Session based recommendation system on Spotify's music streaming sessions dataset

Yoav Navon
Pontificia Universidad Católica de Chile
Santiago, Chile
yanavon@uc.cl

Matías Andrade Guzmán
Pontificia Universidad Católica de Chile
Santiago, Chile
mandrade2@uc.cl

ABSTRACT

In this paper, we evaluate a model with the Spotify Music Sessions Data set, for a next item prediction task. In the session recommendation setting, it is only important the data in the current session, so past user data becomes irrelevant. We propose a model based on the GRU4REC model, that uses a RNN for session recomendation. We evaluate the model with different variations of the model, including content-based representations, and different types of loss functions. We find that the original GRU4REC performs well in the data set, outperforming more elaborated versions of the same model.

KEYWORDS

neural networks, session based recommendation, recommender systems

ACM Reference Format:
Yoav Navon and Matías Andrade Guzmán. 2019. Session based recommendation system on Spotify's music streaming sessions dataset . In . ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Nowadays, people hear a lot of music in platforms like Spotify or Youtube. For this platforms to introduce new music to it's users they have to perform a recommendation based on past behavior. This kinds of recommendation commonly use collaborative filtering or content based methods, but these methods fail to take advantage of short term listening patterns. Ignoring what the user is hearing now and taking it's hearing history as a whole.

In this paper, a session based recommendation system is going to be trained and tested on Spotify's Music Streaming Sessions Data set [1].

2 RELATED WORK

2.1 Session-Based Recommendations With Recurrent Neural Networks[3]

This model was the first one that leveraged the capacities of recurrent neural networks for item recommendation. It consists of a GRU cell that it's trained for session based recommendation. The original model was trained on a data set of 37.000 items, and used one-hot representations of items. Our proposed solution it's based on this model, with the difference of the input layer, in which we used embeddings and content-based vectors in addition of the one-hot representation. Moreover, our model is given the task of classification of 500.000 different items.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
, ,
© 2019 Association for Computing Machinery.
https://doi.org/10.1145/1122445.1122456

3 SOLUTION

As it was mentioned in section 2.1, our model is based on the work of [3], called GRU4REC. The original model was fed with one-hot representations of the items, and we expanded on this to use dense embeddings, and content based vectors; with the embedding layer shown in figure 1.

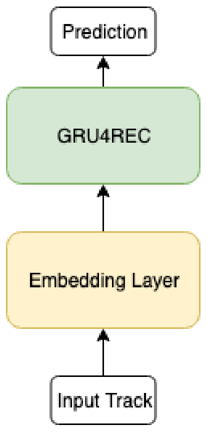

Figure 1: Block model of proposed solution

3.1 Embedding Layer

We used a few variations of the embedding layer, which are explained below.

- **One-Hot**: This is the original input layer in [3], it's the one-hot representation of the track number. (GRU4REC one-hot)
- **Embedding**: Consists on a look up embedding layer before the GRU cell, where each track is mapped to a **d** dimensional vector. The weights of the layer are optimized in training for good latent representations of the tracks. (GRU4REC embedding)
- **Content-Based**: The data set utilized had 28 features of metadata for each track. For this layer we standardize the features, and fed the model a 28-dimensional vector. (GRU4REC content-based)

- **MLP** This is an extension of the content-based method. The 28 features in the metadata might not be equally relevant for recommendation, so we let the model optimize a dense representation based on the features. To accomplish this, we added a MLP of 2 layers and 64 units each, and we gave it the content based vector. This way, the GRU cell is given the resulting 64-dimensional vector from the MLP. (GRU4REC content-mlp)

3.2 Loss Function

A good loss function is fundamental for a good performance in any Machine Learning task. For this reason, we used 3 different loss functions to optimize our model in next item prediction.

- **Cross Entropy**: This method treats the task as a classification problem between all the tracks in the data set. It's commonly used in multi-class classification problems.
- **BPR**: This loss function is proposed in [3], based in the original BPR paper [4]. It is defined as follows, where N is the number of samples, r_i is a positive sample, r_j is a negative one, and σ is the sigmoid function.

$$L_{bpr} = -\frac{1}{N}\sum_{j=1}^{N}\log\sigma(r_i - r_j)$$

As noted in [4] the negative sample method is crucial for a good performance of the function. In this case, we used as negative samples the other items in the batch, assuming that they are indeed negative items (we can make the assumption because of the sparsity of the data).

- **BPR-max**: In [2], the authors noted a few flaws of the BPR function L_{bpr}. The problem is that most of the samples are irrelevant, so $r_i \gg r_j$, adding nothing to the total gradient. Nevertheless, the gradient it's always discounted by the total number of negative samples ($\frac{1}{N}$ factor), so the gradient vanishes. To account for this, in [2] it's proposed a correction of BPR to fix this issue.

$$L_{bpr-max} = -\log\frac{1}{N}\sum_{j=1}^{N}s_j\sigma(r_i - r_j)$$

The parameters are the same, but the coefficient s_j it's the value of a softmax between all the negative samples. This way, bad negative samples have no effect in the overall loss function.

4 DATA SET

The complete data set consists of 150.000.000 listening sessions on the spotify platform which we'll call the Full Data set. We created a sub sample of this data set, to make the training feasible. This sub sample consisted in 1/330 of the Full Data set.

	Events	**Tracks**	**Sessions**
Full Data set	2,250,000,000	3,700,000	150,000,000
Sub Sample	6,066,614	505,195	365,448

Table 1: Data set Statistics

For each session in the data set, there's a minimum of 10 events and a maximum of 20. We checked that the events are contiguous in the data set. The data set is highly sparse, where 79% of the tracks, only appear 10 times or less.

5 METHODOLOGY

The model was trained in the dataset shown in the previous section (Sub Sample). Following [3], the model was trained using session-parallel mini-batches to feed the model multiple sessions at the same time.

The Spotify Dataset was designed for the Skip Prediction Challenge, a different task than the next item prediction that we want to perform. Because of this, there isn't any baseline in this particular dataset, hence we implemented the Most Popular baseline to validate all the results of our model. Most Popular it's a common non-personalized baseline in recommender systems, it's simple but some models struggle to beat it's results.

For all the different variants of the model, we implemented a grid search approach to find the best parameters. In the Results section it is show the best model for each variant.

After training the model, we evaluated it in another sub sample of the Full Data set, with the same size of the training sample. We filtered the tracks that didn't appear in training and then obtained some metrics. To measure the performance of our models, we employed two common metrics: Recall@20 and MRR@20 (Mean Reciprocal Rank).

6 PARAMETER TUNING

The proposed model has different variants, but the main variables that we tuned were the type of input (one-hot, embedding, content-based, content-mlp), loss function (cross-entropy, bpr, bpr-max), regularization, learning rate and batch size. For the case of learning rate and batch size, for all models the best results were obtained with values of 0.0001 and 64 respectively. Moreover, the weights regularization resulted in negative improvements, so it was removed from all layers.

7 RESULTS

In table 2 are shown the result of the best version of each model. Surprisingly, the most simple version of the model (GRU4REC one-hot) got the best results.

Another insight, is that the bpr loss function performs rather poorly with respect to cross-entropy. Nevertheless, we can generally observe that bpr-max is competitive with cross-entropy. That validates the claims of the authors in [2], and the use of bpr should come with a good negative sampling strategy.

We can see that the content-mlp outperforms the pure content-based version for cross-entropy and bpr-max, showing that assigning equal relevance to every feature in the metadata was wrong.

8 CONCLUSIONS

First of all, should be noted that the GRU4REC model performed worst that it was shown in [3]. That was expected, as the number of items is a huge factor in item recommendations. It wouldn't be possible to use this model for the Full Data set, because the increase in items would result in poor results. Probably a two stage model

	cross-entropy		**bpr**		**bpr-max**	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
Most Popular	0.0696	0.0095	0.0696	0.0095	0.0696	0.0095
GRU4REC one-hot	0.24	0.024	0.1278	0.0170	0.2132	0.0205
GRU4REC embedding	0.2273	0.023	0.1017	0.0130	0.1293	0.0126
GRU4REC content-based	0.1533	0.0177	0.0707	0.0095	0.1045	0.0110
GRU4REC content-mlp	0.1640	0.0191	0.0532	0.0082	0.1101	0.0119

Table 2: The model results on all its variants

would be needed, in a way that you first reduce the number of candidates, and then perform the recommendation; this method is explored in [6].

Nevertheless, the model comfortably outperform the Most Popular Baseline, validating it's potential as a personalized recommender system.

Finally, we argue that the metadata provided by Spotify wasn't specially helpful, and the most important part of the data set are the session interactions. We argue that 28 features are not enough to perform content-based recommendation. Of the 28 features, 8 were acoustic vector components extracted with deep learning methods[5]. Even though those acoustic vector where 64-dimensional, we were only given 8 components for each track vector. A good content-based method would need the full set of data.

REFERENCES

[1] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. 2019. The Music Streaming Sessions Dataset. In *The World Wide Web Conference*. ACM, 2594–2600.

[2] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 843–852.

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

[4] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[5] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in neural information processing systems*. 2643–2651.

[6] Lin Zhu and Yihong Chen. 2019. Session-based Sequential Skip Prediction via Recurrent Neural Networks. *arXiv preprint arXiv:1902.04743* (2019).