SELF-ATTENTIVE SEQUENTIAL RECOMMENDATION(ICDM, 2018)

2021-06-07 지능정보융합학과 이원도

SELF-ATTENTIVE SEQUENTIAL RECOMMENDATION

Intro

- temporal/sequential RS
- attention mechanism

SASRec

- model, training
- discussion

Experiments & Evaluation

- performance evaluation, abalation study
- visualizing attention weights

1. INTRO

General RS

- Matrix Factorization, Item Similarity, recent trend of using Deep Learning

Temporal RS

- TimeSVD++, concepts such as temporal 'drift',

<u>Sequential RS</u>

- model sequential patterns, context
- Markov Chains : most are first order transitions model, some are high order models, MCs are effective in sparse data
- RNNs : deep learning models such as GRU4Rec, recurrent structure, effective in dense data

1. INTRO

Attention Mechanism

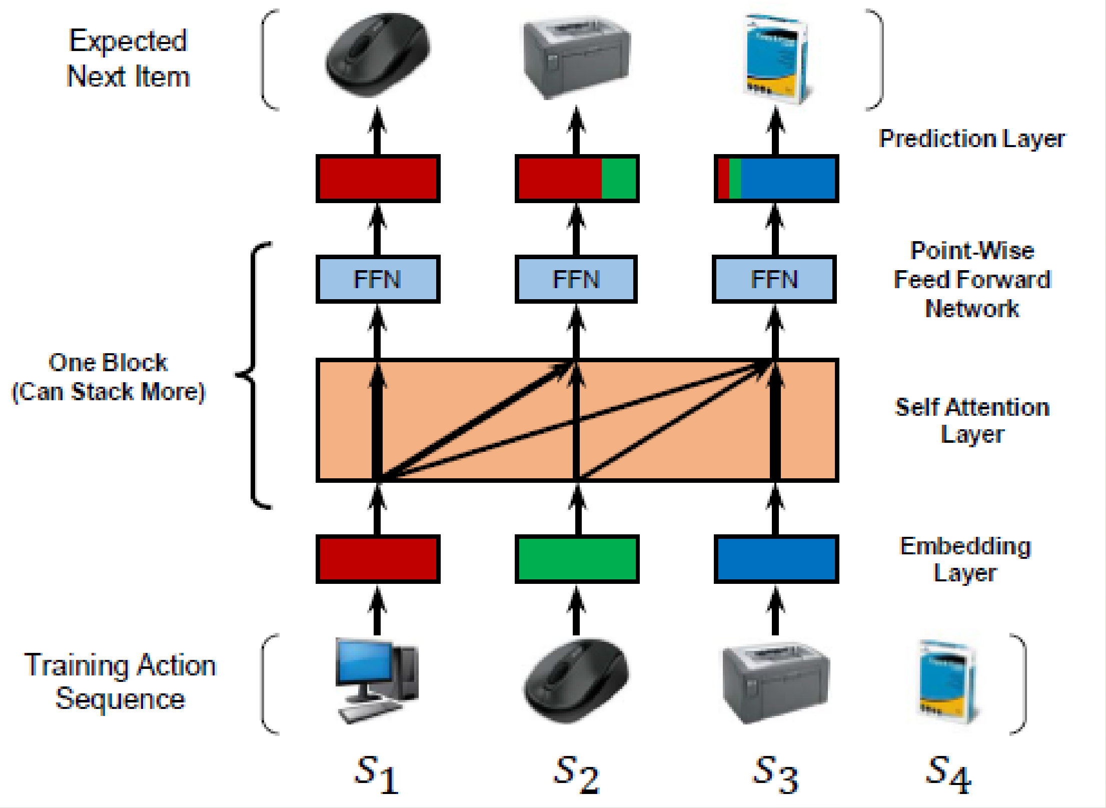
- Effective use in image captioning, machine translation tasks
- Idea is to let each component focus on relevant parts of input
- Somewhat interpretable
- Recently, it is used additionally on existing recommender systems

- Recent NLP sota Transformer model relies purely on ‘self-attention’
- → inspired by Transformer, new sequential RS model SASRec based on self-attention is proposed

2. SASREC

- SASRec model overview
  Prediction layer
  ↑ Self-attn block (b)
  …
  ↑ Self-attn block (1)
  ↑ Embedding layer
  ↑ Input

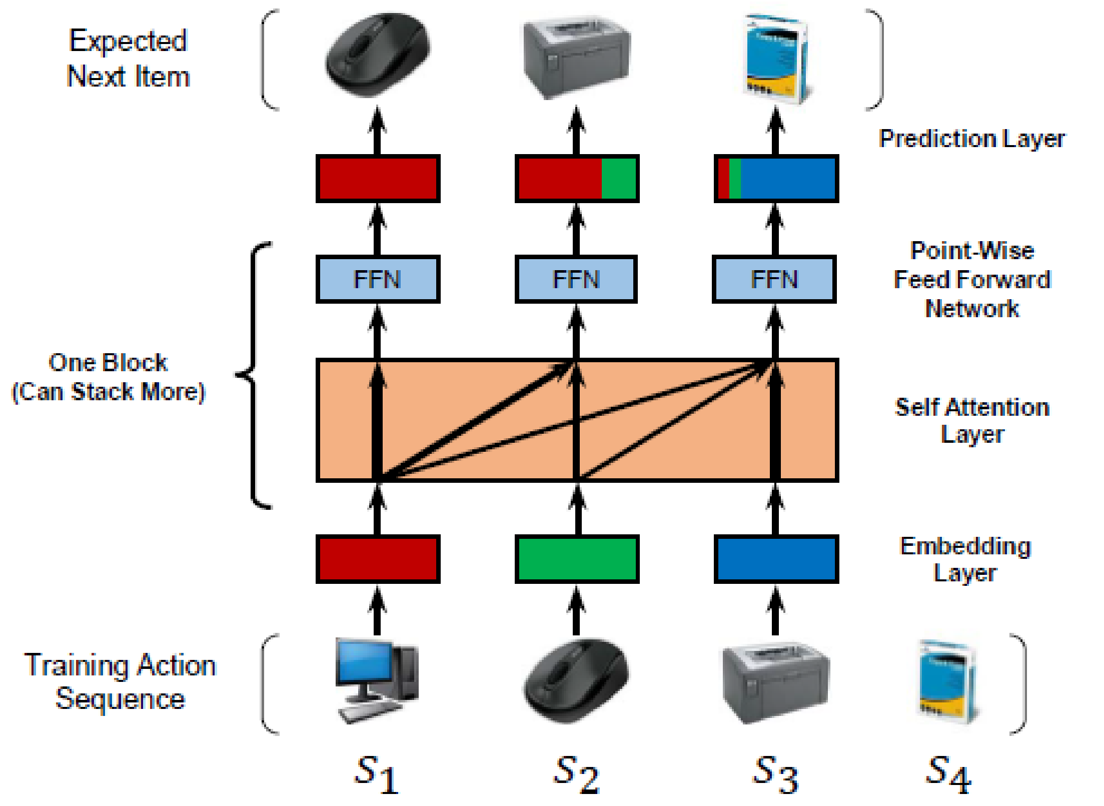
- Training
  Input : [a, b, c, d]
  → Want Output : [b, c, d, e]



2. SASREC

Table I: Notation.

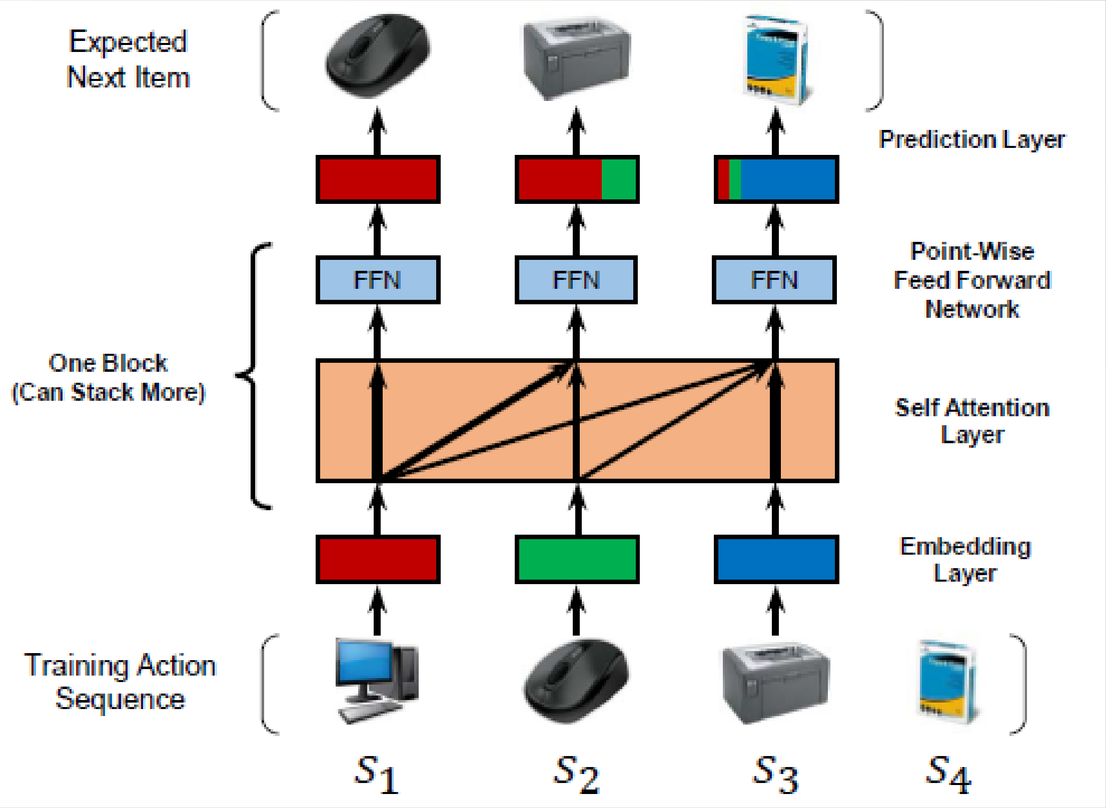
Notation	Description
$\mathcal{U}, \mathcal{I}$	user and item set
$\mathcal{S}^u$	historical interaction sequence for a user u: $(\mathcal{S}_1^u, \mathcal{S}_2^u, ..., \mathcal{S}_{\lvert\mathcal{S}^u\rvert}^u)$
$d \in \mathbb{N}$	latent vector dimensionality
$n \in \mathbb{N}$	maximum sequence length
$b \in \mathbb{N}$	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



2. SASREC

- Input
  Data : user's action sequence
  $S^u = (S_1^u, S_2^u, \dots, S_{|S^u|}^u)$

- Set maximum length of input : n
  If data is greater than n, use most recent n,
  If data is shorter than n, use padding to the left
- Ex1 : $s = (s_1, s_2, \dots s_n)$
- Ex2 : $s = (0, 0, \dots, 0, s_{n-1}, s_n)$



2. SASREC

- Embedding Layer

  Embed input s with item embedding matrix $M \in R^{|I| \times d}$

  $s = (s_1, s_2, \dots s_n)^t$

  $\rightarrow E = (e_1, e_2, \dots, e_n)^t$, where $e_i \in R^d$

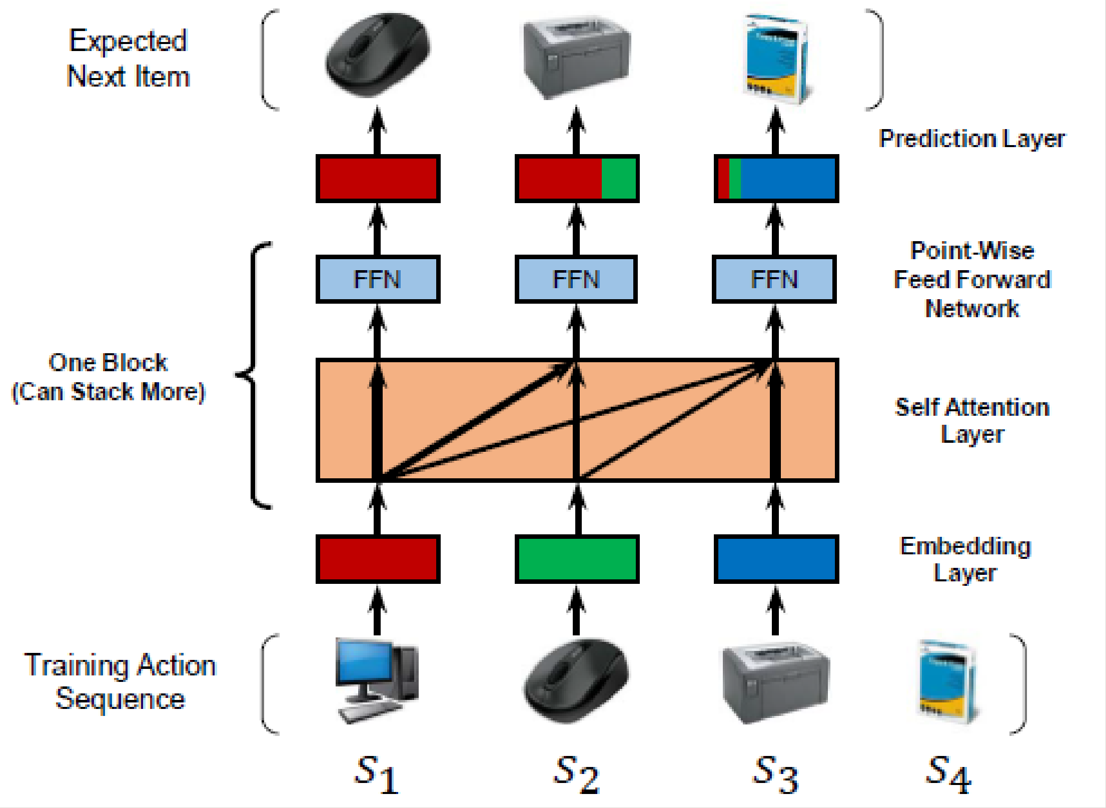
- Positional Embedding

  Model don't have positional components

  $\rightarrow$ (learnable) position parameter P is added

  $E = (e_1, e_2, \dots, e_n)^t$

  $\rightarrow \hat{E} = (e_1 + P_1, \dots, e_n + P_n)^t$, where $P_i \in R^d$



2. SASREC

- Self-Attn Block(= Self-Attn layer + Feed Forward)

- Definition of Attention

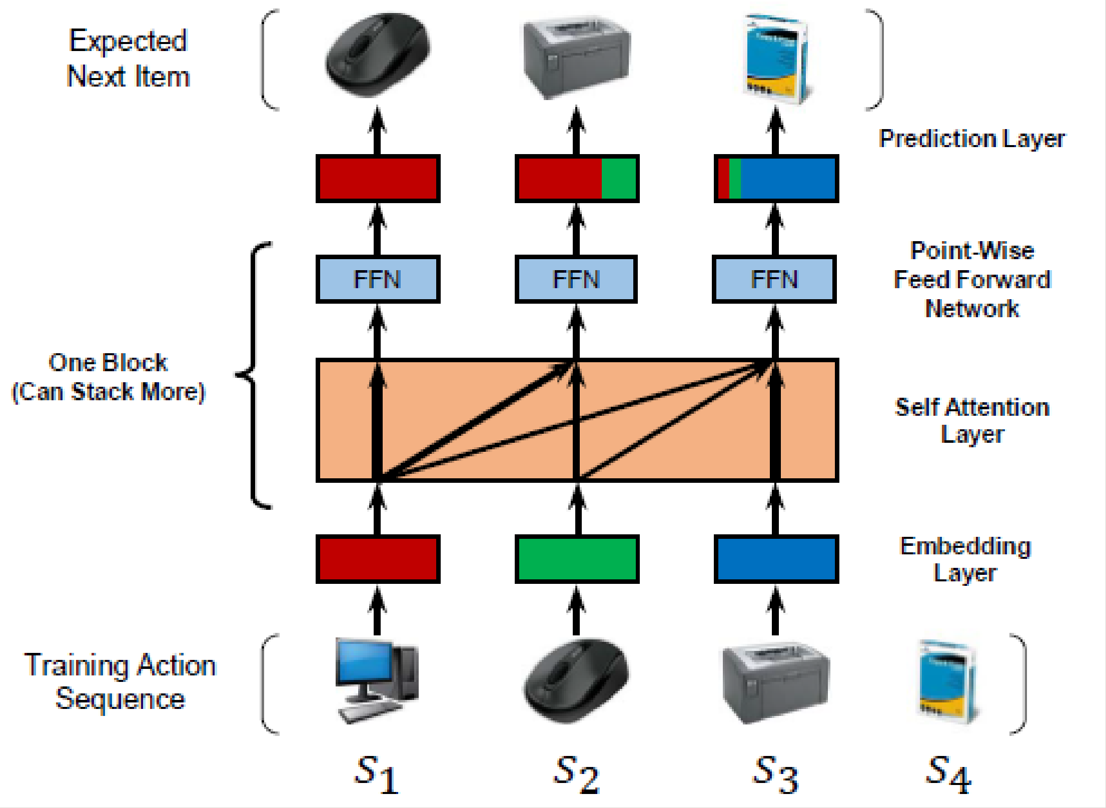
  $\text{Attn}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V$

- Self-Attn Layer

  $\hat{E} = (\widehat{e_1}, \ldots, \widehat{e_n})^t$

  $\rightarrow$ Self-Attn($\hat{E}$) = Attn($\hat{E}W^Q, \hat{E}W^K, \hat{E}W^V$)

  where $W^Q, W^K, W^V \in R^{d \times d}$



2. SASREC

- Self-Attn Layer
  $\hat{E} = (\widehat{e_1}, \dots, \widehat{e_n})^t$

$$\hat{E} = \begin{bmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \end{bmatrix}$$

- $\rightarrow$ Self-Attn($\hat{E}$) = Attn($\hat{E}W^Q, \hat{E}W^K, \hat{E}W^V$) = Attn($\hat{E}_Q, \hat{E}_K, \hat{E}_V$)

  = softmax($\frac{\hat{E}_Q {\hat{E}_K}^T}{\sqrt{d}}$)$\hat{E}_V$

$$= \mathrm{softmax}\left[\begin{bmatrix} e_{q_1} \\ e_{q_2} \\ e_{q_3} \end{bmatrix} \begin{bmatrix} e_{k_1} & e_{k_2} & e_{k_3} \end{bmatrix}\right] \times \begin{bmatrix} e_{v_1} \\ e_{v_2} \\ e_{v_3} \end{bmatrix}$$

$$= \mathrm{softmax} \begin{array}{c} \begin{array}{ccc} e_{v_1} \downarrow & e_{v_2} \downarrow & e_{v_3} \downarrow \end{array} \\ \begin{bmatrix} e_{q_1}e_{k_1} & e_{q_1}e_{k_2} & e_{q_1}e_{k_3} \\ e_{q_2}e_{k_1} & e_{q_2}e_{k_2} & e_{q_3}e_{k_3} \\ e_{q_3}e_{k_1} & e_{q_3}e_{k_2} & e_{q_3}e_{k_3} \end{bmatrix} \end{array}$$

$$= \begin{bmatrix} p_{11}e_{v_1} + p_{12}e_{v_2} + p_{13}e_{v_3} \\ p_{21}e_{v_1} + p_{22}e_{v_2} + p_{23}e_{v_3} \\ p_{31}e_{v_1} + p_{32}e_{v_2} + p_{33}e_{v_3} \end{bmatrix}$$

2. SASREC

- Self-Attn Block(= Self-Attn layer + Feed Forward)

- Definition of Attention

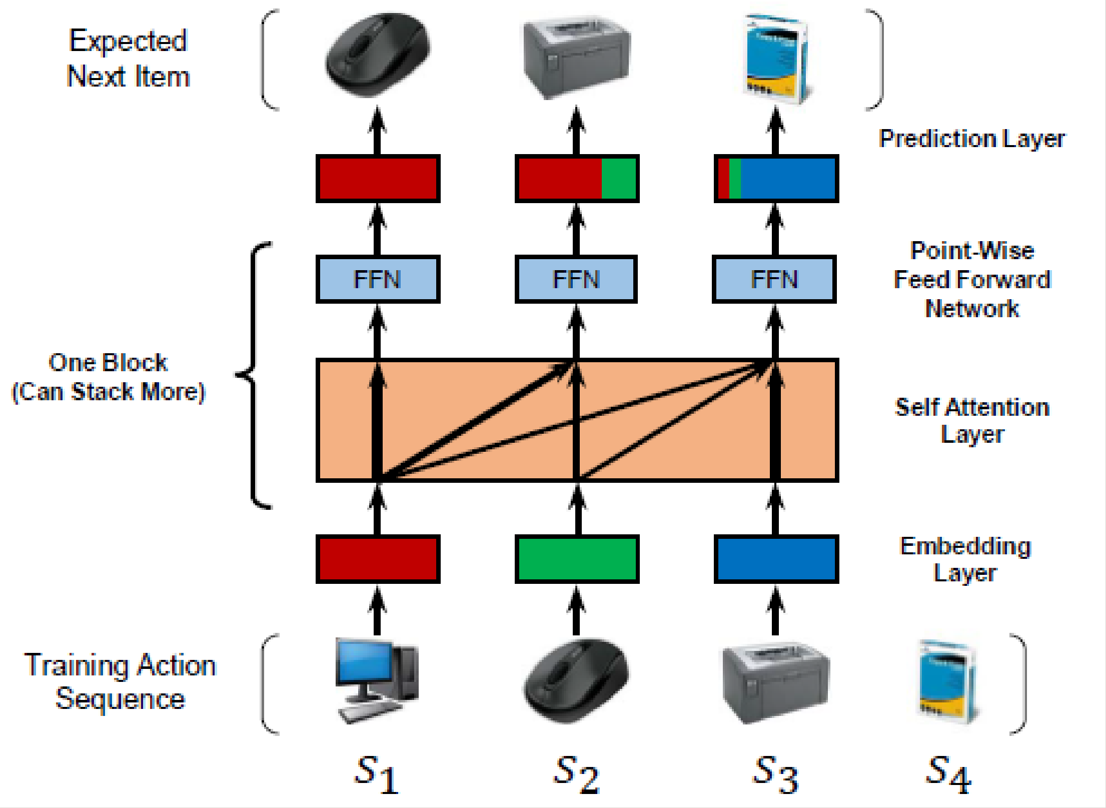
  $\text{Attn}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V$

- Self-Attn Layer

  $\hat{E} = (\widehat{e_1}, \ldots, \widehat{e_n})^t$

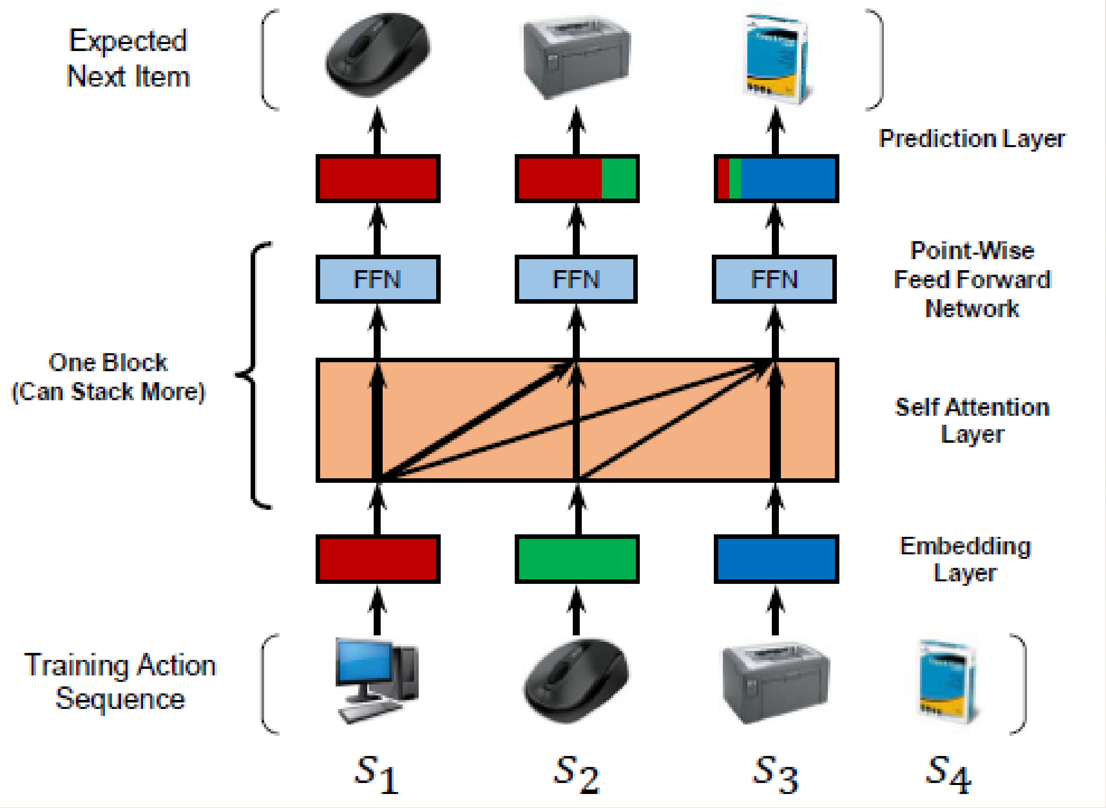
  $\rightarrow$ Self-Attn($\hat{E}$) = Attn($\hat{E}W^Q, \hat{E}W^K, \hat{E}W^V$)

  where $W^Q, W^K, W^V \in R^{d \times d}$



2. SASREC

- Self-Attn layer - Causality
  → to predict 't+1' item, only use first 't' items
  Modify by forbidding links between Q_i, K_j where $(i < j)$



2. SASREC

- Self-Attn Layer
  $\hat{E} = (\widehat{e_1}, \dots, \widehat{e_n})^t$

$$\hat{E} = \begin{bmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \end{bmatrix}$$

- $\rightarrow$ Self-Attn($\hat{E}$) = Attn($\hat{E}W^Q, \hat{E}W^K, \hat{E}W^V$) = Attn($\hat{E}_Q, \hat{E}_K, \hat{E}_V$)

  = softmax$\left(\frac{\hat{E}_Q \hat{E}_K{}^T}{\sqrt{d}}\right)\hat{E}_V$

$$= \text{softmax}\left[\begin{bmatrix} e_{q_1} \\ e_{q_2} \\ e_{q_3} \end{bmatrix} \begin{bmatrix} e_{k_1} & e_{k_2} & e_{k_3} \end{bmatrix}\right] \times \begin{bmatrix} e_{v_1} \\ e_{v_2} \\ e_{v_3} \end{bmatrix}$$

$$= \text{softmax} \begin{array}{c} \begin{array}{ccc} e_{v_1} & e_{v_2} & e_{v_3} \end{array} \\ \begin{bmatrix} e_{q_1}e_{k_1} & & \\ e_{q_2}e_{k_1} & e_{q_2}e_{k_2} & \\ e_{q_3}e_{k_1} & e_{q_3}e_{k_2} & e_{q_3}e_{k_3} \end{bmatrix} \end{array}$$

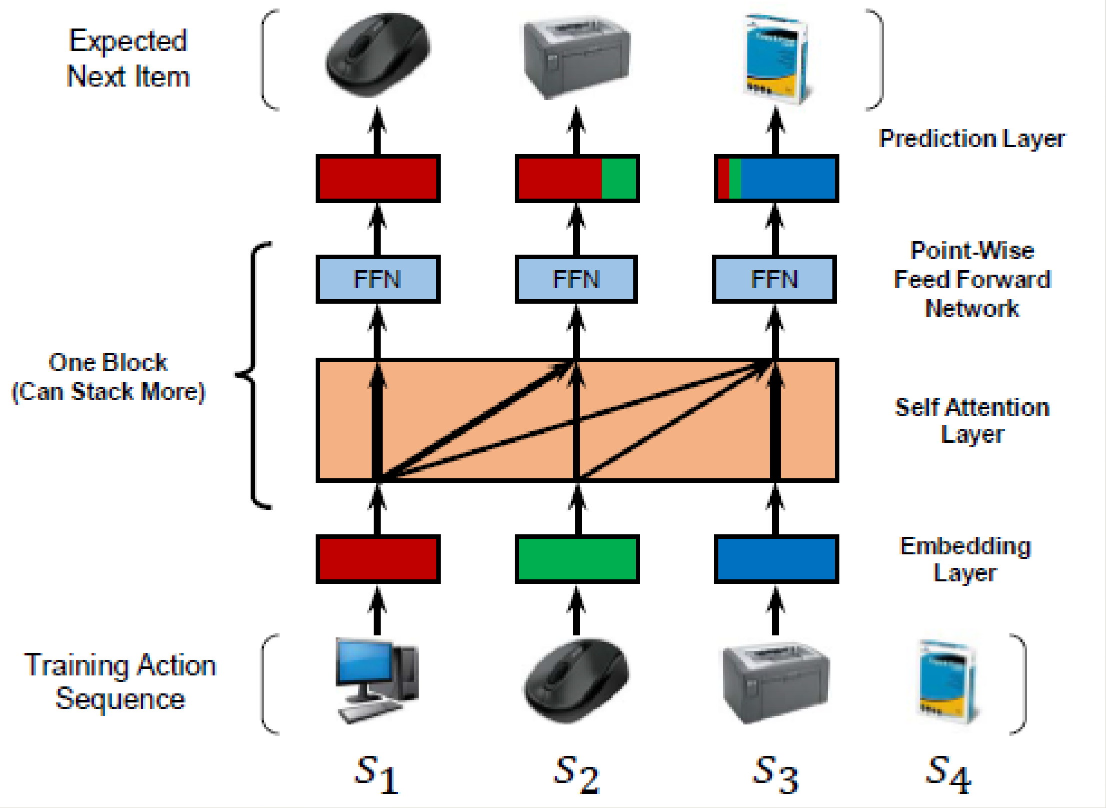
$$= \begin{bmatrix} p_{11}e_{v_1} + \\ p_{21}e_{v_1} + p_{22}e_{v_2} \\ p_{31}e_{v_1} + p_{32}e_{v_2} + p_{33}e_{v_3} \end{bmatrix}$$

2. SASREC

- Self-Attn Block(= Self-Attn layer + Feed Forward)

- Attn layer is linear layer
  → Apply pointwise non-linearity to attn layer result

- Attn$(\hat{E}W^Q, \hat{E}W^K, \hat{E}W^V)$ = S $= (s_1, s_2, \ldots, s_n)$
  → $\mathrm{FFN}(S_i) = \mathrm{ReLU}(S_i W_1 + b_1)W_2 + b_2 = F_i$



2. SASREC

- Stacking Self-Attention Blocks

  stack multiple self-attn blocks b times

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

- But deeper network leads to overfitting, unstability, etc.   Hence use,

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

- residual connections

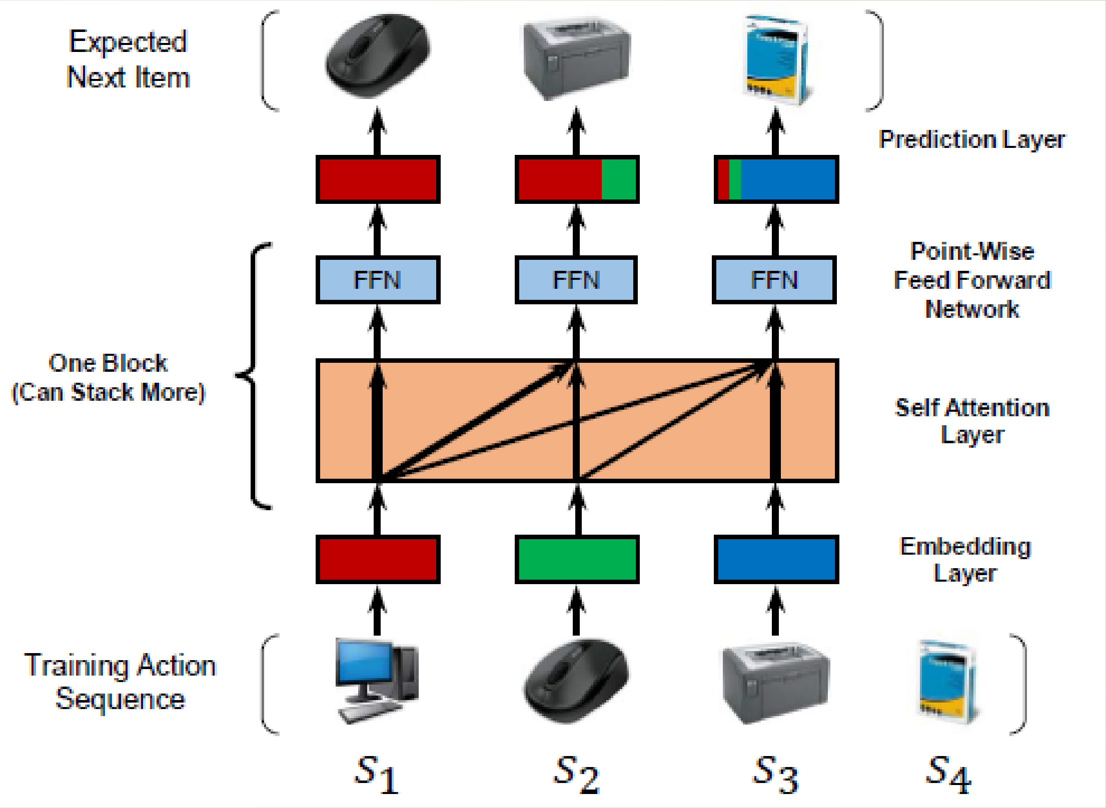
  - direct use of lower level feature is useful

- layer normalization

  - for stabilize

- dropout

  - prevent overfitting



2. SASREC

- Prediction Layer
  predict t+1 items with final $F_t^{(b)}$
  $N \in R^{|I| \times d}$ is item embed matrix

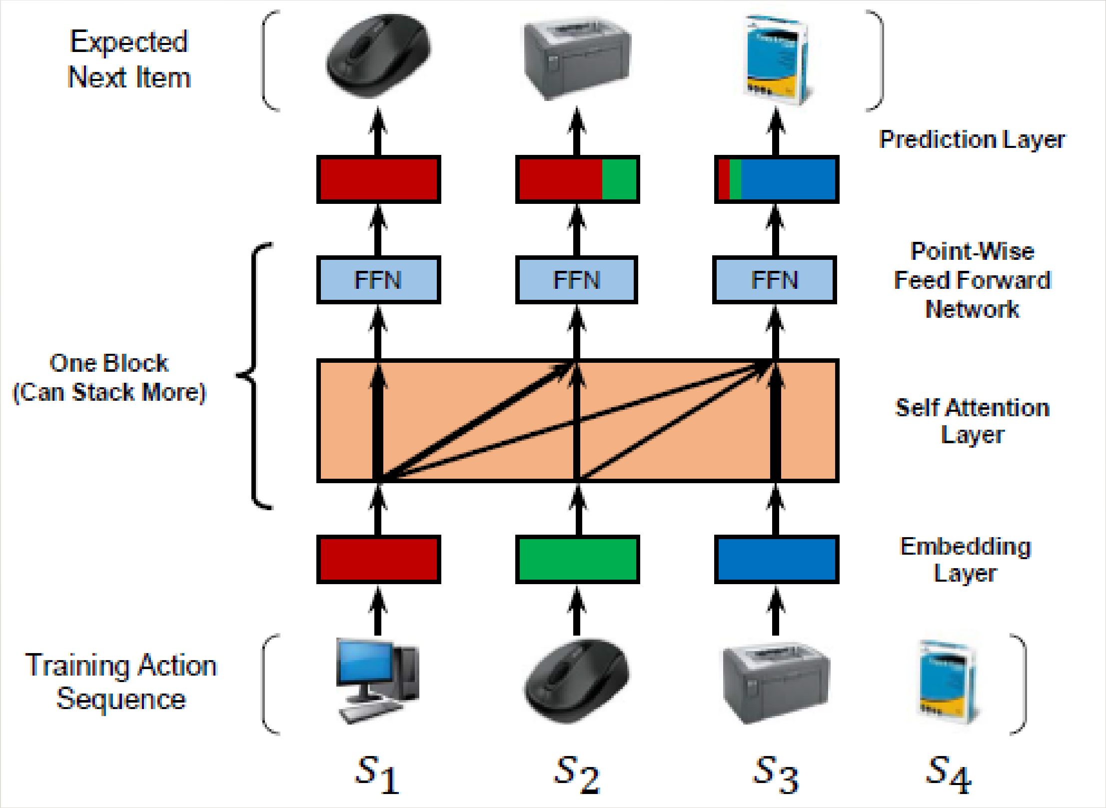
$$r_{i,t} = \mathbf{F}_t^{(b)} \mathbf{N}_i^T,$$

- Shared Item Embedding
  Reuse initial item embed matrix M

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

- Explicit User Modeling
  But does not bring improvement

$$r_{u,i,t} = (\mathbf{U}_u + \mathbf{F}_t^{(b)}) \mathbf{M}_i^T$$



2. SASREC

- Training

  Data : $[S_1^u, S_2^u, \dots, S_{|S^u|-1}^u]$

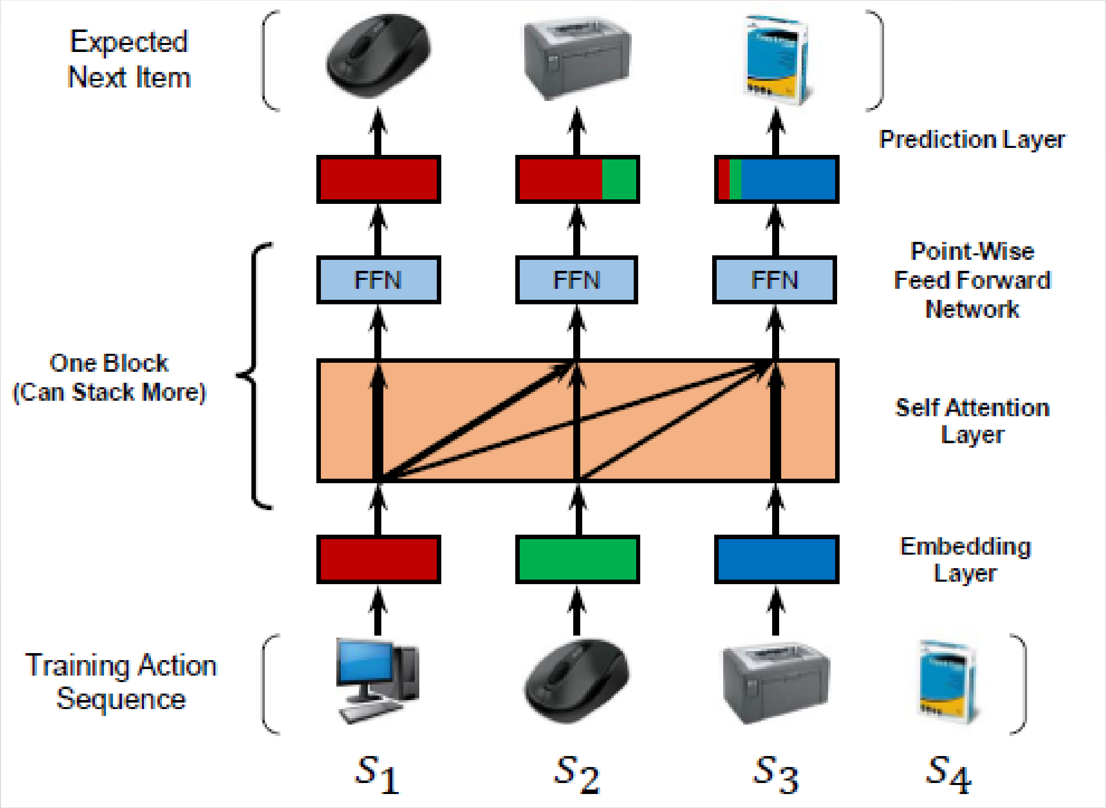
  Input : $s = [s_1, s_2, \dots, s_n]$

  Output

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

- Loss : binary cross entropy loss

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



2. SASREC

Complexity analysis

- Space Complexity(# of params)
- - $O(|I|d + nd + d^2)$, does not grow with number of users(not bad)

Time Complexity

- - $O(n^2 d + nd^2)$, where self-attn is dominant term
- - however computation is parallelizable(very good), 10 times faster than RNN models

Problem : Can't Scale to Very Long Sequence

- - solution 1) use restricted self-attn,   solution 2) split into shorter sequence

2. SASREC

Compared with Markov Chain-based RS

- - by removing all attn-block, pos embedding → SASRec reduces to MC model
- - first order MCs perform well on sparse datasets, but higher order MCs does not show big improvement
- - SASRec shows flexible attention to recent & distant items

Compared with RNN-based RS

- - RNNs are suited to modeling sequences, but can't parallel compute
- - self-attention model is gaining popularity, also can parallel compute
- - RNN has $O(n)$ maximum path length(from input node to related output node), SASRec has $O(1)$ maximum path length, and can learn long-range dependencies

3. EXPERIMENTS & EVALUATION

- Evaluation
- Datasets & Preprocess
  - sparse : Amazon Beauty, Amazon Games,
  - dense : Steam, MovieLens

Table II: Dataset statistics (after preprocessing)

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

- Preprocess
  1) treat review, rating as implicit user-item feedback
  2) use timestamps to determine sequence
  3) discard users, items with fewer than 5 interaction
  4) for sequence of length k,
  split train : 1 ~ k-2, val : k-1, test : k

- Implementation Details
  Two self-attn blocks, learnable position embeddings, shared item embedding weights in input / predict layer

3. EXPERIMENTS & EVALUATION

- Comparison models

  - general : PopRec, Bayesian Personalized Ranking

  - MC based* : FMC, FPMC, TransRec

  - DL based : GRU4Rec, GRU4Rec+, Caser

  - models such as timeSVD++ is not considered

- Evaluation metrics

  - Hit Rate@10, NDCG@10 with 100 random negative samples

FMC : Factorized Markov Chain
FPMC : Factorized Personalized Markov Chain
TransRec : Translation-based Recommendation

3. EXPERIMENTS & EVALUATION

- Markov Chain based performs well on sparse data
- DL based performs well on dense data

Table III: Recommendation performance. The best performing method in each row is boldfaced, and the second best method in each row is underlined. Improvements over non-neural and neural approaches are shown in the last two columns respectively.

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	<u>0.4607</u>	0.2125	0.3949	0.4264	**0.4854**	5.4%	13.8%
	NDCG@10	0.2277	0.2183	0.2477	0.2891	<u>0.3020</u>	0.1203	0.2556	0.2547	**0.3219**	6.6%	25.9%
Games	Hit@10	0.4724	0.4853	0.6358	0.6802	<u>0.6838</u>	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	<u>0.4759</u>	0.3214	**0.5360**	14.5%	12.6%
Steam	Hit@10	0.7172	0.7061	0.7731	0.7710	0.7624	0.4190	<u>0.8018</u>	0.7874	**0.8729**	13.2%	8.9%
	NDCG@10	0.4535	0.4436	0.5193	0.5011	0.4852	0.2691	<u>0.5595</u>	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	<u>0.7886</u>	**0.8245**	8.5%	4.6%
	NDCG@10	0.2377	0.3287	0.4676	0.5176	0.3969	0.3381	0.5513	<u>0.5538</u>	**0.5905**	14.1%	6.6%

3. EXPERIMENTS & EVALUATION

- Increasing hidden dimension d shows consistent improvement

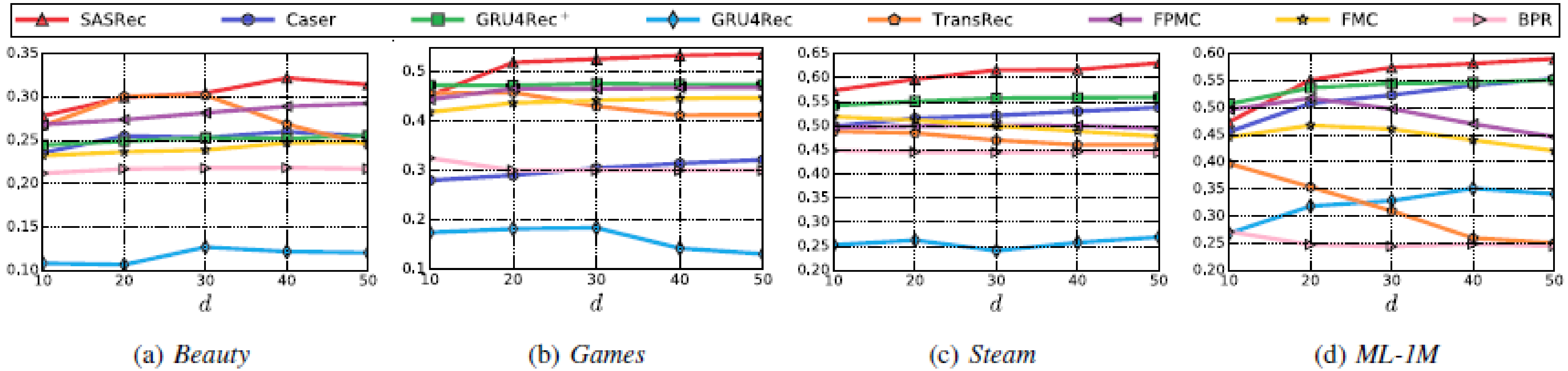


Figure 2: Effect of the latent dimensionality d on ranking performance (NDCG@10).

3. EXPERIMENTS & EVALUATION

- SASRec shows faster and efficient training compared to other models (left)
- SASRec can easily scale for longer sequence length n (right)

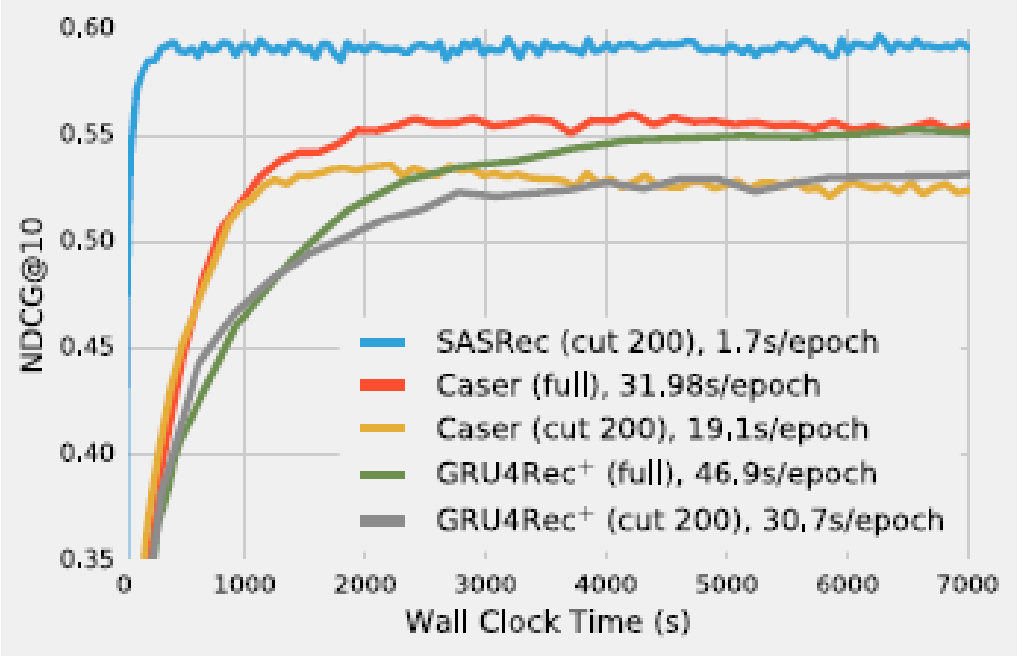


Table V: Scalability: performance and training time with different maximum length n on *ML-1M*.

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

3. EXPERIMENTS & EVALUATION

- Abalation Study
- 1) Remove Positional Embedding
  - without pos emb, there is no order info for past item sequence
  - performance increases for sparse dataset, worsens for dense dataset
- 2) Unshared Item Embedding
  - performance worsens, possibly overfitting
- 3) Remove Residual Connections
  - performance is slightly worse

Table IV: Ablation analysis (NDCG@10) on four datasets. Performance better than the default version is boldfaced. '↓' indicates a severe performance drop (more than 10%).

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

3. EXPERIMENTS & EVALUATION

- Abalation Study
- 4) Remove Dropout

  - performance worsens

- 5)-7) Number of blocks

  - 0 block is worst, 2~3 blocks shows similar performance

- 8) Multi-head attention

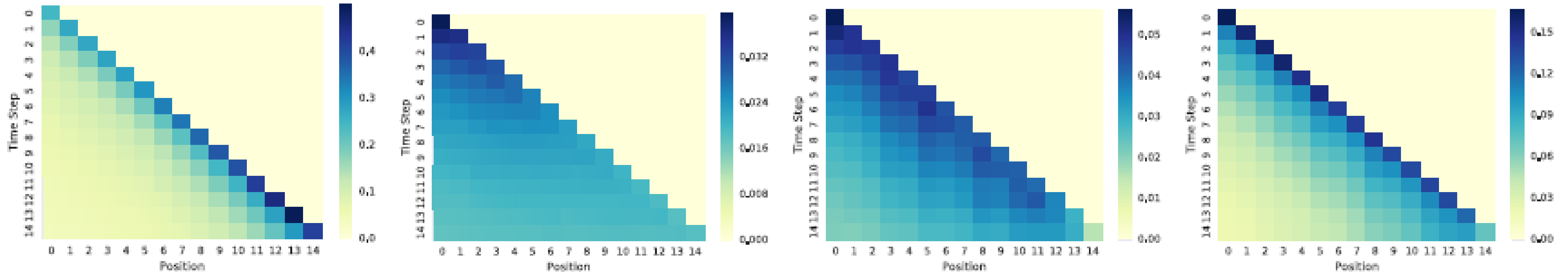
  - multi-head is worse than single-head, maybe because model is too small for multi-head

Table IV: Ablation analysis (NDCG@10) on four datasets. Performance better than the default version is boldfaced. '↓' indicates a severe performance drop (more than 10%).

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

3. EXPERIMENTS & EVALUATION

- Visualizing Attention Weights



(a) *Beauty*, Layer 1 (b) *ML-1M*, Layer 1, w/o PE (c) *ML-1M*, Layer 1 (d) *ML-1M*, Layer 2

Figure 4: Visualizations of average attention weights on positions at different time steps. For comparison, the heatmap of a first-order Markov chain based model would be a diagonal matrix.

3. EXPERIMENTS & EVALUATION

- Visualizing Attention Weights
- (a),(c) : for beauty dataset, attention on recent item is enough compared to MovieLens
- → shows why MC model can be effective & shows SASRec is adaptive
- (b),(c) : without positional information, attention is spread uniformly
- (c),(d) : attention varies for different blocks, 1st attn layer considers distant items, higher layer considers recent item

3. EXPERIMENTS & EVALUATION

- Summary
- - self attention based sequential model SASRec
- - pos embedding layer, self-attn layer
- - models the entire sequence with self-attention, no recurrent element
- - faster, better performance

- Future works
- - incorporate rich context information(dwell time, action types, locations, devices), long sequences(clicks)

MISC

- Personal thoughts on temporal / sequential recsys
- - diverse dynamics, patterns can be observed and modelled compared to non-temporal models
- - especially data of different domains will show different patterns

감사합니다