Open Source Repository Recommendation in Social Coding

Jyun-Yu Jiang
Department of Computer Science
University of California, Los Angeles
jyunyu.jiang@gmail.com

Pu-Jen Cheng
Department of Computer Science
National Taiwan University
pjcheng@csie.ntu.edu.tw

Wei Wang
Department of Computer Science
University of California, Los Angeles
weiwang@cs.ucla.edu

ABSTRACT

Social coding and open source repositories have become more and more popular. Software developers have various alternatives to contribute themselves to the communities and collaborate with others. However, nowadays there is no effective recommender suggesting developers appropriate repositories in both the academia and the industry. Although existing one-class collaborative filtering (OCCF) approaches can be applied to this problem, they do not consider particular constraints of social coding such as the programming languages, which, to some extent, associate the repositories with the developers. The aim of this paper is to investigate the feasibility of leveraging user programming language preference to improve the performance of OCCF-based repository recommendation. Based on matrix factorization, we propose language-regularized matrix factorization (LRMF), which is regularized by the relationships between user programming language preferences. Extensive experiments have been conducted on the real-world dataset of GitHub. The results demonstrate that our framework significantly outperforms five competitive baselines.

CCS CONCEPTS

•**Information systems** →**Social recommendation;** *Geographic information systems;* •**Human-centered computing** →**Social recommendation;** *Social networking sites;*

KEYWORDS

open source repository; repository recommendation; user programming language preference; manifold regularization

ACM Reference format:
Jyun-Yu Jiang, Pu-Jen Cheng, and Wei Wang. 2017. Open Source Repository Recommendation in Social Coding. In *Proceedings of SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan, ,* 4 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080753

1 INTRODUCTION

Social coding, such as the services provided by SourceForge [1] and GitHub[2], has become one of the most important collaborative approaches for open-source software development. By social coding, software projects called "repositories" can be publicly created and easily contributed by collaborators from the whole world. However, currently there is not any personalized repository recommender to suggest appropriate repositories from countless candidates to users. If repositories can be recommended to suitable users early, the development of open source software can speed up. How to precisely recommend repositories to users is, therefore, an important issue.

One possible solution to repository recommendation is one-class collaborative filtering (OCCF) [14] because users' past activities reveal what repositories they might be interested in. Without negative instances (i.e., the capacity of observing a user dislikes an item), OCCF captures implicit feedback and ranks items in a reasonable order. For example, some works [20] focus on the pairwise relations between observed and non-observed items. Although conventional OCCF approaches can recommend items with only positive feedback, none of them considers the particular constraints in the relationship between social coding and repositories.

Programming language[3] is one of the constraints in social coding. The language usage of a collaborator may depend on her coding abilities and experiences [12]. Programming language is also an essential part for repositories because programmers invariably have to choose languages to develop projects. For instance, a front-end application could be implemented in JavaScript and CSS; the applications for Apple iOS devices are implemented in Swift or Objective-C. Some interpreted languages like Python are adopted due to their convenience; others like C are utilized for execution efficiency. The language preference coming from both users and repositories implies that the repositories of interest to a user are limited. It is expected that the performance of CF could be further boosted by considering not only the repository preference of user neighbors but also the language preference between them.

A possible solution that exploits the language information is to directly treat languages as tags or categorical features [4]. For example, a recommender can recommend repositories with languages identical to users' used languages [22]. Some conventional factorization models can be expanded as tensors[7] or factorization machines [15]. However, language information is generally too sparse as categorical features because each repository is designated for only few languages and only few languages are used by a user [12]. Moreover, users' language preference may potentially affect the relationship between other users and repositories in CF. Such phenomenon is not taken into account by previous methods, which consequently do not fully take the advantage of the language information for repository recommendation.

In this paper, we focus mainly on exploiting user language preference to improve the performance of repository recommendation

[1] SourceForge: https://sourceforge.net/
[2] GitHub: https://github.com/

SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan

DOI: http://dx.doi.org/10.1145/3077136.3080753

[3] *Programming language* is hereinafter referred to as *language* for simplicity.

on GitHub. Given the user language preference and preferred repositories in the training data, the goal of this paper is to rank all other repositories so that the top-ranked repositories are more likely to be interesting to the user. Because language preference may implicitly represent several user attributes, the difference of language preference between users is taken into account. By assuming that users with similar language preferences may also have similar repository preferences, we propose a regularization framework LRMF formed with latent factor based OCCF models. Extensive experiments have been conducted on the four-year GitHub logs. The experimental results show that our approach significantly outperforms five competitive baseline methods and their variations. Such improvements are consistent across subsets of the testing set with different user and repository distributions.

In the literature, social coding has been popular and discussed in many aspects, such as transparency [3] and collaboration [9]. To the best of our knowledge, currently there is no thorough published study about personalized repository recommendation. Although Suchal and Návrat [18] mentioned "recommendation," they only built a full-text search engine but not a personalized recommender. Orii [13] applied topic models to recommend repositories, but the performance is poor and the results are not published. Yu *et al.* [21] utilized the simple TF-IDF measure to recommend expert reviewers for pull-request. Although their solution can be reversed to repository recommendation, only the experts who can become reviewers will be recommended. Hence, a thorough study of repository recommendation and an effective recommender are necessary for social coding.

2 REPOSITORY RECOMMENDATION WITH USER LANGUAGE PREFERENCE

We first formally define the problem of repository recommendation. Let U be the user set and I be the repository set. Suppose there are a training set of observations $S \subseteq U \times I$, and a testing set $T \subseteq (U \times I) \setminus S$. Each entry (u, i) in S and T represents that the user u prefers the repository i. For simplicity, repositories preferred by a user u in the training and testing sets are denoted as $S(u)$ and $T(u)$. The language set is represented as L. For a repository $i \in I$, $L(i)$ denotes its applied language. Given a user $u \in U$ and the training data S, our goal is to give a ranking to repositories $i \in I \setminus S(u)$ so that ones with higher rankings are more likely to be in $T(u)$.

Here we propose **Language-Regularized Matrix Factorization (LRMF)**. The framework starts from matrix factorization for obtaining knowledge from collaborative filtering. We assume that users with similar language preference may also have similar repository preference, and add regularization terms to reflect the concepts. To estimate the similarity of language preference, two kinds of preference representations are proposed.

2.1 Matrix Factorization

LRMF rests upon general matrix factorization (MF) [8] learning from collaborative filtering. Actually, almost all of latent factor based models can be utilized in our framework. Hence, theoretically our model can at least perform as well as the applied models.

In this paper, we choose Bayesian personalized ranking matrix factorization (BPRMF) [16] because it is one of the state-of-the-art OCCF models. As a MF model, BPRMF treats all predictions $\hat{r}_{ui}$ as a $|U| \times |I|$ target matrix $\hat{\mathbf{R}}$, and predicts the matrix by a $k \times |U|$ matrix $\mathbf{P}$ and a $k \times |I|$ matrix $\mathbf{Q}$ as $\hat{\mathbf{R}} = \mathbf{P}^{\mathrm{T}}\mathbf{Q}$, where k is the dimension of hidden factors. Each column $\mathbf{p}_u$ in $\mathbf{P}$ represents the user vector of the user u, and each column $\mathbf{q}_i$ in $\mathbf{Q}$ is the item vector of the item i. Optimizing AUC (area under the ROC curve) [5], the objective function of BPRMF can be written as follows:

$$\begin{aligned} &\underset{\mathbf{P},\mathbf{Q}}{\operatorname{argmin}} \text{ BPR-OPT} \qquad (1)\\ &= \underset{\mathbf{P},\mathbf{Q}}{\operatorname{argmin}} \sum_{u \in U} \sum_{(i,j) \in D_s(u)} -\ln(\sigma(\hat{r}_{uij})) + \frac{\lambda_1}{2}\left(||\mathbf{P}||_F^2 + ||\mathbf{Q}||_F^2\right), \end{aligned}$$

where $D_s(u)$ is the set of preferred and unpreferred item pairs for the user u; σ is the logistic sigmoid; $||\cdot||_F$ represents the Frobenius norm for regularization of two matrices; and λ_1 is the regularization parameter. With optimization methods such as gradient descent, the model can reach a local minimum and optimize the errors.

In our implementation, the bootstrapping-based stochastic gradient descent approach is applied as [16] recommends. In each training iteration, we randomly sample an unpreferred repository $j \in I \setminus S(u)$ for every observation $(u, i) \in S$, and then optimize the model. Hence, the optimization of BPRMF can be done in $O(k|S|)$ for each training iteration.

2.2 User Language Preference Regularization

The main principle in this paper is that user language preference is really helpful for the repository recommendation. Here we propose two intuitive assumptions for establishing the regularization framework:

Assumption 1. *If two users have similar languages preference, they will also have similar preference in repositories.*

Assumption 2. *Given a well-trained matrix factorization model, if two users x and y have similar preference in repositories, they will also be close to each other in their user latent factors $\mathbf{p}_x$ and $\mathbf{p}_y$.*

By the first assumption, user language preference can be linked to how users prefer repositories, which is the core of personalized repository recommendation. However, we have to encode this concept into the model, so the other assumption is needed. The second assumption is more intuitive and almost equivalent to the low-rank assumption in collaborative filtering. Hence, by two assumptions, we can connect the actual objective function in the optimization with user language preference.

To add user language preference into the model, we propose to apply the manifold regularization framework and improve model by the geometrical structure of the data [2]. Taking the assumptions, the framework preserves the local invariance [1], i.e., nearby language preferences lead to the similar user latent factors. Suppose the language preference of a user u can be represented as a vector $\boldsymbol{\theta_u}$, the preference similarity for two user can be measured by any of well-defined similarity functions. For two users x and y, we simply use the cosine similarity as the similarity function $w\left(\boldsymbol{\theta_x}, \boldsymbol{\theta_y}\right)$ By the manifold regularization framework, the proposed assumptions can be characterized as a regularizer as follows:

$$\frac{1}{2} \sum_{x \in U} \sum_{y \in U \setminus \{x\}} \left(\mathbf{p}_x - \mathbf{p}_y\right)^2 w\left(\boldsymbol{\theta_x}, \boldsymbol{\theta_y}\right), \qquad (2)$$

where $\mathbf{p}_x$ and $\mathbf{p}_y$ are the user latent factors of users x and y. By combining Eq. (1) and (2), we finally can write the objective function of our model as follows:

$$\underset{\mathbf{P},\mathbf{Q}}{\text{argmin}}\ \text{BPR-OPT} + \frac{\lambda_2}{2} \sum_{x \in U} \sum_{y \in U \setminus \{x\}} \left(\mathbf{p}_x - \mathbf{p}_y\right)^2 w\left(\boldsymbol{\theta}_{\boldsymbol{x}}, \boldsymbol{\theta}_{\boldsymbol{y}}\right),$$

where λ_2 is also a parameter for regularization. By introducing the regularizer into the objective function, the matrix $\mathbf{P}$ can be optimized more smoothly with respect to the relationship between users' language preferences.

However, the time complexity will be so large as $O(|U|^2)$ if we directly optimize the regularizer. To train the model more efficiently, we apply bootstrapping-based stochastic gradient descent again by sampling $c|U|$ user pairs, where c is a constant for adjustment. In this paper, we set $c = 100$. Therefore, the optimization of the language preference regularization can be done in $O(|U|)$ for each training iteration.

2.3 Preference Representation

In this section, we propose two kinds of representations for user language preference, including the vector space model (VSM) and the latent factor model (LFM).

Vector Space Model (VSM). VSM has been widely used in information retrieval [17]. VSM applies a vector whose dimensions correspond the weights of separate terms to represent a document. This concept is also useful in collaborative filtering [10]. Treating each language as a term and observations in $S(u)$ as the document, $\boldsymbol{\theta}_{\boldsymbol{u}}$ can be represented as a $|L|$-dimensional vector $\boldsymbol{\theta}_{\boldsymbol{u}}$ by TF-IDF in the vector space of languages. In other words, VSM takes user's past language usage into account to represent the language preference.

Latent Factor Model (LFM). However, VSM may have room for improvement because (1) users may tend to apply few languages; (2) VSM does not consider the relationship between languages. One feasible solution is to reduce the dimensions of vectors for more representative representations. LFM is one of solutions for dimension reduction. If a user preferred a repository of a language, we can say that the user also preferred such language. Hence, the language preference can be treated as a OCCF problem. In this paper, we apply BPRMF [16] again to derive user latent factors as language preferences. Suppose the user preference for each language can be described in a $|U| \times |L|$ matrix $\mathbf{R}_2$, BPRMF decomposes the matrix by a $k \times |U|$ matrix $\mathbf{G}$ and a $k \times |L|$ where k is also the dimension of latent factors. Each column $\mathbf{g}_u$ in $\mathbf{G}$ represents the user vector of the user u. Finally, the language preference $\boldsymbol{\theta}_{\boldsymbol{u}}$ is assigned by the user factor $\mathbf{g}_u$ in the latent factor model. By reducing the dimensions from $|L|$ to k, the language preference can be represented precisely within few dimensions. Moreover, each dimension of the vector can further explain the preference for multiple programming languages.

3 EXPERIMENTS

3.1 Experimental Settings and Datasets

We conduct experiments on data collected from GitHub Archive[4], which comprises all public user events on GitHub from 12 February 2011 to 31 December 2014. In this paper, a user is considered to prefer a repository if the user does any of the five activities including `PushEvent`, `IssueEvent`, `ForkEvent`, `PullRequestEvent` and `WatchEvent`, whose definitions can be found in GitHub. Every repository has some metadata including the programming language for implementation. To be fair to previous work, our experimental settings is exactly identical to [16]. Users and repositories with less than 10 occurrences in the logs are dropped because they may not be active. A subsample of 10,000 users and 10,000 repositories with 616,633 observations is created so that every user prefers at least 10 repositories, and every repository is preferred by at least 10 users.

A 10-fold cross-validation scheme is applied to evaluate the performance. For all models with latent factors, the latent dimension k is set to 30 because a larger k leads to similar performance. The parameters λ_1 and λ_2 are automatically tuned by a grid search. The learning rate is fine-tuned by line search in each iteration based on the validation performance. All latent factors based baselines are tuned by the same method.

To evaluate performance on different situations, seven testing datasets are generated as shown in Table 1 . In addition to the overall testing set, we evaluate the performance for users, repositories and languages with different popularity.

Table 1: The description of each testing subset.

Dataset	#(users)	#(repos)	Description
Overall	10,000	10,000	All users and all repositories
Head Users	2,000	10,000	Users with top 20% frequency
Tail Users	2,000	10,000	Users with bottom 20% frequency
Head Repos	10,000	2,000	Repos with top 20% frequency
Tail Repos	10,000	2,000	Repos with bottom 20% frequency
Head Langs	10,000	9,584	Languages with top 20% frequency
Tail Langs	10,000	74	Languages with bottom 20% frequency

3.2 Baseline Methods

Our approach is compared to five baselines:

- **Most-Popular (MP):** MP is the naïve method which simply ranks all repositories by their frequencies.
- **Weighted Regularized MF (WRMF):** WRMF [6, 14] applies implicit feedback with weighted regularization for unknown preferences.
- **Maximum Margin MF (MMMF):** MMMF [20, 20] exploits ordinal ranking for recommendation.
- **Bayesian Personalized Ranking MF (BPRMF):** BPRMF [16] is one of the state-of-the-art approach for ranking in the OCCF problem as described in Section 2.1.
- **Factorization Machine (FM):** FM [15] mimics factorization models by feature engineering. Hence it can consider additional features such as language preference into the model. Note that FM is the only baseline considering the information of programming languages in this paper.

3.3 Experimental Results

In our experiments, *area under the ROC curve* (AUC) [5], *R-Precision* (RP) [11] and *mean reciprocal rank* (MRR) [19] are applied to evaluate the quality of repository recommendation. Table 2 shows the experimental results. Among all baselines, FM performs best

[4]GitHub Archive: https://www.githubarchive.org/

Table 2: The performance of five methods and their variations with seven datasets. All improvements of LRMF against the best baseline are significant differences at 99% level in a paired t-test.

Dataset		MP	WRMF	MMMF	BPRMF	FM	LRMF (VSM)	LRMF (LFM)
Overall	AUC	0.7898	0.8815	0.8821	0.8912	0.8968	0.9165	**0.9217**
	RP	0.0247	0.0231	0.0334	0.0353	0.0366	0.0579	**0.0644**
	MRR	0.0900	0.0842	0.1144	0.1182	0.1203	0.1803	**0.2033**
Head Users	AUC	0.7810	0.8804	0.8726	0.8829	0.8903	0.9090	**0.9150**
	RP	0.0385	0.0500	0.0572	0.0619	0.0653	0.0902	**0.1000**
	MRR	0.1750	0.1549	0.2130	0.2135	0.2187	0.3164	**0.3544**
Tail Users	AUC	0.7971	0.8723	0.8856	0.8930	0.8968	0.9193	**0.9228**
	RP	0.0127	0.0068	0.0203	0.0200	0.0153	0.0363	**0.0393**
	MRR	0.0399	0.0362	0.0607	0.0593	0.0523	0.0933	**0.1028**
Head Repos	AUC	0.6954	0.8386	0.8404	0.8491	0.8501	0.8814	**0.8851**
	RP	0.0275	0.0263	0.0395	0.0428	0.0399	0.0670	**0.0754**
	MRR	0.0935	0.0906	0.1245	0.1299	0.1249	0.1877	**0.2099**
Tail Repos	AUC	0.4250	0.7942	0.7943	0.8073	0.8114	0.8477	**0.8695**
	RP	0.0000	0.0108	0.0058	0.0068	0.0044	0.0124	**0.0186**
	MRR	0.0019	0.0401	0.0313	0.0343	0.0246	0.0497	**0.0649**
Head Langs	AUC	0.7885	0.8819	0.8823	0.8912	0.8968	0.9167	**0.9218**
	RP	0.0236	0.0231	0.0328	0.0338	0.0360	0.0570	**0.0638**
	MRR	0.0862	0.0834	0.1112	0.1145	0.1187	0.1758	**0.1993**
Tail Langs	AUC	0.6813	0.7799	0.7711	0.7712	0.8310	**0.8440**	0.8405
	RP	0.0259	0.1336	0.0819	0.1056	0.1897	0.1746	**0.2134**
	MRR	0.1350	0.2842	0.2161	0.2326	0.3237	0.3320	**0.3662**

because it further considers language information. BPRMF outperforms other MF models because it optimizes the measure about rankings directly. MMMF beats WRMF because WRMF only pointwisely optimize the model so that it cannot learn rankings well. LRMF significantly outperforms all baselines over all evaluation measures for both VSM and LSM. The model using LFM is better than the one using VSM because LFM represents preferences more precisely. Although FM encodes languages as features, the complex preferences are not learned. In contrast, LRMF captures the preference directly by observing user relationships. Among different testing datasets, it is obvious that models perform better with head users/repos than tail users/repos on RP and MRR because there are more observations in head users and head repos training sets. AUC of head users is worse than AUC of tail users because tail users only preferred few repositories. That is also why AUC is sometimes not suitable for evaluation. RP and MRR in the set of tail languages are higher than others because the number of candidate repositories is few.

4 CONCLUSIONS

In this paper, we propose a pairwise regularization framework LRMF for open source repository recommendation aiming to rank users' preferred repositories in high positions. Compared to conventional recommenders, LRMF attempts to learn users' language preference and then enhance matrix factorization. Specifically, we make an assumption that users with similar language preference may also have similar preference in repositories. Following the assumption, the user latent factors can be regularized by the manifold regularization with the preference of other users. Moreover, we further propose two ways to estimate the user language preference. The results of extensive experiments show that our methods can significantly outperform the existing state-of-the-art OCCF models. Such improvements are consistent across different testing sets with different user and repository distributions. The data analysis also supports that user language preference is important.

There are two ways for future work: (1) analyze contents of social coding (i.e., codes in repositories) for further improvements; and (2) exploit social networks in social coding (i.e., user collaborations) for other applications like link prediction.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their helpful comments. The work is partially supported by NIH U01HG008488, NIH R01GM115833, NIH U54GM114833, and NSF IIS-1313606.

REFERENCES

[1] Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering.. In *NIPS 2001*. 585–591.
[2] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR* 7 (2006), 2399–2434.
[3] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *CSCW 2012*. ACM, 1277–1286.
[4] Antonina Dattolo, Felice Ferrara, and Carlo Tasso. 2010. The role of tags for recommendation: a survey. In *Proceedings of 3rd International Conference on Human System Interaction*. IEEE, 548–555.
[5] Alan Herschtal and Bhavani Raskutti. 2004. Optimising area under the ROC curve using gradient descent. In *ICML 2004*. ACM, 49.
[6] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM 2008*. IEEE, 263–272.
[7] Tamara G Kolda and Jimeng Sun. 2008. Scalable tensor decompositions for multi-aspect data mining. In *ICDM 2008*. IEEE, 363–372.
[8] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.
[9] Antonio Lima, Luca Rossi, and Mirco Musolesi. 2014. Coding together at scale: Github as a collaborative social network. In *ICWSM 2014*.
[10] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE* 7, 1 (2003), 76–80.
[11] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, and others. 2008. *Introduction to information retrieval*. Vol. 1. Cambridge university press Cambridge.
[12] Leo A Meyerovich and Ariel S Rabkin. 2013. Empirical analysis of programming language adoption. In *SIGPLAN 2013*. ACM, 1–18.
[13] Naoki Orii. 2012. Collaborative Topic Modeling for Recommending GitHub Repositories. (2012). http://www.cs.cmu.edu/~norii/pub/github-ctr.pdf.
[14] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *ICDM 2008*. IEEE, 502–511.
[15] Steffen Rendle. 2012. Factorization Machines with libFM. *ACM TIST* 3, 3, Article 57 (2012), 22 pages.
[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI 2009*. 452–461.
[17] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Comm. of ACM* 18, 11 (1975), 613–620.
[18] Ján Suchal and Pavol Návrat. 2010. Full text search engine as scalable k-nearest neighbor recommendation system. In *Artificial Intelligence in Theory and Practice III*. Springer, 165–173.
[19] Ellen M Voorhees and others. 1999. The TREC-8 Question Answering Track Report.. In *Trec*, Vol. 99. 77–82.
[20] Markus Weimer, Alexandros Karatzoglou, and Alex Smola. 2008. Improving maximum margin matrix factorization. *Machine Learning* 72, 3 (2008), 263–276.
[21] Yue Yu, Huaimin Wang, Gang Yin, and Charles X Ling. 2014. Reviewer Recommender of Pull-Requests in GitHub. In *ICSME 2014*. IEEE, 609–612.
[22] Shiwan Zhao, Nan Du, Andreas Nauerz, Xiatian Zhang, Quan Yuan, and Rongyao Fu. 2008. Improved recommendation based on collaborative tagging behaviors. In *IUI 2008*. ACM, 413–416.