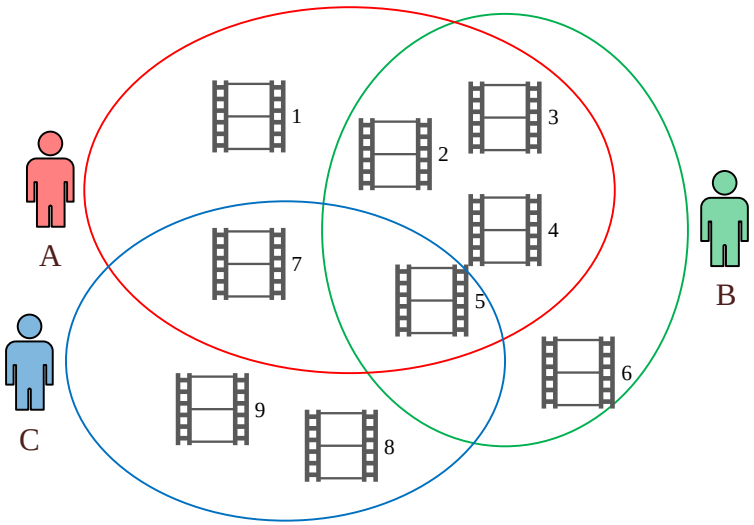
# Self-Attentive Sequential Recommendation

Jing Lin (revised by Weike Pan)

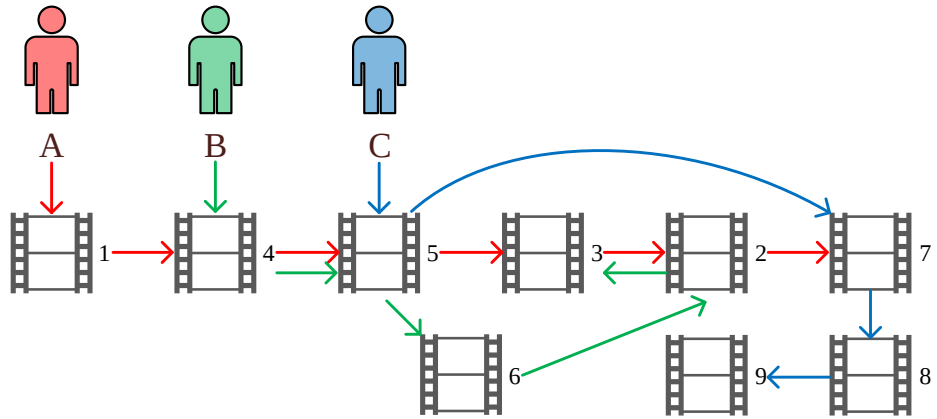
College of Computer Science and Software Engineering
Shenzhen University

**Reference:** Self-Attentive Sequential Recommendation (ICDM 2018) by Wang-Cheng Kang and Julian J. McAuley

# Problem Definition




(a) general recommendation




(b) sequential recommendation

**Sequential Recommendation (Next-Item Recommendation)**

- **Input:** $(u, \mathcal{S}^u)$, i.e., a sequence of items for each user $u$, where $u \in \mathcal{U}$ and $\mathcal{S}^u = \{s_1^u, s_2^u, \ldots, s_{|\mathcal{S}^u|}^u\}$, $s_\cdot^u \in \mathcal{I}$.
- **Goal:** 1) Predict the preference of user $u$ to item $i$ at the $(\ell+1)$th time step, i.e., $\hat{r}_{\ell+1,i}^u$, where $i \in \mathcal{I} \backslash \mathcal{S}^u$ and $\ell \in \{1, \ldots, |\mathcal{S}^u|\}$. 2) Provide a top-N recommendation list for each user $u$, in which we expect the real next interacted item $s_{|\mathcal{S}^u|+1}^u$ to appear and be ranked as high as possible.

# Notations (1/2)

Table: Some notations.

| | |
|---|---|
| $\mathcal{U}$ | the whole set of users |
| $\mathcal{I}$ | the whole set of items |
| $u \in \mathcal{U}$ | user ID |
| $i \in \mathcal{I}$ | item ID |
| $\mathcal{S}^u \subseteq \mathcal{I}$ | a set/sequence of items that have been interacted by user $u$ |
| $s^u_\ell \in \mathcal{S}^u$ | the $\ell$th item interacted by user $u$ |
| $\hat{r}^u_{\ell+1,i}$ | predicted preference of user $u$ to item $i$ at the $(\ell+1)$th time step |

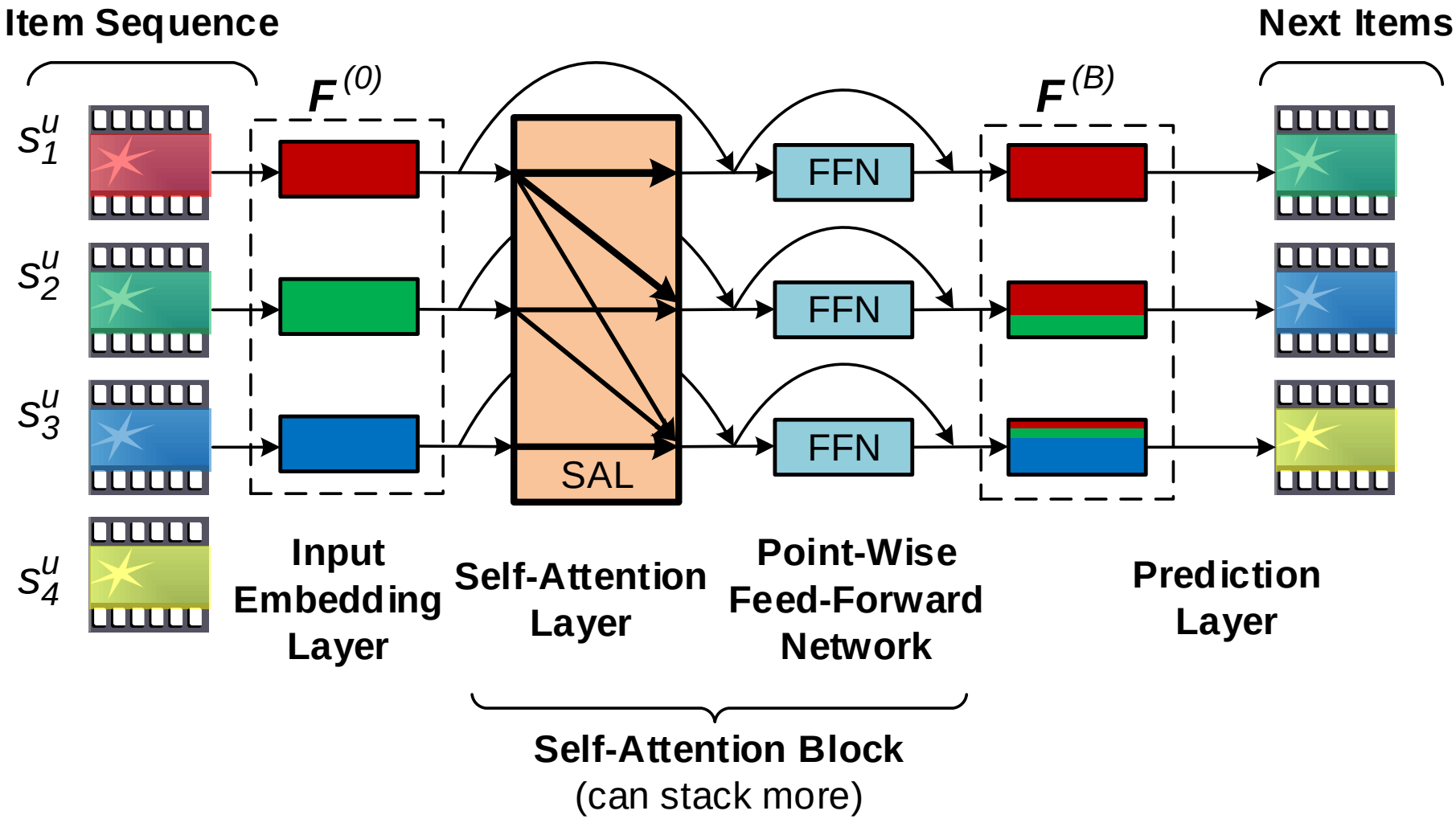
# Notations (2/2)

Table: Some notations (cont.).

| | |
|---|---|
| $L \in \mathbb{N}$ | maximum sequence length |
| $d \in \mathbb{N}$ | latent vector dimensionality |
| $B \in \mathbb{N}$ | number of self-attention blocks |
| $M \in \mathbb{R}^{\lvert\mathcal{I}\rvert \times d}$ | learnable item embedding matrix |
| $\boldsymbol{P} \in \mathbb{R}^{L \times d}$ | learnable position embedding matrix |
| $\boldsymbol{F}^{(0)} \in \mathbb{R}^{L \times d}$ | input embedding matrix for the first self-attention block (SAB) |
| $\boldsymbol{F}^{(b)} \in \mathbb{R}^{L \times d}$ | output embedding matrix of the $b$th SAB, $b \geq 1$ |

Note that we use capital letters in bold to denote matrices and their lowercase form to denote the corresponding row vectors.

# Illustration



# Input Embedding Layer

- We fix the input sequence of each user $u$ by extracting his/her latest $L$ behaviors, which is abbreviated as $\mathcal{S}^u = \{s_1^u, s_2^u, \ldots, s_L^u\}$. Note that padding items are appended at the beginning of the sequences if they are too short.
- The input embedding matrix $\boldsymbol{F}^{(0)} \in \mathbb{R}^{L \times d}$ of $\mathcal{S}^u$ to be fed into the self-attention network, i.e., the first self-attention block (SAB), is as follows,

$$\boldsymbol{F}^{(0)} = \begin{bmatrix} \boldsymbol{m}_{s_1^u} + \boldsymbol{p}_1 \\ \boldsymbol{m}_{s_2^u} + \boldsymbol{p}_2 \\ \ldots \\ \boldsymbol{m}_{s_L^u} + \boldsymbol{p}_L \end{bmatrix}, \tag{1}$$

where $\boldsymbol{m}_{s_\ell^u} \in \mathbb{R}^{1 \times d}, \ell \in \{1, 2, \ldots, L\}$ is the learnable item embedding of item $s_\ell^u$ at the $\ell$th step of the sequence $\mathcal{S}^u$, and $\boldsymbol{p}_\ell \in \mathbb{R}^{1 \times d}$ is the learnable position embedding at the $\ell$th step shared by all sequences.

# Self-Attention Layer (1/2)

- With an input embedding matrix denoted as $\boldsymbol{X} \in \mathbb{R}^{L \times d}$, a self-attention layer (SAL) is defined as follows,

$$SAL(\boldsymbol{X}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{\Delta} \cdot \boldsymbol{V}, \tag{2}$$

where $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q$, $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K$ and $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V$ with $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d \times d}$ are the projected query, key and value matrices, respectively, to improve the flexibility, and $\boldsymbol{\Delta}$ is a unit lower triangular matrix of size $L \times L$, which represents the causality mask to prevent the transitions from the future.

# Self-Attention Layer (2/2)

- For better propagating the low-layer features and stabilizing, we adopt the residual connection and layer normalization after each sub-layer in an SAB. Specifically, for a self-attention layer, we have

$$\boldsymbol{X}' = SAL_{con}(\boldsymbol{X}) = LayerNorm(\boldsymbol{X} + Dropout(SAL(\boldsymbol{X}))), \quad (3)$$

where for each row vector $\boldsymbol{x} \in \mathbb{R}^{1\times d}$ in the matrix $\boldsymbol{X} \in \mathbb{R}^{L\times d}$

$$LayerNorm(\boldsymbol{x}) = \boldsymbol{\alpha} \otimes \frac{\boldsymbol{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \boldsymbol{\beta}, \quad (4)$$

where $\otimes$ denotes the element-wise product operation, $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}$ are the mean and variance of $\boldsymbol{x}$, $\boldsymbol{\alpha} \in \mathbb{R}^{1\times d}$ and $\boldsymbol{\beta} \in \mathbb{R}^{1\times d}$ are the scaling factors and biases to be learned [Ba et al., 2016].

# Point-Wise Feed-Forward Network

- With an input embedding matrix denoted as $\boldsymbol{X}' \in \mathbb{R}^{L \times d}$, a point-wise feed-forward network (FNN) is defined as follows,

$$FFN(\boldsymbol{X}') = ReLU(\boldsymbol{X}'\boldsymbol{W}_1 + \mathbf{1}^T\boldsymbol{b}_1)\boldsymbol{W}_2 + \mathbf{1}^T\boldsymbol{b}_2, \tag{5}$$

  where $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{d \times d}$ and $\boldsymbol{b}_1, \boldsymbol{b}_2 \in \mathbb{R}^{1 \times d}$ are weights and biases to be learned, and $\mathbf{1}$ is a row vector of ones of size $1 \times L$.
- Note that unlike in the SAL, the correlations among different time steps is no longer considered. And the FNN can also be viewed as two convolutions with kernel size 1 applied on each of the row vectors.
- Similarly, we append a connection layer at the end of a feed-forward network as follows,

$$FFN_{con}(\boldsymbol{X}') = LayerNorm(\boldsymbol{X}' + Dropout(FFN(\boldsymbol{X}'))). \tag{6}$$

# Stacking Self-Attention Blocks (1/2)

- Note that in our implementation, we use a connection layer as that in [Vaswani et al., 2017], i.e.,

$$g_{con}(\boldsymbol{X}) = LayerNorm(\boldsymbol{X} + Dropout(g(\boldsymbol{X}))), \tag{7}$$

  where $g(\cdot)$ can be $SAL(\cdot)$ or $FNN(\cdot)$ in the previous pages.
- Actually, a dropout and normalization layer is also appended at the end of the input embedding layer, (i.e., before it is fed into the sefl-attention network),

$$\boldsymbol{F}^{(0)} \leftarrow LayerNorm(Dropout(\boldsymbol{F}^{(0)})). \tag{8}$$

# Stacking Self-Attention Blocks (2/2)

- We obtain the output embedding matrix $\boldsymbol{F}^{(b)} \in \mathbb{R}^{L \times d}$ of the $b$th self-attention block as follows,

$$\boldsymbol{F}^{(b)} = SAB^{(b)}(\boldsymbol{F}^{(b-1)}) \tag{9}$$

$$= FFN^{(b)}_{con}(SAL^{(b)}_{con}(\boldsymbol{F}^{(b-1)})), \quad b \in \{1, 2, \ldots, B\}. \tag{10}$$

# Prediction Layer

- We predict the preference of user $u$ to item $i$ at the $(\ell+1)$th time step based on the output vector $\boldsymbol{f}_{\ell}^{(B)} \in \mathbb{R}^{1\times d}$ from the self-attention network as follows,

$$\hat{r}_{\ell+1,i}^{u} = \boldsymbol{f}_{\ell}^{(B)} \boldsymbol{m}_i^{T}. \tag{11}$$

# Network Training

- We train SASRec by minimizing the binary cross-entropy loss with the Adam optimizer. The loss function is as follows:

$$\mathcal{L} = -\sum_{u \in \mathcal{U}} \sum_{\ell=1}^{L-1} \delta(s_{\ell+1}^u)[\log(\sigma(r_{\ell+1,s_{\ell+1}^u})) + \log(1 - \sigma(r_{\ell+1,j}^u))], \quad (12)$$

where $j \in \mathcal{I} \backslash S^u$ is a negative item randomly sampled for each prediction. The indicator function $\delta(s_{\ell+1}^u) = 1$ only if $s_{\ell+1}^u$ is not a padding item, otherwise 0.

# Datasets

We follow [Kang and McAuley, 2018] and use Steam in the experiments. We preprocess the dataset as follows:

1. We treat the presence of the review behaviors as positive feedback and order them by the timestamps.
2. We discard later duplicated user-item pairs in order to predict new items.
3. We successively discard items and users with fewer than 5 records to maintain sequentiality.
4. We adopt the leave-one-out evaluation by splitting the dataset into three parts, i.e., the last interaction of each user for test, the penultimate one for validation and the rest for training. Note that during testing, the input sequences contain training actions and the validation action. Note that cold-start items in the test and validation data are also removed.

| Dataset | # Users | # Items | # Interactions | Avg. Length | Density |
|---|---|---|---|---|---|
| Steam | 281,428 | 13,044 | 3,488,899 | 12.40 | 0.10% |

# Evaluation Metrics

- Rec@10
- NDCG@10
- To reduce computation, we follow [Kang and McAuley, 2018] and prearrange a candidate list with 100 randomly sampled un-interacted items for each user according to item popularity.

# Baselines

- Four matrix factorization (MF) based methods:
  - BPRMF [Rendle et al., 2009].
  - FISM [Kabbur et al., 2013].
  - FPMC [Rendle et al., 2010].
  - Fossil [He and McAuley, 2016].
- Two deep learning (DL) based methods:
  - GRU4Rec+ [Hidasi and Karatzoglou, 2018]. An RNN-based model.
  - Caser [Tang and Wang, 2018]. A CNN-based model.

# Implementation Details (1/2)

- We implement the MF-based models with the codes provided by [He and McAuley, 2016] for the research of Fossil[1], and run the DL-based methods GRU4Rec+[2] and Caser[3] with the codes released by the authors of the original papers.
- We implement SASRec with the code for FISSA[4] [Lin et al., 2020], which is based on python 3.5+ and TensorFlow 1.2.1+.

[1] https://cseweb.ucsd.edu/~jmcauley/
[2] https://github.com/hidasib/GRU4Rec
[3] https://github.com/graytowne/caser_pytorch
[4] http://csse.szu.edu.cn/staff/panwk/publications/FISSA/

# Implementation Details (2/2)

- For fair comparison, we fix the item embedding dimensionality $d = 50$ in all models. Other key parameters such as the MC orders ($\in \{1, 2, \ldots, 9\}$ for Fossil and Caser), negative sampling numbers (2048 for GRU4Rec+), filter sizes (4 and 16 for the vertical and horizontal filters, respectively in Caser) and so on are all tuned on the validation data according to the suggestions in the corresponding papers.
- For SASRec, we set the sequence length $L$ to 50, the batch size to 128, the learning rate to 0.001 and the dropout rate to 0.5, and use single-head self-attention layers. The number of blocks $B$ is searched from $\{1, 2, 3\}$.

# Results

Table: Recommendation performance of SASRec and six baselines on Steam.

| Metric | MF-based | | | | DL-based | | |
|---|---|---|---|---|---|---|---|
| | BPRMF | FISM | FPMC | Fossil | GRU4Rec+ | Caser | SASRec |
| Rec@10 | 0.1023 | 0.3183 | 0.1735 | 0.2926 | 0.3177 | 0.2686 | **0.3886** |
| NDCG@10 | 0.0468 | 0.1703 | 0.0849 | 0.1546 | 0.1707 | 0.1342 | **0.2144** |

# Conclusions

- The self-attention based sequential model is effective.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016).

Layer normalization.

He, R. and McAuley, J. (2016).

Fusing similarity models with Markov chains for sparse sequential recommendation.
In *Proceedings of the 16th IEEE International Conference on Data Mining*, ICDM '16, pages 191–200.

Hidasi, B. and Karatzoglou, A. (2018).

Recurrent neural networks with top-k gains for session-based recommendations.
In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 843–852.

Kabbur, S., Ning, X., and Karypis, G. (2013).

FISM: Factored item similarity models for top-N recommender systems.
In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 659–667.

Kang, W. and McAuley, J. J. (2018).

Self-attentive sequential recommendation.
In *Proceedings of the 18th IEEE International Conference on Data Mining*, ICDM '18, pages 197–206.

Lin, J., Pan, W., and Ming, Z. (2020).

FISSA: Fusing item similarity models with self-attention networks for sequential recommendation.
In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, pages 130–139.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009).

BPR: Bayesian personalized ranking from implicit feedback.
In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461.

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2010).

Factorizing personalized Markov chains for next-basket recommendation.
In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 811–820.

Tang, J. and Wang, K. (2018).

Personalized top-N sequential recommendation via convolutional sequence embedding.
In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 565–573.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017).
Attention is all you need.
In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NeurIPS '17, pages 6000–6010.