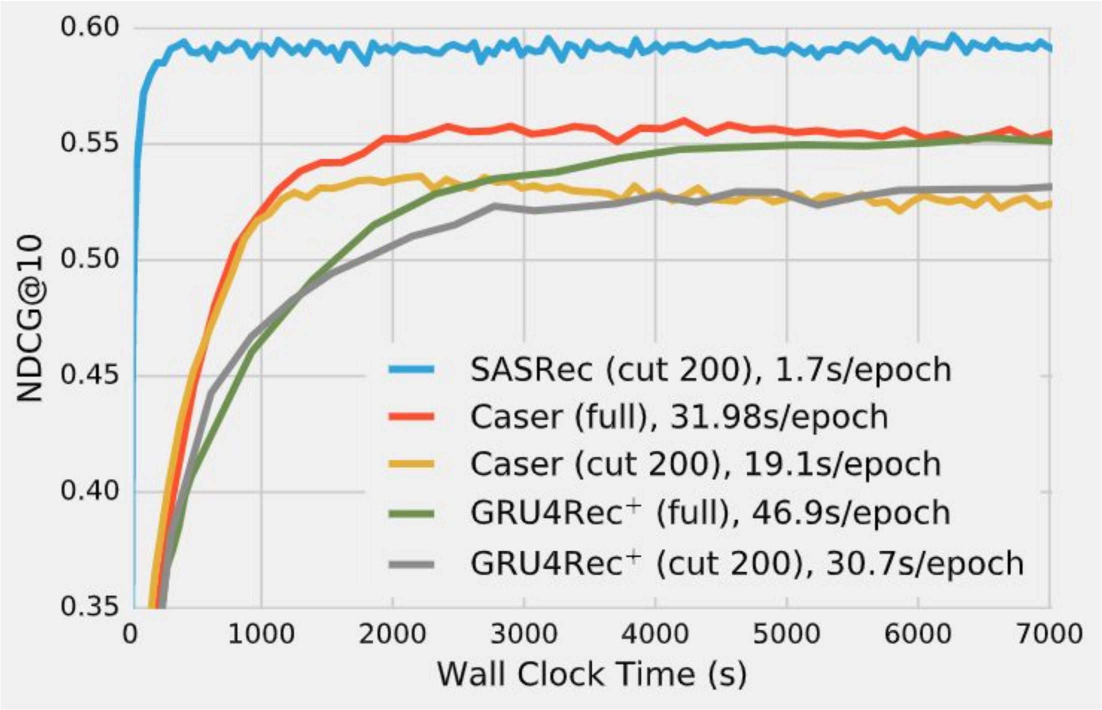
Experiments

RQ2: What is the influence of various components in the SASRec architecture?

Architecture	*Beauty*	*Games*	*Steam*	*ML-1M*
(0) Default	0.3142	0.5360	0.6306	0.5905
(1) Remove PE	**0.3183**	0.5301	0.6036	0.5772
(2) Unshared IE	0.2437↓	0.4266↓	0.4472↓	0.4557↓
(3) Remove RC	0.2591↓	0.4303↓	0.5693	0.5535
(4) Remove Dropout	0.2436↓	0.4375↓	0.5959	0.5801
(5) 0 Block (b=0)	0.2620↓	0.4745↓	0.5588↓	0.4830↓
(6) 1 Block (b=1)	0.3066	**0.5408**	0.6202	0.5653
(7) 3 Blocks (b=3)	0.3078	0.5312	0.6275	**0.5931**
(8) Multi-Head	0.3080	0.5311	0.6272	0.5885

Experiments

RQ3: What is the training efficiency and scalability (regarding) of SASRec?



Experiments

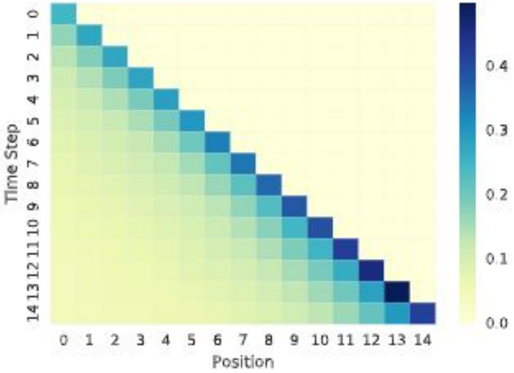
RQ3: What is the training efficiency and scalability (regarding) of SASRec?

Table V: Scalability: performance and training time with different maximum length n on *ML-1M*.

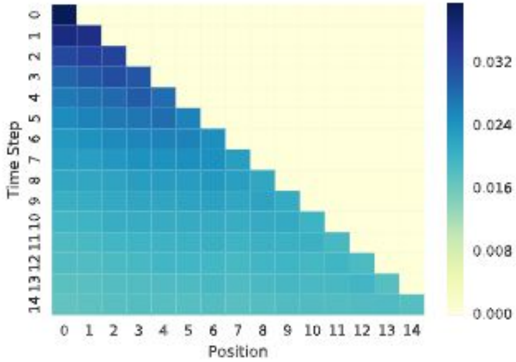
n	10	50	100	200	300	400	500	600
Time(s)	75	101	157	341	613	965	1406	1895
NDCG@10	0.480	0.557	0.571	0.587	0.593	0.594	0.596	0.595

Experiments

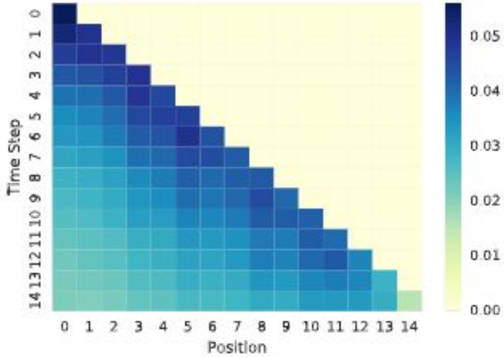
RQ4: Are the attention weights able to learn meaningful patterns related to positions or items’ attributes?



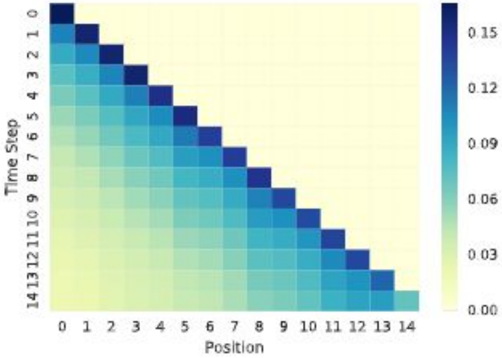
(a) *Beauty*, Layer 1



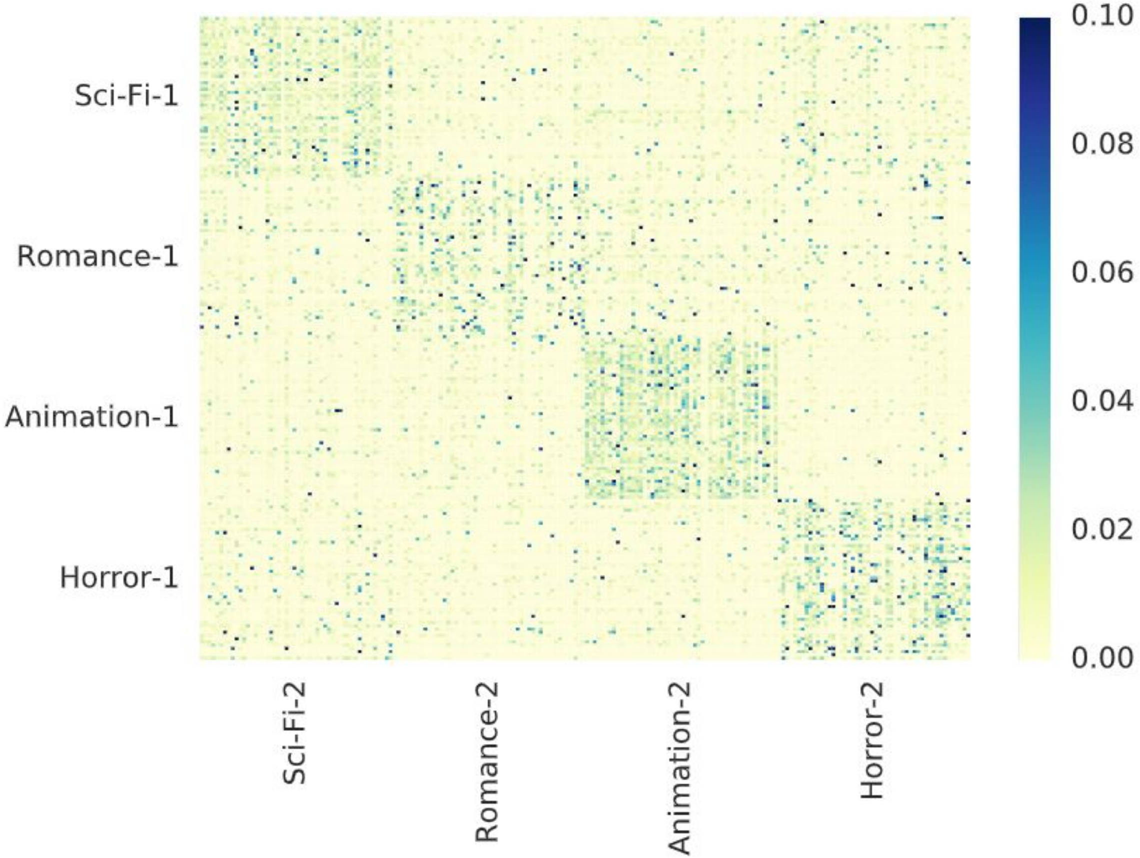
(b) *ML-1M*, Layer 1, w/o PE



(c) *ML-1M*, Layer 1



(d) *ML-1M*, Layer 2


0.10
0.08
0.06
0.04
0.02
0.00
Sci-Fi-1
Romance-1
Animation-1
Horror-1
Sci-Fi-2
Romance-2
Animation-2
Horror-2

Спасибо за внимание

Докладчик: Челпанов Василий

Ссылки

1. Оригинальная статья
2. Реализация на TensorFlow 1.12 и Python 2
3. Реализация на Pytorch v1.6*
4. Transformer, explained in detail | Igor Kotenkov | NLP Lecture (in Russian)
5. The Illustrated Transformer
6. The Illustrated GPT-2
7. The Transformer Family