Poster: A Recommender System for Developer Onboarding

Chao Liu, Dan Yang
Xiaohong Zhang, Haibo Hu
School of Software Engineering
Chongqing University, China.
{liu.chao, dyang, xhongz, hbhu}@cqu.edu.cn

Jed Barson
Baishakhi Ray
Department of Computer Science
University of Virginia, USA.
{jb3bt, rayb}@virginia.edu

ABSTRACT

Successfully onboarding open source projects in GitHub is difficult for developers, because it is time-consuming for them to search an expected project by a few query words from numerous repositories, and developers suffer from various social and technical barriers in joined projects. Frequently failed onboarding postpones developers' development schedule, and the evolutionary progress of open source projects. To mitigate developers' costly efforts for onboarding, we propose a ranking model NNLRank (Neural Network for List-wise Ranking) to recommend projects that developers are likely to contribute many commits. Based on 9 measured project features, NNLRank learns a ranking function (represented by a neural network, optimized by a list-wise ranking loss function) to score a list of candidate projects, where top-n scored candidates are recommended to a target developer. We evaluate NNLRank by 2044 succeeded onboarding decisions from GitHub developers, comparing with a related model LP (Link Prediction), and 3 other typical ranking models. Results show that NNLRank can provide developers with effective recommendation, substantially outperforming baselines.

CCS CONCEPTS

• **Software and its engineering → Open source model**;

KEYWORDS

Developer Onboarding, Recommender System, Learning to Rank

ACM Reference Format:
Chao Liu, Dan Yang, Xiaohong Zhang, Haibo Hu, Jed Barson, and Baishakhi Ray. 2018. Poster: A Recommender System for Developer Onboarding. In *Proceedings of 40th International Conference on Software Engineering Companion (ICSE '18 Companion).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3194989

1 INTRODUCTION

Open source software ecosystems like GitHub attract numerous developers to pursue their interests and goals [1] in software development. However, developers often face difficulties to successfully onboard open source projects, namely contributing many commits on joined projects, due to various technical and social barriers [1]. Frequently failed onboarding is harmful not only to developers but also to the open source projects. It increases new developers' efforts for searching a good project for themselves and learning [6]. It also postpones the development schedule of existing developers of a project and hurts the evolutionary progress of open source projects [7]. As a result, the delayed development would take more time and efforts to make up. In this work, we propose a novel recommender system to help developers onboarding new GitHub projects.

To capture developers' onboarding pattern, we develop 9 project features that may affect the onboarding process. We consider an onboarding is successful when a developer at least makes 6 commits (number of median commits per project in our studied data) to the project. Next, we propose a *learning to rank* model called NNLRank (Neural Network for List-wise Ranking) to recommend projects where developers are likely to contribute many commits. NNLRank learns a ranking function (represented by a neural network [4] that is optimized by a list-wise ranking loss function [8]) to score a list of candidate projects, and the top-n scored candidates are recommended to a developer. NNLRank is verified by investigating 2044 successful onboarding decisions from GitHub developers. We compare NNLRank with the LP model [6], and 3 other typical learning to rank models, SVMRank [5], BPNet [4], and SVM (Support Vector Machine). By evaluating models with a commonly used performance metric MRR (Mean Reciprocal Rank) with 5-fold cross-validation, we show that NNLRank achieves the best performance (mean MRR = 0.466), outperforming the best baseline SVMRank by 29.81%.

2 DATA COLLECTION

We briefly describe sampled datasets and measured features:

Datasets. To simulate the practical usage of NNLRank, we collect developers' onboarding data from GHTorrent [2], a mirror of GitHub data with structured form. We downloaded the database dump dated 11/01/2016, and sampled 6,343 active projects [1] from the latest 5 years, and sampled 17,219 active developers [1]. We consider a joining with at least 6 commits (median of the sampled projects) as a successful onboarding. Finally, we collected 2,044 successful onboarding decisions, relating to 1,070 active projects and 1,672 active developers. Each decision involves 257-2794 candidate projects, a total of about 2.9 million different instances.

Dependent Variable. The outcome of the model is a rank list of onboarded projects recommended to a developer at onboarding time [1]. The proposed model aims to give higher score to the onboarded project so that the project to be onboarded can have a higher ranking.

Independent Variables. To capture developers' onboarding pattern, we represent candidate projects by 9 features: *1)* the number of developer's joined projects collaborated with the owner of target

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3194989

project; each collaborated project is weighted by the reciprocal of member count [1]; *2)* the number of commits in the target project [3]; *3)* the number of developer's joined projects whose programming language matches the target project [1]; *4)* the duration from developer's onboarding time to the time of creation, *5)* first commit, *6)* latest commit, and *7)* first membership. *8)* latest membership in the target project [3]; *9)* the number of members who are company colleagues of the developer.

3 METHODOLOGY

The NNLRank is a neural network based model, and it is optimized by a list-wise ranking loss function. Simply sorting predicted scores for candidate projects can make a recommendation for developer onboarding.

Neural Network. We build a 4-layer neural network [4] (i.e., a ranking function f) with 2 hidden layers, where each hidden layer contains 5 nodes. The input and output layers respectively have 9 (the number of features) and 1 node(s). By inputting a list of candidate projects, $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\} \in \mathbb{R}^{9\times n}$, to the network, it gives the scores of the candidate projects $f(\mathbf{X}) = \{f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_n)\}$ in the last layer. In the m-th layer, its input is multiplied by a weight matrix $\mathbf{W}^{(m)}$, added by a bias vector $\mathbf{b}^{(m)}$, and processed by an activation function (we use the Arctan function).

Network Optimization. To optimize the built network, the weights are randomly initialized and the bias are set to zero at first. We then iteratively input lists of candidate projects to the network, and update the m-th layer of the network based on a loss function proposed by [8]. We simplify the loss function as: $\arg\min_{\mathbf{W},\mathbf{b}} \sum_{i=2}^{n}(i-1)f(\boldsymbol{x}_i) + \frac{1}{2}\sum_{m=1}^{M}\left(\|\mathbf{W}^{(m)}\|_F^2 + \|\mathbf{b}^{(m)}\|_2^2\right)$. The optimization stops when the loss function converges to a small value (we set 0.01), or all the inputted lists have been used [8].

4 EARLY RESULT

We conduct an early experiment on our proposed model:

Evaluation Criteria. We assess our model by a widely used performance metric MRR (Mean Average Precision). We also use an auxiliary measure, MS (Match Score). MS calculates the percentage of onboarded projects ranked among top-k (k=1,5,10) recommendations for all lists in testing data. Larger MS indicates more developers can find expected projects by reviewing top-k recommended projects at most.

Validation Data. To suppress the effect of outliers, we normalize 9 features among each list of candidate projects (2044 in total) by the min-max normalization. We then perform a 5-fold cross-validation.

Results and Discussion. Table 1 presents the mean MRR and MS of 6 models. Results show that NNLRank achieves the best mean MRR (0.466), outperforming the best baseline SVMRank by 29.81%. The main advantage of NNLRank over other baselines are resulted from the better MS at top-1 and top-5, where 35.28% and 57% of developers can find their expected projects to onboard by reviewing at most 1 and 5 recommended projects, respectively.

Table 1: Prediction accuracy (MRR and MS at top-1/5/10) comparison of NNLRank and 5 baseline models.

Model	MRR	MS_1	MS_5	MS_{10}
NNLRank	0.466	35.28%	57.00%	62.52%
SVMRank	0.359	19.46%	53.94%	65.57%
BPNet	0.022	00.69%	02.59%	03.52%
SVM	0.007	00.15%	00.44%	00.73%
Random	0.018	00.24%	01.86%	02.45%
LP	0.004	00.00%	00.00%	00.00%

5 FUTURE WORK

In our future work, we plan to explore more questions:

How can the proposed model work for developers with different prior experience? Some features are extracted from developers' prior experience, such as social tie, we thus plan to investigate the recommendation for developers with different levels of project experience in GitHub.

What other measurable features can help capture developers' onboarding pattern? Other features, like project domain, may also affect developers' onboarding, but they are not easy to be measured. We will investigate more measurable features.

What are the most important features that affects developers' onboarding? Among measured features, some may affect the model more than others, we will explore the relative importance of these features by sensitivity analysis.

ACKNOWLEDGMENTS

The work described in this paper was partially supported by the Fundamental Research Funds for the Central Universities of China (Grant No. 106112017CDJXSYY002), the National Natural Science Foundation of China (Grant no. 61772093), and Chongqing Research Program of Basic Science & Frontier Technology (Grant No. cstc2017jcyjB0305), Scientific and Technological Research Program of Chongqing Municipal Education Commission (Grant No. KJ1501504).

REFERENCES

[1] Casey Casalnuovo, Bogdan Vasilescu, Premkumar Devanbu, and Vladimir Filkov. 2015. Developer onboarding in Github: the role of prior social links and language experience. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 817–828.

[2] Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: GitHub's data from a firehose. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*. IEEE Press, 12–21.

[3] Jungpil Hahn, Jae Yun Moon, and Chen Zhang. 2008. Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties. *Information Systems Research* 19, 3 (2008), 369–391.

[4] Robert Hecht-Nielsen et al. 1988. Theory of the backpropagation neural network. *Neural Networks* 1, Supplement-1 (1988), 445–448.

[5] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 217–226.

[6] Tadej Matek and Svit Timej Zebec. 2016. GitHub open source project recommendation system. *arXiv preprint arXiv:1602.02594* (2016).

[7] Igor Steinmacher, Marco Aurelio Graciotto Silva, Marco Aurelio Gerosa, and David F Redmiles. 2015. A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology* 59 (2015), 67–85.

[8] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*. ACM, 1192–1199.