Selected Self-Attentive Sequential Recommender models: Comparison

Oriol Camps Pérez
University of Würzburg

ABSTRACT

Sequential recommender systems aim to capture the context of users activities in order to predict their next action, basing these predictions on the user's previous actions. There are different approaches which show a certain ability to interpret the dynamics of sequential patterns (e.g. Markov Chains and Recurrent Neural Networks), but that have limitations depending on, for example, the dataset density. In this paper, two performant Self-Attention Sequential Recommender models that escape from these issues are presented. The first one, SASRec, was conceived with this exact purpose, while the second, TiSASRec, can be seen as an evolution of the first one, thanks to its time interval awareness which enables it to find concealed patterns.

Keywords: user behaviour, user historical actions, ranking, recommender system, sequential prediction, self-attention, position-aware, time-aware

1 INTRODUCTION

With the passage of time, more and more online services are being used and developed. The personalization of the experience for each user is a feature that can certainly be very profitable for both companies, in terms of monetary profit, and users, in terms of time and quality, since a personalized service will fit better their expectations. For this purpose, recommender systems appear to be very useful. This is the reason why they are present in many different services, from e-commerce to video streaming services [3].

Sequential recommendation systems can be utilized in order to predict the behaviour of the users and, therefore, to create recommendations which can be useful for them. The goal of these systems is to capture and model useful patterns of user's historical behaviour based on their past actions, keeping in mind the context of recent actions. This can potentially become a really ambitious task, considering that the input data would possibly grow exponentially, as the past actions, which are being used as context, increase in number [5].

Different approaches have been taken in order to mitigate that inconvenience. For instance, Markov Chain (MCs) models predict the user's next action only based on a specific number of last actions [8], so that an L-order MC bases its recommendations on the previous L actions [6]. Another approach to consider is the Recurrent Neural Networks (RNNs). This approach can be employed with satisfactory results in session-based recommendations [3], nevertheless it should be taken into account that it might require large amounts of data because of its low performance on sparse datasets [5].

This sequential data can be processed with an attention mechanism that relates the different positions in the sequence for the purpose of computing a representation of the sequence. That is the reason why it is called *self-attention* or, at times, *intra-attention* [11]. This mechanism has been seen to be very efficient when it comes to capturing syntactic and semantic patterns between the words of a sentence, for instance using it for machine translation tasks [5]. Based on that, W. Kang and J. McAuley [5] applied this method to create a sequential recommendation system model called *Self-Attention based Sequential Recommendation* (SASRec). This model is the starting point of study in this paper.

2 MAIN CONCEPTS

2.1 General Recommendation

Modelling the compatibility between users and items, based on historical feedback and interactions such as clicks, purchases, likes, etc., is the main goal of a recommender system. But modelling that, can be difficult and challenging, especially for certain feedbacks. There are two types of user feedback, *explicit* feedbacks that are basically any type of direct rating (stars, like or dislike thumbs, etc.) and that are easily interpretable, and *implicit* feedbacks like purchase history, comments, browsing history and search patterns or even mouse movements that could give valuable information about users behaviour [4] but which present a bigger challenge in order to interpret them. That is even more accentuated because of the ambiguity of interpreting *non-observed* data, that is, for example, non purchased items [5].

To model users' preferences based on their interaction, recommendation systems usually use Collaborative Filtering (CF), and among different CF methods, Matrix Factorization (MF) is the most utilized one [9]. These MF methods look to discover latent dimensions to represent the preferences of the user and the properties of an item, as well as to approximate interactions between them [5]. By projecting items and users into a shared vector space, the user's preference on an item can be estimated through the inner product of their vectors [9].

2.2 Sequential Recommendation

Sequential recommender systems aim to predict and capture sequential patterns of successive items and the user's interactions through time by modelling item-item transition matrices [5], [6], [9].

Aside from MCs-based approaches, there are other approaches which model complete user behaviour sequences via RNNs [6], [9], such as *Gated Recurrent Unit* (GRU) [9]. This methods' general idea is to encode the user's past records into a vector.By doing so, the representations of the user's predilections can be used, for instance, to predict future tendencies [9]. Overall, these methods have a good performance on dense datasets, but have an inadequate performance when it comes to sparse datasets [6].

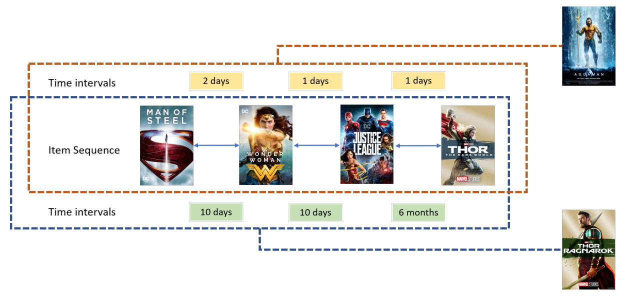

Figure 1: Given the same item sequence with different time intervals, a Time Interval-Aware Self-Attention model like the one proposed by J. Li et al. [6] **(TiSASRec) will yield a different prediction for the next item.**

There are more sequential recommendation methods other than the MC and the RNN-based methods such as, Convolutional Neural Network (CNN) based methods, where various precedent items are treated as if they were an image in order to process them [10].

2.3 Self-Attention Mechanisms

Originally from Vaswani et al. [11], these mechanisms consist in treating unequally the values of the input, detecting the most relevant parts, and giving weights to the input according to that relevance.

There are many tasks for which attention mechanisms have been proven to be effective [5], [6], specially regarding natural language processing tasks [1], [9] but are also present in other tasks like image captioning [13] and, of course, in recommender systems [12], [5], [6].

2.4 Time Interval-Aware Self-Attention

Other users' context, aside from the order of users' interactions, can also be taken into account. There are approaches that propose to keep in mind the time relation between the actions also affecting the output (see Figure 1). So that given the same item sequence for two datasets but with different time intervals between their items, the resulting prediction or recommendation would be different. So, using *timestamps* (see 5.2.5) on the items of a dataset, allows us to capture the temporal dynamics of users and items [6]. That can initially seem unnecessary, but it is understandable that the intervals between the different inputs can be very important, and also that the more users' context if taken into account, the more accurate recommendations can be. The popularity of an item can vary during different time slots, as well as the average ratings of a user might increase or decrease over time [6], just like personal likings. This approach is taken by Li et al. [6] to create TiSASRec (see Section 5).

3 PRECEDING WORK: TRANSFORMER

Vaswani et al. [11] created a purely attention-based sequence-to-sequence method called *Transformer* that has shown good results on machine translation tasks from English to German and from English to French.

This mechanism can find patterns using weights on the input, but it is also faster and more efficient to train in comparison to, for example, RNNs based methods, which evaluate the sequence step-by-step while in self-attention mechanisms this is done simultaneously through different matrix operations.

Heavily relying on the proposed self-attention modules in order to interpret complex structures in sentences, this attention-based method achieved state-of-the-art performance and efficiency. This mechanism is nowadays used by other models and for multiple purposes (see 2.3), including recommendation problems [1], [5], [6], [11], [13].

4 SASRec

W. Kang and J. McAuley [5] have presented the SASRec (Self-Attention based Sequential Recommendation) model that, at each time step, designates different weights to the preceding items adaptively, applying self-attention mechanisms to sequential recommendation problems with the aim to mitigate some of these limitations that RNNs and MCs present. SASRec was conceived with the idea of being able to fathom out context from all the past actions, like RNNs, but even with sparse datasets; as well as being capable to be trained more efficiently.

4.1 Methodology

In this subsection, it is summarized how W. Kang and J. McAuley [5] have built their SASRec model with an embedding layer, several self-attention blocks, and a prediction layer.

Given a sequence of a user's actions $S^u = (S_1^u, S_2^u, ..., S_{|S^u|-1}^u)$, the model aims to predict the next item $S_{|S^u|}^u$. With this in mind, during the training of the model, it uses the previous t items in order to predict the next one at time step t. Taking this into account, it would be pertinent to think of the output of the model as the input sequence $S^u = (S_1^u, S_2^u, ..., S_{|S^u|-1}^u)$ but shifted to the next item (i.e. the prediction) obtaining a sequence $S^u = (S_2^u, S_3^u, ..., S_{|S^u|}^u)$ [5], just as represented in Figure 2.

4.1.1 ***Embedding Layer****.* The input training sequence is transformed from $S^u = (S_1^u, S_2^u, ..., S_{|S^u|-1}^u)$ into the fixed-length sequence $s = (s_1, s_2, ..., s_n)$, being n the exact length of the sequence. This is done via truncation when the sequence has more items than n, or padding items (in that case, constant zero vectors) when the sequence has fewer items than n.

In order to construct the embedding matrix E, an item embedding matrix M to encode item similarity is created, and the input embedding is retrieved. Also, a positional embedding matrix P to let E take into account the positions of items inside the sequence is injected into the input embedding, since the self-attention model is not aware of the positions of previous items [5].

$$E = \begin{bmatrix} M_1 & + & P_1 \\ M_2 & + & P_2 \\ & ... & \\ M_n & + & P_n \end{bmatrix} \tag{1}$$

4.1.2 ***Self-Attention Block****.* The attention layer calculates a weighted sum of all values using the *Scaled Dot-Product Attention* presented by A. Vaswani et al. [11], where the weight between

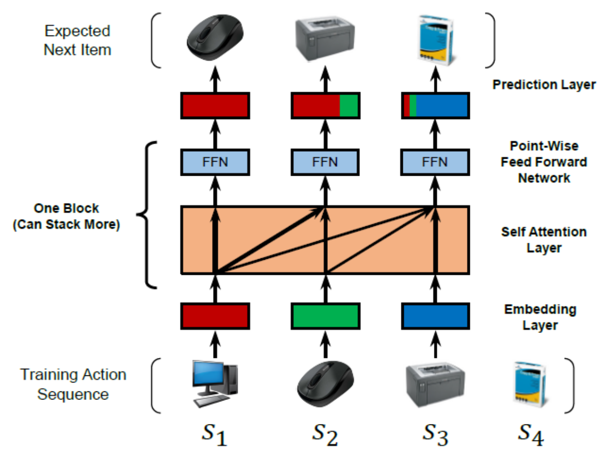

Figure 2: In this diagram, a simplified representation of the training process of the SASRec model is shown. The model uses attention to focus on items that are relevant to the next action from considering all previous items at each time step. It can be seen than the output sequence corresponds to the initial sequence but shifted to the next (i.e. the predicted) item as explained in 4.1 [5].

query i and value j relates to the interaction between query i and key j [5].

With a dimension of d_k, Q represents the matrix of queries, K the matrix of keys and V the matrix of values, so that the output matrix is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

The scaling factor of $\frac{1}{\sqrt{d_k}}$ is used to counteract the bad effects of having high dimensionalities [11].

The self-attention operation takes as an input the initial embedding matrix, and uses linear projections to take the vectors Q, K and V. In other words, those three vectors are different projections from the same matrix, different views from the same object, hence, the use of the term self-attention. Then, these three vectors are used by the attention layer. The model becomes flexible due to the projections. For instance, the model can learn asymmetric interactions (i.e., <query i, key j> and <query j, key i> can have different interactions) [5].

Only the first t items should be considered by the model when predicting the $(t + 1)$-st item. However, the t-th output of the self-attention layer (S_t) contains embeddings of subsequent items, which makes the model ill-posed. Therefore, the attention is altered by forbidding all links between Q_i and K_j $(j > i)$.

Then, a point-wise two-layer feed-forward network is applied to all item embeddings in order to arm the model with nonlinearity and to take into account interactions between distinct latent dimensions [5].

4.1.3 *Stacking Self-Attention Blocks.*

After the first self-attention block, all the previous item embeddings are aggregated using a point-wise two-layer feed-forward network, which ensures that there is no interaction between embeddings from different items. In this stage, layer normalization is applied in order to avoid the problems that can appear or worsen when the point-wise two-layer feed-forward network goes deeper. Like when 1) there is overfitting due to the increase of the model capacity; 2) the training process stops being stable (due to, for example, vanishing gradients); and 3) more training time is often required for models with more parameters. Layer Normalization stabilizes and accelerates the neural network training [5].

4.1.4 *Prediction Layer.*

In order to predict the relevance $r_{i,t}$ of an item i given the first t items, a matrix factorization layer is adopted, after that the self-attention blocks have adaptively and hierarchically extracted information of previously consumed items. An elevated interaction score $r_{i,t}$ translates to a high relevance, so it is possible to generate recommendations by ranking the obtained scores, selecting the most probable next item [5].

With the purpose of reducing the model size as well as alleviating overfitting, W. Kang and J. McAuley [5] have considered a scheme that uses a single embedding item, *Shared Item Embedding*.

4.1.5 *Network Training.*

Using a fixed length sequence of a user's items, as explained in 4.1.1, o_t is the expected output at time step t:

$$o_t = \begin{cases} S_{t+1} & 1 \le t \le n \\ S^u_{|S^u|} & t = n \end{cases} \tag{3}$$

Being j a randomly-generated negative item, the binary cross entropy loss function is adopted as the objective function:

$$-\sum_{S^u \in S} \sum_{t \in [1,2,\ldots,n]} \left[\log(\sigma(r_{o_t,t})) + \sum_{j \notin S^u} \log(1 - \sigma(r_{j,t})) \right] \tag{4}$$

4.2 Experimental Evaluation

4.2.1 *Datasets.*

The methods were evaluated on four datasets which differed in platforms, domains, and sparsity. All four datasets were from three real world applications [5].

- *Amazon*: It consists of an ensemble of datasets with a considerable high variability and sparsity. It incorporates a compilation of reviews from product crawled from *Amazon.com*. A separate dataset was considered for the top-level product categories. Two categories were considered for the evaluation of the method, *Amazon Beauty* and *Amazon Games*. Both of the considered categories have, on average, the fewest actions both, per item and per user (6.9 and 7.6 respectively for *Amazon Beauty*, and 12.1 and 9.3 respectively for *Amazon Games*), in comparison to the other two datasets [5].
- *Steam*: This dataset was crawled from the wide online distribution platform for video games, Steam. The platform has a lot of potentially useful information. This dataset has, in average, an elevated number of actions per item, with a value of 282.5, but a smaller number of average actions per user, with a value of 11.0 [5].
- *MovieLens*: It consists of a benchmark dataset extensively used to evaluate collaborative filtering algorithms. This is

Table 1: SASRec recommendation performance [5]

Dataset	Metric	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	Improvement vs.	
		PopRec	BPR	FMC	FPMC	TransRec	GRU4Rec	GRU4Rec$^+$	Caser	SASRec	(a)-(e)	(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 boldfaced values correspond to the best performing method in each row, and the underlined ones represent the second-best method in each row. It is clearly visible that the performance of SASRec is superior. The last two columns correspond to the percentage of improvement that SASRec shows over non-neural approaches, (a)-(e), and neural approaches, (f)-(h), respectively [5]. *(See 4.2.4)*

the heaviest of the four datasets, with an average of actions per user of 163.5, and of actions per item of 289.1 [5].

4.2.2 ***Comparison Methods.***
The authors of [5] incorporated three groups of recommendation baselines so as to study the efficacy of their method.

The first group does not take into consideration the action's sequence order, but rather just the user feedback. They do so by the use of general recommendation methods:

- *PopRec*: It consists of a simple baseline which ranks items depending on their popularity [5].
- *Bayesian Personalized Ranking* (BPR) [7]: It is a classic method to learn the personalized rankings from implicit feedback [5].

The next group bases their sequential recommendation methods on first order MCs, considering only the item that was visited last.

- *Factorized Markov Chains* (FMC): A first-order chain method which uses two item embeddings to factorize an item transition matrix. It only uses the last visited item to generate recommendations [5].
- *Factorized Personalized Markov Chains* (FPMC) [8]: A method which captures the long-term preferences of the users and the item-to-item transitions with a matrix factorization combined with a factorized first-order Markov chains as its recommender [5].
- *Translation-based Recommendation* (TransRec) [2]: It is defined as a first-order sequential recommendation method. It models as a translation vector each user with the aim to capture the transition from the present item to the following [5].

The third and final group, takes into account many, if not all, the items which were previously visited.

- *GRU4Rec* [3]: This seminal method models user action with RNNs for session-based recommendation [5].
- *GRU4Rec*$^+$: It is an upgraded version of GRU4Rec, with a modified sampling strategy and loss function [5].
- *Convolutional Sequence Embeddings* (Caser) [10]: This is a method based on CNN, which applies convolutional operations on the embedding matrix of the L most recent items, capturing high-order Markov chains. It was shown to attain an adequate sequential recommendation performance [5].

4.2.3 ***Evaluation Metrics.***
Two common Top-N metrics were adopted for the evaluation of the recommendation performance.

- *Hit@10*: it estimates the fraction of times which the ground-truth next item is present amidst the top 10 items [5].
- *NDCG@10*: it consists of a position-aware metric that awards higher scores to higher ranked items [5].

W. Kang and J. McAuley [5] sampled randomly 100 negative items and ranked them with the ground-truth item, and evaluated *Hit@10* and *NDCG@10* based on these rankings.

4.2.4 ***Recommendation Performance.***
In Table 1 the recommendation performance on the four datasets for all the methods are presented. A general pattern appears when taken into consideration the second best methods from all datasets, with non-neural methods showing a better performance on sparse datasets, and neural approaches with a better performance on denser datasets. This can be attributed to the higher number of parameters of the neural approaches to capture high order transitions. On the contrary, simpler models, even when carefully designed, have a higher efficiency in high-sparsity settings [5]. W. Kang and J. McAuley [5] SASRec method surpasses all baselines for both dense and sparse datasets. On average, the model gains against the strongest baseline 6.9% Hit Rate and 9.6% NDCG improvements. A probable cause is that the mentioned model is able to, for various datasets, attend items within different ranges adaptively. It should be noted that it was seen that the SASRec model typically profits from larger numbers of latent dimensions.

5 TiSASRec

Li et al. [6] presented TiSASRec (Time Interval aware Self-attention based sequential recommendation), a model which can be easily seen as an evolution of SASRec.

TiSASRec uses self-attention mechanisms, and its internal structure is very similar to SASRec, along with its training, adopting binary cross entropy as the loss function as well. But it additionally takes into account the time intervals between interactions in the sequence of users' actions.

Table 2: TiSASRec recommendation performance [6]

Dataset	Metric	Pop	BPR	FPMC	TransRec	GRU4Rec$^+$	Caser	MARank	TiSASRec	Improvement
MovieLens-1m	Hit@10	0.4386	0.5952	0.6182	0.4478	0.6522	0.7517	<u>0.7652</u>	**0.8038**	5.04%
	NDCG@10	0.2389	0.3421	0.3917	0.2488	0.4334	0.5011	<u>0.5199</u>	**0.5706**	9.75%
Amazon CDs&Vinyl	Hit@10	0.3335	0.5627	0.5122	0.3356	0.3536	0.3865	<u>0.6464</u>	**0.7212**	11.57%
	NDCG@10	0.1862	0.3626	0.3355	0.1893	0.2005	0.2285	<u>0.4355</u>	**0.5047**	15.89%
Amazon Movies&TV	Hit@10	0.4576	0.5537	0.5104	0.4793	0.4848	0.5015	<u>0.6584</u>	**0.7125**	9.21%
	NDCG@10	0.2723	0.3482	0.3376	0.2891	0.2987	0.3212	<u>0.4568</u>	**0.4974**	8.89%
Amazon Beauty	Hit@10	0.3215	0.2554	0.2806	0.3016	0.3184	0.2639	<u>0.3472</u>	**0.4345**	25.14%
	NDCG@10	0.1758	0.1523	0.179	0.1704	0.1743	0.1446	<u>0.2127</u>	**0.2818**	32.49%
Amazon Game	Hit@10	0.4006	0.4417	0.5222	0.3943	0.3971	0.4474	<u>0.6498</u>	**0.7087**	9.06%
	NDCG@10	0.23	0.2731	0.341	0.2292	0.2321	0.2661	<u>0.4437</u>	**0.4797**	8.11%
Steam	Hit@10	0.6611	0.5941	0.6017	0.6594	0.6605	0.7135	<u>0.751</u>	**0.8103**	7.9%
	NDCG@10	0.4136	0.3598	0.4083	0.412	0.4138	0.4787	<u>0.5135</u>	**0.5897**	14.84%

The boldfaced values correspond to the best performing method in each row, and the underlined ones represent the second-best method in each row. It is clearly visible that the performance of TiSASRec is superior. The last column correspond to the percentage of improvement that TiSASRec shows over other approaches [6]. (*See 5.2.4*)

TiSASRec uses the mechanisms from SASRec to take into account the position of the items, i.e. the order of the sequence, but also adds time-awareness to the model, and so, it captures more information of the users' context to discover better or more complex patterns that can influence the decision of a user, and thus, that are relevant when recommending an item.

5.1 Methodology

This model includes personalized time interval processing (see 5.2.5), an embedding layer, time-aware self-attention stacked blocks and a prediction layer, again, being very similar to SASRec (see 4.1). For instance, a n-length-fixed sequence S^u is used just like in SASRec as well as a time sequence T^u that works the same. To combine representation of items, positions and time intervals, Time-aware self-attention blocks are used and similarly stacked for the prediction layer, which uses a user's preference score $R_{i,t}$ where the time intervals are included [6].

5.1.1 ***Timestamp methods***. Three kinds of methods were discussed for the processing of timestamps.

(1) The first method directly uses the timestamps as the features. The minimum timestamps in the datasets were subtracted in order to let all the timestamps start from 0.
(2) The second one uses unscaled time intervals. The minimum timestamps in the sequence are subtracted with the aim to let user's timestamps start from 0.
(3) The third and final one uses personalized time intervals. The intervals are divided by the smallest intervals for every time-interval of a user. (See 5.2.5).

5.1.2 ***Time Interval-Aware Self-Attention***. Since there can be multiple items with the same timestamp (such as items with "daystamps"), to consider only timestamps instead of both positions and time intervals would not improve the model in regard to SASRec (see Section 6).

Instead of feeding the self-attention layer with linear projections of the embedding matrix, different embedding matrices are constructed using distinct positional embedding matrices.

5.2 Experimental Evaluation

5.2.1 ***Datasets***. The methods were evaluated on six public datasets with different domains, size, and sparsity, all of which were from three real world applications [6]. These three applications were the same three that W. Kang and J. McAuley [5] used in their experiment: *Amazon*, *Steam* and *MovieLens*, as explained in 4.2.1. Nonetheless, it must be noted that for the *Amazon* platform four categories were considered instead of the two from the previous experiments, which were *CDs and Vinyl*, *Movies and TV*, *Beauty* and *Video Games*. All of the datasets had the specific date, or timestamps, of the interactions [6]. Like W. Kang and J. McAuley [5], J. Li et al. [6] also treated as implicit feedback the existence of reviews or ratings for all the mentioned datasets [6].

5.2.2 ***Comparison Methods***. The authors of [6] compared the *Time Interval-aware Self-Attention based Sequential Recommendation* model (TiSASRec) with several models from four different groups of recommenders. Most of these models are also considered for comparisons by W. Kang and J. McAuley [5] and thus, are explained in 4.2.2. Therefore, there are not detailed explanations in this section, but only a condensed list.

The first two methods, use classic general recommendation, which does not consider sequential patterns: *Pop* (ranks items in all user's training sets by their popularity) and *Bayesian Personalized Ranking* (BPR) [7].

The next group consist of sequential recommendation methods that are based on first-order MCs, so only considering the last visited item: *Factorized Markov Chains* (FMC) [8] and *Translation-based Recommendation* (TransRec) [2].

The third group consist of Neural Networks (NN) -based methods: *GRU4Rec*$^+$ (for session-based recommendation [6]), *Convolutional Sequence Embeddings* (Caser) [10], and *Multi-order Attentive*

Table 3: Comparison of three timestamp processing methods (NDCG@10) [6]

Dataset	Method (1)	Method (2)	Method (3)
MovieLens-1m	0.5643	0.5658	**0.5706**
Amazon CDs&Vinyl	0.5001	0.4931	**0.5047**
Amazon Movies&TV	0.4838	0.4826	**0.4974**
Amazon Beauty	0.2692	**0.2825**	0.2818

The boldfaced values correspond to the best performing method in each row [6]. (*See 5.2.5*)

Ranking Model (MARank) [14], the only one which is not used for comparisons with SASRec [5].

In this method, both individual- and union-level item interaction are merged into a preference inference model from several views. The aim is to represent the user's short-term preferences, by embedding the user along with a set of present items into multi-order features from intermedia hidden status of a deep neural network [14].

The last compared method by Li et al. [6] is none other than SASRec (see Section 6).

5.2.3 ***Evaluation Metrics.*** The model comparison is performed using the two common Top-N metrics Hit Rate@10 (Hit@10) and NDCG@10, similar to SASRec, which have been explained in 4.2.3.

5.2.4 ***Recommendation Performance.*** In Table 2 the recommendation performance on the six datasets of all methods is shown. Compared to other baselines, MARank [14] shows a relatively good performance. NN-based methods show an accentuated better performance than MC-based methods for dense datasets. This is explained due to the stronger ability to capture long-term sequential patterns. Nevertheless, MC show a better performance on sparse datasets due to their focus on dynamic transition of items. TiSASRec shows an improvement in comparison to the best baseline methods concerning the two metrics on all dense and sparse datasets. While the other models only consider the different items and the absolute positions so as to adapt weights, the TiSASRec model also considers the time intervals, taking advantage of the attention mechanism. By taking advantage of the attention mechanism, the TiSASRec model is able to attend to items within different ranges in a more adaptative matter for different datasets [6].

5.2.5 ***Personalized time intervals.*** In Table 3 the results of the different methods for the processing of the timestamps are presented (see 5.1.1). It should be considered that the highest NDCG are highlighted with boldface. As it can be seen, in the first three datasets, the method which achieves the best performance is the third one.

6 COMPARISON: SASREC AND TISASREC

Before the comparison, Li et al. [6] discussed whether those two models are comparable or not. The reason is that SASRec only considers the absolute position of the items in the sequence, but TiSASRec combines both the absolute position and the relative time intervals. On that account, Li et al. [6] modified TiSASRec and created TiSASRec-R, which only contemplates relative time intervals, in order to purely compare the absolute-position (SASRec) and the time-interval (TiSASRec-R) approaches, and their combination (TiSASRec).

Table 4: Comparison between time intervals and absolute positions [6]

Dataset	Metric	SASRec	TiSASRec-R	TiSASRec
MovieLens	Hit@10	0.7929	**0.8031**	0.8038
	NDCG@10	0.5524	**0.5648**	0.5706
CDs&Vinyl	Hit@10	0.7001	**0.7176**	0.7212
	NDCG@10	0.4880	**0.4978**	0.5047
Movies&TV	Hit@10	**0.7035**	0.701	0.7125
	NDCG@10	**0.4882**	0.4853	0.4974
Beauty	Hit@10	0.4185	**0.4223**	0.4345
	NDCG@10	0.2722	**0.2748**	0.2818
Game	Hit@10	**0.6952**	0.6939	0.7087
	NDCG@10	**0.4714**	0.4711	0.4797
Steam	Hit@10	**0.8058**	0.7937	0.8103
	NDCG@10	**0.5801**	0.5731	0.5897

The best results except for TiSASRec are highlighted in bold. *(See 6)*

The comparison between these three methods, using the already mentioned datasets and metrics, is shown in Table 4. One might notice that SASRec and TiSASRec-R have similar performance, in the sense that each one is the second-best method for half of the datasets. Li et al. [6] discussed that since self-attention mechanisms do not take into account the positions of the items by default, which is why positional embedding matrices are used (see 4.1.1), TiSASRec-R is reduced to self-attention without any positional information. In other words, SASRec and TiSASRec-R are modelling one aspect of the users' context, while TiSASRec is modelling both aspects and so, discovering richer patterns from richer item relations.

7 CONCLUSION

This paper performs an overview of some recommender models, focusing on two specific performant self-attentive sequential recommender models that consider previous actions of a user in order to recommend the next item. While some previous recommender models use RNNs, for large datasets on session-based recommendations, or CNNs, treating the recent actions as *images*, SASRec implements self-attention mechanisms in action sequences that were firstly used for translation tasks and shows better results independently of the density of the dataset. The other compared model, TiSASRec, can be easily seen as an evolution of the first one, adding a time-awareness to the model that allows the recommendation to be "conscious" of the nuances that two actions performed in the same order but in different time intervals implies. Probably for that reason, it shows even a superior performance in comparison to all previous methods. It can be concluded that the more nuances of the context of the actions are captured and modelled, the better recommendations will potentially be given. Therefore, maybe new lines of study could explore that.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[2] Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based recommendation. *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017.

[3] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *CoRR*, abs/1511.06939, 2016.

[4] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, 2008.

[5] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206, 2018.

[6] Jiacheng Li, Yujie Wang, and Julian McAuley. Time interval aware self-attention for sequential recommendation. *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020.

[7] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *ArXiv*, abs/1205.2618, 2009.

[8] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW '10*, 2010.

[9] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

[10] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018.

[11] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[12] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI*, 2017.

[13] Ke Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[14] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. Multi-order attentive ranking model for sequential recommendation. In *AAAI*, 2019.