Self-Attentive Sequential Recommendation

ICDM 2018, Kang and McAuley

Contents

Abstraction

- Sequential Dynamics
  - A key feature of many modern recommender systems
  - To capture the ‘context’ of users’ activities
    based on actions they have performed recently

- Previous Approaches
  - Markov Chains
    - Good to extremely sparse datasets
  - Recurrent Neural Networks
    - Good to denser datasets for complex modeling

Abstraction

- Self-Attention based Sequential Model (SASRec)
  - Balanced between sparse and dense datasets
  - Capture long-term semantics (like an RNNs)
  - Predict based on relatively few actions by self-attention mechanism

- SOTA on All Datasets, All Metrics

Introduction

- Markov Chains
  - Assume that the next action is conditioned on only the previous action (or previous few)
  - Have been successfully adopted to characterize short-range item transitions
  - Perform well in high-sparsity settings
  - Rendel et al., “Factorizing personalized markov chains for next-basket recommendation”, WWW, 2020
- Recurrent Neural Networks
  - Summarize all previous actions via a hidden state
  - Require large amounts of data
  - Hidasi et al., “Session-based recommendations with recurrent neural networks”, ICLR 2016

Introduction

- Transformer
  - Is highly efficient and capable of uncovering syntactic and semantic patterns by attention mechanism, called self-attention
  - Vaswani et al., “Attention is all you need”, NIPS, 2017

- SASRec
  - Applies self-attention mechanisms to sequential recommendation problems
  - To draw context from all actions in the past
  - To frame predictions in terms of just a small number of actions

Related Work - General Recommendations

- Matrix Factorization
  - Uncovers latent dimensions to represent users’ preferences and items’ properties
  - Estimates interactions through the inner product between the user and item embeddings
- Item Similarity Models
  - Learn an item-to-item similarity matrix
  - Estimate a user’s preference toward an item

    via measuring its similarities with items

    that the user has interacted with before

Related Work - Temporal Recommendation

- TimeSVD++
  - Splits time into several segments and model users and items separately in each
  - Exhibits significant temporal ‘drift’
    - “How have movie preferences changed in the last 10 years”
    - “What kind of businesses do users visit at 4pm”

Related Work - Sequential Recommendation

- Sequantial Rec. Differs from Temporal Rec.
  - Only considers the order of actions
  - Models sequential patterns independent of time
  - Tries to model the 'context' of users' actions
    based on their recent activities
- Markov Chains
  - Assume the next action item is related to several previous actions
  - First-order or higher-order MCs
  - Shows strong performance especially on sparse datasets

Related Work - Sequential Recommendation

- Convolutional Sequence Embedding (Caser)
  - Views the embedding matrix of L previous items as an ‘image’
  - Applies convolutional operations to extract transitions

- RNN-based Methods
  - GRU4Rec
    - Uses GRUs to model click sequences
  - Less efficient because of their difficulty of the parallelism

Related Work - Attention Mechanisms

- Ideas behind the Attention Mechanism
  - Sequential outputs each depends on ‘relevant’ parts of some input
    that the model should focus on successively
  - More interpretable (by their attention weights)
- Attention Mechanisms in Recommender Systems
  - Chen et al., “Attentive collaborative filtering”, SIGIR, 2017
  - Xiao et al., “Attentional factorization machines: Learning the weight of feature interactions via attention networks”, IJCAI, 2017
  - Wang et al., “Attention-based transactional context embedding for next-item recommendation”, AAAI, 2018

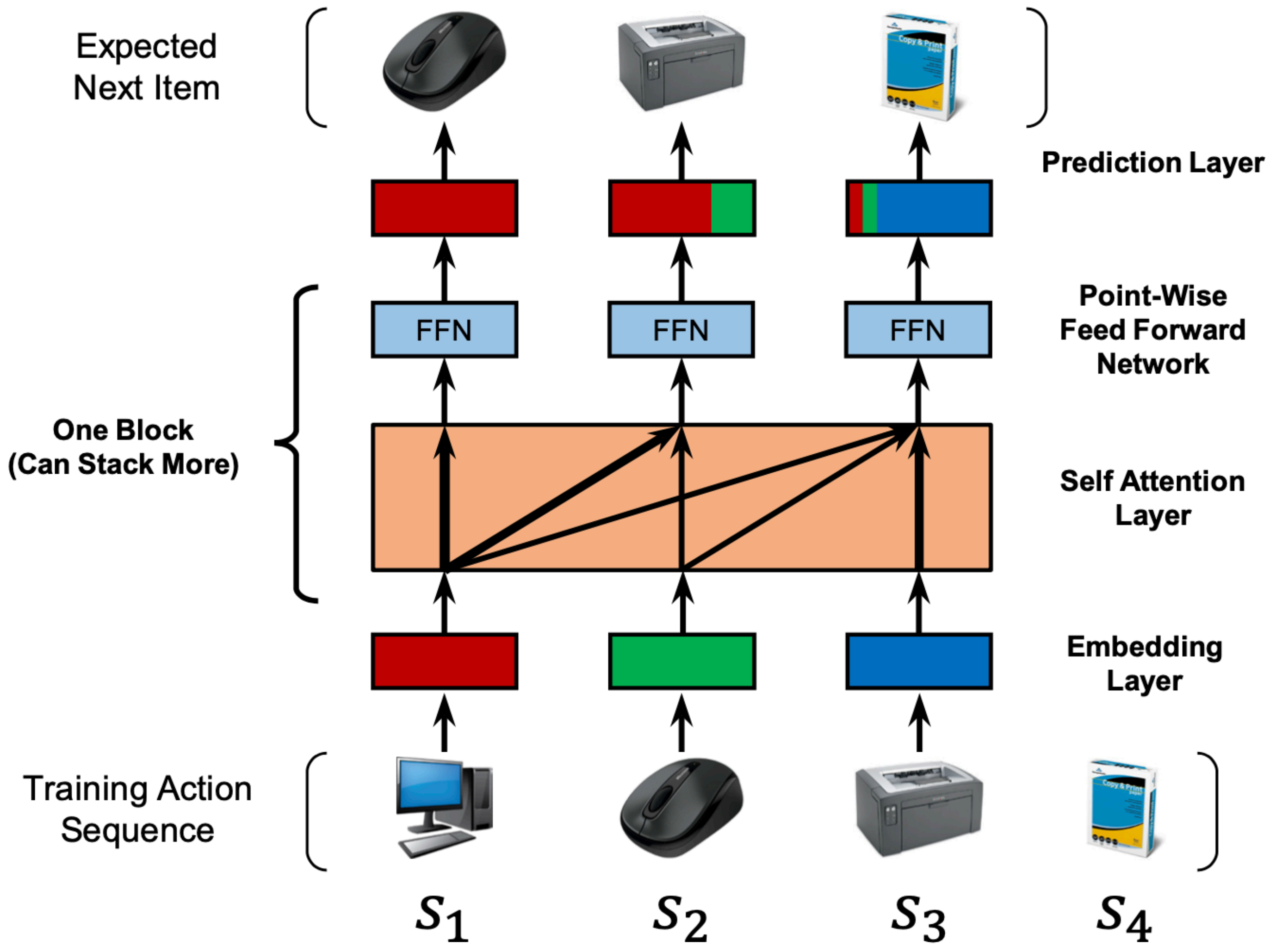
Related Work - Attention Mechanisms

- Difference from SASRec
  - Previous attention mechanisms are additional components to the original model
  - But the Transformer relies heavily on the ‘self-attention’ modules

- SASRec
  - Inspired by the Transformer
  - Builds a new sequential recommendation model

    based on the self-attention approach

Methodology - The Training Process



Methodology - 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

Methodology - Embedding Layer

- Item Embedding Matrix
  - $M \in \mathbb{R}^{|\mathcal{I}| \times d}$
- Positional Embedding Matrix
  - $P \in \mathbb{R}^{n \times d}$
- Input Embedding

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

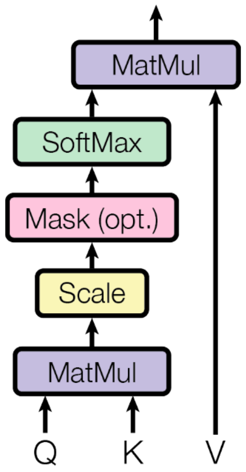
Methodology - Self-Attention Block

- Scaled Dot-product Attention
  - $Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$
- Self-Attention Layer
  - $S = SA(\hat{E}) = Attention(\hat{E}W^Q, \hat{E}W^K, \hat{E}W^V)$
- Causality
  - The model should consider only the first t items when predicting the (t+1)-st item
  - By masking the key vectors as zero between Q_i and K_j ($j > i$)

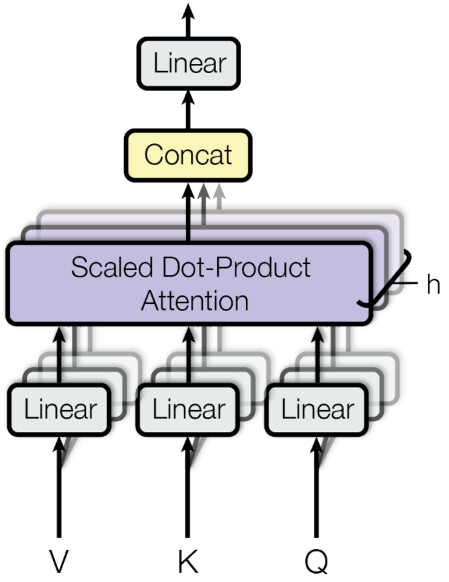
Methodology - Self-Attention Block

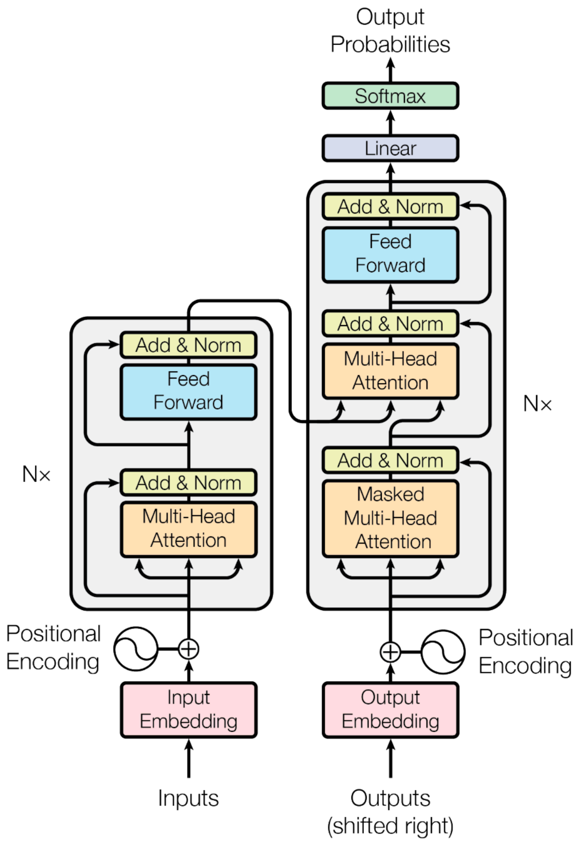
- Point-wise Feed-forward Network
  - $F_i = FFN(S_i) = ReLU\left(S_i W^{(1)} + b^{(1)}\right) W^{(2)} + b^{(2)}$

Scaled Dot-Product Attention



Multi-Head Attention





Methodology - Stacking Self-Attention Block

- The b-th Block
  - $S^{(b)} = SA\left(F^{(b-1)}\right)$
  - $F_i^{(b)} = FFN\left(S_i^{(b)}\right)$
    - $\forall i \in \{1,2,\dots,n\}$
- Block to Block Operations
  - $g(x) = \underbrace{x}_{\text{Residual Connection}} + \underbrace{Dropout}_{\text{Dropout}}(g(\underbrace{LayerNorm(x)}_{\text{Layer Normalization}}))$

Residual Connection Dropout Layer Normalization

- where g(x) represents the self attention layer or feed-forward network layer
- $LayerNorm(x) = \alpha \odot \frac{X-\mu}{\sqrt{\sigma^2+\varepsilon}} + \beta$

Methodology - Stacking Self-Attention Block

- Why Block to Block Operations?
  - Residual Connection, Layer Normalization
    - The training process becomes unstable (vanishing gradients)
    - Models with more parameters often require more training time
  - Dropout
    - The increased model capacity leads to overfitting

Methodology - Prediction Layer

- MF Layer to Predict the Relevance of Item i
  - $r_{i,t} = F_t^{(b)} N_i^T$
    - where $N \in \mathbb{R}^{|\mathcal{I}| \times d}$ is an item embedding matrix
  - Generates recommendations by ranking the scores
- Shared Item Embedding
  - $r_{i,t} = F_t^{(b)} M_i^T$
    - which means $N_i^T = M_i^T$
  - An issue on homogeneous item embedding
    - Inner products cannot represent asymmetric item transitions
      - "item i is frequently bought after j, but not vise versa"
    - But SASRec can learn a nonlinear transformation easily
      - since $FFN(M_i)M_j^T \neq FFN(M_j)M_i^T$

Methodology - Prediction Layer

- Explicit User Modeling
  - $F_t^{(b)}$ is some kind of “implicit” user embedding from previous actions
  - Adding “explicit” user embedding at the last layer
  - So the relevance again is
    - $r_{u,i,t} = \left(U_u + F_t^{(b)}\right) M_i^T$
      - where U is user embedding matrix

Methodology - Network Training

- The Expected Output as Time Step t

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

- The Objective Function
  - Binary cross entropy

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

Methodology - Complexity Analysis

- Space Complexity
  - The total number of parameters
  - $O(|\mathcal{I}|d + nd + d^2)$

- Time Complexity
  - $O(n^2d + nd^2)$
  - Dominant term is $O(n^2d)$ from the self-attention layer
    - Moreover, it can be fully parallelizable

Methodology - Discussion

- Factorized Markov Chains
  - $P(j|i) \propto M_i^T N_j$
  - SASRec can be reduced by
    - Setting the self-attention block to zero
    - Using unshared item embeddings
    - Removing position embedding
- Factorized Personalized Markov Chains
  - $P(j|u,i) \propto [U_u, M_i^T] N_j$
  - SASRec can be extended by
    - Adding an explicit user embedding

Methodology - Discussion

- Factorized Item Similarity Models
  - $P(j|u) \propto \left(\frac{1}{|S^u|}\sum_{i\in S^u} M_i\right) M_j^T$
  - SASRec can be reduced by
    - Using only one self-attention layer excluding the feed-forward network
    - Setting uniform attention weights
    - Using unshared item embeddings
    - Removing position embedding

Experiments

- Research Questions
  - RQ1: Does SASRec outperform state-of-the-art model including CNN/RNN based methods?
  - RQ2: What is the influence of various components in the SASRec architecture?
  - RQ3: What is the training efficiency and scalability (regarding n) of SASRec?
  - Are the attention weights able to learn meaningful patterns related to positions or items' attributes?

Experiments - Datasets

- 4 Datasets from 3 Real World Applications

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

Experiments - Comparison Methods

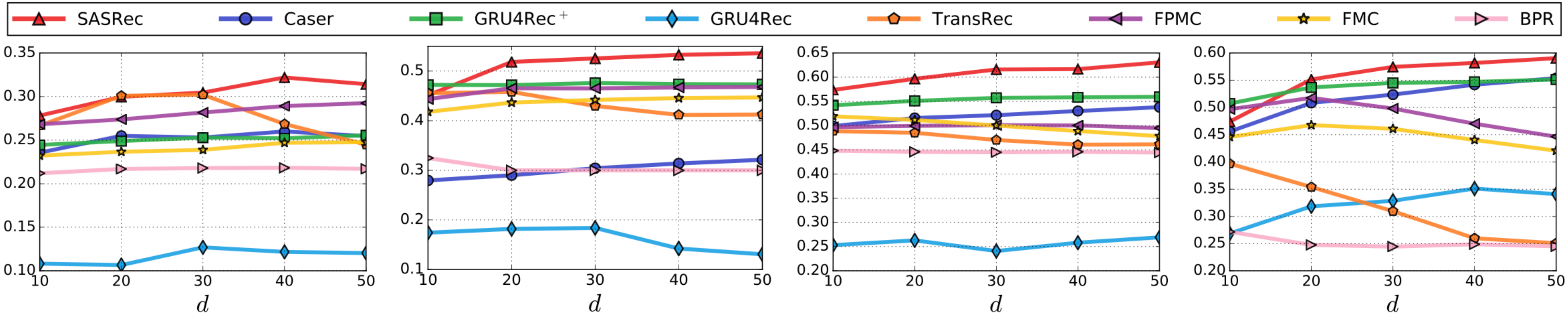
- General Recommendation Methods
  - PopRec
  - Bayesian Personalized Ranking (BPR)
- First-order Markov Chains
  - Factorized Markov Chains (FMC)
  - Factorized Personalized Markov Chains (FPMC)
  - Translation-based Recommendation (TransRec)
- Deep Learning based Sequential Rec. Systems
  - GRU4Rec
  - GRU4Rec$^+$
  - Convolutional Sequence Embeddings (Caser)

Experiments - Evaluation Metrics

- Hit Rate@10
  - Counting the fraction of times that the ground-truth next item is among the top 10
  - Equivalent to Recall@10
- NDCG@10
  - A position-aware metric which assigns larger weights on higher positions
- Sampling Strategy
  - For each user u, we randomly sample 100 negative items and rank these items with the ground-truth item

Experiments - Recommendation Performance (RQ1)

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%



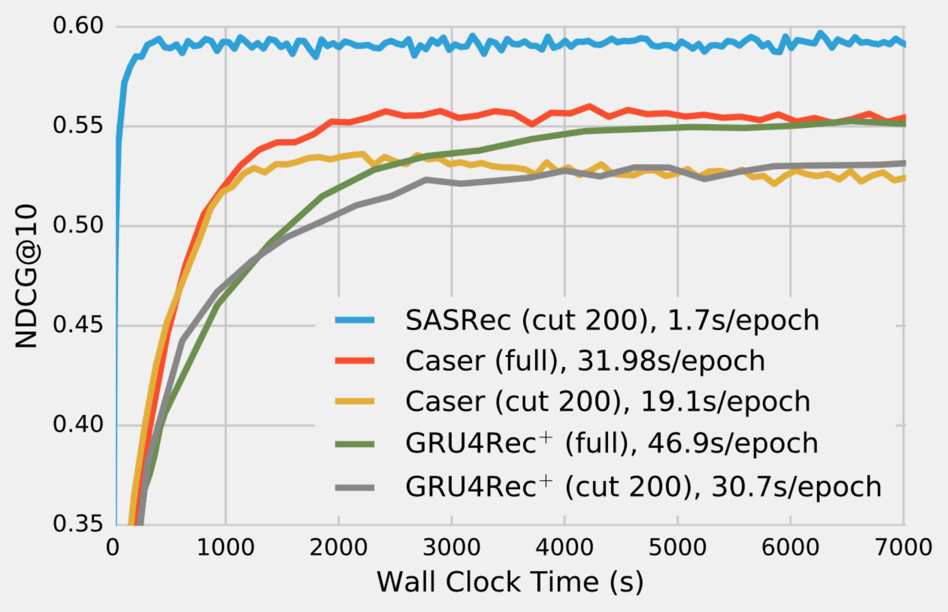
(a) *Beauty* (b) *Games* (c) *Steam* (d) *ML-1M*

Experiments - Ablation Study (RQ2)

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

- NDCG@10 on 4 datasets
- ↓ indicates a severe performance drop (more than 10%)

Experiments - Training Efficiency & Scalability (RQ3)



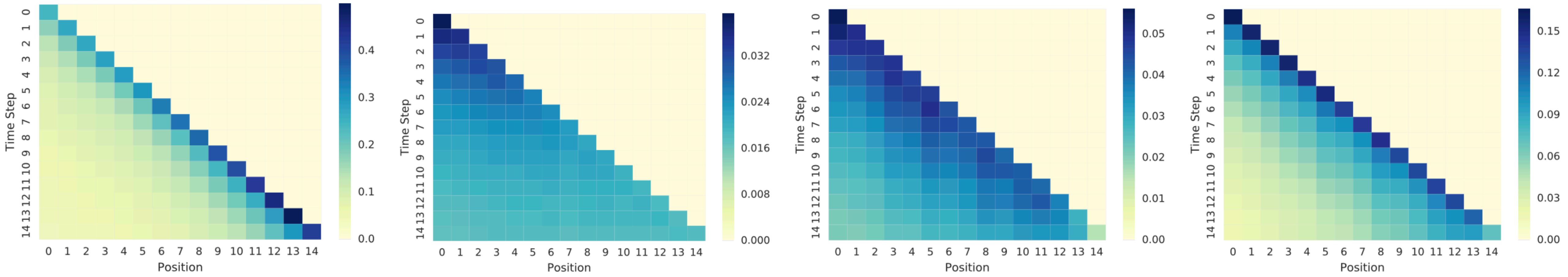
- Training time per epoch
- Total training time

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

- Performance and training time with max sequence length n

Experiments - Visualizing Attention Weights (RQ4)

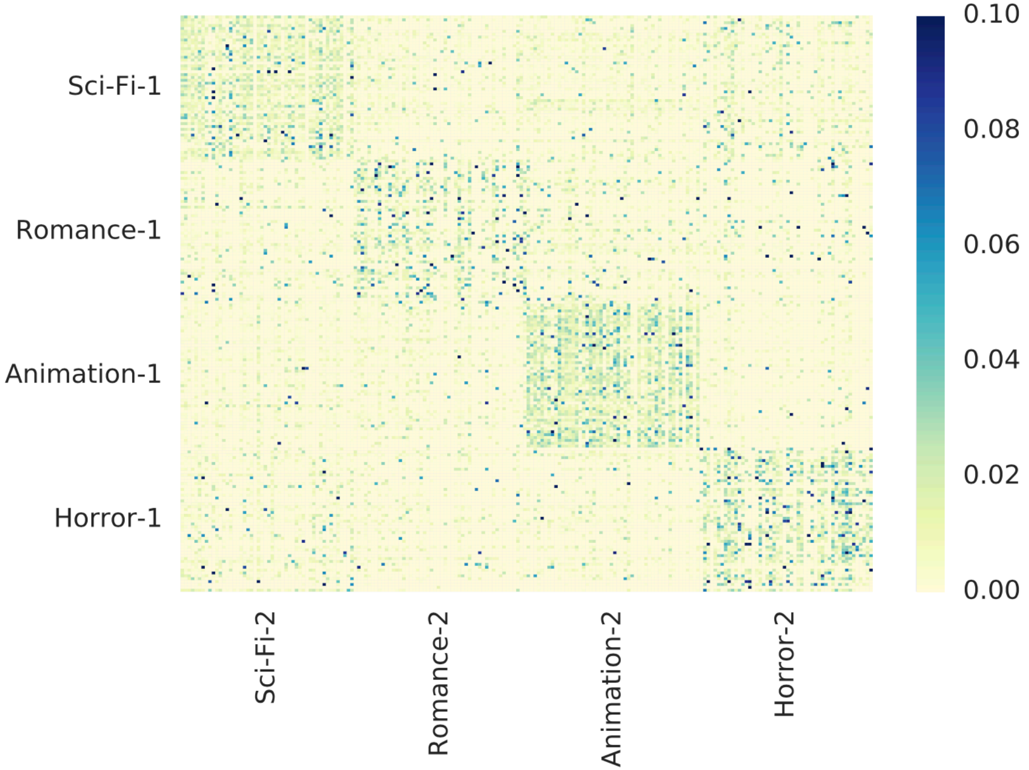
- Attention on Positions
  - (a) vs (c): sparse vs dense datasets
  - (b) vs (c): the effect of using positional embeddings
  - (c) vs (d): lower vs higher layers



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

Experiments - Visualizing Attention Weights (RQ4)

- Attention Between Items
  - MovieLens-1M, 200 movies from 4 categories
  - Heatmap of average attention weight between query and key sets



Conclusion

- A novel self-attention based sequential model "SASRec" for next item recommendation
- The state-of-the-art performance on both sparse and dense datasets
- Faster speed than CNN/RNN based approaches

Thank you!