GAT-Team: A Team Recommendation Model for Open-Source Software Projects

Qing Qi†, Jian Cao†*, Duo Wang†
*Corresponding author
†Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
Email: {qi_ng616, cao-jian, wd_doylle}@sjtu.edu.cn

Abstract—The active participation of contributors is key to the success of any open source software (OSS) projects. In the open source community, it can be found that some developers have participated together in multiple projects and also built social connections so that they form an independent team. Independent teams can be high-quality developer candidates for OSS projects. By modeling the developers and independent teams, we design approaches to recommend independent teams to OSS projects. We used the interaction data between developers to build the developers' social connection network, on which independent teams can be discovered by utilizing the community discovery algorithm. Based on the graph attention network, we designed a team recommendation model GAT-team. The performance of the GAT-team is evaluated on a real-world dataset. GAT-team achieves much better results compared with the other algorithms in the experiments. We provide the data and the code at https://doi.org/10.5281/zenodo.4646651.

Index Terms—Open Source Software Project, Independent Team, Team Recommendation, Graph Attention Network

I. Introduction

Finding and attracting developers with relevant expertise and interest to contribute to open-source software (OSS) projects is crucial to their sustainable development [1]. However, developers often find it hard to obtain suitable resources from tremendous OSS projects. This largely discourages their participation in OSS projects. Therefore, many approaches have been proposed to recommend developers to OSS projects [2]–[5].

Software development is a sophisticated task which inherently requires teamwork. Researchers have proposed approaches for team formation, task assignment, conflict management, collaboration mechanisms and information sharing [6]–[9]. Comparing with teamwork in traditional projects, an OSS project team is often composed of unknowns and volunteers [10], who are often distributed in different places. Therefore, improving the collaboration efficiency of virtual teams working on OSS projects is a more significant challenge [11].

In OSS communities, the same group of developers can be found having participated in several projects. If these developers have built social connections and cooperate with each other directly, they form an independent team. Compared with project teams that only focus on the tasks of the specific project, members of an independent team are more likely to share common interests and develop long-lasting cooperative relationships.

It can be assumed that independent teams have higher productivity so as to they can contribute significantly to the OSS projects. There are several reasons for it. Firstly, the social connections between team members are beneficial to their collaboration. Secondly, they have lower start-up costs. Corbitt et al. [12] studied the amount of time spent at each of the four classic team development stages (forming, storming, norming and performing for virtual teams), which is 30%, 18%, 17%, and 35%, respectively. This indicates that stable and long-lasting teams are more efficient than temporary teams because the formation cost of a virtual team is considerably large.

As an empirical study, we identified independent teams based on the data from GitHub and compared their contribution rates with other contributors (the approach to identify independent teams will be introduced later) The contribution data provided by GitHub REST APIs is adopted to measure members' contribution rates. We normalized the data by dividing them by the average contribution of all contributors in each repository. Then a student's t-test is applied to compare the contribution rates of independent team members and other contributors. The conclusion is the contribution rate of independent team member is expected to be 2.57-2.60 times higher than the average with 95% confidence.

This observation motivates us to recommend independent teams to OSS projects. Currently, few insights are available on recommending development teams to open-source projects. The contributions of this paper are as follows:

1) We provide an approach to identify independent teams in OSS communities.
2) We design GAT-team, a graph attention network based algorithm to recommend independent teams to OSS projects efficiently.
3) We perform extensive experiments on the real-world dataset and prove that GAT-team is significantly better than other models.

The paper is organized as follows. Section 2 reviews the related work. Section 3 presents an approach to discover independent teams in OSS communities. Section 4 introduces GAT-team, a team recommendation approach

DOI reference number: 10.18293/SEKE2023-084

for OSS projects. Experiments are reported in Section 5. Finally, we conclude the paper in Section 6.

II. Related Work

A. Teamwork in Open Source Software Projects

Researchers are interested in enhancing the efficiency of teamwork for OSS development. For example, in [13], the influence of developers' different involvements is investigated. Grottke et al. study the relationships between team factors and the efficiency of failure processing [14]. In [8], the mechanism of conflict management is proposed.

It is well known that relationship building and group cohesion are very important for virtual teams to be effective in accomplishing tasks. Research shows that developers are more likely to join projects that are initiated by those with whom they have socially connected in the past. In addition, social networks are the most common approach for finding cooperative groups [15]. McDonald found two social networks in an organization [16]. One is called the work group graph (WGG) which reflects shared work contexts. The other is based on the successive pile sort (SPS) which reflects workplace sociability. These two approaches to construct social networks are both adopted in our research. Moreover, Bird et al. [17] prove that the organizational structure in OSS projects can be reflected by the social connections among project participants. We therefore detect teams based on the resulting social networks. Our research is dedicated to recommending stable and context-independent developer teams.

B. Member Recommendation for Open Source Software Projects

Developers can be recommended to a project based on their expertise or past experience [2]. Many past research studies have focused on recommendation problems on GitHub [3]–[5]. A common approach is leveraging the interaction data between the developer and projects and the similarity data between projects to find projects similar to the ones on which the developer has already participated. In the work by Xu et al., it extracts TF-IDF features from the projects' text files to compute the similarity between projects. After this, the Create, Fork, and Star actions of the developers are mined, weights to these actions are assigned, and the interaction matrix between developers and projects is obtained [3]. Other works try to solve the problem with neural network models. For example, Zhou et al. point out that due to the severe sparsity of recommendation problems on GitHub, the tradition collaborative filtering cannot achieve satisfying results. From this observation, they propose the Hierarchical Collaborative Embedding Model, which compensates for the data sparsity by constructing a knowledge network among projects [4].

In this study, we will recommend independent teams rather than individual members because members of independent teams are more productive than individual members.

C. Group Recommendation

Team recommendation is also related to the group recommendation problem. The difference is group recommendation approaches try to recommend items to a group while team recommendation approaches try to recommend teams to a project. However, they share a similarity in that potential connections between a group (or a team) and an item (or a project) are predicted.

A traditional approach for the group recommendation problem is to use a predefined consensus function to aggregate the recommendation results for each member into the results for the group. Gartrell et al. use this tradition approach by combining the social connections between users into the consensus function, which achieves increased performance [18]. Recently, many works use the attention mechanism in group recommendation problems. The MoSAN model designed by Vinh Tran et al. assigns a child attention network for each member in the group and learns the preference of the members using the attention mechanism [19]. The work by Cao et al. combines neural collaborative filtering and the attention mechanism. The social connections between group members are also exploited to improve the performance [20]. Their work proves that the attention mechanism is superior to the several traditional consensus functions used in group recommendation such as Least Misery and Maximum Satisfaction.

III. Independent Teams in OSS Communities: Definition and Detection Approach

An independent team is a virtual team in OSS communities that has worked on more than one project and their members are socially connected. Developers on OSS platforms can have different types of social connections such as following each other, contributing to the same OSS project, and discussing in issue resolution processes. However, these connections are context-dependent and they do not necessarily indicate mutual connections between developers. For example, following can be uni-directional. Contributors under the same repository do not necessarily know each other, even if they have has jointly participated into the resolution process of the same issue. At the same time, developers can contact with each other beyond open source platforms, such as in google group or through emails. These communication channels may provide direct proofs that they are socially connected. Unfortunately, to obtain this data is not easy and has privacy issues.

Fortunately, there are other cooperation mechanisms provided by the open source platforms that can be directly adopted as an evidence, for example, mention action on GitHub. When you mention @ a GitHub username in the context of an issue or pull request, that person is notified and subscribed to future updates. When two developers

have mentioned each other, we can assume they have set up social connections. Therefore, our model relies on mention to detect social connections.

We can identify independent teams from OSS communities according to the definition. As the first step, we construct a developer social network (DSN). In this network: (1) Each node represents a developer. (2) For any two nodes, there is an edge linking them if and only if: a) the two developers have contacted with each other in some ways; b) the two developers have jointly contributed to multiple projects.

After constructing the DSN, we use community detection algorithms to detect teams with strong intra-connections and weak inter-connections in the network. Since a developer can belong to multiple independent teams, we choose the Order Statistics Local Optimization Method (OSLOM) to detect independent teams that are represented by overlapping communities [21].

The information used in the recommendation of independent teams includes: (1) Interest vectors of independent teams and developers; feature vectors of project repositories. (2) Project repositories to which each team or developer contributes to and their contribution values in each of the project repositories. (3) Teams to which each developer belongs, contribution rate and structural information (degree in the team sub-network of the DSN) of each member in each of the teams. (4) Social connections among developers (the DSN).

The feature vectors of project repositories include numeric features (size, forks, watchers and subscribers) and non-numeric features (language, topics).The interest vector of a developer is an aggregation of the feature vectors of all the project repositories on which the developer has worked. The contribution value of an independent team in a hosting project repository is the sum of the contribution values of all its members. The contribution rate of a team member is represented by the percentage of contributions the member made to the hosting project repositories of the team. The structural information of a team member is represented by its degree in the corresponding sub-network of the whole DSN.

IV. GAT-team: A GAT-based Team Recommendation Model

A. Model Design

Since the participation and collaboration of developers in OSS communities can be directly modeled as a graph, we try to make use of the graph neural network (GNN) to support team recommendation. Graph attention (GAT) network introduces attention mechanisms to the propagation step, which compute the hidden representations of each node in the graph, by attending over its neighbors, following a self-attention strategy. GAT is efficient because the operation is parallelizable across node-neighbor pairs. In addition, it can be applied to graph nodes having different degrees [22]. In order to thoroughly mine the relational information in OSS communities to support team recommendation, we designed a GAT-based team recommendation model (GAT-team). The architecture of the model is shown in Fig. 1.

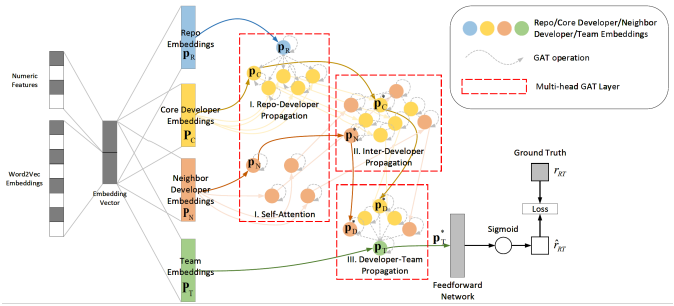


Fig. 1. The GAT-based Team Recommendation Model: GAT-team

The input features (Repo Embedding, Core Developer Embedding, Neighbor Developer Embedding and Team Embedding) consist of numeric features and Word2Vec embedding. For the three graph attention layers in the model, the input graphs are connections between the project repository and core developers, connections between core developers and their immediate neighbors, and connections between developers and teams.

Multi-head graph attention layers are used as basic units of GAT-team. In every multi-head graph attention layer, multiple parallel graph attention operations are computed on the input graph and the input embedding.

The outputs of the parallel graph attention units are aggregated and fed into the output attention unit to compute the final output embedding. For multi-head graph attention layers, common aggregation methods include averaging and concatenation. Our model uses concatenation. Finally, the output embedding of the multi-head graph attention layer is computed.

B. Propagation Mechanism

The first multi-head graph attention layer is responsible for the weight propagation between project repositories and core developers. The input graph contains the connections between the project repository and its core developers. It is necessary to mention that because a self-connection is attached to each node on the repository-developer graph, we compute self-attention for all the neighbor developers for the sake of consistency between the output embedding of core developers and neighbor developers. The second multi-head graph attention layer is responsible for the weight propagation between developers. The input graph is the DSN, which models the social connections between developers. And the output of inter-developer propagation is computed. The third multi-head graph attention layer is responsible for the weight propagation between developers and teams. The input graph is the connections between developers and teams. The output is the developer-team propagation.

C. Output Layer

The final module of the model consists of a feed forward network and a Sigmoid activation. The output of the model is computed by Equation (1):

$$\hat{r}_{\mathrm{RT}} = \sigma(\mathbf{A}_{ff}\mathbf{P}_{\mathrm{T}}^{*} + \mathbf{b}_{ff}) \tag{1}$$

where $\mathbf{A}_{ff}$ and $\mathbf{b}_{ff}$ are the learnable parameters of the model. We choose the binary cross entropy-loss as the loss function of GAT-team.

V. Experiments

A. Data Set

Our experiment is based on the data set of all the issue comment events on GitHub from 01/01/2015 to 06/30/2020. We also collect the information of 312,934 repositories and 1,384,736 users. The DSN is constructed according to the definition. The network has 719,202 nodes and 1,121,778 edges in total. A total number of 22,875 independent teams are found in the network.

B. Metrics

The discounted cumulative gain (DCG) measures the quality of the results in a ranked list, where items in that list are graded in some way. The grading is a relevance judgment for each result. DCG accumulated at a particular rank position p is defined in Equation (2):

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{2}$$

where rel_i is the relevance score of ith element.

By computing the DCG value of the optimal recommendation result, one gets the ideal discounted cumulative gain (IDCG) of a single query. Normalized Discounted Cumulative Gain (nDCG) is computed by dividing the DCG by the IDCG. The higher the nDCG, the better ranked list.

C. Comparison Approaches

We developed a series of algorithms to compare them with GAT-team. Moreover, we select a group recommendation approach SoAGREE [20] for comparison.

1) Interest-Vector-Based Recommendation: An intuitive approach is to compute the similarity between the feature vectors of the teams and the project repositories. Since the connections between developers and teams are known, group recommendation techniques can be employed to improve the performance.

Interest Matching: For each team in the dataset, we calculate their suitability to project repositories using the following approach. For an interest vector of a team (V_T) and a feature vector of a project repository (V_R): (1) For numeric features, calculate the Euclidean distance between the two vectors. (2) For non-numeric features A and B, calculate their Jaccard distance. (3) Calculate the distance between the two vectors by using the two aforementioned distances. (4) Take the reciprocal of the distance to get the similarity of the two vectors.

For each project repository, we sort the candidate teams in descending order of their suitability, and then take the first-k teams as the recommendation result.

Group Aggregation: In group recommendation problems, the interest of the team is an aggregate of interests of all members. Using the known information in the dataset, we designed the following aggregation methods: (1) Interest Aggregation: Aggregate the interest vectors of the members into the interest vector of the team. Calculate the similarity between the team's interest vector and the project repository's feature vector. (2) Mean Aggregation: After calculating the suitability of all the members, take the mean value as the suitability of the team. (3) Least Misery: After calculating the suitability of all the members, take the minimum as the suitability of the team. (4) Maximum Satisfaction: After calculating the suitability of all members, take the maximum as the suitability of the team. (5) Expertise-Based Aggregation: Take a weighted sum as the suitability of the team. We adopted two strategies to weight each member, i.e., weight assignment based on contribution rate and weight assignment based on members' degree in the team social network.

The recommendation algorithms designed based on the aforementioned aggregation methods are referred to as sim_interest, sim_mean, sim_lm, sim_ms, sim_exp_contri, and sim_exp_degree respectively in the experiments.

2) Core-Developer-Based Social Recommendation: Social connections between team members are essential information in team recommendation problems. From the aforementioned observations, we designed the Core-Developer-Based Social Recommendation algorithm: (1) For each repository, obtain a list of its core developers. (2) For each candidate team, obtain a list of its members. Count the number of social connections between team members and core developers of the repository. Take this number as the strength of the connection between the team and the repository. (3) Sort the candidate teams in the descending order of their connection strength with the repository. Take the first-k teams in the sequence as the recommendation results.

The core-developer-based social recommendation is referred to as Social in the experiments.

3) SoAGREE: We also compared GAT-Team with the recommendation model based on social connections and graph attention networks: Social-Enhanced Attentive Group Recommendation (SoAGREE) [20]. We used the implementation shared by the authors and adjusted the hyperparameters to fit the dataset used in this study.

D. Ablation Study

In order to verify the architecture of GAT-Team, we employed the ablation study method. We designed the

following two alternations of the original model to verify the validity of the three-layer structure:

(1) ALT1: Combine the Repo-Developer layer and the Inter-Developer layer. The data inputs to the new GAT layer are Repo Embedding and Developer Embedding. The input graph to the new GAT layer is a combination of the Repo-Developer graph and Inter-Developer graph.
(2) Combine the Repo-Developer layer, the Inter-Developer layer, and the Team-Developer layer. The data inputted to the new GAT layer are Repo Embedding, Developer Embedding, and Team Embedding. The graph inputted to the new GAT layer is a combination of the Repo-Developer graph, Inter-Developer graph, and Developer-Team graph.

The two alternatives are referred to as GAT-team_alt1 and GAT-team_alt2 in the experiments.

In summary, we implemented 11 different recommendation algorithms in total (Interest_team, Interest_mean, Interest_lm, Interest_ms, Interest_exp_contri, Interest_exp_degree, Social, GAT-team, GAT-team_alt1, GAT-team_alt2, SoAGREE). We tested all the algorithms on the test set and used nDCG as the evaluation metric.

E. Experiment Results

The results of the experiments are shown in Fig.2. Figures 2(a) through 2(k) show the distribution of nDCG values achieved by all the algorithms when the value of k is 30. Fig. 2(l) and Table I show the relationship between the average nDCG value achieved by the algorithms and the value of k. Table II shows the training errors and test errors of GAT-team, GAT-team_alt1, GAT-team_alt2, and SoAGREE.

As one can see from the experimental results, the GAT-team and the Core-Developer-Based Social Recommendation algorithm designed in our study achieve significantly better performance than other algorithms. For GAT-based models, the original architecture performs better than the two alternative architectures. While the performance of GAT-team is better than Social, GAT-team_alt2 is significantly out-performed by Social. Although SoAGREE achieves better performance than interest-vector-based recommendation algorithms, it is outperformed by the other algorithms by a large margin.

The success of GAT-team mainly lies in two aspects. Firstly, it effectively utilizes the information in a comprehensively way. Secondly, it thoroughly exploits the social connections with weight propagation and graph attention operations on the developer social network.

The SoAGREE model proposed by Cao et al. achieves outstanding performance on the MaFengWo dataset and CAMRa2011 [20], but it does not perform well in the experiments in our study. The reason is SoAGREE learns group embeddings only by aggregating the embeddings of users while neglecting more comprehensive information in OSS projects.

VI. Conclusion

This study focuses on how to recommend independent teams to OSS projects. For the first time, we suggest that independent teams can be high quality developer candidates for OSS projects. We designed and implemented a GAT-based recommendation model and compared it with the Interest-Vector-Based Recommendation algorithm, Core-Developer-Based Social Recommendation algorithm and SoAGREE, a recommendation model which works well on traveling information websites and movie rating websites. Our GAT-based model and the Core-Developer-Based Social Recommendation both achieve satisfying performance in the experiments, out-performing SoAGREE by a large margin. Additionally, we conducted an ablation study on the GAT-based model and verified the validity of our architecture.

References

[1] Zhang, Xunhui et al. "Devrec: a developer recommendation system for open source repositories." Mastering Scale and Complexity in Software Reuse: 16th International Conference on Software Reuse, ICSR 2017.

[2] Sun, Xiaobing et al. "Enhancing developer recommendation with supplementary information via mining historical commits." Journal of Systems and Software 134 (2017): 355-368.

[3] Xu, Wenyuan et al. "Scalable relevant project recommendation on GitHub." Proceedings of the 9th Asia-Pacific Symposium on Internetware. 2017.

[4] Zhou, Zili et al. "Knowledge-based recommendation with hierarchical collaborative embedding." Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part II 22. Springer International Publishing, 2018.

[5] Zhou, Yuqi et al. "Ghtrec: a personalized service to recommend github trending repositories for developers." 2021 IEEE International conference on web services (ICWS). IEEE, 2021.

[6] Majumder, Anirban et al. "Capacitated team formation problem on social networks." Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining. 2012.

[7] Soares, Daricélio M. et al. "What factors influence the reviewer assignment to pull requests?." Information and Software Technology 98 (2018): 32-43.

[8] Huang, Wenjian et al. "Effectiveness of conflict management strategies in peer review process of online collaboration projects." Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work Social Computing. 2016.

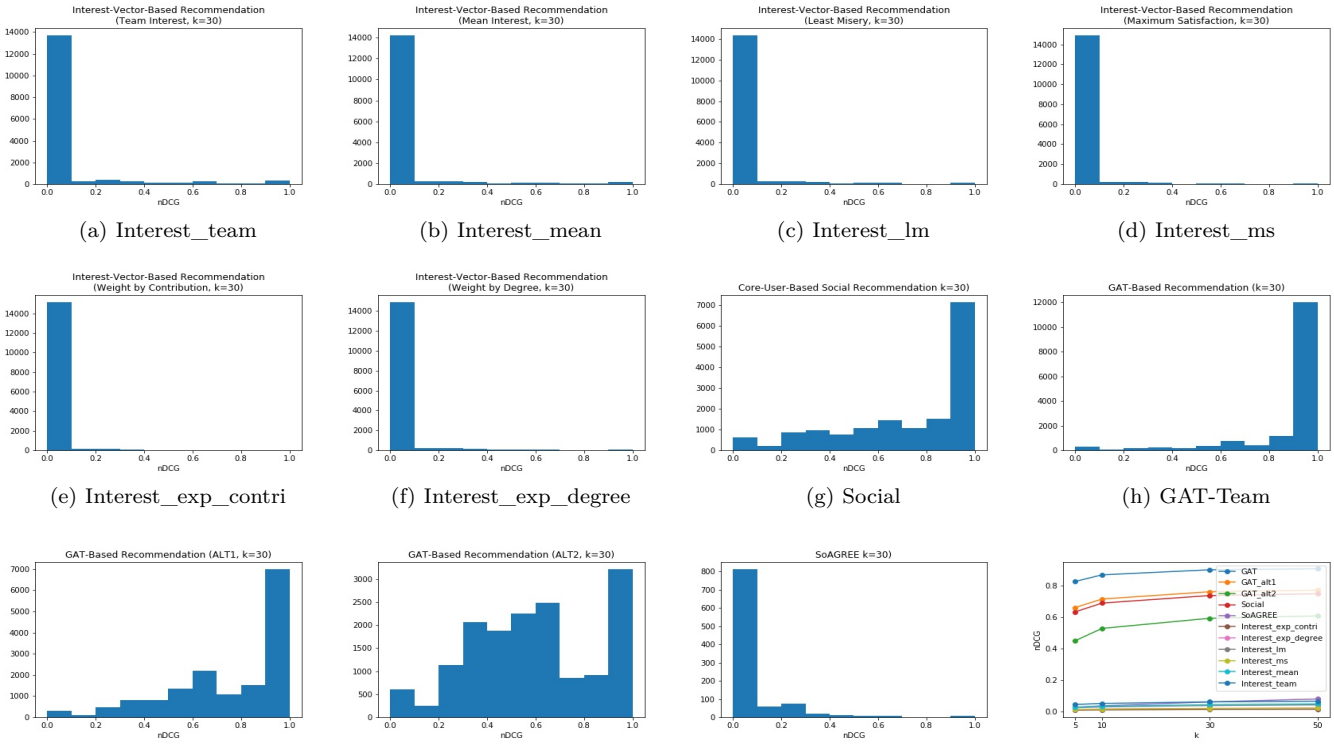
[9] Wang, Duo et al. "Investigating cross-repository socially connected teams on GitHub." 2019 26th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2019.

[10] Chopra, Deepti and Kaur, Arvinder. "Predicting Group Size for Software Issues in an Open-Source Software Development Environment." International Conference on Innovative Computing and Communications: Proceedings of ICICC 2020, Volume 2. Springer Singapore, 2021.

[11] Sohal, Amitpal Singh et al. "Trust in open source software development communities: A comprehensive analysis." Research Anthology on Usage and Development of Open Source Software. IGI Global, 2021. 200-220.

[12] Corbitt, Gail et al. "A comparison of team developmental stages, trust and performance for virtual versus face-to-face teams." Proceedings of the 37th Annual Hawaii International Conference on System Sciences. IEEE, 2004.

[13] Rebouças, Marcel et al. "How does contributors' involvement influence the build status of an open-source software project?." 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR). IEEE, 2017.



(a) Interest_team (b) Interest_mean (c) Interest_lm (d) Interest_ms

(e) Interest_exp_contri (f) Interest_exp_degree (g) Social (h) GAT-Team

(i) GAT-Team_alt1 (j) GAT-Team_alt2 (k) SoAGREE (l) Comparisons

Fig. 2. Experimental Results

TABLE I
Experimental Results

Algorithm/Model	avg. ndcg@k=5	avg. ndcg@k=10	avg. ndcg@k=30	avg. ndcg@k=50
Interest_team	0.04306	0.04986	0.05987	0.06415
Interest_mean	0.02597	0.03182	0.04187	0.04699
Interest_lm	0.02319	0.02805	0.03657	0.04069
Interest_ms	0.01065	0.01325	0.01688	0.01887
Interest_contri	0.00678	0.00836	0.01096	0.01208
Interest_degree	0.01184	0.01433	0.01847	0.02066
Social	0.63339	0.68949	0.73760	0.74919
SoAGREE	0.02511	0.03466	0.05931	0.07837
GAT-team	0.82743	0.86948	0.90172	0.90812
GAT-team_alt1	0.66080	0.71542	0.76180	0.77193
GAT-team_alt2	0.44937	0.52811	0.59278	0.60764

TABLE II
Training and Testing Error

Model	training error	test error
GAT-team	0.01431	0.01966
GAT-team_alt1	0.02733	0.03712
GAT-team_alt2	0.04592	0.07248

[14] Grottke, Michael et al. “Team factors and failure processing efficiency: An exploratory study of closed and open source software development.” 2010 IEEE 34th Annual Computer Software and Applications Conference. IEEE, 2010.

[15] Yin, Xiaoyan et al. “Social connection aware team formation for participatory tasks.” IEEE Access 6 (2018): 20309-20319.

[16] McDonald, David W. “Recommending collaboration with social networks: a comparative evaluation.” Proceedings of the SIGCHI conference on Human factors in computing systems. 2003.

[17] Bird, Christian et al. “Latent social structure in open source projects.” Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering. 2008.

[18] Gartrell, Mike et al. “Enhancing group recommendation by incorporating social relationship interactions.” Proceedings of the 16th ACM international conference on Supporting group work. 2010.

[19] Vinh Tran, Lucas et al. “Interact and decide: Medley of sub-attention networks for effective group recommendation.” Proceedings of the 42nd International ACM SIGIR conference on research and development in information retrieval. 2019.

[20] Cao, Da et al. “Social-enhanced attentive group recommendation.” IEEE Transactions on Knowledge and Data Engineering 33.3 (2019): 1195-1209.

[21] Lancichinetti, Andrea et al. “Finding statistically significant communities in networks.” PloS one 6.4 (2011): e18961.

[22] Velickovic, Petar et al. “Graph attention networks.” stat 1050.20 (2017): 10-48550.