
Shuai Zhang\textsuperscript{1}, Yi Tay\textsuperscript{2}, Lina Yao\textsuperscript{1}, Bin Wu\textsuperscript{3} and Aixin Sun\textsuperscript{2}

\textsuperscript{1}University of New South Wales
\textsuperscript{2}Nanyang Technological University
\textsuperscript{3}Zhengzhou University

\{shuai.zhang@student., lina.yao@\}unsw.edu.au,
\{ytay017@e., axsun@\}ntu.edu.sg, wubin@gs.zzu.edu.cn

Abstract

Deep learning based recommender systems have been extensively explored in recent years. However, the large number of models proposed each year poses a big challenge for both researchers and practitioners in reproducing the results for further comparisons. Although a portion of papers provides source code, they adopted different programming languages or different deep learning packages, which also raises the bar in grasping the ideas. To alleviate this problem, we released the open source project: DeepRec. In this toolkit, we have implemented a number of deep learning based recommendation algorithms using Python and the widely used deep learning package - Tensorflow. Three major recommendation scenarios: rating prediction, top-N recommendation (item ranking) and sequential recommendation were considered. Meanwhile, DeepRec maintains good modularity and extensibility to easily incorporate new models into the framework. It is distributed under the terms of the GNU General Public License. The source code is available at github: https://github.com/cheungdaven/DeepRec

1 Introduction

In recent years, deep learning has achieved tremendous success in many application domains, such as computer vision, natural language processing, and speech recognition. Inspired by the effectiveness of deep neural networks, researchers and practitioners started to apply deep learning techniques to recommendation tasks and have achieved a huge improvement on many recommendation tasks. Meanwhile, the number of publications on deep neural network based recommendation models also increases exponentially [Zhang et al., 2019]. Moreover, many companies resort to deep learning techniques to improve their recommendation quality for better user experience and increasing revenue [Covington et al., 2016; Cheng et al., 2016]. In general, deep learning techniques can be utilized to learn representations for user and item side information (e.g., user profile, item content, etc.), introduce nonlinearities to recommendation models and model the sequential patterns of user consumption history. In addition, deep neural networks are of high flexibility, which makes it possible to develop more powerful models by combining different neural networks.

However, this hype also poses a big challenge to researchers and practitioners, that it, reproducing results for those proposed deep learning based models. Although some authors provide source code for reproducibility, many of them are written in different programming languages, or use different deep learning packages, let alone a large portion of work do not provide source code, which makes it hard to understand and hinder the process of justifying newly proposed methods. To this end, we start the project - DeepRec. Our goal is to make reproducing results easier and lower the bar of developing deep learning based recommendation models and increase their practicality.

Although there are some open source recommendation packages available, such as MyMediaLite\textsuperscript{1}, LibRec [Guo et al., 2015], Surprise\textsuperscript{2} and OpenRec [Yang et al., 2018], most of them only include traditional recommendation algorithms or have not updated for a long time. The OpenRec library do provide several deep neural network based recommendation models, the number is still very small (only four models implemented) compared with DeepRec.

To this end, we develop an open-source toolkit named ‘DeepRec’. This toolkit provides a flexible framework and unified interfaces for deep learning based recommendation models. The prime features of DeepRec are twofold:

- In this toolkit, we addressed three common scenarios in recommendation research: rating prediction, top-n recommendation (item ranking) and sequence-aware recommendation. It offers state-of-the-art algorithms for those three tasks.

- DeepRec is extensible, adaptable and efficient. Users could add algorithm easily. Moreover, DeepRec is able to make use of the computation power of GPUs, making it scalable to large-scale datasets.

\textsuperscript{1}http://mymedialite.net/index.html
\textsuperscript{2}http://surpriselib.com/
2 Introduction to DeepRec

In this section, we briefly introduce the architecture of the DeepRec, plus implemented recommendation algorithms and comparison with other frameworks.

2.1 Architecture of DeepRec

DeepRec is made up of three components: recommender, backend and utility. We adopt the widely used deep learning framework Tensorflow and several other frequent used scientific calculation libraries such as Numpy, Scipy, Sklearn and Pandas as the core backend; A number of representative and advanced algorithms deep learning based recommendation models forms the recommender components. We split the algorithms into three categories: rating prediction, top-n recommendation and sequence-aware recommendation. We put the common evaluators and data preprocess functions into the utility component.

2.2 Implemented Algorithms

A number of algorithms that were implemented in the current version are listed below. Note that a few non deep learning methods (e.g., biasedSVD, Factorization Machines, BPRMF) are also implemented with Tensorflow.

Rating Prediction Models

In this category, we implemented methods like classical method like biasedSVD (or MF) [Koren et al., 2009], Factorization Machines (FM) [Rendle, 2010], user and item based AutoRec [Sedhain et al., 2015], neural network matrix factorization (NNMF) [Dziugaite and Roy, 2015], neural rating regression (NRR) [Li et al., 2017], neural factorization machines (NFM) [He and Chua, 2017] and attentive factorization machines (AFM) [Xiao et al., 2017]. For factorization machines, we use the same feature format as libfm [Rendle, 2012] to make them compatible.

Top-n Recommendation Models

Here, we included classical pairwise personalized ranking methods BPRMF [Rendle et al., 2009], collaborative metric learning (CML) [Hsien et al., 2017], collaborative denoising auto-Encoder (CDAE) [Wu et al., 2016], multilayer-perceptron (MLP), generalized matrix factorization (GMF) and neural collaborative filtering (NeuMF) [He et al., 2017], NeuRec [Zhang et al., 2018b] and latent relational metric learning [Tay et al., 2018].

Sequential Recommendation Models

For sequence-aware recommendation models, we implemented the personalized ranking metric embedding model (PRME) [Feng et al., 2015], LSTM and GRU based methods [Hidasi et al., 2015], convolution based model (Caser) [Tang and Wang, 2018], and self-attention based approach (AttRec) [Zhang et al., 2018a].

2.3 Evaluation Metrics

Several commonly used evaluation metrics were included in this toolkit. For rating prediction, we can use Root Mean Square Error (RMSE) and Mean Average Error (MAE) to measure the effectiveness. For top-n recommendation and sequence-aware recommendation, we included several ranking-aware metrics, including, Recall@n, Precision@n, Normalized Discounted Cumulative Gain (NDCG) and Mean Reciprocal Rank (MRR) to measure both the accuracy and quality of ranking lists. For detailed definitions, readers are referred to [Shani and Gunawardana, 2011].

2.4 Comparison with Other Frameworks

<table>
<thead>
<tr>
<th>Library</th>
<th>backend</th>
<th>GPUs</th>
<th>3 scenarios</th>
<th>#DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suprise</td>
<td>SgKit</td>
<td>N</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>MyMediaLite</td>
<td>N/A</td>
<td>N</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>LibRec</td>
<td>N/A</td>
<td>N</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>OpenRec</td>
<td>Tensorflow</td>
<td>Y</td>
<td>N</td>
<td>4</td>
</tr>
<tr>
<td>DeepRec</td>
<td>Tensorflow</td>
<td>Y</td>
<td>Y</td>
<td>&gt;20</td>
</tr>
</tbody>
</table>

Table 1: Comparison with existing libraries. “3 scenarios” means whether the library can deal with the above mentioned three different tasks. “#DL” means the number of neural networks based methods.
References


[Feng et al., 2015] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. Personalized ranking metric embedding for next new poi recommendation. In Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.


