Sequential recommenders for MeLiDataChallenge 2020

Sarah Everke
Nro. alumno 1563759j
sveverke@uc.cl

Vicente Aguilera
Nro. alumno 16637860
vjaguilera@uc.cl

***Abstract*—MercadoLibre has provided a sequential recommendation challenge. They request a recommendation based on the modeling of users' interactions with the platform. In this paper we apply RNN models: GRU4Rec, a GRU-based RNN, and SASRec, a self-attention based sequential system. We also use MostPopular as a base line method. Finally, we prove that SASRec has a better performance and is faster than GRU4Rec.**

I. Introduction

MeLiDataChallenge is a recommendation challenge proposed by MercadoLibre, a well-known e-commerce and fintech platform in Latin America. MercadoLibre shows over 1.000.000 posts on products and services. Users navigate that content to buy new products or search for new ideas for their next purchase. [1] MercadoLibre submitted this challenge to study what may interest customers while they use their web page. So they provide a record of a user's browsing period and request a prediction of the ten products most likely to be chosen by the user's next purchase. This challenge is a session-based recommender problem since a user's session may be modeled as a sequence of events.

There are multiple studies for sequential recommendations. They provide a better user understanding because the series of interactions often reveal the development of users' interests models have been adapted to this recommendation setting. In this recommendation approach, the initial input of the RNN is the first event. Then each consecutive interaction will generate an output that depends on the past users' history [2]. We use GRU4Rec, an RNN based on multiple layers, and SASRec, a self-attention-based sequential system, to generate recommendations.

II. State of art

A. GRU4Rec

GRU4Rec is a GRU-based RNN for session-based recommendations. A GRU is an RNN unit that learns when and how much to update its hidden state [2]. The activation of a GRU corresponds to an interpolation between the last activation and the candidate activation $\hat{h_t}$ [2]:

$$\hat{h_t} = (1 - z_t)h_{t-1} + z\hat{h_t} \tag{1}$$

if the input of the unit at time t is x_t, then the update gate is given by:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{2}$$

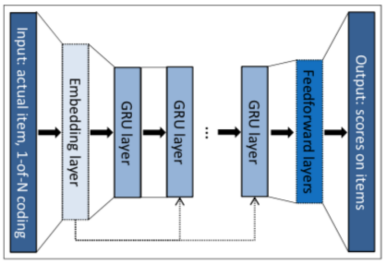

Fig. 1. Architecture of GRU4Rec [2]

In this model the input of the network is the actual state of a session. And the output corresponds to the item of the next event in the session. The architecture Fig.1 consists of multiple GRU layers and an optional feedforward between the last GRU layer and the output.

B. SASRec

SASRRec a self-attention-based sequential system for session-based recommendations. This model takes advantage of the attention mechanism that aims to capture relevant items in the user's past interaction, to predict the following product [3]. It receives as input a user's action sequence to predict the following item. While training, at time steps t, the model predicts the next item depending on the last t elements. The architecture of SASRec Fig.2 consists of an embedding layer, self-attention layer, point-wise feed-forward network, and a prediction layer.

III. Dataset

MercadoLibre provides two datasets [1]. The first dataset contains the buying history of 413.163 users. It specifies which item was bought and the events that lead to the purchase. There are 11.999.166 interactions and on average a user performs 29 events to buy an item. There are two types of events: view and search. While a view event shows the id of the product, a search event provides the query string. About half of the interactions are of the search type. The second dataset describes 2.102.277 products offered by the platform. It indicates the id, price, title, category, domain, and condition of each item.

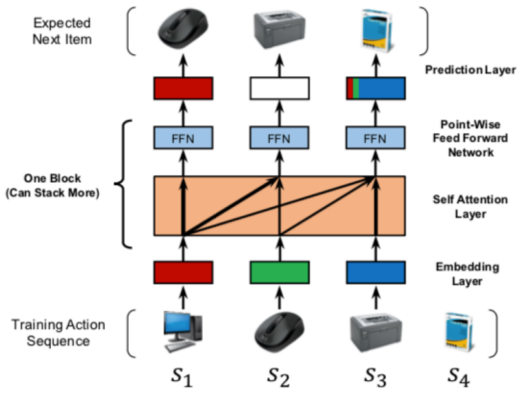

Fig. 2. Architecture of SASRec [2]

IV. Methodology

We used MostPopular as a baseline. The input of this model is the user id and the item purchased by that person. Since products are not ranked, we decided that if an item was purchased, then its ranking is 1.

Each RNN model used received the users´ history navigation. Each model used received the users‘ history navigation. The training set consists of 80% of the users‘ sessions. Meanwhile, the testing set corresponds to the remaining sessions. If the event was a view, the model received the products' id indicated in the dataset.But if it was a search event, there were two options:

1) Every search event is removed from the training and testing sets.
2) Every search event is eliminated only from the training set. If there is a search event in the testing set, then the model got the items' id with the most similar title to the search query. We use word2vec [4] to generate vectors for the item's titles in the dataset. Using k-means we generate 200 clusters. Then we get the cosine similarity between each vectorized query and every vectorized cluster centroid. Once the most similar cluster is computed, we calculate the most similar item inside that cluster.

V. Evaluation methodology

The metrics chosen to evaluate the models where:

- NDCG@10, which corresponds to the metric chosen by MercadoLibre to rank the contestants' submissions. This accuracy metric is useful because it privileges that relevant recommendations appear earlier in the recommendation set. It is calculated as:

$$\text{NDCG}_{10} = \frac{\text{DCG}_{10}}{\text{iDCG}_{10}} \tag{3}$$

- Recall@20, which represents the proportion of relevant items found in the top-20 recommendations. The formula is given by:

$$\text{recall} = \frac{|\text{relevant items} \cap \text{recommended items}|}{|\text{relevant items}|} \tag{4}$$

VI. Results

We used a Keras implementation for GRU4Rec [5], we created 2 models with diferent epochs (1 and 3) to check if this affects the performance in someway. We adapted a PyTorch implementation available in python2.7 [6] to run in python3.8 [7], created 2 diferent models to analyze the relevance of the maximum sequence length . Finally, we calculate MostPopular with pyreclab [8]. The results of the options are:

A. Testing without word2vec

TABLE I
Metrics

Model	Metrics		
	Recall@20	***NDCG@10***	***time***
MostPoular	0,02	0,06	<5 min.
GRU4Rec 1 epoch	0,24	-	2 hrs.
GRU4Rec 3 epochs	0,37	-	5 hrs. 48 min.
SASRec 50 maxlines	0,87	0,75	6 hrs. 45 min.
SASRec 200 maxlines	0,86	0,74	7 hrs. 35 min.

B. Testing with word2vec

The main objective of this experiment was to use all available data for training a model, and check if the search portion of the events can induce a different behavior, a better or worst performance. Due to time and resource limitations, we could not test the models with this methodology. But the implementation is available on our Github for you to make the predictions taking the searched query into account.

VII. Conclusions

In this work, we applied 2 of the most relevant sequential recommender systems in the MeLiDataChallenge context. We also check the outperformance of SASRec over GRU4Rec in both time and accuracy dimensions, in a specific context. In this way, we experiment with both models and find out that SASRec takes less time to train than GRU4Rec, and still have better results. This happens due to the efficiency of the self-attention component.

References

[1] MercadoLibre. Mercadolibre data challenge 2020. https://ml-challenge.mercadolibre.com/. Accessed: 2020-12-16.

[2] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

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

[4] TensorFlow. Word2vec. https://www.tensorflow.org/tutorials/text/word2vec. Accessed: 2020-12-16.

[5] paxcema. Gru4rec in keras. https://github.com/paxcema/KerasGRU4Rec. Accessed: 2020-12-16.
[6] pmixer. Sasrec.pytorch. https://github.com/pmixer/SASRec.pytorch. Accessed: 2020-12-16.
[7] pmixer. Sasrec.pytorch modified. https://github.com/vjaguilera/SASRec.pytorch. Accessed: 2020-12-16.
[8] gasevi. pyreclab: Recommendation lab for python. https://github.com/gasevi/pyreclab. Accessed: 2020-12-16.