Open-source project recommendation model

Huihui Zhu, *Pu* Peng*, and *Huan* Xu

East China Normal University, China

Keywords: github, recommendation model, collaborative filtering, light-GCN, embedding.

Abstract. Open-source has become a very important topic in this era. the number of open-source projects on github Shows a huge growth trend. Facing so many open-source projects, it's not easy to find the projects and topics that the developers interested in. so, it is necessary to model the user's behavior data,So as to automatically recommend projects to developers. to explore this problem, we constructed a dataset of 90w users and 461w projects based on github log and did a lot of cleaning work on the data. finally, we model the data through the improvement of the Light-GCN model to recommend relevant open-source projects to users. The experimental results show that the accuracy of our model is more than 15%.

1 Introduction

As the world's largest code hosting platform, the data shows that in the past 2020, there were 56 million developers establisged over 60 million projects, these projects cover many areas of our real life. Facing so Many projects and communities, it is hard for developers to choose the project among tens of millions of open source projects which they are instrested in, so it is necessary to build a recommendation system based on github log data.

In recent years, recommendation algorithms have achieved good applications in both academia and industry, such as The early BPR[2] matrix factorization method, recently, some researchers have applied GCN to the recommendation system, such as DGCF[4] and NGCF[3], light-GCN[1] eliminates redundant nonlinear transformations and feature changes in NGCF,It makes the model more concise and easy to train.

collaborative filtering(CF), which focuses on exploiting the past user-item interactions to achieve the prediction, remains to be a fundamental task towards effective personalized recommendation [9,10,11,12],However, few people apply the recommendation model to open -source Project recommendation.

We proposes a GCN-based open-source project recommendation model, which builds based on user behavior data. In addition, this model has better recall while ensuring training speed.

The model has been applied to github behavior log data and achieved good results. The other parts of this article are organized as follows: Section 2 introduces the model , and

* Corresponding author: ppu@cc. ecnu. edu. cn

Section 3 gives the experimental steps And the results. finally, the finally part summarizes the full paper.

2 Methodologies

2.1 Data aggregation

When we built the recommendation dataset, we found that the data is extremely sparse, which will affect The performance of embedding, and sparseness will also affect the recommended performance. We use data aggregation to solve such problem, More specifically, We extract the features of open source projects from README and label, and then compare the similarities of these features,When the similarity between them is close to a certain threshold, the two projects will be merged into one project. This alleviates the extremely sparse nature of the data.

2.2 Embedding

Various methods of graph embedding have been proposed in the machine learning literature [13,14,15].

The NGCF and DGCF are all inherited from GCN, The basic idea of GCN is to learning representation for nodes by smoothing features over the graph [7,8], NGCF contains the characteristics of GCN For operations such as transformation and nonlinear activation functions, the propagation rules of NGCF are define as:

$$\begin{aligned}\mathbf{e}_u^{(k+1)} &= \sigma\left(\mathbf{W}_1\mathbf{e}_u^{(k)} + \textstyle\sum_{i\in\mathcal{N}_u}\frac{1}{\sqrt{|\mathcal{N}_u||\mathbb{N}_i|}}\left(\mathbf{W}_1\mathbf{e}_i^{(k)} + \mathbf{W}_2\left(\mathbf{e}_i^{(k)}\odot\mathbf{e}_u^{(k)}\right)\right)\right)\\ \mathbf{e}_i^{(k+1)} &= \sigma\left(\mathbf{W}_1\mathbf{e}_i^{(k)} + \textstyle\sum_{u\in N_i}\frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\left(\mathbf{W}_1\mathbf{e}_u^{(k)} + \mathbf{W}_2\left(\mathbf{e}_u^{(k)}\odot\mathbf{e}_i^{(k)}\right)\right)\right)\end{aligned} \tag{1}$$

The propagation of embedding is carried out through neural network operations such as feature transformation and nonlinear activation function, which is different from traditional methods. Compared with the method, NGCF has achieved great success, but for the recommended data, The ID does not contain more features, It shows a one-hot code, which has no special meaning, but the information of each node on the image contains very rich information,So the characteristic transformation and nonlinear activation function in GCN may not work. After simplified, LGC dissemination method is:

$$\begin{aligned}\mathbf{e}_u^{(k+1)} &= \sum_{i\in\mathcal{N}_u}\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}\mathbf{e}_i^{(k)}\\ \mathbf{e}_i^{(k+1)} &= \sum_{u\in\mathcal{N}_i}\frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}}\mathbf{e}_u^{(k)}\end{aligned} \tag{2}$$

The feature transformation and nonlinear activation function are directly discarded. Embedded communication becomes more concise and easier to train.

After a rigorous ablation experiment, Light-GCN shows that the training speed and training accuracy of the Light-GCN model are both better than NGCF.

After K rounds of propagation, a total of K+1 embedding representations are obtained. If the last embedding is directly used as the finally embedding, the information embedded before the tune will be lost, so the combination of embedding will be introduced in the next section.

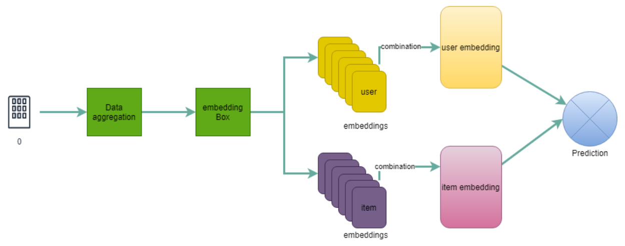

Fig. 1. Model overview.

2.3 Embeddings combibation and prediction

After K propagations of Light-GCN, a total of K+1 embeddings are obtained, and the final embedding is obtained by multiple parties. The problem comes that how shoule we combination this emneddings? One method is to directly take the last embedding step to make the most final embedding, but as the number of layers increases, The embedding is too smooth, and the semantics of different embedding layers are different, so the last layer is directly embedded As the final embedding, it directly affects the performance of embedding. Another way is to give each layer of embedding a weighted. For Light-GCN, in each layer of embedding The weights are 1/(K+1), we believe that the propagated embedding should be better than the previous embedding,Therefore, the later embeddings should be given higher weights. Unlike Light-GCN, in our model, The final embedding of our users and projects is defined as:

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)} \tag{3}$$

$\alpha = \{1 + (k/h)\}/(K+1)$, in this way, the later embeddings are given higher hyperparameters, where h is one of the models A hyperparameter that can be trained represents the weight of the embedding layer.

Finally, the prediction score is calculated by the inner product of the user's items:

$$\hat{y}_{ui} = \mathbf{e}_u^T \mathbf{e}_i \tag{4}$$

2.4 Training

For our model, after giving the embedding of the 0th layer, the rest can be directly calculated,So the parameters that can be trained are the parameters of the 0th layer, and the parameter h of the embedding combination can be adjusted manually Hyperparameters, the loss function uses BPR loss:

$$L_{BPR} = -\sum_{u=1}^{M} \sum_{i\in\mathcal{N}_u} \sum_{j\notin\mathcal{N}_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda\|\mathbf{E}^{(0)}\|^2 \tag{5}$$

3 Experriments

In order to verify the execution efficiency of the model proposed in this paper, we trained it on the ubuntu 20. 04. 1 operating system Tesla V100 32G graphics card accelerates training, using recall, mrr, and ndcg hit precision indicators to measure the performance of our model.

3.1 Dateset

Our original data comes from the API interface of user behavior logs open on github. The first problem we face is how to store T-level behavioral data. For github logs, we need update it frequently , and traditional relational database may face frequent disk I/O. The distributed open-source database clickhouse supports high-speed data compression, multi-core parallel processing and other features, which can solve the problems above. so we chooses clickhouse to store log data.

Table 1. Performance of different models

Model	recall	mrr	ndcg	hit	precision
BPR	0. 5673	0. 6349	0. 5563	0. 7183	0. 1384
NGCF	0. 3979	0. 4388	0. 3695	0. 5582	0. 0971
DGCF	0. 5681	0. 5979	0. 5372	0. 7232	0. 1414
LINE	0. 522	0. 5922	0. 5078	0. 6843	0. 1162
MultiDAE	0. 578	0. 7124	0. 6131	0. 7217	0. 1466
LIGHT-GCN	0. 5659	0. 5593	0. 5096	0. 7203	0. 1430
our-model	**0. 5725**	**0. 6238**	**0. 5207**	**0. 7211**	**0. 1433**

The original data records the user's issue, pr, merge and other operations. This provides us with a basis for scoring ,Therefore, specifically, IssueCommentEvent is counted as 1 point, IssuesEvent is counted as 2 points, fork Scored as 3 points, PullRequestEvent but not merge scored as 4 points, PullRequestReviewCommentEvent is scored as 5 points, and PullRequestEvent is The merge is recorded as 6 points. Therefore, the data are equally divided into "1-6". For multiple operations of a user, Determine the final score by averaging. This data and filter all the data for the first half of 2020,At the sametime, in order to ensure the reliability of the data, the user data with less than 15 operation records is eliminated. We have carried out a series of data preprocessing, and the specific flow chart is as figure 2.

Finally, after filtering and conversion, we got a dataset with 914,871 users and 4,619,089 projects with a total of 12,871,935 interactive, the Table 2 is the detailed information of the data set.

Table 2. Indicators of dataset.

users	914871
items	4619089
Average actions of items	2. 786683215387973
The number of inters	12871935
The sparsity of the dataset	99. 99969540157988%

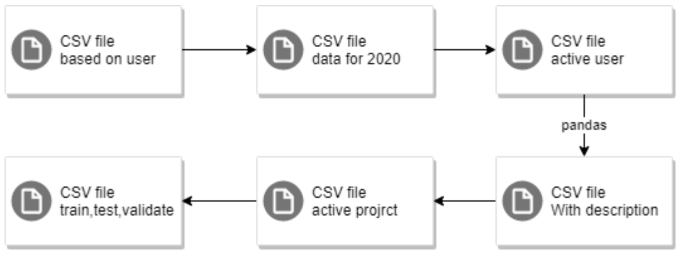

Fig. 2. Data cleaning process.

3.2 Result

The experimental results are shown in the Table 1. This article is based on the light-GCN model to make it more suitable For our experiment, we performed an aggregation operation on interaction-based data to alleviate the problem of extremely sparse data.

3.2.1 Compare model

In the integration based on the data set we built, we ran our data on the following models:

• **BPR**: A classic Bayesian personalized ranking based on implicit feedback.

• **NGCF**: An advanced model that uses neural networks to build user and project embeddings.

• **DGCF**: In the process of modeling, pay more attention to the user-project relationship of user intent.

• **LINE**: LINE's main research is to embed super-large information networks into bottom-dimensional vectors. We think this is consistent with our embedding process, so we also build a recommendation model based on this type. [5]

• **MultiDAE**: A collaboration that extends Variational Autoencoder (VAE) to be used for implicit feedback Filter method. [6]

• **LIGHT-GCN**: The basic method of embedding used in this article.

3.2.2 Compare result

Light-GCN is a variant model based on NGCF, and our model is based on the modification of Light-GCN,Therefore, this article focuses on comparing the relationship between the three, and the results obtained by other models are only for reference. As can be seen from the table, Light-GCN has made relatively high progress in various indicators compared to NGCF.

This has been demonstrated in the author's article, and compared to Light-GCn, our model,The verification results are better than the previous model, @recall increased by 0. 0066, @mrr increased Increased by 0. 0645, @ndcg increased by 0. 0111, @hit increased by 0. 008, @precision increased 0. 0003. More specific information is in Table 1.

4 Conclusion

This paper implements a recommendation model based on the improvement of Light-GCN and applies the model to the github item, In the purpose of recommendation, it shows good performance and can be applied to actual scenarios. At the same time, we built a large dataset based on user behavior log data of open source projects. In future research, we Talk about further improving our model to make it show better performance.

References

1. Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China. ACM, New York, NY, USA,10 pages. https://doi. org/10. 1145/3397271. 3401063
2. Steffen Rendle,et al,BPR: Bayesian personalized ranking from implicit feedback,Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence,2009,452-461.
3. Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. InProceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France. ACM, New York, NY, USA, 10 pages. https://doi. org/10. 1145/3331184. 3331267

4. Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China. ACM,New York, NY, USA, 10 pages. https://doi. org/10. 1145/3397271. 3401137 [5]Jian Tang. LINE: Large-scale Information Network Embedding,Proceedings of the 24th International Conference on World Wide Web,2015,1067-1077 DOI:https://doi. org/10. 1145/2736277. 2741093Conference Name:ACM Woodstock conferenceConference Short Name:WOODSTOCK'18
5. Jian Tang. LINE: Large-scale Information Network Embedding,Proceedings of the 24th International Conference on World Wide Web,2015,1067-1077 DOI:https://doi. org/10. 1145/2736277. 2741093
6. Dawen Liang,Variational Autoencoders for Collaborative Filtering,Proceedings of the 2018 World Wide Web Conference,2018,689-698. DOI:https://doi. org/10. 1145/3178876. 3186150Conference Name:ACM Woodstock conferenceConference Short Name:WOODSTOCK'18
7. Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR. Conference Location:El Paso, Texas USA
8. Felix Wu, Amauri H. Souza Jr. , Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In ICML. 6861–6871.
9. Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In SIGIR. 515–524
10. Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In WWW. 173–182. .
11. Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In WWW. 689–698.
12. Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In SIGIR. 165–174.
13. T. F. Cox and M. A. Cox. Multidimensional scaling. CRC Press, 2000.
14. J. B. Tenenbaum, V. De Silva, and J. C. Langford. Aglobal geometric framework for nonlinear dimensionality reduction. Science, 290(5500):2319–2323, 2000.
15. M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In NIPS, volume 14, pages 585–591, 2001.