Team Expansion in Collaborative Environments

Lun Zhao[1], Yuan Yao[1], Guibing Guo[2], Hanghang Tong[3], Feng Xu[1], and Jian Lu[1]

[1]State Key Laboratory for Novel Software Technology, Nanjing University, China
[2]Software College, Northeastern University, China
[3]Arizona State University, USA
zhaolun@smail.nju.edu.cn,{y.yao,xf,lj}@nju.edu.cn
guogb@swc.neu.edu.cn,hanghang.tong@asu.edu

Abstract. In this paper, we study the team expansion problem in collaborative environments where people collaborate with each other in the form of a team, which might need to be expanded frequently by having additional team members during the course of the project. Intuitively, there are three factors as well as the interactions between them that have a profound impact on the performance of the expanded team, including (1) the task the team is performing, (2) the existing team members, and (3) the new candidate team member. However, the vast majority of the existing work either considers these factors separately, or even ignores some of these factors. In this paper, we propose a neural network based approach TECE to simultaneously model the interactions between the team task, the team members as well as the candidate team members. Experimental evaluations on real-world datasets demonstrate the effectiveness of the proposed approach.

Keywords: Team expansion, candidate member prediction, collaborative environments, neural networks

1 Introduction

In many application domains, people tend to frequently collaborate with others in the form of teams for specific tasks. For example, in open source software community, developers distributed worldwide work with each other by forming developer teams for specific projects; in research community, researchers form research teams and they collaborate with each other for research projects/papers; in the film industry, the crew works together as a team for a film shooting. In this work, we refer to these application domains as *collaborative environments*.

One of the key features in these collaborative environments is *team mobility*, i.e., teams are formed upon specific tasks and individuals can participate in several teams depending on their own interest and capabilities. In such environments, we usually need to expand the team by adding new members during team formation or when the existing team encounters difficulties on the task. In this paper, we put our focus on the team expansion problem in collaborative environments.

Broadly speaking, team expansion is related to several existing lines of research (please refer to the related work section for more details): (1) people and task matching which searches for an optimal match between people capabilities and task requirements [3, 23], (2) recommendation which recommends items to users [18, 11], and (3)

social proximity analysis which computes the proximities between users [21, 13]. However, they all suffer from some limitations for the team expansion problem studied in this paper: (1) people and task matching methods need concrete descriptions about people capabilities and task requirements, while such descriptions are usually unavailable; (2) recommendation methods mainly focus on recommending items for users, while team expansion aims to recommend users for items (i.e., tasks) where the 'chemistry' between existing users and the new user matters; (3) social proximity analysis methods analyze the social connections between users, while the matching between users and tasks are widely ignored.

In this paper, we propose a neural network based approach (TECE) for team expansion in collaborative environments. The basic considerations of TECE are three-fold: (1) no concrete requirements and capabilities for the candidate team member are needed, (2) the candidate should match the task, and (3) the candidate should match the existing team members. To this end, we propose to automatically match the candidate members to both team tasks and existing team members, based on the existing interactions between them. To match a candidate member with the given task, we exploit the collaborative filtering idea from recommender systems by mining the existing interactions between individuals and tasks; to match a candidate member with the existing team members, we take the team leader as a proxy of the team members and incorporate the social connections between the candidate and the team leader into the model. Additionally, we adopt deep models with multiple non-linear neural layers to capture the complex relationships between candidate members, team leaders, and team tasks.

The main contributions of this paper include:

- We formally define the team expansion problem in collaborative environments, which has a wide range of applications.
- We propose the TECE model to solve the team expansion problem. The proposed TECE simultaneously considers three important factors (team task, existing team members, and candidate team member) as well as their interactions.
- Experimental results on two real-world datasets show that the proposed method can outperform several competitors in terms of accurately identifying candidate members. For example, TECE can achieve up to 22.1% improvement compared with its best competitors.

The rest of the paper is organized as follows. Section 2 defines the team expansion problem. Section 3 describes the proposed approach. Section 4 presents the experimental results. Section 5 covers related work, and Section 6 concludes.

2 Problem Statement

In this section, we present the problem definition. Without loss of generality, we assume that each team corresponds to a unique task. Therefore, we use t to denote both the team task and the team itself. We assume that there are m teams/tasks and n unique individuals, and we use $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ and $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ to denote the set of teams and individuals, respectively. The existing interactions between team tasks and individuals are contained in matrix $\mathbf{R}$. For example, $\mathbf{R}(t, i) = 1$ means that individual i

belongs or once belonged to team t, and $\mathbf{R}(t, i) = 0$ indicates otherwise. For a specific team t, we use $\mathcal{I}_t$ to denote the existing members in the team. Specially, we assume that there is a team leader $o_t \in \mathcal{I}_t$ (e.g., the owner of the team) in each team. With the above notations, we define the team expansion problem in collaborative environments as follows.

Problem 1. Team Expansion Problem in Collaborative Environments (TECE)

Given: (1) a collection of teams/tasks $\mathcal{T}$ each of which has a team leader o_t, (2) a collection of individuals $\mathcal{I}$, (3) the existing interactions $\mathbf{R}$ between teams/tasks and individuals, and (4) a team $t_{test} \in \mathcal{T}$ that is about to expand;
Find: the candidate member to join team t_{test}.

The above team expansion problem can be formulated as estimating the fitness scores of unobserved entries in $\mathbf{R}$, which resembles the recommendation problem. However, different from traditional recommendation problem, when recommending a candidate member to a team, we need to pay special attention to the 'chemistry' between the candidate member and existing team members. The goal of team expansion is to generate a ranked list of candidates for the team that is about to expand. The ranked list is determined by the estimated scores of unobserved entries in $\mathbf{R}$. For example, suppose team t is the team to expand. We take team t itself, the team leader o_t, and a candidate team member i as input, and the goal is to learn a mapping function f to obtain the estimated fitness score between candidate i and the team t with leader o_t. Formally, we have $\hat{\mathbf{R}}(t, i) = f(t, o_t, i|\Theta)$, where $\hat{\mathbf{R}}(t, i)$ denotes the estimated fitness score, and Θ contains the model parameters. In the next section, we will show how we construct the mapping function f and learn its parameters.

3 The Proposed Approach

In this section, we present the proposed TECE model, followed by some discussions and generalizations.

3.1 The TECE Model

Figure 1 shows the overview of the proposed approach. As we can see, TECE takes (the one-hot encodings of) team task id, team leader id, and candidate member id as input, and embeds each of them into a low-dimensional vector. After that, the resulting embeddings are fed into several non-linear layers to learn the complex interactions between them. To be specific, TECE exploits the collaborative filtering idea from recommender systems to model the interactions between candidate team members and team tasks; it treats the team leader as a proxy of the team members and model the social interactions between candidate members and team leaders; it also models the interactions between team tasks and team members as team state which could impact the ideal candidate member. The last layer contains the final high-level features from the above interactions for predicting the fitness score between the input candidate and the input team (with its team leader). Finally, the output layer produces the estimated fitness score.

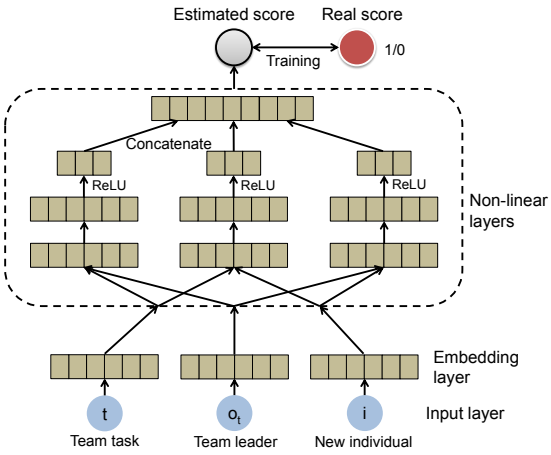


Fig. 1. The overview of TECE.

Next, we present the details of the TECE model. As mentioned in introduction, team expansion needs to consider the interactions between candidate member, team task, and existing team members. In the following, we first separately consider these interactions, and then describe the neural network based objective functions.

(A) Matching Candidate Team Member with Team Task. For the matching between candidate team member and team task, we first denote the embedding vectors/features of team task and candidate member as $\mathbf{p}_t \in \mathbb{R}^d$ and $\mathbf{q}_i \in \mathbb{R}^d$, respectively, where d is the vector dimensionality. Then, we model the interaction vector $\mathbf{r}_{ti}$ between team task t and candidate i as follows,

$$\mathbf{r}_{ti} = \mathbf{p}_t \odot \mathbf{q}_i \tag{1}$$

where $\odot$ denotes element-wise multiplication. For simplicity, we assume all the embedding vectors are row vectors in this paper.

(B) Matching Candidate Team Member with Existing Team Members (Team Leader). For the matching between candidate member and existing members, we pay special attention to the team leader as he/she plays important roles in building the team spirit and improving team performance [4, 19]. Therefore, we use the team leader as a proxy of the team members in this work. Based on this, we model the interactions between team leader o_t and candidate i as the likelihood of their cooperation in the current team. We use $\mathbf{c}_{ti}$ to stand for the cooperation likelihood, and it can be computed as

$$\mathbf{c}_{ti} = \mathbf{q}_i \odot \mathbf{q}_{o_t} \tag{2}$$

where we still use the element-wise multiplication, and $\mathbf{q}_i$ and $\mathbf{q}_{o_t}$ are the embedding vectors for candidate i and team leader o_t, respectively. Note that team leader is one of the individuals in $\mathcal{I}$.

(C) Team State Building. In addition to matching the candidate with team task and team leader, we consider to build the team state vector which describes the interactions

between team task and team leader. Such interactions could impact what is the best way the ideal candidate member should interact with team state. For example, if the current team state is harmonic, the ideal candidate should maintain the existing 'chemistry'; if there is a lack of timely communication in the current team state, the ideal candidate should act as a communication bridge. Specially, we define the team state vector $\mathbf{s}_t$ as

$$\mathbf{s}_t = \mathbf{p}_t \odot \mathbf{q}_{o_t} \tag{3}$$

where $\mathbf{p}_t$ is the embedding vector of team task t, and $\mathbf{q}_{o_t}$ is the embedding vector of team leader o_t.

(D) Single-layer Modeling. Next, based on the above three intermediate vectors, we can build a neural network to compute the fitness score $\hat{\mathbf{R}}(t, i)$. In the simplest case, we can concatenate these vectors and feed the resulting vector into a dense (fully-connected) layer, whose output score $\hat{\mathbf{R}}(t, i)$ can be computed as

$$\hat{\mathbf{R}}(t, i) = f^{out}([\mathbf{r}_{ti}, \mathbf{c}_{ti}, \mathbf{s}_t]\mathbf{x}^T) \tag{4}$$

where $\mathbf{x}$ is the weight vector, and f^{out} can be set as the sigmoid function $f^{out}(x) = \frac{1}{1+e^{-x}}$.

Note that for the network we describe in Eq. (4), it can be seen as a generalization of the traditional collaborative filtering based recommendation. It can be degenerated to traditional collaborative filtering if we take only the $\mathbf{r}_{ti}$ as input, define $\mathbf{x}$ as a row vector of $1s$, and define f^{out} as the identify function. In other words, compared to traditional recommendation, we further consider the current team state ($\mathbf{s}_t$) and the interactions between candidate members and team leaders ($\mathbf{c}_{ti}$).

(E) Multi-layer Modeling. For single-layer network, simply using a vector concatenation as final features is insufficient to represent the complex interactions between candidate team members, team leaders, and team tasks. To address this issue, we add multiple non-linear layers to model these interactions. Take the cooperation likelihood vector $\mathbf{c}_{ti}$ as an example, the multi-layer modeling for fitness score $\hat{\mathbf{R}}(t, i)$ can be defined as follows:

$$\begin{aligned}
\mathbf{z}^{(1)} &= f^{(1)}(\mathbf{W}^{(1)}\mathbf{c}_{ti} + \mathbf{b}^{(1)}) \\
\mathbf{z}^{(2)} &= f^{(2)}(\mathbf{W}^{(2)}\mathbf{z}^{(1)} + \mathbf{b}^{(2)}) \\
&\cdots \\
\mathbf{z}^{(L)} &= f^{(L)}(\mathbf{W}^{(L)}\mathbf{z}^{(L-1)} + \mathbf{b}^{(L)}) \\
\hat{\mathbf{R}}(t, i) &= f^{out}(\mathbf{z}^{(L)}\mathbf{x}^T)
\end{aligned} \tag{5}$$

where L is the layer number, $\mathbf{W}^{(i)}$, $\mathbf{b}^{(i)}$, and $f^{(i)}$ denote the weight matrix, bias vector, and activation function for the corresponding layer, respectively. f^{out} and $\mathbf{x}$ are defined in Eq. (4). For the activation function, we choose the ReLU (Rectified Linear Units) function for $f^{(i)}$. Similarly, we can add multiple non-linear layers for each of the three intermediate vectors as defined in Eq. (1) - (3), and we may also add multiple non-linear layers for the concatenated vector as defined in Eq. (4).

(F) Objective Function. Finally, we define the objective function to learn the embedding vectors $\mathbf{p}_t$, $\mathbf{q}_i$, and $\mathbf{q}_{o_t}$, as well as the other model parameters Θ. Specially, we adopt the following logistic-like objective function

$$\underset{\mathbf{P},\mathbf{Q},\Theta}{\operatorname{argmax}} \sum_{(t,i)\in\mathbf{R}} \mathbf{R}(t, i) \log(\hat{\mathbf{R}}(t, i)) + (1 - \mathbf{R}(t, i)) \log(1 - \hat{\mathbf{R}}(t, i)) \tag{6}$$

where $\mathbf{P}$ contains the embedding vectors $\mathbf{p}_t$, $\mathbf{Q}$ contains the embedding vectors $\mathbf{q}_i$ and $\mathbf{q}_{o_t}$, $\mathbf{R}(t,i)$ is the real fitness score between team task t and candidate i, and $\hat{\mathbf{R}}(t,i)$ is estimated fitness score by our model.

Here, the ground truth is contained in the existing interaction matrix $\mathbf{R}$ with $\mathbf{R}(t,i) = 1$ as positive label and $\mathbf{R}(t,i) = 0$ as negative label. The remaining problem is to sample the negative (t,i) pairs. Directly choosing all the possible (t,i) pairs is computational expensive (quadratic time complexity). Choosing only the positive labels (i.e., $\mathbf{R}(t,i) = 1$) would lead to trivial solutions (i.e., the feature values towards infinity). In this work, we keep all the positive labels, and randomly sample r (sampling ratio) negative labels (e.g., $\mathbf{R}(t,j) = 0$) in terms of team task t for each positive label. Based on the sampling strategy, a stochastic gradient ascent learning algorithm can be applied for optimization.

3.2 Generalizations and Discussions

The TECE model is open for some reasonable adjustments on the model architecture. Here, we discuss some possible generalizations of the proposed method.

First, the proposed model is flexible with several special cases. For example, we may consider the single-layer objective function as discussed above. Such treatment may improve the training efficiency while lower the prediction accuracy. Besides, for the three inputs, we can delete either the team task or the team leader which will degenerate the model to social proximity model and recommendation model, respectively. We will experimentally evaluate some the above special cases in the next section.

Second, as a common practice in neural networks, we use element-wise multiplication in Eq. (1) - (3) to model the interactions between team task, team leader, and candidate team member. In addition to this operation, other operations such as inner product, concatenation, average pooling, and max pooling can also be used. In fact, we can simultaneously use element-wise multiplication and concatenation to obtain more intermediate vectors. In this work, we omit such extensions for brevity.

Third, in our model, when matching the candidate member with team members, we take team leader as a proxy. We can also incorporate all the existing team members into the model by average pooling or max pooling. Take average pooling as an example. We can flatten the embedding vectors of existing team members by computing the average vector. We will experimentally evaluate this in the experimental section.

Fourth, in this work, we employ the team-individual interaction history only as input for the team expansion problem. Actually, we can consider much richer information such as the text descriptions of team tasks, member profiles, and temporal effects when such information is available. We leave these extensions as future work.

4 Experiments

4.1 Experimental setup

Datasets. We conducted experiments on two real datasets: GitHub[1] and DBLP[2]. GitHub is an open-source software development platform. The data contains the information

[1] http://ghtorrent.org/downloads.html.
[2] https://cn.aminer.org/billboard/citation.

Table 1. Statistics of the datasets.

Dataset	# of team tasks	# of individuals	Avg. members per team	Avg. teams per individual
GitHub	10,505	30,258	31.56	10.96
DBLP	17,838	29,423	3.59	2.18

about projects, developers, and the actions from developers to projects. Since there are many toy projects and inactive users, we filter the data by deleting the developers who have contributed to less than five projects, and the projects whose 'star' is no more than five and whose team member number is less than five. DBLP is an open database that collects scientific articles in the field of computer science [20]. We treat each article as a team task, authors as team members, and the first author as the team leader. We use a subset of the whole dataset, and the subset contains the areas of Data Mining, Machine Learning, Database, and Artificial Intelligence. Similar to the processing steps above, we filter out the authors who have published less than three articles, and the articles whose author number is less than three. Overall, both software development and research article publication can be seen as collaborative environments where people frequently collaborate with others in the form of teams for specific tasks. Both datasets are publicly available, and the statistics are listed in Table 1.

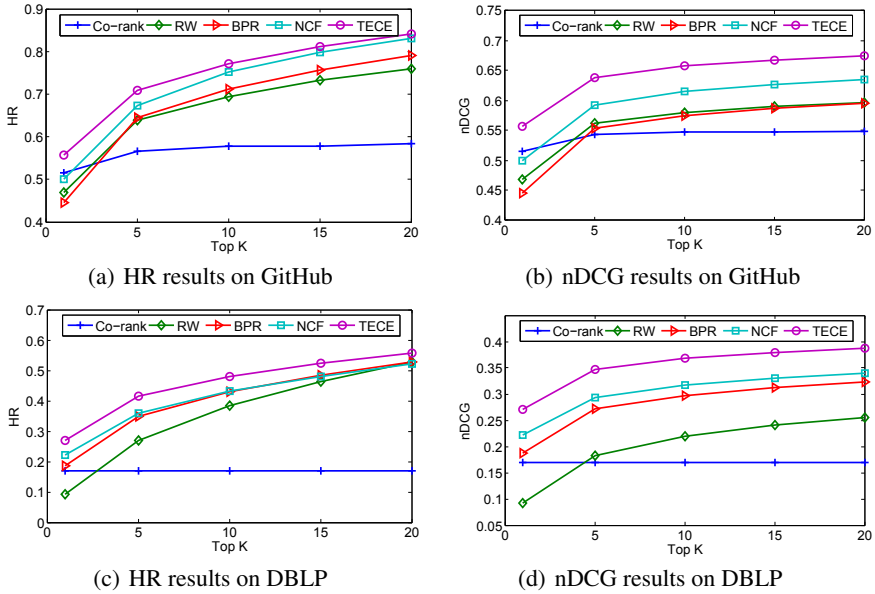
Compared Methods. We compare TECE with the following methods including two social proximity analysis methods and two recommendation methods.

- Co-rank. It is a heuristic method. The basic idea is to rank the candidate team members based on their cooperation times with the team leader.
- RW [21]. This is a random walk method that computes the proximities between users in a network. We adapt the method to the bipartite network of individuals and tasks.
- BPR [17]. This is a classic recommendation model for the one-class feedback case. It directly optimizes the rankings between an observed feedback and an unobserved feedback.
- NCF [9]. NCF is a recent neural network based model designed for one-class recommender systems. It treats the recommendation problem as a binary classification problem, and adopts neural networks to model the interactions between users and items.

Evaluation Metrics. To evaluate the effectiveness of the compared methods, we adopt the following two widely used evaluation metrics. Specifically, we output the top K candidate members in a ranked list, and compute the HR and nDCG metrics as follows.

$$HR@K = \frac{1}{|T|}\sum_{t=1}^{|T|} hit_t, \qquad nDCG@K = \frac{1}{|T|}\sum_{t=1}^{|T|} \frac{\log 2}{\log(r_t + 1)}$$

where T is the test set of teams, $hit_t \in \{0, 1\}$ is a binary value indicating whether the ground-truth candidate is in the top K list, and $r_t \in \{1, 2, ..., K\}$ is the ranking of the ground-truth candidate in the ranked list. $r_t = 0$ if the candidate is not in the top K list. In this work, we set K to 1, 5, 10, 15, and 20.



(a) HR results on GitHub (b) nDCG results on GitHub

(c) HR results on DBLP (d) nDCG results on DBLP

Fig. 2. Effectiveness comparisons. TECE generally outperforms the compared methods in both HR and NDCG on both datasets.

Experimental Settings. To setup the experiments, we randomly select one team member in each team as the test set. The training data is used to train the model where a ranked list of possible candidates for each team will be generated. Here, we basically assume that the existing member is a suitable match for the corresponding team task. While this is not always true in reality, we mitigate the issue by filtering the datasets and keeping the teams that can be considered as good teams (e.g., software projects that have been starred several times, or scientific papers that have been published in top venues). During the testing stage, since predictions on all the candidates would be time-consuming, we randomly select 100 negative samples for each ground-truth candidate.

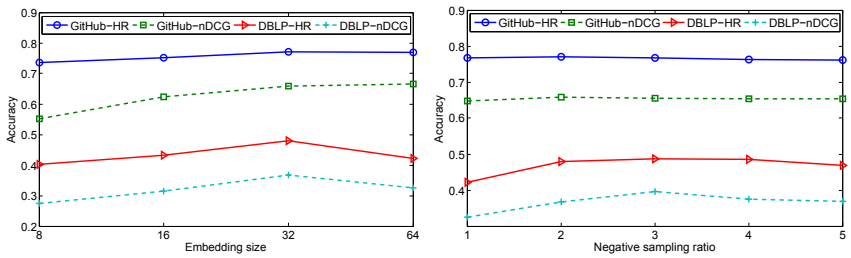
For the parameters, we either follow the default setting or set them equally. For example, we set the embedding dimensionality to 32 for all the methods (Co-rank and RW are not applicable). For the two non-linear layers of TECE, we set the embedding size as 16 and 8, respectively.

4.2 Experimental Results

(A) Effectiveness Comparisons. We first compare TECE with the existing methods, and report the results in Figure 2. First of all, we can observe from the figures that the proposed TECE generally outperforms the compared methods in terms of the two evaluation metrics on both datasets. For example, TECE improves the best competitor (NCF) by up to 11.3% on the GitHub data, and by up to 22.1% on the DBLP data. Basically, TECE is better than NCF and BPR as it further considers the matching between

Table 2. Performance gain analysis. The team leader, team task, and multi-layer modeling are all helpful to improve the prediction accuracy.

Method	TECE	TECE-tc	TECE-lc	TECE-sl	TECE-em	TECE-con
HR@10	0.7711	0.7359	0.7556	0.7550	0.7351	0.7299
nDCG@10	0.6579	0.5962	0.6404	0.6553	0.6040	0.5949



(a) Embedding size d (b) Negative sampling ratio r

Fig. 3. Parameter study. TECE is robust to the two parameters in a relatively wide range.

candidate member and team leader; TECE is better than RW and Co-rank as it further considers the matching between candidate member and team task.

Second, we can also observe that the overall results on the GitHub data are better than those on the DBLP datasets. This is due to the fact that the DBLP data is much sparser, making the problem more challenging. Third, for the Co-rank method, it can identify the candidate team member with a relatively high accuracy at top 1 ($K = 1$), while the accuracy increases slowly as K increases. The reason is that Co-rank can only find the candidates in a local neighborhood (i.e., previously cooperated), while the global perspective of task-candidate matching is ignored.

(B) Performance Gain Analysis. Next, we analyze the proposed TECE by checking the performance of its components and variants. For its components, we delete the team leader, team task, and multi-layer modeling of TECE, and obtain the TECE-tc (task and candidate), TECE-lc (leader and candidate), and TECE-sl (single-layer) method, respectively. For its variants, we consider to substitute team leader with the average-pooling on all the existing team members, and substitute the element-wise product with concatenation. The resulting methods are referred to as TECE-em (existing member) and TECE-con (concatenation), respectively. For brevity, we report the HR@10 and nDCG@10 results on the GitHub data in Table 2.

First, we can observe from the table that all the three components (i.e., team leader, team task, and multi-layer) of TECE are helpful to improve the prediction accuracy. For example, when team leader, team task, and multi-layer modeling are incorporated, the HR@10 performance of TECE improves by 4.8%, 2.1%, and 2.1%, respectively. Second, TECE is better than its two variants TECE-em and TECE-con. The improvement over TECE-em indicates the important role of the team leader, and the improvement over TECE-con indicates that the element-wise multiplication is a better way to model the interactions between team tasks, team leaders, and candidate team members.

(C) Parameter Study. Finally, we conduct a parameter study of the proposed method in terms of the embedding size d and the negative sampling ratio r. The results are shown in Figure 3 where we still report the HR@10 and nDCG@10 results. As we can see from Figure 3(a), the prediction accuracy generally improves when the embedding size d grows from 8 to 32. No significant improvement can be observed when d becomes larger. For the negative sampling ratio r, slight improvement can be observed when $r = 3$ for the DBLP data. In general, TECE is robust to the two parameters in a relatively wide range. In this paper, we fix embedding size d to 32 and sampling ratio r to 2 for simplicity.

5 Related Work

In this section, we briefly review the related work including people and task matching, recommender systems, social proximity analysis, etc.

People and Task Matching. In the operations research community, the people and task matching problem has been extensively studied [5, 3, 23]. Typically, the matching problem is often formulated as an integer linear program, and the goal is to search for an optimal match between people capabilities and task requirements. This line of work needs explicit and concrete descriptions of people capabilities and task requirements, while such descriptions are usually unavailable or inaccurate in many real applications.

Recommender Systems. Team expansion is related to recommender systems. One of the branches in recommender systems takes collaborative filtering as the model basis, and recommends items to users based on the existing interactions/feedback between users and items [11, 17]. Later, some researchers further incorporate social connections between users into the model [15, 25], and some others adopt deep neural networks. For example, normal deep networks [9], stacked auto-encoder [22], convolutional neural network [10], and recurrent neural network [24] have been used for modeling the user feedback, item content, temporal effect, etc. Different from the existing recommendation methods whose goal is to recommend items to users, team expansion aims to recommend users to items/tasks, where the 'chemistry' between existing users and the candidate user matters.

Social Proximity Analysis. Since the social connections between the candidate member and the existing members matter for the team expansion problem, our work is also related to existing social proximity analysis work [21, 14, 2]. For example, Tong et al. [21] propose fast random walks based on which the social proximity between two nodes in a network can be computed; Cummings and Kiesler [6] find that prior working experience is the best predictor for the collaborative tie strength; recently, Han and Tang [8] propose the social group invitation problem, and solve the problem from a group evolution viewpoint. This line of work mainly focuses on the proximity analysis between users in the social network, while the matching between individuals and tasks are widely ignored.

Team Formation and Optimization. The team formation or team expansion problem has been studied in the entrepreneurial context [5, 7], where interpersonal attraction, knowledge and communication skills make the essential factors for a successful team. In computer science, the team formation problem [1, 16] has also been studied.

However, existing work still requires the explicit descriptions of task requirements and user skills. The most related work is perhaps the recent work by Li et al. [12]. They propose to reformulate the team expansion problem as a team replacement problem by defining a virtual member with the desired skill set and communication structure, and then replacing this member with a most similar substitute. In contrast to their work, we do not require the descriptions of skill set and communication structure, and our focus is to find a candidate team member by exploiting the interactions between candidates, team leaders, and team tasks.

6 Conclusions

In this paper, we have proposed the team expansion problem in collaborative environments, and proposed a neural network based approach TECE for the problem. The key idea of TECE is to match the candidate team member with both team task and team leader. Additionally, TECE models the non-linear interactions between them via a multi-layer architecture. Experimental evaluations on real-world datasets demonstrate that the proposed approach can outperform several competitors in terms of accurately identifying candidate members. Future directions include exploring richer information such as task descriptions and member profiles for the team expansion problem, and handling the cold-start cases of the team expansion problem.

Acknowledgments. This work is supported by the National Natural Science Foundation of China (No. 61690204, 61672274, 61702252), the National Key Research and Development Program of China (No. 2016YFB1000802), the Fundamental Research Funds for the Central Universities (No. 020214380033), and the Collaborative Innovation Center of Novel Software Technology and Industrialization. Guibing Guo is partially supported by the National Natural Science Foundation for Young Scientists of China (No. 61702084). Hanghang Tong is partially supported by NSF (IIS-1651203, IIS-1715385, CNS-1629888 and IIS-1743040), DTRA (HDTRA1-16-0017), ARO (W911NF-16-1-0168), and gifts from Huawei and Baidu.

References

1. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Online team formation in social networks. In: WWW. pp. 839–848. ACM (2012)
2. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: WSDM. pp. 635–644. ACM (2011)
3. Baykasoglu, A., Dereli, T., Das, S.: Project team selection using fuzzy optimization approach. Cybernetics and Systems: An International Journal 38(2), 155–185 (2007)
4. Bradley, J.H., Hebert, F.J.: The effect of personality type on team performance. Journal of Management Development 16(5), 337–353 (1997)
5. Chen, S.J., Lin, L.: Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. IEEE Transactions on Engineering Management 51(2), 111–124 (2004)

6. Cummings, J.N., Kiesler, S.: Who collaborates successfully?: prior experience reduces collaboration barriers in distributed interdisciplinary research. In: CSCW. pp. 437–446. ACM (2008)
7. Forbes, D.P., Borchert, P.S., Zellmer-Bruhn, M.E., Sapienza, H.J.: Entrepreneurial team formation: An exploration of new member addition. Entrepreneurship Theory and Practice 30(2), 225–248 (2006)
8. Han, Y., Tang, J.: Who to invite next? predicting invitees of social groups. In: AAAI. pp. 3714–3720 (2017)
9. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: WWW. pp. 173–182 (2017)
10. Kim, D., Park, C., Oh, J., Lee, S., Yu, H.: Convolutional matrix factorization for document context-aware recommendation. In: RecSys. pp. 233–240. ACM (2016)
11. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)
12. Li, L., Tong, H., Cao, N., Ehrlich, K., Lin, Y.R., Buchler, N.: Enhancing team composition in professional networks: Problem definitions and fast solutions. IEEE Transactions on Knowledge and Data Engineering 29(3), 613–626 (2017)
13. Li, L., Yao, Y., Tang, J., Fan, W., Tong, H.: Quint: On query-specific optimal networks. In: KDD. pp. 985–994. ACM (2016)
14. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the Association for Information Science and Technology 58(7), 1019–1031 (2007)
15. Ma, H., Yang, H., Lyu, M.R., King, I.: Sorec: social recommendation using probabilistic matrix factorization. In: CIKM. pp. 931–940. ACM (2008)
16. Rangapuram, S.S., Bühler, T., Hein, M.: Towards realistic team formation in social networks based on densest subgraphs. In: Proceedings of the 22nd international conference on World Wide Web. pp. 1077–1088. ACM (2013)
17. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: UAI. pp. 452–461. AUAI Press (2009)
18. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM 40(3), 56–58 (1997)
19. Soomro, A.B., Salleh, N., Mendes, E., Grundy, J., Burch, G., Nordin, A.: The effect of software engineers? personality traits on team climate and performance: A systematic literature review. Information and Software Technology 73, 52–65 (2016)
20. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 990–998. ACM (2008)
21. Tong, H., Faloutsos, C., Pan, J.y.: Fast random walk with restart and its applications. In: ICDM. pp. 613–622 (2006)
22. Wang, H., Wang, N., Yeung, D.Y.: Collaborative deep learning for recommender systems. In: KDD. pp. 1235–1244. ACM (2015)
23. Wi, H., Oh, S., Mun, J., Jung, M.: A team formation model based on knowledge and collaboration. Expert Systems with Applications 36(5), 9121–9134 (2009)
24. Wu, C.Y., Ahmed, A., Beutel, A., Smola, A.J., Jing, H.: Recurrent recommender networks. In: WSDM. pp. 495–503. ACM (2017)
25. Yao, Y., Tong, H., Yan, G., Xu, F., Zhang, X., Szymanski, B.K., Lu, J.: Dual-regularized one-class collaborative filtering. In: CIKM. pp. 759–768. ACM (2014)