CSCE 676 600: REPORT 3

Name: Rohan Chaudhury

UIN: 432001358

21st April 2022

Paper Title: Self-Attentive Sequential Recommendation

Paper Citation: Kang, Wang-Cheng and McAuley, Julian, "Self-Attentive Sequential Recommendation", in arXiv, 2018, https://doi.org/10.48550/arxiv.1808.09781

1. SUMMARY:

a. Motivation:

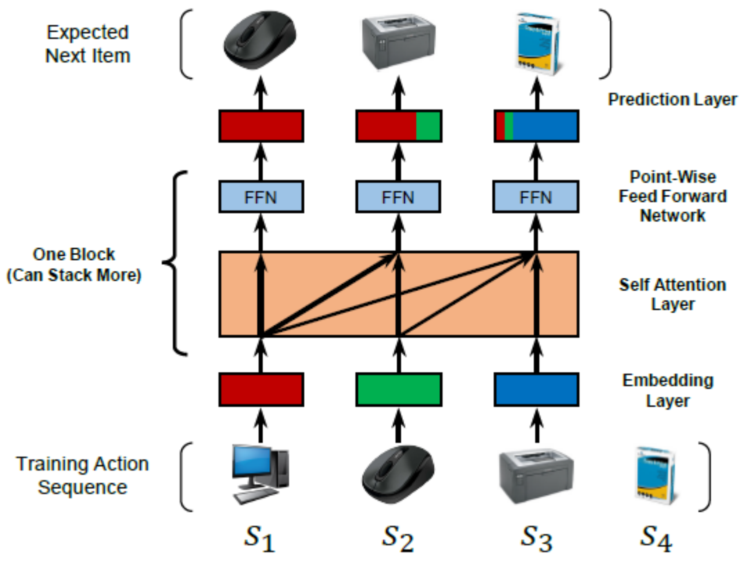
In recent years, recommender systems have in general used dynamics which are sequential in nature to identify the user's context on the basis of their most recent actions. Markov Chains (MCs) and Recurrent Neural Networks (RNNs) are two popular ways of capturing such patterns where (1) MCs presume that a user's next behavior can be anticipated based on their prior or last few actions, and (2) RNNs, in theory, can account for the identification of longer-term semantics. However, RNNs in general show better performance when the dataset available is dense in nature where it is feasible to increase the complexity of the model, while MC-based approaches show the best performance in datasets that are extremely sparse where the parsimony of the model is a crucial element. So now, the author's work aims to strike a balance between these two goals by introducing a self-attention-based sequential model (SASRec) that enables them to capture long-term semantics (like an RNN) while making predictions based on a relatively few numbers of actions (like an MC). SASRec, with the use of an attention mechanism, attempts to determine which actions from a user's activity history are 'relevant' at each time step and uses them to anticipate the next action. On both sparse and dense datasets, the strategy described in this paper beats several state-of-the-art sequential models (including MC/CNN/RNN-based methods), according to extensive empirical experiments.

b. Main technical contributions and concepts used:

An overview of the approach used by the authors:

The paper describes that the primary aim of a sequential recommender system is to combine a user's behavior characteristics based on their historic activities with their current 'context' which is based on the recent actions of the user. The task of capturing relevant information from sequential dynamics is difficult owing to the fact that the space of the input data expands in an exponential manner as we increase the number of past actions that we want to utilize as context. Markov Chains (MCs) try to get around this bottleneck by considering that the next action is a function of only the prior or previous few actions. With this methodology, MCs have been successful in modeling short-range recommendation transitions. RNNs on the other hand summarize all prior actions via a hidden state, which is then used to forecast the following actions. As evident, both MC and RNNs have their own pros and cons. MCs function well in sparse situations since they make strong simplification assumptions, but they may miss the nuanced dynamics of more complicated situations. RNNs, on the other hand, while powerful, require a considerable quantity of dense data to surpass even simpler baselines.

In order to solve the issues in both MCs and RNNs, the authors use the ‘self-attention’ mechanism which is the working principle of the sequential model called ‘Transformers’. The ‘self-attention’ mechanism can detect semantic and syntactic patterns between the words in sentences very efficiently. They intended to design their model to be able to do two things: 1) be able to draw context from all past behavior, 2) should be able to predict based on a limited amount of important actions, and they named it SASRec (Self-Attention based Sequential Recommendation) model based on its inherent properties. Below is a simplified illustration depicting SASRec's training process. As we can see, the model evaluates all prior things at each time step and utilizes the attention mechanism to 'focus on' items important to the next action by assigning weights to preceding items adaptively at each time step.



A brief overview of the Attention Mechanism:

The basic premise of attention mechanisms is that consecutive outputs are all dependent on 'relevant' elements of some input that the model should focus on in order. These mechanisms have been demonstrated to be useful in a variety of tasks, including picture captioning and machine translation. Attention-based approaches are frequently more interpretable, which is an added benefit. Transformers is a sequence-to-sequence method based solely on attention that had produced state-of-the-art performance and efficiency on machine translation tasks formerly dominated by RNN/CNN-based methods. To capture complicated patterns in sentences and retrieve important words for creating the next word, the Transformer model significantly depends on the 'self-attention' modules. The scaled dot product attention is given as:

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

Where,

Q refers to queries,

K refers to keys, and

V refers to values and the items are represented by the individual row

Basically, the attention layer computes a weighted total of all values, in which the weight for a given query I and the value j corresponds to the connection between query I and key j. The scaling factor $\sqrt{d}$ is used to prevent the inner product from having excessively large values, notably when the dimensionality is large.

A brief overview of their methodology:

In essence, the problem statement is the prediction of a user's next action based on their previous action sequence $\mathcal{S}^u = (\mathcal{S}_1^u, \mathcal{S}_2^u, \ldots, \mathcal{S}_{|\mathcal{S}^u|}^u)$ which is known in a sequential recommendation setting. Following are the blocks that the authors utilized to build this sequential recommendation model:

1. *Embedding Layer:*

The authors convert the training action sequence into a fixed sequence of length n, $s = (s_1, s_2, \ldots, s_n)$, (by either truncating the incoming sequence or padding it with zeros to the left as per requirement) where n is the maximum capacity that their model can currently handle. An item embedding matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{I}| \times d}$ and input embedding matrix $\mathbf{E} \in \mathbb{R}^{n \times d}$ were then created, where, d is the latent dimensionality and $\mathbf{E}_i = \mathbf{M}_{s_i}$. A learnable position embedding $\mathbf{P} \in \mathbb{R}^{n \times d}$ is added to the input embedding to make the self-attention model consider the previous item's positions.

2. *Self-Attention Block*

Attention (Q, K, V), as defined in the previous section, is the attention block here. Using linear projections, the self-attention operation converts the input embedding $\widehat{\mathbf{E}}$ into 3 matrices which are then fed into an attention layer.

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

The projection matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d}$ and they increase the flexibility of the model.

Since the self-attention model is linear, the authors have applied a point-wise two-layer feed-forward network to all shared parameters identically to equip the model with nonlinearity and to consider interactions between distinct latent dimensions.

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

Where, $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}$ are $d \times d$ matrices and $\mathbf{b}^{(1)}, \mathbf{b}^{(2)}$ are d-dimensional vectors.

3. *Stacking Self-Attention Blocks*

For the purpose of learning more complicated item transactions, the authors stack the self-attention blocks and the b-th block is defined as:

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

To ease the problems of overfitting, vanishing gradients, and increased training time with an increase in the depth of the network, the authors have used the following operation as defined in [2].

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

Where g(x) is the representation of the feed-forward network or the self-attention layer. That is, the authors perform layer normalization (which accelerates and stabilizes the training of the neural network) to the input x before sending it into g, apply dropout (solves the problem of overfitting) to g's output, then add the input x to the final outcome for layer g in each block.

The authors have also used residual connections to propagate potentially useful low-layer features to higher layers.

4. *Prediction Layer*

The authors use $\mathbf{F}_t^{(b)}$ to forecast the next item (given the first t items) after b self-attention blocks that flexibly and hierarchically harvest information from prior consumed items. In order to anticipate the significance of item i the authors use an MF layer:

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

where N is an item embedding matrix and $r_{i,t}$ is the relevance of item i being the next item given the prior t items. As a result, a high interaction score $r_{i,t}$ indicates a high level of relevance, and the authors derived suggestions by rating these scores. To reduce overfitting and model size the authors used shared item embeddings.

5. *Network training*

The authors had described the expected output at time step t as follows:

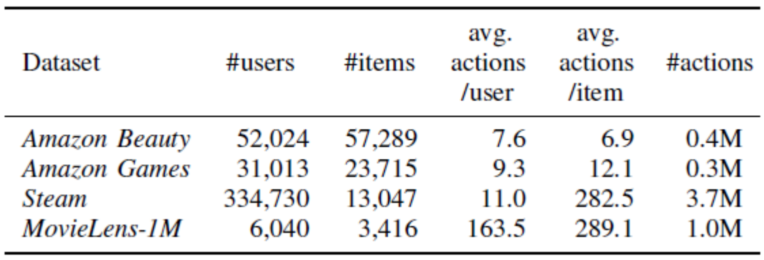
$$o_t = \begin{cases} \text{<pad>} & \text{if } s_t \text{ is a padding item} \\ s_{t+1} & 1 \le t < n \\ \mathcal{S}^u_{|\mathcal{S}^u|} & t = n \end{cases},$$

Where the padding item is indicated by <pad>. Their model takes sequence s described above as input, the sequence corresponding to this input as expected output o. As the objective function, the authors utilized the binary cross-entropy loss defined below and used Adam Optimizer to optimize the network.

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

c. Interesting experimental result/theoretical finding:

The authors evaluated their models on four different datasets. The data statistics after preprocessing are shown below:



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

The authors include three groups of recommendation baselines to demonstrate the efficacy of their approach.

A. *Group 1:*

This category comprises generic recommendation systems that rely only on user input and ignore the order in which actions are performed (such as **PopRec** and **Bayesian Personalized Ranking(BPR)**).

B. *Group 2:*

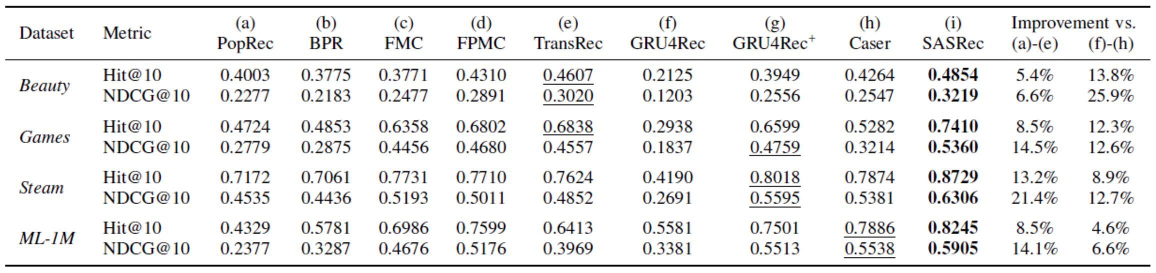
This category includes sequential recommendation algorithms based on first-order Markov chains that take into account the most recently visited item (such as **Factorized Markov Chains (FMC), Factorized Personalized Markov Chains (FPMC), Translation-based Recommendation (TransRec)**)

C. *Group 3:*

This category includes sequential recommender systems based on deep learning that take into account numerous (or all) prior items (such as **GRU4Rec, GRU4Rec$^+$, Convolutional Sequence Embeddings (Caser)**).

The performance of all approaches on the four datasets is presented in the table below.

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	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%

The following can be observed from the table above:

i. On both sparse and dense datasets, the proposed SASRec approach beats all baselines, gaining a 6.9% Hit Rate and 9.6% NDCG increases (on average) over the strongest baseline. One possible explanation is that their model might be attending items in various ranges on different datasets (for example, only the preceding item on sparse datasets and more on dense datasets).

ii. Non-neural techniques (i.e., (a)-(e)) perform better on sparse datasets, whereas neural approaches (i.e., (f)-(h)) score higher on denser datasets when considering the second-best performing methods across all datasets. This is due to the fact that neural techniques have more parameters to capture high-order transitions (but easily overfit), whereas well built but simpler models perform better in high-sparsity environments.

The authors also observed that having a higher number of latent dimensions enhances the performance of their model. With d larger than or equal to 40, their model performs satisfactorily across all datasets.

2. MY THOUGHTS:

a. Pros and challenges they address:

i. The authors have used a very novel method to address the issues of sequential recommendation by using the self-attention mechanism earlier used in Transformers which allows them to draw 'context' from all prior actions and do the final prediction based on only a few important actions.

ii. Due to the self-attention mechanism, SASRec tends to evaluate long-range relationships on dense data whereas concentrating on more recent events on sparse datasets which proves to be critical for adaptively managing datasets of variable density.

iii. The computations of the self-attention block of SASRec can be parallelized which makes it faster than the other RNN/CNN options.

iv. The ablation study done by the authors in the paper was very effective in showing how different components of the model affect its behavior. Visualization of the attention weights in the paper is very helpful in giving the readers of the paper an intuitive look into the model's characteristics.

v. The addition of residual connections enables the model to propagate useful low-layer features to higher layers. This concept is especially important in recommendation systems in cases such as it helps in the consideration of the elevated importance of the very last item visited by the user in the model and allows this useful feature to be propagated through all the stacked self-attention layers.

b. Limitations:

i. The author is not considering the explicit user embeddings and relies entirely on the implicit user embeddings for the recommendation task.

ii. The authors are taking only up to a fixed length of the sequence n (n most recent actions) in the first embedding layer as that is the maximum length that their model can handle. If the value of n is less then it might adversely affect the model's performance. Their approach cannot be scaled to handle sequences that are very long.

iii. A fixed position embedding embedded into the input embedding gave worse results in their model. Fixed position embeddings can have the advantage of faster training of the proposed models. Some minor changes can be introduced to make the model work better with fixed position embeddings.

iv. During preprocessing of the data, the authors are discarding the users and items with less than five related actions. This action is not desirable as certain discarded actions can be very useful.

3. FOLLOW UP/NEXT STEPS:

a. Future work can be taken up by considering explicit user embeddings together with the self-attention mechanism.
b. Research can be undertaken to find efficient ways to increase the value of **n** (number of recent actions) in the first embedding layer (to consider longer sequences) as a large value of **n** will mean that the model is able to take more prior actions into consideration while predicting. One direction can be to split the longer sequences into short segments and then use them for training. Another could be to use restricted self-attention, which will only consider recent acts rather than all actions, and distant actions can be evaluated in higher layers.
c. Research can be undertaken to improve the performance of the model while using fixed position embeddings as done in [2] in the input embedding.
d. The model may be improved by including extensive context information (e.g., dwell time, locations, and so on), as well as looking into ways to manage extremely lengthy sequences (e.g. clicks).

4. REFERENCES:

[1] Kang, Wang-Cheng and McAuley, Julian, “Self-Attentive Sequential Recommendation”, in arXiv, 2018, https://doi.org/10.48550/arxiv.1808.09781
[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in NIPS, 2017.