Dynamic Memory based Attention Network for Sequential Recommendation

Qiaoyu Tan [1], Jianwei Zhang [2], Ninghao Liu [1], Xiao Huang [3]
Hongxia Yang [2], Jingren Zhou [2], Xia Hu [1]
[1] Texas A&M University, [2] Alibaba Group, [3] The Hong Kong Polytechnic University
{qytan,nhliu43,xiahu}@tamu.edu, xiaohuang@comp.polyu.edu.hk
{zhangjianwei.zjw, yang.yhx, jingren.zhou}@alibaba-inc.com

Problem Formulation

- **Sequential Recommendation:** Assume $\mathcal{U}$ and $\mathcal{V}$ denote the sets of users and items, respectively. $S = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{|S|}\}$ represents the behavior sequence in chronological order of a user. $\boldsymbol{x}_t \in \mathcal{V}$ records the t-th item interacted by the user. Given an observed behavior sequence $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{|S|}\}$, the sequential recommendation task is to predict the next items that the user might be interacted with.

Challenges

- Given that the response time in real-world systems is limited, it has become expensive to scan over the entire behavior sequence at each prediction time.

- It is crucial to model the whole behavior sequence for a more accurate recommendation.

- It is hard to explicitly control the contributions of long-term or short-term interests for user modeling.

Methodology

- Truncating the whole user behavior sequence into several successive sub-sequences and optimizes the model sequence by sequence.

- A recurrent attention network and a long-term attention network with dynamic memory are derived for short-term and long-term interest modeling.

- A gate mechanism is applied to adaptively control the importance of the above two interests combination.

Dynamic memory-based self-attention network (DMAN)

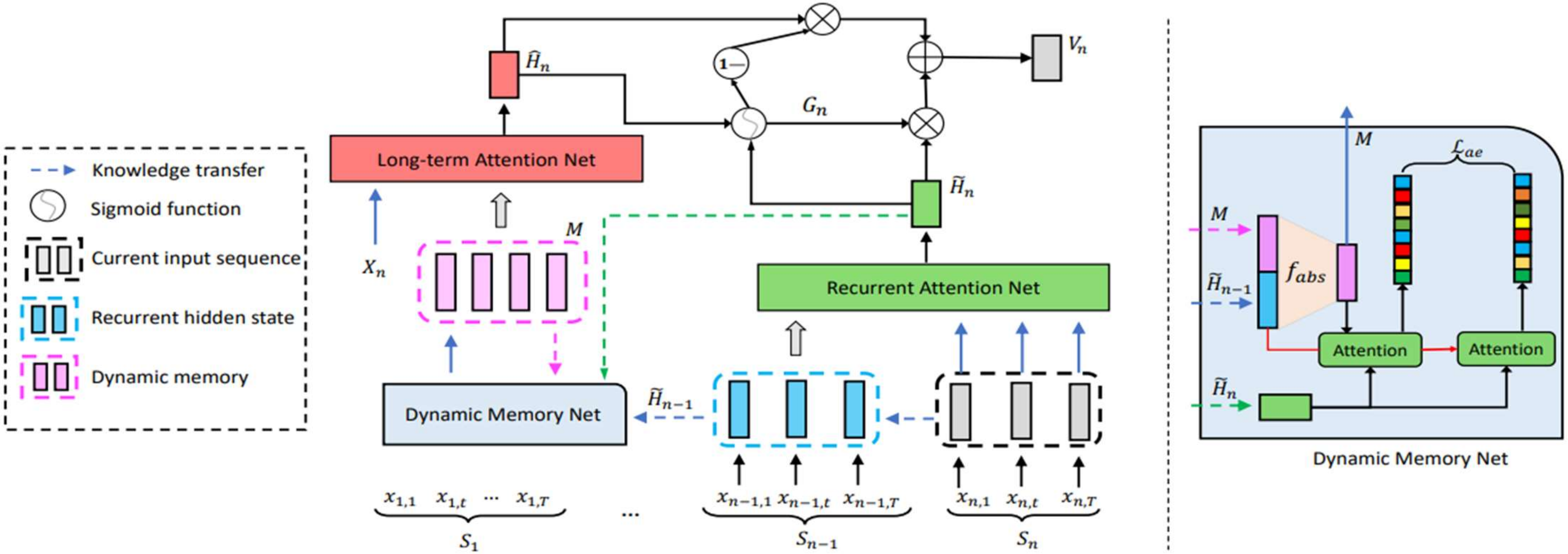
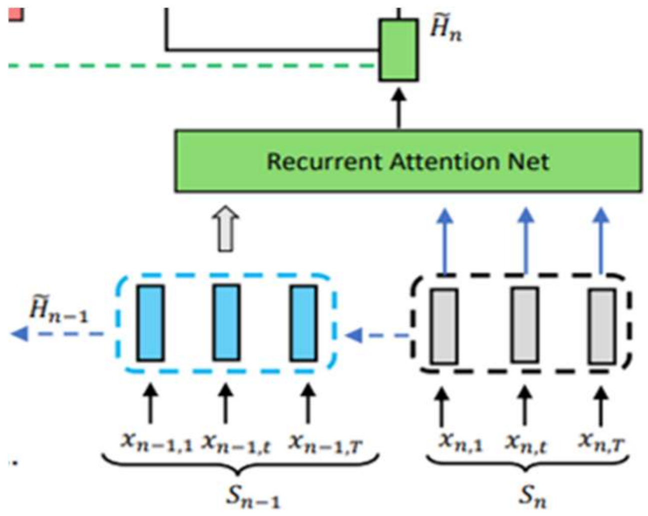


Figure 1: Illustration of DMAN for one layer. It takes a series of sequences as input and trains the model sequence by sequence. When processing the n-th sequence $\mathcal{S}_n$, the recurrent attention network is applied to extract short-term user interest by using the previous hidden state $\widetilde{\mathbf{H}}_{n-1}$ as context. Meanwhile, the long-term attention network is utilized to extract long-term interest based on the memory blocks $\mathbf{M}$. Next, the short-term and long-term interests are combined via a neural gating network for joint user modeling. Finally, the dynamic memory network updates the memory blocks via fusing the information in $\widetilde{\mathbf{H}}_{n-1}$, and the model continues to process the next sequence. The overall model is optimized by maximizing the likelihood of observed sequence, while the dynamic memory network is trained based on a local reconstruction loss $\mathcal{L}_{ae}$.

Recurrent Attention Network



Truncating accumulated behavior sequence S into a series of successive sub-sequences with fixed window size T, i.e., $S = \{S_n\}_{n=1}^{N}$. Thus, $S_n = \{\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}, \dots, \boldsymbol{x}_{n,T}\}$ denotes the n-th sequence. Let $\widetilde{\mathbf{H}}_{n-1}^{l} \in \mathbb{R}^{T\times D}$ denote the l-th layer hidden state produced for sequence S_{n-1}. S_n can be calculated as follows.

$$\widetilde{\mathbf{H}}_n^l = \mathrm{Atten}_{rec}^l(\widetilde{\boldsymbol{Q}}_n^l, \widetilde{\boldsymbol{K}}_n^l, \widetilde{\boldsymbol{V}}_n^l) = \mathrm{softmax}\left(\widetilde{\boldsymbol{Q}}_n^l \left(\widetilde{\boldsymbol{K}}_n^l\right)^T\right)\widetilde{\boldsymbol{V}}_n^l$$

$$\widetilde{\boldsymbol{Q}}_n^l = \widetilde{\mathbf{H}}_n^{l-1}\widetilde{\mathbf{W}}_Q^T, \qquad \widetilde{\mathbf{K}}_n^l = \mathbf{H}_n^{l-1}\widetilde{\mathbf{W}}_K^T, \qquad \widetilde{\boldsymbol{V}}_n^l = \mathbf{H}_n^{l-1}\widetilde{\mathbf{W}}_V^T$$

$$\mathbf{H}_n^{l-1} = \widetilde{\mathbf{H}}_n^{l-1} || \mathrm{SG}(\widetilde{\mathbf{H}}_{n-1}^{l-1})$$

The function $\mathrm{SG}(\cdot)$ stands for stop-gradient from previous hidden state $\widetilde{\mathbf{H}}_{n-1}^{l-1}$. The input of the first layer is the sequence embedding matrix $\mathbf{X}_n = \{\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}, \dots, \boldsymbol{x}_{n,T}\} \in \mathbb{R}^{T\times D}$. The final short-term interest embedding is defined as $\widetilde{\mathbf{H}}_n = \widetilde{\mathbf{H}}_n^L$.

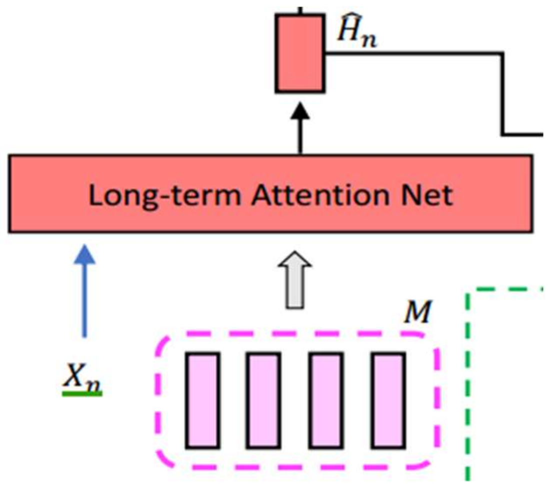
Long-term Attention Network

Define the l-th layer memory matrix $\mathbf{M}^{l} \in \mathbb{R}^{T\times D}$ to explicitly memorize a user's long-term preferences, where m is the number of memory slots. The long-term hidden state $\widehat{\mathbf{H}}_{n}^{l}$ can be estimated as follows.

$$\widehat{\mathbf{H}}_{n}^{l} = \mathrm{Atten}^{l}(\widehat{\boldsymbol{Q}}_{n}^{l}, \widehat{\boldsymbol{K}}_{n}^{l}, \widehat{\boldsymbol{V}}_{n}^{l}) = \mathrm{softmax}\left(\widehat{\boldsymbol{Q}}_{n}^{l}, \left(\widehat{\boldsymbol{K}}_{n}^{l}\right)^{T}\right)\widehat{\boldsymbol{V}}_{n}^{l}$$

$$\widehat{\boldsymbol{Q}}_{n}^{l} = \widehat{\mathbf{H}}_{n}^{l-1}\widehat{\mathbf{W}}_{Q}^{T}, \qquad \widehat{\boldsymbol{K}}_{n}^{l} = \mathbf{M}^{l-1}\widehat{\mathbf{W}}_{K}^{T}, \qquad \widehat{\boldsymbol{V}}_{n}^{l} = \mathbf{M}^{l-1}\widehat{\mathbf{W}}_{V}^{T}$$

The input of the first layer of query is $\mathbf{X}_{n}$. The final long-term interest embedding is denoted as $\widehat{\mathbf{H}}_{n} = \widehat{\mathbf{H}}_{n}^{L}$.



Dynamic Memory Network

To abstract a user's long-term interests from the past actively, the memory is updated by an abstraction function $f_{abs}: \mathbb{R}^{(m+T)\times D} \to \mathbb{R}^{m\times D}$,

$$\mathbf{M}^{\boldsymbol{l}} \leftarrow f_{abs}(\mathbf{M}^{\boldsymbol{l}}, \widetilde{\mathbf{H}}_{n-1}^{l})$$

- **Abstraction function** f_{abs}

f_{abs} is parameterized with the dynamic routing model in CapsNet for user's divers interest capture. Referring capsules from the first layer and second layer as primary capsules (input vector $\mathbf{x}_i, i \in \{1, \dots, T+m\}$) and interest capsules (output vector $\bar{\mathbf{x}}_j, j \in \{1, \dots, m\}$). The dynamic routing logit b_{ij} between primary capsule i and interest capsule j is computed by

$$b_{ij} = \bar{\mathbf{x}}_j^T \mathbf{W}_{ij} \mathbf{x}_i, \qquad \bar{\mathbf{x}}_j = \operatorname{squash}(\mathbf{s}_j) = \frac{||\mathbf{s}_j||^2}{1 + ||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||}$$

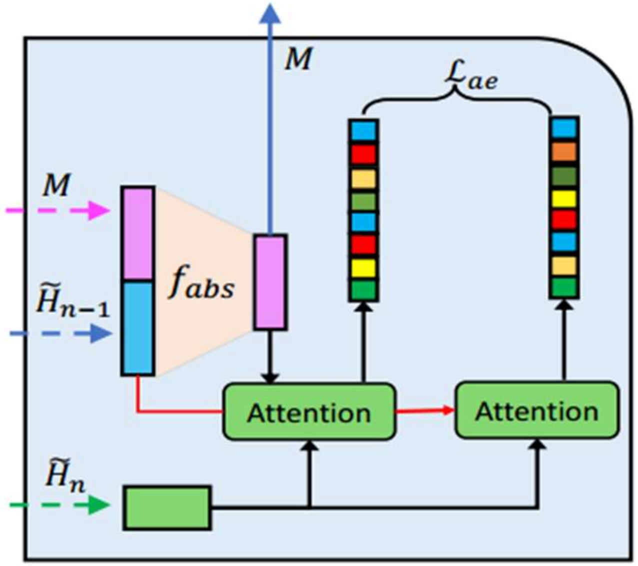
$$\mathbf{s}_j = \sum_{i=1}^{m+T} \alpha_{ij} \mathbf{W}_{ij} \mathbf{x}_i\,, \qquad \alpha_{ij} = \exp(b_{ij}) \,/ \sum_{j'=1}^{m+T} \exp(b_{ij'})$$

The routing process usually repeats three times to converge. When routing finishes, the output interest capsules of user u are then used as the memory, i.e., $\mathbf{M} = [\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_m]$.

Dynamic Memory Network

- **Attention-based reconstruction loss $\mathcal{L}_{ae}$**

To make the primary interests can be extracted by f_{abs} as much as possible, an auxiliary attention-based reconstruction loss $\mathcal{L}_{ae}$ is designed as follows.



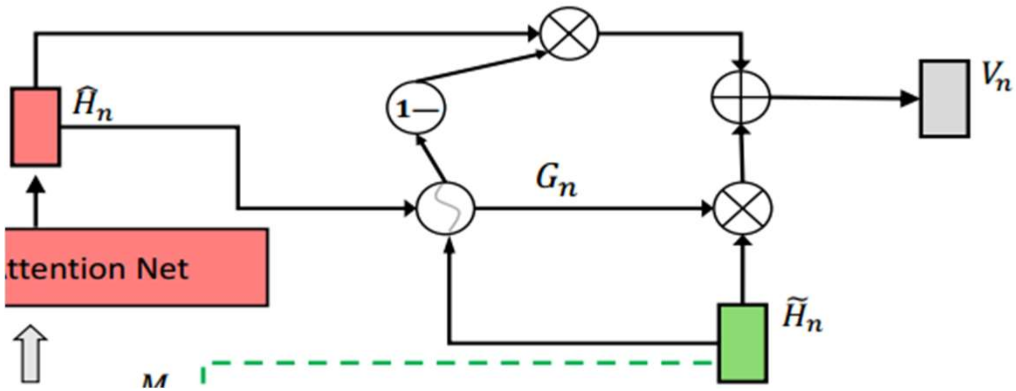
Dynamic Memory Net

$$\mathcal{L}_{ae} = \min \sum_{l=1}^{L} ||\text{atten}_{rec}^{l}(\widetilde{\boldsymbol{Q}}^{l}, \widetilde{\boldsymbol{K}}^{l}, \widetilde{\boldsymbol{V}}^{l}) - \text{atten}_{rec}^{l}(\widetilde{\boldsymbol{Q}}^{l}, \widehat{\boldsymbol{K}}^{l}, \widehat{\boldsymbol{V}}^{l})||_F^2$$

$$\widetilde{\boldsymbol{Q}}^{l} = \widetilde{\mathbf{H}}_{n}^{l}, \qquad \widetilde{\boldsymbol{K}}^{l} = \widetilde{\boldsymbol{V}}^{l} = \mathbf{M}_{pre}^{l}||\widetilde{\mathbf{H}}_{n-1}^{l}, \qquad \widehat{\boldsymbol{K}}^{l} = \widehat{\boldsymbol{V}}^{l} = \mathbf{M}_{up}^{l}$$

The recurrent attention network is reused here but keep the parameters fixed and not trainable.

Neural Gating Network



Considering that a user's future intention can be influenced by early behaviors, while short-term and long-term interests may contribute differently for next-item prediction over time, a neural gating network is applied to adaptively control the importance of the two interest embeddings.

$$\mathbf{V}_n = \mathbf{G}_n \odot \widetilde{\mathbf{H}}_n + (1 - \mathbf{G}_n) \odot \widehat{\mathbf{H}}_n$$

$$\mathbf{G}_n = \sigma(\widetilde{\mathbf{H}}_n \mathbf{W}_{short} + \widehat{\mathbf{H}}_n \mathbf{W}_{long})$$

The final user embedding $\mathbf{V}_n \in \mathbb{R}^{T \times D}$ is obtained by a feature-level weighted sum of two types of interest embeddings controlled by the gate.

Model Optimization

Given the training sample (u, t) in a sequence S_n with the user embedding vector $\mathbf{V}_{n,t}$ and target item embedding $\boldsymbol{x}_t$, the model aims to minimize the following negative likelihood

$$\mathcal{L}_{like} = -\sum_{u\in\mathcal{U}}\sum_{t\in S_n} logP\left(\boldsymbol{x}_{n,t}\middle|\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}, \dots, \boldsymbol{x}_{n,t-1}\right) = -\sum_{u\in\mathcal{U}}\sum_{t\in S_n} log\frac{\exp(\boldsymbol{x}_t^T\mathbf{V}_{n,t})}{\sum_{j\in\mathcal{V}}\exp(\boldsymbol{x}_j^T\mathbf{V}_{n,t})}$$

A negative sampling strategy is adopted to approximate the softmax function in experiments. The $\mathcal{L}_{ae}$ and $\mathcal{L}_{like}$ are separately updated in order to preserve long-term interests better. Specifically, the $\mathcal{L}_{like}$ is updated first by feeding a new sequence and then updating the abstraction function's parameters by minimizing $\mathcal{L}_{ae}$.

Experiments

Table 3: Sequential recommendation performance over three benchmarks. $*$ indicates the model only use the latest behavior sequence for training; otherwise, the whole behavior sequence. The second best results are underlined.

Models	MovieLens			Taobao			JD.com		
	HR@10	HR@50	NDCG@100	HR@50	HR@100	NDCG@100	HR@10	HR@50	NDCG@100
GRU4Rec*	17.69	43.13	16.90	10.42	14.01	4.23	27.65	38.73	23.40
Caser*	18.98	45.64	17.62	13.71	16.51	6.89	29.27	40.16	24.25
SASRec*	21.02	47.28	19.05	16.41	22.83	9.23	33.98	44.89	27.41
GRU4Rec	19.78	47.40	18.75	13.48	16.53	5.81	35.28	47.52	27.64
Caser	20.80	48.12	19.28	15.55	17.91	7.35	36.76	49.13	28.35
SASRec	22.96	50.09	20.36	20.47	24.48	9.84	38.99	52.64	31.32
SHAN	21.34	49.52	19.55	18.87	21.94	8.73	37.72	50.55	29.80
HPMN	22.84	50.54	19.77	19.98	24.37	9.66	39.14	53.22	32.24
SDM	<u>23.42</u>	<u>51.26</u>	<u>20.44</u>	<u>21.66</u>	<u>25.42</u>	<u>10.22</u>	<u>40.68</u>	<u>55.30</u>	<u>34.82</u>
DMAN	**25.18**	**53.24**	**22.03**	**24.92**	**29.37**	**11.13**	**44.58**	**58.82**	**36.93**
Improv.	7.51%	3.86%	7.77%	15.05%	15.53%	8.90%	9.58%	6.36%	6.05%

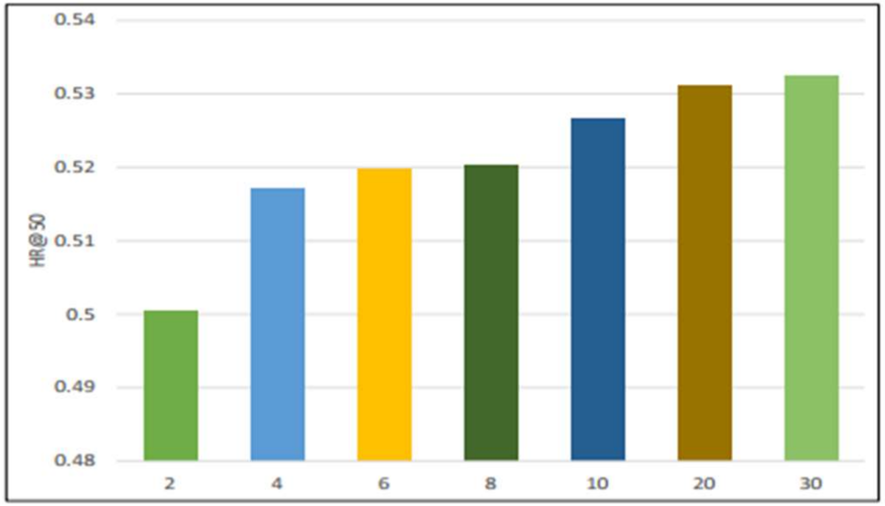
Table 4: Performance on long user behavior data XLong.

Method	Recall@200	Recall@500
GRU4Rec*	0.079	0.098
Caser*	0.084	0.105
SASRec*	0.105	0.123
GRU4Rec	0.046	0.063
Caser	0.023	0.041
SASRec	0.061	0.096
SHAN	0.091	0.115
HPMN	0.118	0.136
SDM	0.107	0.129
DMAN	**0.132**	**0.163**

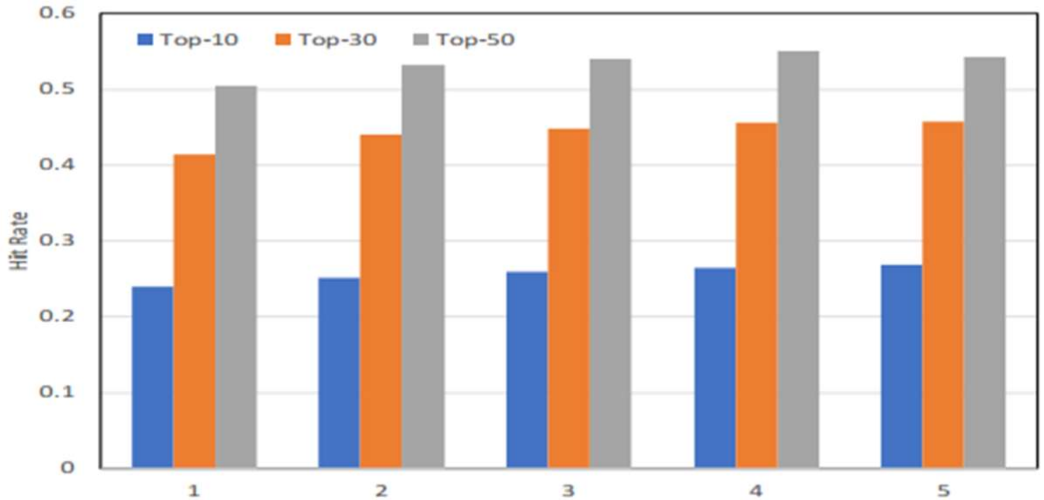
Table 5: Ablation study of DMAN.

Dataset	Method	Recall@100	NDCG@100
Taobao	**DMAN-XL**	0.237	0.094
	DMAN-FIFO	0.263	0.108
	DMAN-NRNA	0.257	0.104
	DMAN	0.293	0.111
XLong	**DMAN-XL**	0.021	0.013
	DMAN-FIFO	0.036	0.017
	DMAN-NRAN	0.043	0.019
	DMAN	0.054	0.022

Experiments



(a) Memory slots m



(b) Layer size L