RecBoard: A Web-based Platform for Recommendation System Research and Development

Mohit Chawla*
Cornell Tech, Cornell University
New York, New York
mc2683@cornell.edu

Kriti Singh*
Cornell Tech, Cornell University
New York, New York
ks2259@cornell.edu

Longqi Yang
Cornell Tech, Cornell University
New York, New York
ylongqi@cs.cornell.edu

Deborah Estrin
Cornell Tech, Cornell University
New York, New York
destrin@cornell.edu

ABSTRACT

This paper introduces **RecBoard**, a unified web-based platform that facilitates researchers and practitioners to train, test, deploy, and monitor recommendation systems. RecBoard streamlines the end-to-end process of building recommendation systems by providing a collaborative user interface that automates repetitive tasks related to dataset management, model training, visualization, deployments, and monitoring. Our demo prototype demonstrates how RecBoard can empower common tasks in research and development. RecBoard will be open-sourced and publicly available upon publication.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender Systems; Machine Learning; Web Platforms

ACM Reference Format:
Mohit Chawla, Kriti Singh, Longqi Yang, and Deborah Estrin. 2019. RecBoard: A Web-based Platform for Recommendation System Research and Development. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3308558.3314133

1 INTRODUCTION

To innovate on and deploy recommender systems, researchers often need to perform a series of repetitive tasks that involve dataset loading, dataset management, model monitoring, visualization, and deployment, as shown in Fig. 1. These repetitive tasks lead to significant overhead for model development and maintenance.

However, existing platforms are limited in addressing this overhead. On one hand, many solutions are focused on limited stages of the development pipeline (Fig. 1). For example, OpenRec [12] supports systematic model training, TensorFlow serving [9] and Clipper [1] enable model deployment in production, and AutoML [3] automates the process of searching for model configurations. On the other hand, existing systems that attempt to consolidate different development stages have a steep learning curve and are restricted to limited use cases — BigML [5] does not offer any recommendation model; Amazon's SageMaker [4] offers a train-test-deploy pipeline only in a hosted notebook environment, and MLBase [6] requires using a task language to declare machine learning tasks and does not provide deployment supports.

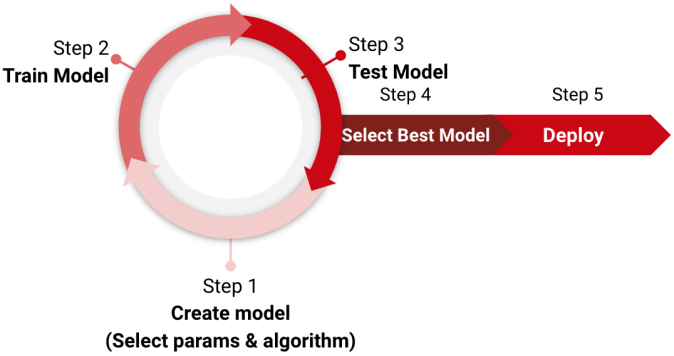

Figure 1: End-to-end process of developing recommender systems. It comprises of various stages: selecting parameters and algorithm, training the model, testing and selecting the best model, and deploying the best model into production.

In this paper, we develop **RecBoard**, an open-sourced and web-based platform that eases the end-to-end process of creating, testing, deploying, and monitoring recommendation models. It unifies five tasks in productionizing a recommendation system: (1) selecting dataset, model and model parameters, (2) training and testing the model, (3) visualizing the training progress (e.g., loss and precision) in real time, (4) collaborating with peers, (5) deploying the model, and (6) monitoring the deployed model. RecBoard is intended for researchers who aspire to create or productionize recommendation systems while collaborating with their peers.

*Both authors contributed equally.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3314133

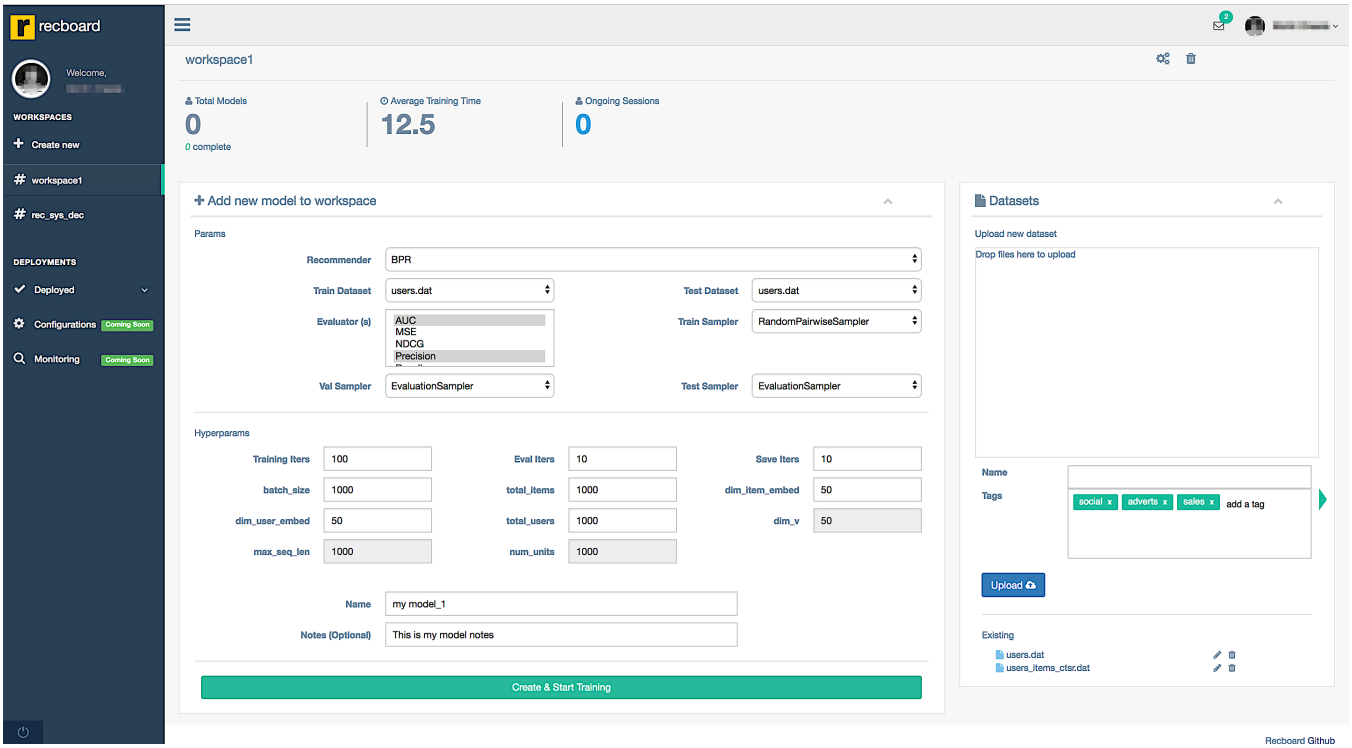

Figure 2: Sample user's workspace. A workspace encapsulates all related datasets and models.

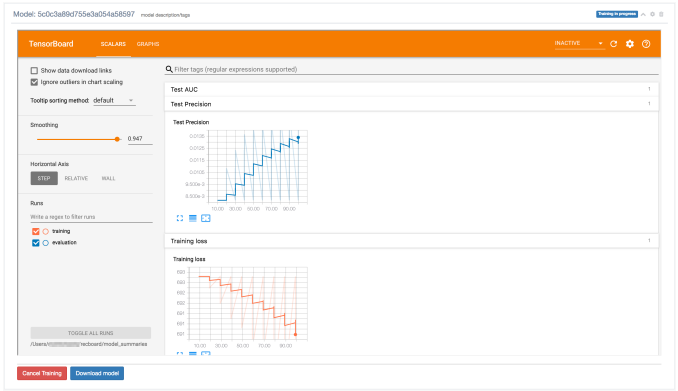

Figure 3: Real-time model variables visualization during training and testing using TensorBoard.

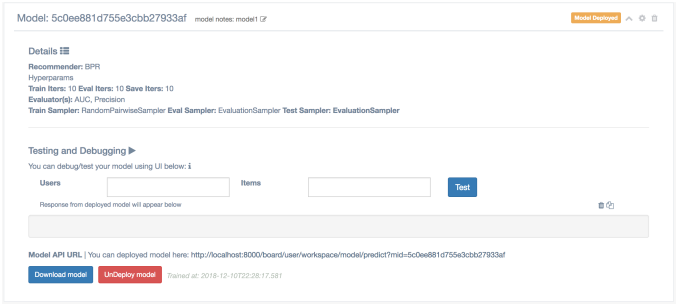

Figure 4: UI (with testing) for a deployed model

2 SYSTEM FEATURES

RecBoard eliminates operational inefficiencies across all stages of productionizing recommendation models. In this section, we present its major functionalities and features.

2.1 Training

RecBoard facilitates users to train models using the user interface as shown in Fig. 2 and enables them to monitor training progress (e.g., loss, precision, and recall) in real-time, as shown in Fig. 3. Since our core training library, OpenRec [12] is based on TensorFlow, we use TensorBoard [11] as the default visualizer. RecBoard allows users to plug in their custom visualization solutions.

2.2 Deployment

RecBoard allows users to deploy a model as a REST API using TensorFlow Serving [9]. This leads to fast, automated and easy to manage deployments. The *Deployment Manager* can be configured to use other deployment solutions, such as Clipper [1].

2.3 Monitoring

RecBoard enables users to monitor metrics and keep track of a model, which is deployed as a REST API. For debugging purposes, an input form is provided, and it enables the user to quickly test if the API of the deployed model is working, as shown in Fig. 4.

2.4 Collaboration

RecBoard enables users to collaborate via features such as dataset sharing, model visualization sharing (Fig. 5).

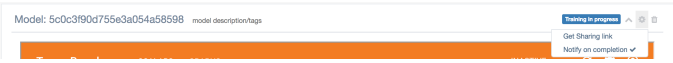

Figure 5: Sharing a model's visualization with peers

2.5 Management

RecBoard provides better model management by introducing workspaces, and model and dataset sharing and tagging.

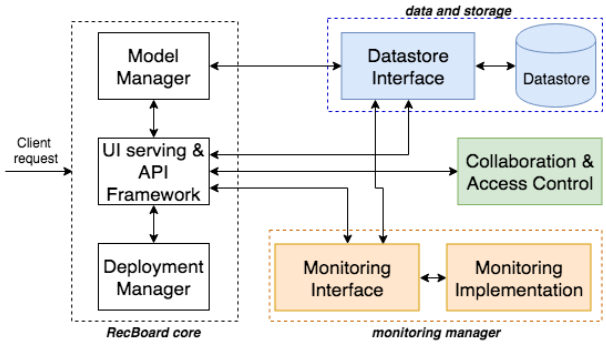

Figure 6: The architecture of RecBoard. All components are designed to automate repetitive tasks in different stages of developing recommender systems.

3 ARCHITECTURE OVERVIEW

The architecture of RecBoard is shown in Fig. 6, which highlights the major components of the system. RecBoard views the process of developing recommender systems as a multi-stage pipeline (Fig. 1) and provides user interfaces to perform tasks relevant to each stage of the pipeline. The tasks include managing datasets, training and testing model(s), organizing models into workspace(s), deploying model(s), and monitoring deployed models. *RecBoard core* is the central component that handles model management (via *Model Manager*), serves the user interface and handles model deployment.

3.1 Modular Design

To make the pipeline efficient, we define reusable components by identifying repetitive tasks and clustering those tasks, as shown in Fig. 7. For example, *Model Manager* automates repetitive tasks such as loading datasets, initializing model, and starting training. Furthermore, RecBoard is modularised in a way that its components can be replaced by custom modules by advanced users. For example, a different database can be connected easily and custom monitoring solutions can be added. Since each model training task is independent of others and is executed outside of the HTTP request-response cycle, we leverage distributed task queues to parallelize model training related operations. A task queue handles invoking

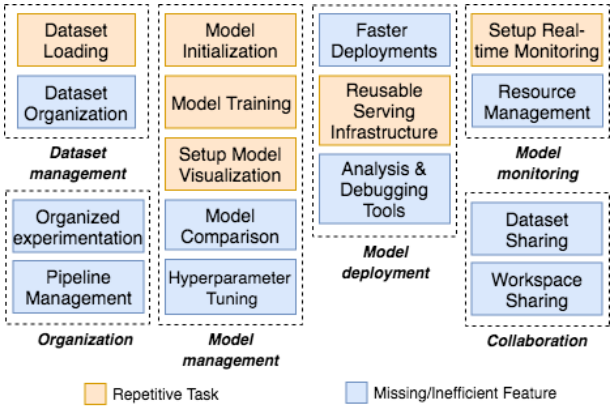

Figure 7: There are a number of repetitive tasks in end-to-end process and some missing features in existing platforms. These can be clustered into reusable components.

code that trains the model, processes results and stores them in a persistent database for later use. This architecture supports an on-premise solution so that users with constraints around data sharing can use RecBoard.

3.2 Components

RecBoard core consists of Model Manager, Deployment Manager and UI Serving & API framework. For a request from client, **UI Serving & API framework** interacts with other components of system and serves the user interface. **Model Manager** provides mechanisms to manage all model related tasks such as training, visualization, testing, management (create, delete, and edit), and download. **Deployment Manager** provides infrastructure for deploying (and un-deploying) a trained model and managing cloud configurations. **Monitoring Manager** provides infrastructure support for monitoring metrics (e.g., requests and up-time) of a deployed model via a monitoring interface. Mechanisms for monitoring are provided by a default implementation. The **Data and storage** component is part of persistence layer and provides an interface which can be implemented by any database wrapper. We used MongoDB in the initial prototype. The **Collaboration and access control** component provides infrastructure for collaboration via features such as: sharing training visualization and sharing datasets.

4 SAMPLE USER WORKFLOWS/EXPERIENCE

This section describes examples of user experience for different tasks performed via RecBoard.

4.1 Developing Recommendation Systems for Online Supermarket

A researcher R_1 at an online supermarket company wants to create a recommender system for clothes and deploy it as a REST API so that other teams in the company can use it to provide clothes' recommendations to users. She opens RecBoard and creates a workspace named "*Clothes Experiments*" to organize all recommender models related to clothes. She splits a dataset into training, validation and testing, uploads them to RecBoard, and adds tags (clothes,

train/test/eval) to them. She decides to create and deploy a Probabilistic Matrix Factorization (PMF) [8] based model. For experimentation, she chooses 3 sets of hyper-parameters to test upon and deploy the best of these. Using the *Add Model to Workspace* Section (Fig. 2), she selects a recommender and other hyper-parameters to add 3 models to the workspace. Now she can monitor key training indicators (e.g., loss, precision, and recall) of these 3 models in real time. After some time, all models are trained and the Model 2's results are the best. She decides to deploy it. To do so, she adds configurations for deployment corresponding to a cloud provider. Next, she clicks the *Deploy* button to deploy the model. After the model is deployed, she sees model testing UI along with API URL (Fig. 4) and decides to test if API is working. To test, she wants to know the output of the model for users with ids 2, 90 for items with ids 3, 81. So, she enters these ids and clicks the *TestAPI* button. She sees responses from the server and verifies that the API is working fine. So, she shares the API URL with other teams.

4.2 Collaborating on Recommendation Systems Development

R_1's team member is a recommender systems developer, R_2. R_2 proposes to experiment with recommenders other than PMF (deployed already). So, R_1 wants to collaborate with R_2. To do so, R_1 shares the "*Clothes Experiments*" workspace (created earlier) using the *Share Workspace* button and entering R_2's email. Now, R_2 has access to the workspace and she creates other models based on Bayesian Personalized Ranking (BPR). She trains new models and finds that, say, one of her models performs better than the model deployed earlier. So, she decides to deploy this model and update other teams with new API URL. Once other teams migrate to this new API, she undeploys previous model using *Undeploy* button (Fig. 4).

5 EVALUATION

In this section, we evaluate RecBoard in terms of efficiency and present examples to demonstrate how users can customize modules and add new recommendation algorithms as per their use case.

5.1 Efficiency: RecBoard as a platform for efficient and organized experimentation

Previously, the end-to-end process was inefficient because it involved writing multiple lines of glue code for repetitive tasks (shown in orange in Fig. 7) and manually setting up pipelines for commonly needed features (shown in blue in Fig. 7). Now, the user can perform the same tasks by a few clicks in the UI, which leads to increased efficiency.

5.2 Extensibility and Maintainability: Adapting RecBoard by customizing existing implementation

It is important that the existing implementation is extensible because users might need additional features and custom implementations of modules. RecBoard provides ways to adapt the platform to custom use cases. For example, additional recommendation algorithms can be added by adding a file similar to the ones in the *recommenders* folder, different datastore (Fig. 6) can be added by extending implementation of the *RecboardDB* class.

As shown above, the existing implementation of RecBoard is extensible and maintainable as it is modular, based on DRY [2] and SOLID [7] principles, and is testable.

6 DEMO OVERVIEW

We plan to demonstrate RecBoard using a user-item interaction dataset with 5551 users and 16980 items. This dataset, *citeulike-a*, was collected by Wang et al. [10] from CiteULike and Google Scholar. CiteULike allows users to create their own collections of articles. There are abstracts, titles, and tags for each article. Auxiliary information such as authors, groups, posting time, and keywords is not used. The details can be found at www.citeulike.ort/faq/data.adp.

The demonstration starts by logging in as a registered user and creating a workspace using *Create New* button in left panel and entering workspace name (e.g., MyWorkspace). This creates a workspace, as shown in Fig. 2. This is followed by uploading the dataset into workspace created above using the *Upload* button. Once the dataset is uploaded, a model is trained by selecting hyper-parameters and other details and clicking *Create and Start Training* button. This would show training process with real-time graphs as shown in Fig. 3. Once training is completed, *Deploy* button is clicked, which deploys the model on local server and presents an API URL along with UI component to test the deployed model's REST API (Fig. 4). We show that deployed model is working by inputing comma-separated sample user-ids and item-ids followed by clicking *Test* button. This shows sample response from the deployed model.

7 CONCLUSION AND FUTURE WORK

RecBoard contributes towards reducing operational inefficiencies in the process of creating recommendation systems. This is made feasible by (1) identifying repetitive tasks in the pipeline, (2) defining reusable components for these tasks, and (3) creating a user interface to automate these tasks using the components defined. This can improve the speed at which recommender models can be productionized, irrespective of production environment or cloud provider. In the future, we plan to improve RecBoard by adding features for dataset visualization, cross-validation and comparison of deployed models. We also plan to explore the idea of a similar platform in other domains, such as Computer Vision, and Natural Language Processing.

8 ADDITIONAL MATERIAL

Screencast video link: http://mohitchawla.in/recboard/index.html

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under grant IIS-1700832 and by Yahoo Research (via the Connected Experiences Laboratory at Cornell Tech). The work was further supported by the small data lab at Cornell Tech, which received funding from NSF, NIH, RWJF, UnitedHealth Group, Google, and Adobe. We thank the anonymous reviewers for their insightful comments and suggestions.

REFERENCES

[1] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J Franklin, Joseph E Gonzalez, and Ion Stoica. 2017. Clipper: A Low-Latency Online Prediction Serving System.. In *NSDI*. 613–627.

[2] Andrew Hunt and David Thomas. 1999. *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[3] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren (Eds.). 2018. *Automatic Machine Learning: Methods, Systems, Challenges*. Springer. In press, available at http://automl.org/book.

[4] Amazon Web Services Inc. 2018. Amazon SageMaker. http://aws.amazon.com/sagemaker/. Online; accessed: 2018-12-08.

[5] BigML Inc. 2018. BigML: Machine Learning made beautifully simple for everyone. https://bigml.com. Online; accessed: 2018-12-08.

[6] Tim Kraska, Ameet Talwalkar, John C Duchi, Rean Griffith, Michael J Franklin, and Michael I Jordan. 2013. MLbase: A Distributed Machine-learning System.. In *Cidr*, Vol. 1. 2–1.

[7] Robert C Martin. 2000. Design principles and design patterns. *Object Mentor* 1, 34 (2000), 597.

[8] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[9] Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li, Vinu Rajashekhar, Sukriti Ramesh, and Jordan Soyke. 2017. TensorFlow-Serving: Flexible, high-performance ML serving. *arXiv preprint arXiv:1712.06139* (2017).

[10] Hao Wang, Binyi Chen, and Wu-Jun Li. 2013. Collaborative Topic Regression with Social Regularization for Tag Recommendation.. In *IJCAI*. 2719–2725.

[11] Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, Dandelion Mané, Doug Fritz, Dilip Krishnan, Fernanda B Viégas, and Martin Wattenberg. 2018. Visualizing dataflow graphs of deep learning models in TensorFlow. *IEEE transactions on visualization and computer graphics* 24, 1 (2018), 1–12.

[12] Longqi Yang, Eugene Bagdasaryan, Joshua Gruenstein, Cheng-Kang Hsieh, and Deborah Estrin. 2018. OpenRec: A Modular Framework for Extensible and Adaptable Recommendation Algorithms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 664–672.