Session-Based Recommendation with Self-Attention

Pham Hoang Anh
Posts and Telecommunications Institute of Technology
hoanganh.pham.12327@gmail.com

Ngo Xuan Bach
Posts and Telecommunications Institute of Technology
bachnx@ptit.edu.vn

Tu Minh Phuong
Posts and Telecommunications Institute of Technology
phuongtm@ptit.edu.vn

ABSTRACT

The goal of session-based recommendation is to predict the next action of a user based on the current session and anonymous sessions before. Recent works on session-based recommendation usually use neural network architectures such as convolution neural networks (CNNs) or recurrent neural networks (RNNs) to extract patterns of sessions. Such features have been shown to give promising results because they can discover the user's sequential behavior and understand the purpose of current session. In this paper, we propose a neural network architecture for session-based recommendation without using convolution or recurrent neural networks. Our model is inspired by the Transformer's design, in which the information of important items is passed directly to the hidden states. Experimental results on two real-world datasets show that our method outperforms several state-of-the-art models.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Session-based recommendation, Sequential behavior, Transformer, Attention mechanism

ACM Reference Format:
Pham Hoang Anh, Ngo Xuan Bach, and Tu Minh Phuong. 2019. Session-Based Recommendation with Self-Attention. In *The Tenth International Symposium on Information and Communication Technology (SoICT 2019), December 4–6, 2019, Hanoi - Ha Long Bay, Viet Nam.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3368926.3369682

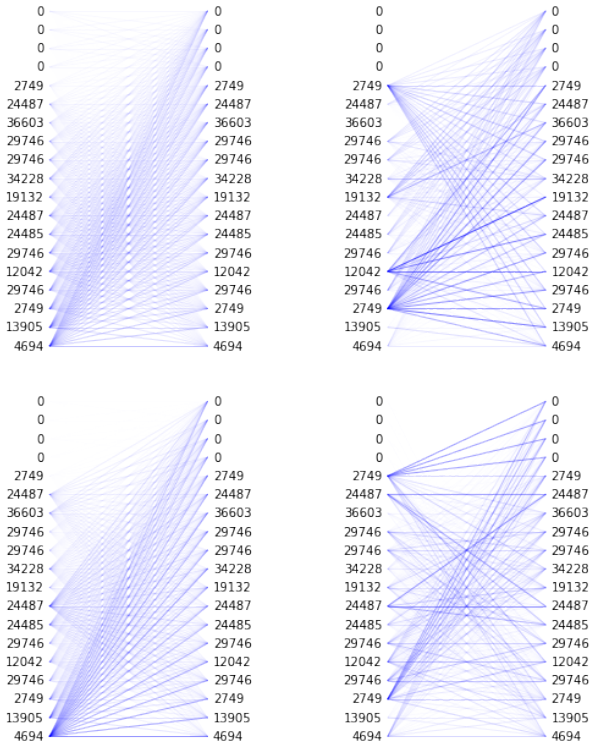


Figure 1: Transformer evaluates the importance of items in sessions to focus on at every head. We visualize the weights of each hidden state from a random layer to its next layer. The darker line shows the more important connection.

1 INTRODUCTION

Nowadays, with the explosive growth of the World Wide Web, the amount of information on the Internet has been increasing rapidly. The bustle of user-item systems improves the satisfaction of users because such systems provide everything they need. When a system becomes enormous, however, it prevents user from approaching new information. For example, the statistics published by Youtube in early 2019 show that about 300 hours of videos are uploaded on their system every minute. So, if a new MV (Music Video) of your favorite singer is uploaded on Youtube, how can you figure it out? The same problem can appear in the Amazon e-commerce system. Because there are millions of items on Amazon sites, users may need to spend their whole day only for searching the things they want. A good recommender system can remove all of the inconvenience and bring the comfort to users.

In the field of recommender systems, session-based recommendation methods recently have attracted much attention from researchers in comparison with the traditional ones like content-based or collaborative filtering [4, 5, 16, 18]. Despite of using only the information bout sessions and do not use other information about users or items, session-based recommendation systems can achieve promising results compared to other methods.

The session-based recommendation task shares similar properties with the text sequence's tasks in natural language processing which have been solved very successfully by the Transformer [17].

SoICT 2019, December 4–6, 2019, Hanoi - Ha Long Bay, Viet Nam

ACM ISBN 978-1-4503-7245-9/19/12…$15.00
https://doi.org/10.1145/3368926.3369682

Transformer is a specially designed architecture for sequence problems without any convolution or recurrent layers. The model based totally on attention with multi-headed and self-attention mechanisms. Following Vaswani et al. 2017 [17], Transformer can be trained significantly faster than architectures based on recurrent or convolution layers. It also achieved new state-of-the-art in the WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks.

There are some reasons for the high performance of the Transformer. First, with self-attention and multi-headed mechanisms, this method is able to find the most important items which have the biggest effect on the prediction's result. In each layer, a hidden state is generated by considering all the hidden states from the previous layer. This will helps hidden states containing more information about the context of all sessions. The self-attention mechanism modifies the effect of each hidden-state on the next hidden-states layers and the multi-headed mechanism allows Transformer to have the ability to focus on different positions of the session. Second, self-attention layers have the minimum path length and minimum number of sequential operators compared to convolution and recurrent layers. Because of this advantage, the information lost through each layer is significantly reduced especially on long sequences. Figures 1 illustrates specifically how items be focused on differently through each layers.

In this paper, we propose a neural network architecture for session-based recommendation which utilizes the Transformer's ability on sequential problems. In particular, we adapt the Transformer to session-based recommendation by adding or modifying several elements of the Transformer. We also investigate the model that combines two Transformer components which are trained separately and each of them has its own responsibility. The first component extracts the features of users' sequential behavior and the other one captures the main purpose of the current session. Experimental results on two real world datasets show that our method outperforms all state-of-the-art baselines with recall and the MRR metric.

2 RELATED WORK

In this section, we review existing session-based recommendation approaches and give a brief introduction of the Transformer, the architecture we employ in our model.

2.1 Conventional methods

Mining Sequential Patterns Agrawal et al. 1995 [1] present an early study on recommendation system based on sequence's analytic. They try to look merely at all the sessions to find the longest patterns which are supported by the minimum number of users. After that, some statistical algorithms are used to find out which items are usually bought together. Although the method can give us a visual look, its performance is not good because the behaviors of users are not always similar.

Item-to-Item Collaborative Filtering is the later approach to solve this problem. Linden et al. [8] focus on finding the most similar items to the current item while Sarwar et al. [13] use many item-based and k-nearest neighbor algorithms.

Sequential recommendation based on Markov chains predicts the next action of user based on the previous ones. Shani et al. [14] introduce Markov decision processes (MDPs) to recommend the next item through computation of the transition probabilities between items. Yap et al. [7] employ a new competence score measure in session pattern mining. Chen et al. [2] propose logistic Markov embedding and model the sequences as Markov chains to learn the representations of items.

2.2 Deep Neural Network Based Methods

Deep neural networks have been used successfully in various fields of computer science such as computer vision, natural language processing, and recommender systems. A lot of deep neural network architectures have been proposed to solve the session-based recommendation task, including recurrent neural networks, convolutional neural networks and their variations.

Recurrent Neural Networks. Hidasi et al. [4] apply recurrent neural networks with GRUs to extract the features of sessions. They also suggest session-parallel mini-batch training for solving the variable length of different sessions. Tan et al. [5] improve the performance of RNNs by using the new data augmentation techniques and training by sequences separately. Li at el. 2017 [6] introduce neural attentive recommendation machine (NARM) which uses RNN based encoder-decoder architecture. RNN models are also modified to incorporate context and user information [10, 11].

Convolutional Neural Networks. Tuan and Phuong [16] propose the first CNN-based model for session-based recommendation, in which 3D CNN is used to model temporal patterns. Tang et al. 2018 [15] introduce CASER network which is a convolution neural network for the same purpose. Yuan et al. 2018 [19] propose convolutional generative networks (Nextitnet) to address the problem of variable length and small filter of CASER.

Graph Neural Networks Graph models have been widely used for conventional recommender systems [9] and graph based neural networks have been used for session-based recommendation in particular. An example is the work of Wu et al. 2019 [18], which takes the advantages of graph neural networks (GNNs) to achieve promising results on session-based recommendation.

2.3 The Transformer

Recently, complex networks with encoders and decoders have been popularly used to solve sequential problems on natural language processing due to their effectiveness. Encoder-decoder architectures are usually based on CNN or RNN layers. The combination of them with an attention mechanism has shown potential results on many tasks. The Transformer, proposed by Vaswani et al. 2017 [17], is a simple network architecture based merely on attention mechanisms without CNN or RNN layers. Experimental results of Transformer show that it requires less time for training and produces much higher results than encoder-decoder models with CNNs or RNNs.

3 PROPOSED METHODS

This section describes the session-based recommendation task and explains our proposed attention architecture to deal with the task.

3.1 Session-based Recommendation

Session-based recommendation systems aim to predict the next action of a user given the current sequence of actions that the user has performed. In this work, an action corresponds to a click to an item. Let $x = \{x_0, x_1, \ldots, x_{t-1}, x_t\}$ be the current user's session where x_t denote an action and $y = \{y_0, y_1, \ldots, y_{n-1}, y_n\}$ be the set of all items. Our goal is to build a model to predict the next action with any given current sequence x. The recommendation system first generates the prediction's result as a ranking list (or a scoring list) of all the candidate items y. The top-k items ($1 \leq k \leq n$) in y with the highest scores are then recommended for the user.

3.2 Model Overview

Our model is inspired by the Transformer (Vaswani et al. 2017 [17]). The main idea is finding the suitable weights to help the decoder to focus on appropriate items in the input sequence.

First, the encoder of our model receives the input session $x = [x_1, x_2, \ldots, x_{t-1}, x_t]$ and converts it into a set of d dimensional item representations $e = [e_1, e_2, \ldots, e_{t-1}, e_t]$. After that, item representations are fed to the transformer's decoder using the self-attention mechanism. The transformer's decoder builds and maintains the next hidden states $h = [h_1, h_2, \ldots, h_{t-1}, h_t]$ to allow the network to incorporate the representations of the previous items into the one we are currently processing. The session feature representation c_t is created from hidden states which are the output of the last layer of two transformer's decoders. Finally, c_t is transformed and go through linear layer which followed by a softmax layer to infer the ranking list of the next click for all items.

One of the most strong points of the self-attention mechanism in Transformer compared to other memorized layers such as recurrent or convolution networks is the path length between two furthest states when we need to process a long sequence. The shorter paths between dependencies solve the forget information problems and make our network easier to learn.

Transformer has been shown to be effective for many sequence tasks in natural language processing, including machine translation and language modeling. Naturally, session-based recommendation is also a sequence task. So, we try to adapt the Transformer to session-based recommendation. We focus on customizing the mask layer and following the NARM (Tan et al. 2017 [6]). We create a model that captures both the user's sequential features and the main purpose of user in this session. In the following, we first describe the mask layer's impact to our method. We then introduce our Transformer's decoders for extracting the features of sequences (Global decoder) and for finding the main objective of the user (Local decoder). Our implementation of Transformer is shown in Figure 2. Finally, we illustrate our model which is the combination of all these parts. The overview of our proposed method is drawn in Figure 4

3.3 Masking Layer

Recently, the transformer has been used successfully in a variety of natural language processing tasks and many of them optimize the transformer as a deep bidirectional model. Following the experimental results of BERT (Devlin et at. 2019 [3]), models with the bidirectional architecture outperform the same methods with

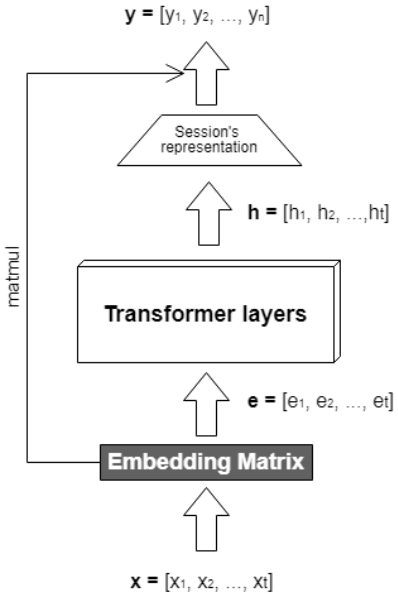


Figure 2: Dataflow of The Transformer applied for session-based recommendation

left-to-right or right-to-left architecture. For the session-based recommendation problem, however, we realize that the calculation's result in earlier hidden-states should not be affected by the later hidden states. For example, with $h = [h_1, h_2, \ldots, h_{t-1}, h_t]$ is the hidden states of the previous layer, transformer will produce the next hidden states $h' = [h'_1, h'_2, \ldots, h'_{t-1}, h'_t]$. In this situation, h'_2 should not be affected by $h_3, h_4, \ldots, h_t$. The reason is information held in h'_2 will be used for predicting h_3 - the next action sequence. If h_3 is joined, the model will focus on h_3. So when we need to predict h_{t+1}, the model will perform badly because we do not have h'_{t+1}.

That's why we need the masking layer to prevent the future seeing mechanism of the model. Masking layer is only allowed to attend to earlier positions in the output sequence. So our model will be forced to learn harder to extract features from the previous hidden states.

3.4 Global Transformer's Decoder

The purpose of global transformer's decoder is to extract the features of a user's sequential behavior in the current session given the click sequence of that user. Both input click sequences and extracted features are sequences of d dimensional vectors. In each block, we keep the original architecture of the transformer, which consists of a masking layer, a self-attention layer, and a feed-forward layer. To predict the next item in the sequence, we only use the last hidden state h_t. The architecture is visualized in Figure 3 which also exactly the same with local decoder's architecture.

$$c_t^g = h_t$$

Global decoder only capture the information of the session. It cannot capture the main purpose of the user in the session. Therefore, we need another decoder to achieve this goal.

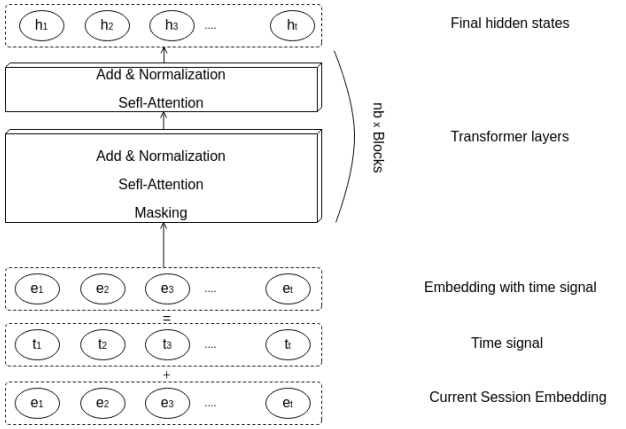


Figure 3: The same architecture of both Global and Local Decoders.

3.5 Local's Transformer Decoder

The architecture and mechanism of the local transformer's decoder is quite the same as the ones of the global decoder. The only difference is that the local one uses all the final hidden states while the global one uses the last hidden state. This helps the local decoder to focus on the main purpose of the current session instead of extracting features from the whole sequence.

$$c_t^l = \sum_{j=0}^{t} \alpha_{tj}.h_j$$

where α_{tj} are important weights. Because all final hidden states appear in the local decoder, we use the average of them.

$$\alpha_{tj} = \frac{1}{t}$$

In another version, α_{tj} can also be computed as the similarity between h_j and the last hidden state h_t

$$\alpha_{tj} = q(h_t, h_j)$$
$$q = v^T sigmoid(A_1 h_t + A_2 h_j)$$

where A_1 and A_2 are used as the weight matrices for the similarity computation.

3.6 Transformers for Session-based recommendation system

In our model, the global Transformer's decoder summarizes the whole session behavior of user into c^g, while the local decoder focuses on the features of some important items to understand the main purpose of the current session and brings them to c^l.

Each vector will have its different role in the session-based recommendation task. In order to have the natural combination, we use both of these by concatenating 2 vectors to c_t.

$$c_t = [c_t^g; c_t^l] = [h_t^g; \sum_{j=0}^{t} \alpha_{tj}.h_j]$$

With this concatenation, we can take the advantages of both decoders. One extract features of the whole sequential and the other will compute from all the previous hidden states. So user's sequential behavior and the main purpose of session can be represented by c_t which is the result of a concatenation between c_t^g and c_t^l.

As we illustrated in the figure, we introduce a special linear method to improve the accuracy of our model. Following the Improving Language Understanding by Generative Pre-Training (Radford at el. [12]), the output of Transformer blocks will be matrix-multiplied with the embedding matrix transformed to create the scoring matrix. This method aims to use a shared-weight of embedding matrix between the first layer and the last layer which improves the results of our model.

$$P(u) = softmax(h_t.E)$$

where h_t is a $|1| \times |d|$ matrix, E is a $|n| \times |d|$, n denotes the number of items, and d is the dimension of embeddings. But our situation has a little different so we can not implement like them directly. Our computation of the scoring matrix is written below

$$S = c_t.(B.E^T)$$

Where B is a $|d| \times |2d|$ matrix which we create to adapt with our problem. Our c_t vector is the result of a concatenation between h_t^g and h_t^l which have them same dimension is $|1| \times |d|$, so our c_t's dimension is $|1| \times |2d|$. So we compute it indirect through matrix B. Then S is entered the $softmax$ layer to represent the probability of items which will occur next. Top-k highest probability is chosen to suggest for user of that session.

To fit with the attention mechanism, we do not train our model in parallel sessions like the GRU4Rec. We instead optimize the model on each sequence $[x_1, x_2, \ldots, x_t]$ with label $[x_{t+1}]$ using mini-batch with the cross-entropy loss.

4 EXPERIMENTS

This section describes the datasets, the state-of-the-art methods for session-based recommendation, and the evaluation metrics that we use. Experimental results are also presented and discussed.

4.1 Datasets

We used two famous real-world datasets for sequential recommendation - YOOCHOOSE and DIGINETICA.

- YOOCHOOSE[1]: a public dataset from RecSys Challenge 2015. This dataset about click-streams of users at a real e-commerce site. To reduce noise, we deleted all sessions of length 1 and items that appear less than 5 times. After preprocessing, we had 7981580 sessions and 37483 items.
- DIGINETICA[2]: the dataset for CIKM Cup 2016. We only used the released transaction data and removed all the rest. Like the previous dataset, we also deleted sessions of length 1 and items that appear less than 5 times. Finally, we had 204771 sessions and 43097 items.

To conduct experiments, we split each dataset into training set and test set. For YOOCHOOSE, we only used the sessions of subsequent day for testing and filtered out clicks from the test set where the clicked items did not appear in the training set. For DIGINETICA, the test sessions were the sessions of subsequent week. For comparison purpose, we only used the users's click-stream data (*session_id* and *item_clicked_id*).

Tan et al. [5] claim that the recommendation models need to adapt for changing user behavior over time and training on the

[1] http://2015.recsyschallenge.com/challenge.html
[2] http://cikm2016.cs.iupui.edu/cikm-cup

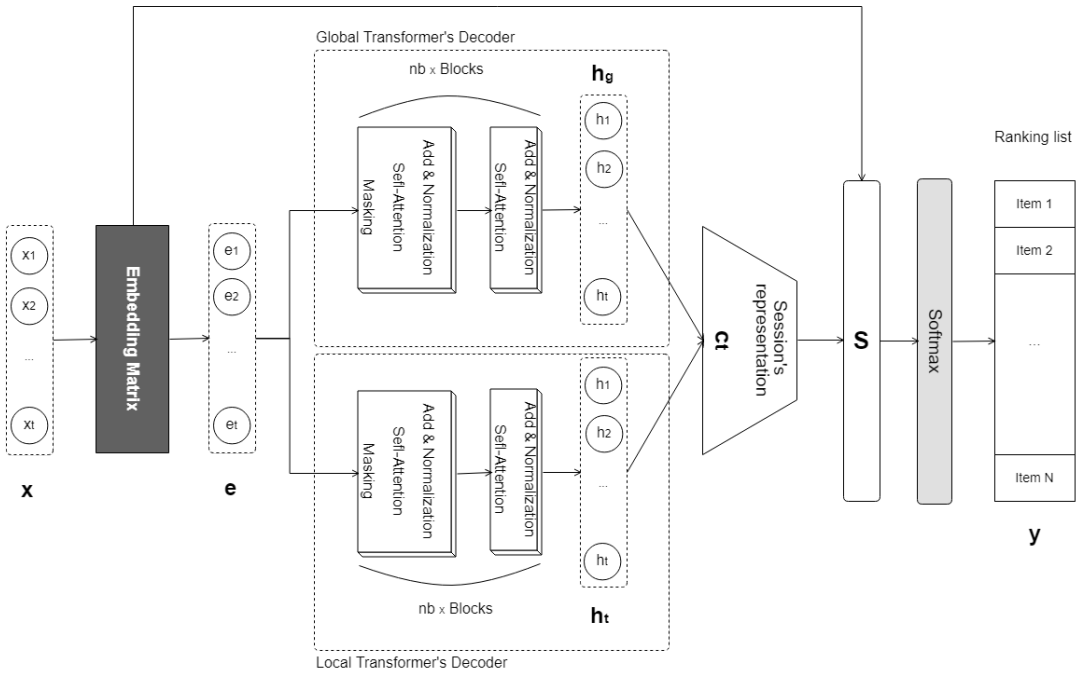


Figure 4: Transformer evaluates the importance of items in sessions to focus on. We visualize the weights of each hidden state from a random layer to its next layer. The darker line shows the more important connection.

entire large dataset like YOOCHOOSE may get worse results than training on more recent parts of the datasets. We, therefore, sorted the training sequences of YOOCHOOSE by time and only trained the model on more recent fractions 1/64 and 1/32 training sequences (denoted by 1/64 YOOCHOOSE and 1/32 YOOCHOOSE, respectively)

To generate training data, we followed the method of Tan et al. [5] and Wu et al. [18]). For each input session $x = [x_1, x_2, \ldots, x_{n-1}, x_n]$, we created a series of prefixes sequences and correspond labels $[([x_1], x_2), ([x_1, x_2], x_3), \ldots, ([x_1, x_2, \ldots, x_{n-1}], x_n)]$, where $[x_1, x_2, \ldots, x_{t-1}]$ is the generated sequence and x_t represents for the next-clicked item, i.e. the label of that prefix sequence. That procedure was applied for creating training data of both YOOCHOOSE and DIGINETICA datasets. The statistic of two datasets is shown in Table 1.

4.2 Other state-of-the-art methods

We compared our model with four state-of-the-art models summarized in the below:

- **Improved GRU4REC** Tan et al. [5] The method use a simple Recurrent Neural Network for this task. Improved GRU4REC contributes an important technical for augmentation the training data and a method to account for shifts in the input session distribution.
- **Nextitnet** Yuan et al. 2018 [19] - A convolutional generative network for next item recommendation. The advantage of this model is the capability of learning high-level representations from both short- and long-range item dependencies
- **NARM** Li et al. 2017a [6] - A neural attentive session-based recommendation model that employs RNNs with an attention mechanism to emphasize the user's main actions and sequential behavior.
- **SR-GNN** Wu et al. 2019 [18] - A session-based recommendation with graph neural networks. Authors introduce a graph-based architecture to obtain accurate item embeddings and take the complex transitions of items into account.

For comparison purpose, we used the codes provided by previous works and ran them on the same training datasets which had been preprocessed before. All the hyper-parameters were set to default values.

4.3 Evaluation Metrics

We used Recall@20 and MRR@20 to evaluate the performance of recommendation methods.

- *Recall@20*: The proportion of cases when the desired items were presented in the top-20 items of all test cases. This metric can be called by hit rate (HR@20) in some documents.
- *MRR@20*: The average of reciprocal ranks of the desired items. The reciprocal rank was set to zero if the rank was larger than 20. MRR takes the rank of the item into account, which is important in settings where the order of recommendations matters.

4.4 Parameter Setup

Our method used 64-dimensional embeddings for the items. For optimization method, we employed Adam optimizer with a learning

Table 1: Statistics of the datasets used in our experiments

Statistic	1/32 YOOCHOOSE	1/64 YOOCHOOSE	DIGINETICA
Numbers of clicks	4336775	557248	982961
Number of training sessions	739718	369859	719470
Number of test sessions	55898	55898	60858
Number of items	20124	16766	43097
Average session's length sessions	5.86	6.16	5.12

rate of 0.001. We used 3 blocks for both global and local decoders with 8-head parallel layers of attention. The inner-layer of each feed forward network has dimensionality dff = 512. Training and testing mini-batch sizes were set to 512. The dropout rate was set to 0.25 for the layer between item embeddings and the first attention layer, and 0.2 for each feed forward network. All sessions were padded zeros or truncated at the earlier items to get fixed 19 time-steps. This was inspired by the settings in many state-of-the-art methods mentioned before.

We trained our model on Google Colab - a free cloud service based on Jupyter Notebooks that supports free GPU by Google. At the time we trained, a GPU supported for all users is Tesla K80. With our experimental setup, model will converge after about 50 epoches for both datasets.

4.5 Comparison between the architectures of Transformer's decoder

First, we investigated our method with different local decoder architectures. We compared experimental results of different variants: 1) model without local decoder; 2) model with local decoder and the local representation vector was the average of all the last hidden states; and 3) model with local decoder and used an additional attention layer for all last hidden states. Experimental results are summarized in Table 2

We can see that the model used local decoder outperformed the model without local decoder on DIGINETICA dataset. The results with DIGINETICA dataset was better 3.6% at Recall@20 metric and 2.4% at MRR@20 metric. The results were still slightly better on 1/64 YOOCHOOSE. So we believe that local decoder had an important role in our method.

Moreover, with the attention weights used additional on local decoder, the results also improved on the 2 datasets. This is understandable because the attention weights enjoys the advantages of adaptively focusing on more value actions to understand the user's main purpose in the current session. We compared experimental results of two models, with and without the masking layer, to prove our assumption before. The experimental results are shown in the Table 3. Using masking layer made a big improvement on both datasets.

4.6 Comparison with other state-of-the-art models

Next, we compared our model with several state-of-the-art models, which were trained and evaluated on the same dataset. The performances of these models are shown in Table 4.

Our model outperformed all these models in term of Recall@20 and MRR@20. Especially on the DIGINETICA datasets, Recall@20 and MRR@20 were improved 2% and 1.5%, respectively, compared with the best results before. The results once again indicated that the attention-based method is more suitable for the sequences task than RNN-based or CNN-based models.

4.7 Model's Analysis

In this section, we take a closer look at our model by extracting information about weights between layers and between items.

Transformer layers's observation. Firstly, we investigate Transformer heads to probe what parts of sequences they have attracted to. We find that there are some common patterns such as focusing on broadly over the whole sessions, skipping the zero padding items, getting information at regular distance points or attending to some cross position previous hidden states only. The visualization of our results is shown in Figures 5 and 6.

These visualization's results are from the weights of head-layer of one test session in DIGINETICA dataset. Because we applied the Masking layer to our network, one hidden state in the next layer could only use the information from the hidden states of the previous layer which had the smaller or equal position to its.

Attention weights'visualization Next we explore the attention weights of the local decoder, which has an important role as mentioned in the previous part. We realize that these attention weights always tried to skip the zeros padding of sequence. This is the great signal which shows that our model learning's process is good. Moreover, some last positions of sequence usually got the more important weights, especially with long sequences.

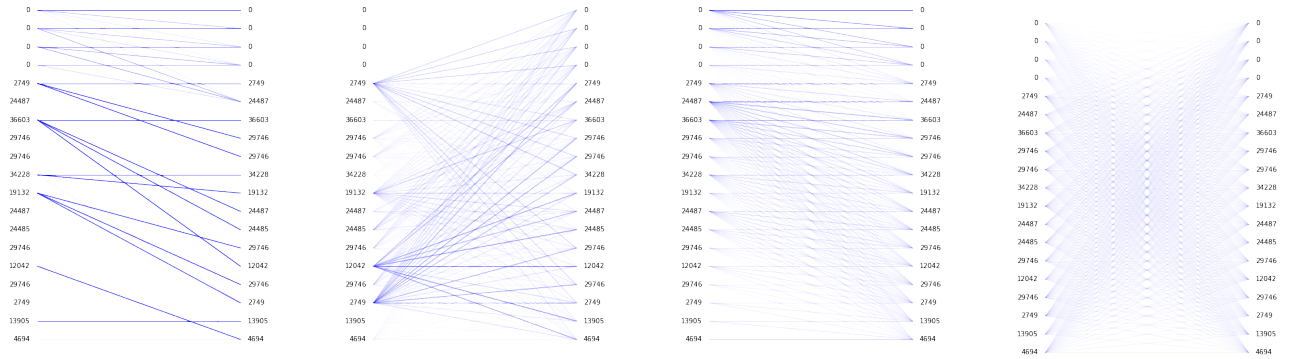
Figure 7 shows attention weights' projection. The darker blue shows the more significant items in sessions. We chose randomly some sessions in test data of DIGINETICA dataset.

5 CONCLUSION

We have presented a Transformer based neural network architecture for session-based recommendation. With two Transformers, our model can extract features of a user's sequential behavior as well as capture the main purpose of the current session. Experimental results showed that our model outperformed several state-of-the-art models on two public datasets.

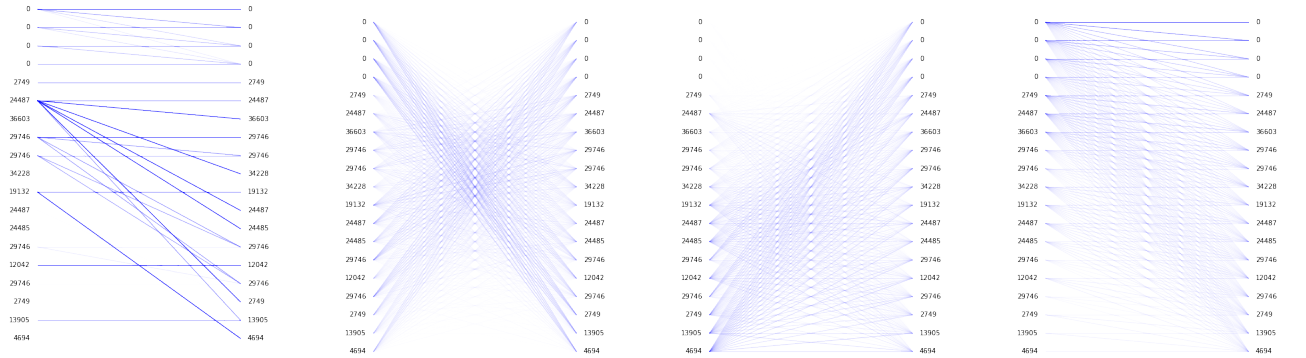
ACKNOWLEDGEMENTS

This work was supported by Ministry of Science and Technology of Vietnam under grant KC.01.23/16-20.



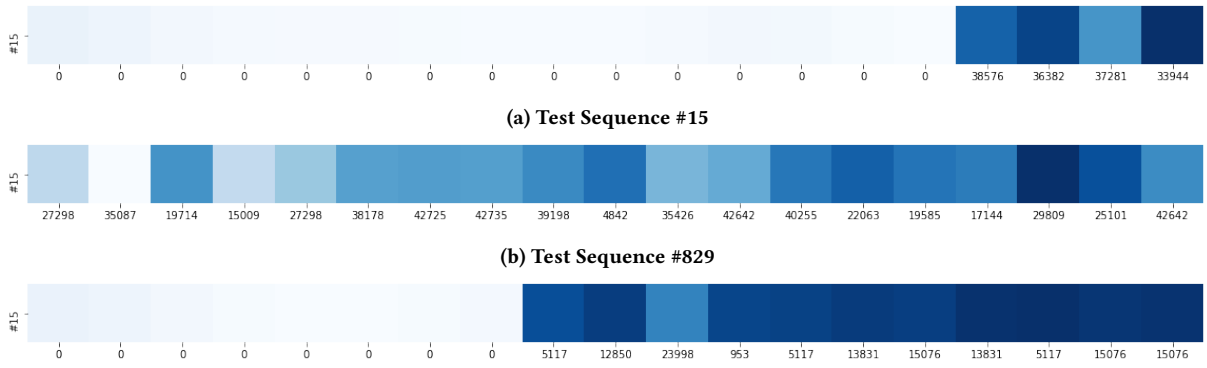
(a) HEAD7-1. Regular distance points (b) HEAD6-2. Skipping Zeros padding (c) HEAD8-5. Skipping Zeros padding (d) HEAD1-6. Attending broadly

Figure 5: Weight Visualization of Global Transformer Decoder



(a) HEAD5-1. Regular distance points (b) HEAD7-4. Cross position states (c) HEAD8-4. Attending broadly (d) HEAD8-5. Attending broadly

Figure 6: Weight Visualization of Local Transformer Decoder



(a) Test Sequence #15

(b) Test Sequence #829

(c) Test Sequence #95

Figure 7: Visualization of Attention Weights Mechanism

Table 2: The comparison between different architectures of Local Transformer's decoders

Local Decoder's Structure	1/32 YOOCHOOSE		1/64 YOOCHOOSE		DIGINETICA	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
No use	70.2	31.0	70.3	30.8	48.2	15.5
Using average all the last hidden states	70.3	31.0	70.5	30.9	**51.8**	**17.9**
Using attention weights for last hidden states	**70.4**	**31.1**	**70.7**	**31.0**	51.8	17.8

Table 3: The comparison of using the masking layer

Model	1/32 YOOCHOOSE		1/64 YOOCHOOSE		DIGINETICA	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
Without masking	69.6	30.3	69.8	30.2	49.8	16.9
With masking	**70.4**	**31.1**	**70.7**	**31.0**	**51.8**	**17.8**

Table 4: Performance comparison between our model and some sate-of-the-art models

Baseline	1/32 YOOCHOOSE		1/64 YOOCHOOSE		DIGINETICA	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
Improved GRU4REC	68.4	30.2	68.2	29.9	42.5	13.3
Nextitnet	63.8	27.1	62.5	26.1	34.6	10.2
NARM	68.6	28.8	68.3	28.6	49.7	16.2
SR-GNN	70.1	30.8	70.4	30.7	49.9	16.4
Transformer	**70.4**	**31.1**	**70.7**	**31.0**	**51.8**	**17.8**

REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining Sequential Patterns. In *ICDE '95*. IEEE Computer Society, Washington, DC, USA, 3–14. http://dl.acm.org/citation.cfm?id=645480.655281

[2] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist Prediction via Metric Embedding. In *KDD '12*. ACM, New York, NY, USA, 714–722. https://doi.org/10.1145/2339530.2339643

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[4] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *arXiv e-prints*, Article arXiv:1511.06939 (Nov 2015), arXiv:1511.06939 pages. arXiv:cs.LG/1511.06939

[5] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. *arXiv e-prints*, Article arXiv:1606.08117 (Jun 2016), arXiv:1606.08117 pages. arXiv:cs.LG/1606.08117

[6] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. *arXiv e-prints*, Article arXiv:1711.04725 (Nov 2017), arXiv:1711.04725 pages. arXiv:cs.IR/1711.04725

[7] Xiaoli li. 2012. Effective Next-Items Recommendation via Personalized Sequential Pattern Mining.

[8] G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (Jan 2003), 76–80. https://doi.org/10.1109/MIC.2003.1167344

[9] Tu Minh Phuong, Do Thi Lien, and Nguyen Duy Phuong. 2019. Graph-based context-aware collaborative filtering. *Expert Systems with Applications* 126 (2019), 9–19. https://doi.org/10.1016/j.eswa.2019.02.015

[10] Tu Minh Phuong, Tran Cong Thanh, and Ngo Xuan Bach. 2018. Combining User-Based and Session-Based Recommendations with Recurrent Neural Networks. In *Neural Information Processing - 25th International Conference, {ICONIP} 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part {I} (Lecture Notes in Computer Science)*, Long Cheng, Andrew Chi-Sing Leung, and Seiichi Ozawa (Eds.), Vol. 11301. Springer, 487–498. https://doi.org/10.1007/978-3-030-04167-0_44

[11] T. M. Phuong, T. C. Thanh, and N. X. Bach. 2019. Neural Session-Aware Recommendation. *IEEE Access* 7 (2019), 86884–86896. https://doi.org/10.1109/ACCESS.2019.2926074

[12] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf* (2018).

[13] Badrul Sarwar, George Karypis, Joseph A Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW '01*. Association for Computing Machinery, Inc, 285–295. https://doi.org/10.1145/371920.372071

[14] Guy Shani, Ronen I. Brafman, and David Heckerman. 2012. An MDP-based Recommender System. *arXiv e-prints*, Article arXiv:1301.0600 (Dec 2012), arXiv:1301.0600 pages. arXiv:cs.LG/1301.0600

[15] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *ACM International Conference on Web Search and Data Mining*.

[16] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D Convolutional Networks for Session-based Recommendation with Content Features. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. ACM, 138–146. https://doi.org/10.1145/3109859.3109900

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv e-prints*, Article arXiv:1706.03762 (Jun 2017), arXiv:1706.03762 pages. arXiv:cs.CL/1706.03762

[18] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based Recommendation with Graph Neural Networks. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI '19)*, Pascal Van Hentenryck and Zhi-Hua Zhou (Eds.), Vol. 33. 346–353. https://doi.org/10.1609/aaai.v33i01.3301346

[19] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2018. A Simple Convolutional Generative Network for Next Item Recommendation. *arXiv e-prints*, Article arXiv:1808.05163 (Aug 2018), arXiv:1808.05163 pages. arXiv:cs.IR/1808.05163