BETA-Rec: Build, Evaluate and Tune Automated Recommender Systems

Zaiqiao Meng
Richard McCreadie
Craig Macdonald
Iadh Ounis
Siwei Liu
Yaxiong Wu
Xi Wang
zaiqiao.meng@gmail.com
University of Glasgow

Shangsong Liang
Yucheng Liang
Guangtao Zeng
Junhua Liang
Sun Yat-sen Univeristy

Qiang Zhang
University College London

ABSTRACT

The field of recommender systems has rapidly evolved over the last few years, with significant advances made due to the in-flux of deep learning techniques. However, as a result of this rapid progress, escalating barriers-to-entry for new researchers is emerging. In particular, state-of-the-art approaches have fragmented into a large number of code-bases, often requiring different input formats, pre-processing stages and evaluating with different metric packages. Hence, it is time-consuming for new researchers to reach the point of having both an effective baseline set and a sound comparative environment. As a step towards elevating this problem, we have developed BETA-Rec, an open source project for Building, Evaluating and Tuning Automated Recommender Systems. BETA-Rec aims to provide a practical data toolkit for building end-to-end recommendation systems in a standardized way. It provides means for dataset preparation and splitting using common strategies, a generalized model engine for implementing recommender models using Pytorch with 9 models available out-of-the-box, as well as a unified training, validation, tuning and testing pipeline. Furthermore, BETA-Rec is designed to be both modular and extensible, enabling new models to be quickly added to the framework. It is deployable in a wide range of environments via pre-built docker containers and supports distributed parameter tuning using Ray. In this demo, we will illustrate the deployment and use of BETA-Rec for researchers and practitioners on a number of standard recommendation datasets. The source code of the project is available at github: https://github.com/beta-team/beta-recsys.

CCS CONCEPTS

• **Information systems → Collaborative filtering**.

KEYWORDS

Recommender Systems, Framework, Open-source, Toolkit

ACM Reference Format:
Zaiqiao Meng, Richard McCreadie, Craig Macdonald, Iadh Ounis, Siwei Liu, Yaxiong Wu, Xi Wang, Shangsong Liang, Yucheng Liang, Guangtao Zeng, Junhua Liang, and Qiang Zhang. 2020. BETA-Rec: Build, Evaluate and Tune Automated Recommender Systems. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3383313.3411524

1 INTRODUCTION

Recommender systems that suggest items of interest to users based on available information such as purchases and interactions histories have been the subject of intensive research by both industry and academia in recent years [3–5, 12]. In particular, deep learning techniques have achieved tremendous success in recommender systems, leading to large and significant improvements in performance being reported [3, 4, 12].

However, partially as a result of this intensive and rapid progress, there are now many barriers-to-entry in the recommendation field for new researchers, as well as increasing challenges when comparing different works from the literature. In particular, current challenges include: 1) the implementations are fragmented across different code repositories and often do not function out-of-the-box; 2) the reported performances across works are often not comparable even when reported on the same dataset and with the same metrics;[1] 3) the usage of different evaluation toolkits is problematic as subtle differences in metric implementations lead to different reported performances; 4) the inconsistency among various tuning strategies of models (notably the omission of baseline tuning) leads to unfair comparisons. Indeed, recent works [1, 6, 7, 11] have demonstrated that, without properly tuning model hyperparemeters, existing state-of-the-art deep learning baselines cannot even consistently outperform a non-neural linear ranking models. Hence, there is a clear need for platforms that provide a common recommendation methodology and pipeline, enabling model comparison across datasets, metrics, and baselines in a sound and fair manner.

[1]Since data pre-processing and splitting techniques differ each other.

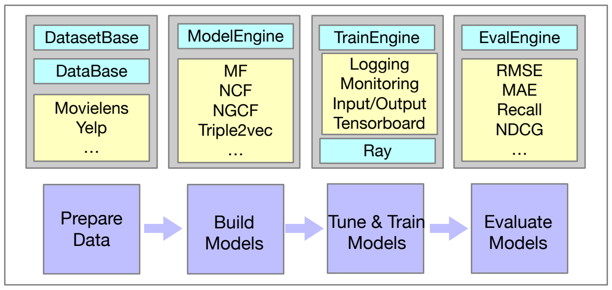

Figure 1: The architecture and workflow of the BETA-Rec project.

To date, a range of platforms have been proposed and developed, including Spotlight [2], Microsoft-Recommenders [3], DeepRec [11] [4], OpenRec [10] and Cornac [5]. However, while these works have greatly aided in making the field more accessible, they are also to blame for some of the challenges summarized earlier, as these platforms are not strongly opinionated about the recommendation methodology and pipeline. For instance, the popular Microsoft-Recommenders platform provides no common framework, resulting in each contained model implementing their own data preparation process. By leaving these aspects at the user's discretion and providing little in the way of guidance/best practices to follow, which means that poor and/or inconsistent methodological choices are sadly commonplace.

Hence, we propose a new platform named **BETA-Rec**, which is a unified platform for building, evaluating and tuning automatic recommender systems, so as to unify the recommendation methodology and pipeline. The primary features of our BETA-Rec are the following:

(1) Contains a convenient and reusable dataset preparing toolkit for processing raw datasets in a standardized way.
(2) Provides a unified framework for training models, monitoring the training processes, as well as validation and testing of the resultant models.
(3) Provides 9 recommendation models that can be used out-of-the-box.
(4) Supports containerized deployment for use in different environments.
(5) Integrates the Ray [6] hyperparameter tuning library.

2 BETA-REC ARCHITECTURE

BETA-Rec provides an end-to-end workflow for researchers and practitioners to build their new models or use the built-in models. It also provides a standardized way to configure model training and evaluate the resultant models under a unified framework. Figure 1 provides the overview of the BETA-Rec architecture, which highlights the major components of the platform. We summarize each of the four main stages as below:

Prepare Data: To make the workflow efficient, we implement two key reusable components for preparing training data for different recommender models. The DatasetBase component provides unified interfaces for processing the raw dataset into interactions and splitting it using common strategies (e.g. leave-one-out, random split or temporal split) into training/validation/testing sets. Meanwhile the DataBase provides the tools to further convert the resultant data sets into usable data structures (e.g. tensors with $< user, item, rating >$ or $< user, positive\_item, negative\_item(s) >$), dependant on the requirements/supported features of the target model. Out-of-the-box we support a number of commonly used datasets, including Movielens_100k, Movielens_1m, Movielens_25m, LastFM. [7]

Build Models: Our project provides a model engine (i.e. represented by the class ModelEngine) for conveniently building a pytorch recommender model in a unified manner. In particular, it provides a unified implementation for saving and loading models, specifying the compute device, optimizer and loss (e.g. BPR loss [5] or BCE loss [3]) to use during training. Out-of-the-box, 9 recommendation models are provided, including classic baselines like MF [6], as well as more advanced neural models such as NCF [3], NGCF [9] and Triple2vec [8].

Train & Tune Models: The TrainEngine component provides unified mechanisms to manage the end-to-end training process. This encompasses: loading the configuration; loading the data; training each epoch; calculating validation performance; checkpointing models; testing early stopping criteria; and calculating the final test performance. The TrainEngine also supports monitoring/visualizing the training progress in real time, including resource consumption and training metrics (such as the training loss and evaluation performance on both the validation and testing sets) of a deployed model via Tensorboard. It can also expose these real-time metrics to a Prometheus time-series data store via an in-built Prometheus exporter, enabling programmatic access to the training state. To support easier and faster hyperparameter tuning for each model, we also integrate the Ray framework [8], which is a Python library for model training at scale. This enables the distribution of model training/tuning across multiple gpus and/or compute nodes.

Evaluate Performance: Three categories of commonly used evaluation metrics for recommender system are included in this platform, namely rating metrics, ranking metrics and classification metrics. For rating metrics, we use Root Mean Square Error (RMSE), R Squared (R2) and Mean Average Error (MAE) to measure the effectiveness. For ranking metrics, we include Recall, Precision, Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP) to measure performance of ranking lists. Model evaluation using build-in classification metrics like Area-Under-Curve (AUC) and Logistic loss are also supported. For detailed definitions of these metrics, readers are referred to [2]. To accelerate the evaluation process, the metric implementations are multi-threaded.

3 CONCLUSION

In this demo, we have presented BETA-Rec, an open source project for Building, Evaluating and Tuning Automated Recommender Systems. We illustrate how to use the new BETA-Rec platform on a (remote) distributed cluster of machines. In particular, we demonstrate the downloading and set-up of the engine, data preparation

[2]https://github.com/maciejkula/spotlight
[3]https://github.com/microsoft/recommenders
[4]https://github.com/cheungdaven/DeepRec
[5]https://github.com/PreferredAI/cornac
[6]https://github.com/ray-project/ray
[7]https://grouplens.org/datasets/.
[8]https://github.com/ray-project/ray

configuration, as well as launching a distributed tuning job for a neural recommender model. We also illustrate how the learning process can be monitored in real-time.

Acknowledgements

The research leading to this project has received funding from the European Community's Horizon 2020 research and innovation programme under grant agreement n° 779747.

REFERENCES

[1] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *RecSys*. 101–109.

[2] Asela Gunawardana and Guy Shani. 2009. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research* 10, Dec (2009), 2935–2962.

[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[4] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *Comput. Surveys* 51, 4 (2018), 66.

[5] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.

[6] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. *arXiv:2005.09683* (2020).

[7] Steffen Rendle, Li Zhang, and Yehuda Koren. 2019. On the difficulty of evaluating baselines: A study on recommender systems. *arXiv preprint arXiv:1905.01395* (2019).

[8] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. 2018. Representing and Recommending Shopping Baskets with Complementarity, Compatibility and Loyalty. In *CIKM*. 1133–1142.

[9] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.

[10] Longqi Yang, Eugene Bagdasaryan, Joshua Gruenstein, Cheng-Kang Hsieh, and Deborah Estrin. 2018. Openrec: A modular framework for extensible and adaptable recommendation algorithms. In *WSDM*. 664–672.

[11] Shuai Zhang, Yi Tay, Lina Yao, Bin Wu, and Aixin Sun. 2019. Deeprec: An open-source toolkit for deep learning based recommendation. *arXiv preprint arXiv:1905.10536* (2019).

[12] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *Comput. Surveys* 52, 1 (2019), 1–38.