SASRec

Francisco Rencoret frencoret@uc.cl - Francisco Pérez faperez10@uc.cl

December 2018

1 Abstract

Self Attentive Sequential Recommendation model uses self-attention mechanisms with matrix factorization techniques to understand user's sequential behaviour and recommend items. In this work we implement our own version of the model, optimize and investigate the correlations of the different model's structures with the desired recommendation task. Specifically, we studied the correlation between self-attention stacked layers with longer data sequences within model's performance. On the other hand, SASRec model is compared with other baseline models available in the OpenRec Library (BPR, PMF, RNNRec and VanillaYoutubeRec), however, SASRec vastly outperforms all baseline methods analyzed in the investigation.

2 Introduction

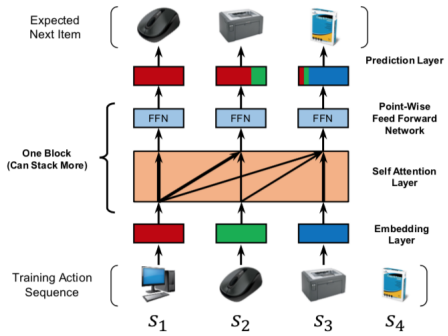


Figure 1: A simplified diagram showing the training process of SASRec. At each time step, the model considers all previous items, and uses attention to 'focus on' items relevant to the next action.

3 Related Work

Recommendation systems seek to model users interaction with items and understand their behaviour. They use historical data to learn and suggest future items to enhance the user's upcoming experiences. Systems can use the explicit (e.g. item ratings) or implicit feedback (e.g. time spent looking at an item), but we will focus on model that use the first option.

Related works can be subdivided by the different approaches they take for the recommendation task. We will cover general, temporal, sequential and attentive models.

General models usually rely on collaborative filtering considering content or context information of items. Content models receive information about the item's structure or composition (e.g. length of movie), while context models receive information about the context in which the user consumed the item (e.g. which device did the user use when consuming the item). When data becomes sparse, they combine these models with Matrix Factorization MF techniques, which uncover latent dimensions to represent user and items relationships [1]. General models where leading recommendation tasks until deep learning techniques appeared presenting better results in different tasks. Multi-layer perceptrons are used for capturing item's content [2] and replacing MF techniques [3].

Temporal models try to capture changes in the consumption of items by the users through time. They are capable of understanding that temporal drift items experience (e.g. how user's preferences has changed for movies). Sequential models focus mainly on the sequence of user's historical data, obviating temporal separation between items consumptions. For example, GRU4Rec [4] uses Recurrent Neural Networks RNN for capturing knowledge from users past historical consumption sequences.

Lastly, attention mechanisms have shown state-of-the-art results for several tasks such as machine translation or visual question and answering tasks. These models consider a query, key and value to learn attention coefficients which relate queries with certain values more than others. Bahdanau [5] present query given attention models while the Tranformer [6] shows methods that enable self-attention within data. This work focuses on the second method mentioned. Although they are not used specifically to recommend, they work as embedding enhancers and complement recommendation models such as RNN or MF.

4 Dataset

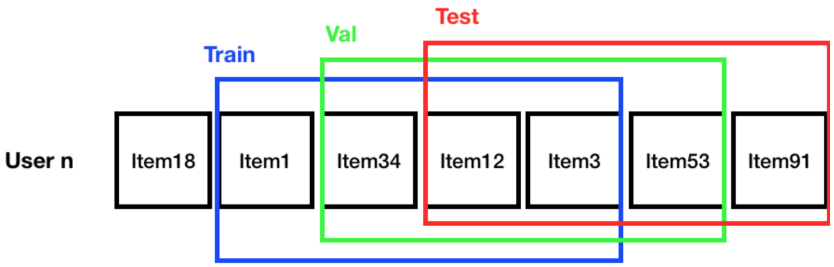
Experiments were done on the MovieLens dataset. This is a well-known dataset in the field of recommendation systems and contains user and movie identifiers as well as ratings and timestamps. The dataset is made available by the GroupLens research group and University of Minnesota [10] from the movie ratings webpage www.movielens.org.

Preliminary experiments were performed on the small MovieLens dataset

which contains 100,000 ratings. However, the final results shown in this paper are done on the 1 million ratings version of the dataset. This change was made because we wanted the results to be comparable to the original paper's results.

The data set was split on training, validation and test subsets. Since the model is trying to predict the next item to be consumed by the user, ratings were ignored.

The training set is made up of items consumed sequentially (ordered by timestamp) by every user excluding the last 2 items, since they are part of the validation and testing sets. Here you can see an example considering a length of sequence equal to 4 (n = 4):



When sequences of consumption are longer that the specified n sequence length, initial items are ignored (in this case Item18).

When a user has consumed less than n items, the sequence is filled until length n is satisfied with zero-padding.

The n hyper-parameter is determined based on the average of ratings per user, which is roughly 165. Although the n parameter is later analysed and iterated on different values to find the best one for the model.

The same data set is used for the SASRec model as well as for the baseline models based on OpenRec included for comparison reasons.

5 Methodology

SASRec is a model that uses self-attention and matrix factorization mechanisms on a sequential recommendation setting. As explained in the previous section, we modeled the data as a sequence of items, where we took the last n items and created a sequence for each user. During the training process, for every user the model tried to predict the next item after the sequence (item in position n+1). In this section we will explain how we integrated a self-attentive model with a matrix factorization technique to predict the most likely product a user should consume after a sequence of n item consumptions.

5.1 Item Embedding Layer

As explained in the section before, items are represented with an index within a dictionary containing all items. We built an item embedding matrix $M \in^{|I|xd}$

which maps a given item i to an embedding vector of latent dimension d. Given a sequence of length n, we created a matrix $E = [M1, M2, M3, ...Mn]$ that contains all the embeddings for the n items of the sequence.

5.2 Positional Embedding Layer

Self-attentive models are proven to lose the sequential information in the data because they don't consider the position of the keys / values in the sequence. The attention coefficients of a query and a particular key won't depend on the position of the key in the sequence. To avoid this, we implemented an Positional Embedding Layer. The Transformer proposed to use a fixed sinusoidal function to provide an positional embedding, but we achieved better results by providing another embedding layer for the positions in the sequence. This position embedding matrix $P \in R^{nxd}$ maps sequence's indices to vectors in the latent dimension d. So, we create a new $E' = [M1+P1, M2+P2, M3+P3...Mn+Pn]$

5.3 Self-Attention Layer

Attention mechanisms use queries, keys and values. As explained in [5], the scaled dot product attention can be calculated as

$$Attention(Q, K, V) = softmax(\frac{Q \cdot K}{\sqrt{d}}) \cdot V$$

The term inside the softmax function basically calculates the attention coefficients that the query assigns for each key. The dot product between the query and the key gives a similarity measure (which is normalized to avoid large values specially with high dimension vectors) which is then passed by a softmax so that they become coefficients that sum 1. Usually keys and values are the same, so then the query becomes the weighted sum of all the values with their attention coefficients. In other words, the query's embedding now considers the rest of the embeddings of the values and combines them using their keys.

Self-attention can be obtained when using the same queries, keys and values. Intuitively, each item's embedding in the sequence (query) now considers the rest of the sequence item's embeddings (keys and values), obtaining more knowledge from the sequence. This helps the model because the item's embeddings are now not independent, but each now relates with the others.

Following the methodology proposed in [5], we pass queries, keys and values through a linear projection (to reduce dimensionality) and feed them to the attention module explained,

$$S = SA(E') = Attention(E'W^Q, E' \cdot W^K, E' \cdot W^V)$$

Following the nature of the recommendation, We structured self-attetion modules so that items consider only previous items within the sequence to pay attention to. Therefore, we modify attention by forbidding all links between $Q_i and K_j$ when $j > i$.

5.4 Point-Wise Feedforward Layer

Though the self-attention module is able to aggregate all items embeddings with adaptive weights, it still achieves linear relationships. In order to enable the model to achieve non-linear interactions between latent dimensions, we integrate a point-wise two layer feedforward network (shared weights for the sequence). As a result, we obtain

$$F_i = FFN(S_i) = RELU(S_i \cdot W_1 + b_1)W_2 + b_2$$

where $W_1, W_2 and d x d matrices and b_1, b_2 are d - dimensional vectors.$

5.5 Stacking Self-Attention modules

One block of self-attention with point-wise feedforward network will be able to aggregate information and obtain relationships within one distance level. In other words, it just captured the relationship between pairs of items. If we stack more modules, the model will be able to capture more complex interactions between triplets, quartets and b-long interactions:

$$S^b = SA(F^{b-1})$$
$$F_i^b = FFN(S_i^b)$$
$$S^1 = S and F^1 = F$$

As the model gets deeper, over-fitting and vanishing gradient problems appear. To avoid this, we implemented:

5.6 Layer Normalization

Layer Normalization helps the model train with stability and avoid vanishing gradients. Each self-attention block has their own Layer Normalization modules with their own weights.

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

where α and β are learnable parameters.

5.7 Dropout

Dropout techniques help solve overfitting problems. With a dropout probability given as a hyperparameter, we randomly turn off neurons during training to reduce the model's capabilities for learning forcing the model not to learn the training data exactly losing generalization performance (overfitting).

5.8 Residual Connections

Even though deeper networks may learn better hierarchical features, the presented many problems until Residual Connections appeared; Residual connections have helped deep networks learn in an optimal way. The connections enable the model to propagate lower-layer features to deeper layers, so that the model can learn from both. This is particularly useful for sequential recommendation. After several self-attention blocks all the embeddings are now entangled with the other embeddings of the sequence, but results have shown that the last item's embeddings of the sequence are important for the prediction. The Residual Connections help the model propagate those last item's embeddings to the final layers and help the prediction.

After each self-attention or point-wise feedforward network we apply these techniques, so that the output would be

$$g(x) = x + Dropout(g(LayerNorm(x)))$$

5.9 Prediction Layer

After the b self-attention and poin-wise feedforward blocks we obtain F^b matrix that contains the aggregated embeddings for each item in the sequence. In other words, we obtain a matrix of dimension nxd . To predict, use a Matrix Factorization technique and calculate the relevance of the item i being the next item to be consumed after consuming the past t items:

$$r_{i,t} = F_t^b N_i^T$$

To reduce the model's parameters and avoid overfitting, we used the same item embedding matrix mentioned at I. Basically, given a F_i we calculate the relevance with all the possible items and recommend the items with the highest relevance.

6 Parameter Analysis

The investigation focused primarily on the analysis of some parameters that are crucial for the working of the SASRec model, mainly the maximum sequence length and the number of stacked attention blocks. The selection of the maximum sequence length in the experiments are based on the presumption that longer sequences with deeper self-attention could result in better results since far apart item consumption patterns could be exploited.

closer to the average number of ratings per user yields better results on testing.

The only practical way of experimenting with parameters is trying the values in a grid, unless there is some plausible heuristic to search for the parameters, but this is not the case. Considering the processing power limitations in the investigation, the values for the parameters that were experimented with were:

Parameter	Values
Maximum Sequence length (n)	[50, 200]
Number of Attention Blocks	[2,4,6]

Table 1: Experiment Values

7 Results

Results can be separated into 2 sections. The first consists of baselines and initial SASRec results. The second in an exploration of results only within SASRec varying the parameters to find the optimal ones.

The main evaluation metrics are NDCG@10 and HitRate@10 these were chosen because in [9] they use these metrics and our investigation wanted to be comparable to the one from the original paper.

7.1 Baselines and SASRec Initial

The baseline comparison algorithms were selected from a pool the OpenRec library offers based on which were the better performing ones.

First, BPR (Bayesian Personalized Ranking) [7], a very common model to learn personalized rankings from implicit feedback.

Second, PMF (Probabilistic Matrix Factorization) [8], a model which works very well on large, sparse and very imbalanced datasets.

Third, RNNRec a model proposed by the creators of the Openrec library that uses Recurrent Neural Networks to model the sequence of users' items consumption.

Forth, VanillaYoutubeRec is a model that fits in the same category as the third and uses multi-layer perceptrons.

The SASRec model is set with the same parameters as [9] proposes.

The results for baselines algorithms and the initial SASRec can be seen in the following table:

Best Baseline (BPR): **NDCG@10: 0.3220 Hit@10: 0.5578**

Best SASRec (Max. Sequence Length = 200 and Num. Attentions Blocks = 2) **NDCG@10: 0.5170 Hit@10: 0.7784**

The best Parameters for SASRec show that a longer sequence length yields better results, while deeper self attention blocks over-fit more, leading to worse results.

7.2 SASRec Parameter Analysis

Since the above results of the SASRec model are with the same parameters as in [9], the investigation focused on tweaking some of the hyper parameters with hopes on finding better results that on [9]. The parameters that were modified and experimented on where 2. First, we modified the maximum length of the

HitRate@10	**n=50**	**n=200**
b = 2	0.7720	**0.7784**
b = 4	0.7460	0.7597
b = 6	0.7153	0.7266

Table 2: Test HitRate Analysis

sequence to predict (from the original 50 to 200). Second, we modified the number of attention blocks (with values 2, 4 and 6).

The following table show the results obtained after 1000 iterations:

The parameter combinations that generated the best results were 2 attention blocks and a maximum sequence length of 200. Our initial hypothesis that more attention blocks in conjunction with longer sequences to exploit far apart item consumption patterns was wrong. However, we were able to find a better performing model with longer maximum sequence lengths.

8 Conclusions

In this investigation, we propose a modification to the original SASRec model by tunning some of the parameters that are chosen in [9]. The results show that original paper's 2 attention blocks stack works well, but choosing a maximum sequence length similar to the average number of ratings per user gives better performance. The initial hypothesis of longer sequences and deeper attention blocks would yield better results gets refused.

A future related investigation could be to try this model with a bigger dataset to alleviate over-fitting and prove out hypothesis fully. Also, the model can be implemented by stacking the same self-attention block and point-wise feed-forward network instead of different ones. In other words, to pass the data several times through the same matrices in stead of learning new weights each time to make the attention block generalize better.

Another posible future work is trying with a bigger dataset, namely the 27 Million ratings version of the MovieLens dataset used. This is with the purpose of preventing over-fitting. One last future suggestion to make the prediction better is to filter the dataset with rating that are better than 3 stars in this way the system will only recommend items that were liked by the users and not just the most probable next item independently of it being a good or bad recommendation.

References

1. Y. Koren and R. Bell, "Advances in collaborative filtering," in Recommender Systems Handbook. Springer, 2011.

2. S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, "What your images reveal: Exploiting visual contents for point-of-interest recommendation," in WWW, 2017.

3. X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in WWW, 2017.

4. B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in ICLR, 2016.

5. 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.

6. D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in ICLR, 2015.

7. Steffen Rendle , Christoph Freudenthaler , Zeno Gantner , Lars Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, p.452-461, June 18-21, 2009, Montreal, Quebec, Canada

8. Ruslan Salakhutdinov , Andriy Mnih, Probabilistic Matrix Factorization, Proceedings of the 20th International Conference on Neural Information Processing Systems, p.1257-1264, December 03-06, 2007, Vancouver, British Columbia, Canada

9. Wang-Cheng Kang, Julian McAuley. 2018. Self-Attentive Sequential Recommendation. arXiv:1808.09781. Retrieved from

   https://arxiv.org/abs/1808.09781

10. F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages.

    DOI=http://dx.doi.org/10.1145/2827872