Vector Representation Learning of Skills for Collaborative Team Recommendation: A Comparative Study

Md Jamil Ahmed, Mahdis Saeedi, and Hossein Fani(✉)

University of Windsor, Windsor, ON, Canada
{ahmed491,msaeedi,hfani}@uwindsor.ca

Abstract. Neural team recommendation models have utilized graph representation learning to achieve state-of-the-art performance in forming teams of experts whose success in completing complex tasks is *almost surely* guaranteed. Specifically, the proposed models frame the problem as an expert recommendation task for a set of required skills whose dense vector representations are transferred from a graph neural network on the collaboration graph. However, there is yet to be a systematic comparative study on the impact of (1) the collaboration graph structure, (2) the node representation learning technique, and (3) the architecture of the final neural recommender on the efficacy of recommended teams. In this paper, we establish a benchmark that includes two heterogeneous collaboration graphs and seven graph representation learning to learn dense vector representations of skills for variational and non-variational neural recommenders. Our experiments on two large-scale datasets from various domains, each with distinct distributions of skills within teams, in a host of classification and information retrieval metrics show that *i*) those graph neural networks that utilize attention on *ii*) heterogeneous collaboration graphs, including expert, team and skill nodes, consistently yield the best dense vector representations of skills for *iii*) neural team recommenders of different architectures *iv*) across datasets. The code to reproduce the experiments reported in this paper is available at https://github.com/fani-lab/OpeNTF/tree/wise24.

Keywords: Team Recommendation · Graph Neural Networks · Social Network Analysis

1 Introduction

As the scope and complexity of tasks surpass the capability of each individual, groups of experts are formed as teams to accomplish the tasks. Teamwork is critical for organizational performance in various fields such as scientific research domains [56], industry [11], manufacturing [36], law [47], freelancing [3] and medical domain [49], and hence, a rich body of research has

M. Barhamgi et al. (Eds.): WISE 2024, LNCS 15437, pp. 193–207, 2025.
https://doi.org/10.1007/978-981-96-0567-5_15

been dedicated to developing computational models that effectively yet efficiently recommend optimum teams whose successes are *almost surely* guaranteed. Examples include operation research techniques [6,22,44,52] and graph-based methods [20,23,30,32,33,46,48], where an optimal team is formed by searching through all possible subsets or subgraphs of experts. However, such methods fall short in real-world scenarios for an overwhelming set of experts. To address scalability while enhancing efficacy, machine learning-based methods, particularly neural models, have been proposed recently [16,25,35,38–43,45], wherein the team recommendation problem has been framed as learning a mapping function from the input subset of required skills to a target subset of optimum experts via a multi-label classification task. To represent the input subset of required skills, as opposed to sparse multi-hot vectors, dense vector representations of required skills have been employed to *i*) accommodate the large set of skills, *ii*) reduce the complexity at the first layer of the neural team recommenders, and more importantly, *iii*) capture the semantic similarity among skills within the context of successful and unsuccessful teams [25,35,38]. For instance, Kaw et al. [25] formed a heterogeneous graph whose nodes were experts, skills, and teams, and the edges linked each team with its subset of experts and required skills. They applied deep graph infomax [50], a graph convolution network with an attention layer as an encoder, to generate dense vector representation of skills so it can be fed into the variational Bayesian neural network to recommend a team. Nguyen et al. [35] augmented the collaboration graph with locations and employed Xiao et al. [54]'s learning-to-propagate, an adaptable message passing for an underlying task, to generate skill vectors. Prior work, however, has been evaluated against a limited number of baselines, overlooking varied ways of forming collaboration graphs, a variety of representation learning methods for skills, and diverse neural team recommenders.

In this paper, we investigate the impact of the collaboration graph structures, skill representation learning methods, and the architectures of neural recommenders on the efficacy of team recommendation. Specifically, we cross-compare (1) two types of heterogeneous collaboration graphs, (2) seven dense vector representation learning methods based on message passing and meta-paths on the collaboration graph as well as document-based vectors and the naive sparse multi-hot vectors for input skills, and (3) neural team recommenders including feedforward and variational Bayesian models.

2 Task Definition

2.1 Team Recommendation Problem

Given a set of skills $\mathcal{S} = \{s_i\}$ and a set of experts $\mathcal{E} = \{e_j\}$, a team t is a set of experts $\mathbf{e} \subseteq \mathcal{E}$ that collectively cover the skill set $\mathbf{s} \subseteq \mathcal{S}$ along with its success status, shown by $t_{\mathbf{s},\mathbf{e},y\in\{0,1\}}$. Further, $\mathcal{T} = \{t_{\mathbf{s},\mathbf{e},y} : \mathbf{s} \neq \emptyset, \mathbf{e} \neq \emptyset\}$ indexes all instances of teams, successful and unsuccessful. For a given set of skills $\mathbf{s}$, the team recommendation problem aims at identifying an optimal subset of experts $\mathbf{e}$ such that their collaboration in the predicted team is successful,

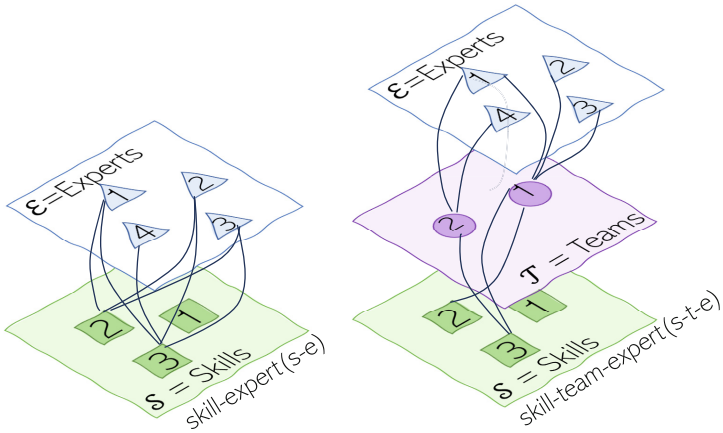

Fig. 1. Heterogeneous collaboration graphs.

that is $t_{\mathbf{s},\mathbf{e},y=1}$, while avoiding a subset of experts $\mathbf{e}'$ resulting in $t_{\mathbf{s},\mathbf{e}',y=0}$. More concretely, the team recommendation problem is to learn a mapping function f of parameters $\boldsymbol{\theta}$ from the powerset of skills to the powerset of experts such that $\forall t_{\mathbf{s},\mathbf{e},y=1}; f_{\boldsymbol{\theta}}(\mathbf{s}) = \mathbf{e}$.

2.2 Collaboration Graph

A collaboration graph is $G(\mathcal{V}, \mathcal{L})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{L}$ is the set of *un*directed links. It is heterogeneous if $\mathcal{V}$ includes nodes of different types. As illustrated in Fig. 1, two mainstream heterogeneous collaboration graph structures include:

- The heterogeneous *bi*partite `skill-expert` graph $G = (\mathcal{V}, \mathcal{L})$ including nodes for skills $\mathcal{S}$ and experts $\mathcal{E}$, i.e., $\mathcal{V} = \mathcal{S} \cup \mathcal{E}$, and $\mathcal{L}$ is the set of links, each of which connects a skill to an expert if they are co-occurring in a team.
- The heterogeneous *tri*partite `skill-team-expert` graph $G = (\mathcal{V}, \mathcal{L})$ including three types of nodes, skills $\mathcal{S}$, teams $\mathcal{T}$ and experts $\mathcal{E}$, i.e., $\mathcal{V} = \mathcal{S} \cup \mathcal{T} \cup \mathcal{E}$. The link set $\mathcal{L}$ connects each team node to all its expert and skill nodes [25,35,38].

2.3 Skill Vector Representation

A vector representation method for a skill s_i is a function $r : \mathcal{S} \rightarrow \mathbb{R}^d, r(s_i) = v_{s_i}$, which maps a skill to a d-dimensional vector. Moreover, a vector representation for a subset of skills $\mathbf{s} \subseteq \mathcal{S}$ for a team $t_{\mathbf{e},\mathbf{s},y}$ can be obtained by summing the vectors of its constituent skills, i.e., $v_{\mathbf{s}} = \sum_{s_i \in \mathbf{s}} r(s_i) = \sum_{s_i \in \mathbf{s}} v_{s_i}$. Hence, the team recommendation aim at estimating $f_\theta(\mathbf{s}) \approx f_\theta(\sum_{s_i \in \mathbf{s}} r(s_i)) = f_\theta(v_{\mathbf{s}})$.

The vector representation method can be as simple as high-dimensional sparse *occurrence* vector representation (one-hot), which is a Boolean vector

of size $d = |\mathcal{S}|$ where $r(s_i)[i] = 1$ and 0 otherwise, or a learnable function parameterized by ϕ on a collaboration graph G, i.e., $r_{\phi,G}(s_i)$, for a dense low-dimensional vector representation, $d << |\mathcal{S}|$, by a graph-based representation learning method [25,38,43]. We experiment two categories of graph-based vector representation learning:

Meta-path-based [9]. As opposed to homogeneous graphs where random walks on any nodes of the same type are permissible to learn node vector representations [15,37], in meta-path-based methods, the skill vectors are learnt such that two skill vectors stay close in the vector space when they co-occur on random walks that are generated by a predefined set of meta-paths to supervise walks between *certain types* of nodes over the heterogeneous graph G.

In the bipartite `skill-expert` graph, a single meta-path is defined as $s_i - e_j - s_{i'}$ to prioritize the co-occurrence of skills for an expert and their potential semantic relatedness. In the tripartite `skill-team-expert` graph, a meta-path is defined as $s_i - t - e_j - t' - s_{i'}$, implying the semantic relatedness of two skills required in different teams that share the same expert. Each mata-path-based random walk is then considered as a document whose words are the nodes followed by `word2vec` [34] to produce d-dimensional vector representations for all types of nodes including skill nodes.

Message-passing-based [2,18,19,51,53,55]. Also referred to as graph neural networks (gnn), in message passing, a node vector is learnt based on a recursive aggregation (`agg`) and a combination (`comb`) of direct (1-hop) or indirect (k-hop) neighbouring nodes' vectors via neural message passing as follows [13]:

$$v_{s_i}^{k+1} = \texttt{comb}(v_{s_i}^{k},\ \texttt{agg}(v_x^k : x \in \mathcal{N}_{s_i}^k)) \tag{1}$$

where $\mathcal{N}_{s_i}^k$ is the k-hop neighbourhood of the skill node s_i, which may include nodes of other types in a heterogeneous graph, `agg` and `comb` are two differentiable functions, which can be fixed like `mean`, or learnable [19,55], and $v_{s_i}^k$ is the learnt skill vector at iteration k. Message-passing-based methods differ in their `aggr` and `comb` functions [17]:

- **Inductive** methods [18], which can infer vector representations for *un*seen skill nodes based on existing nodes residing in the neighbourhood of the unseen skill node, as opposed to transductive methods [4,28], which need the entire graph to be present during vector representation learning and the model should be *re*trained for a newly added node.
- **Attentive** methods [2,51,53], which allocate attention coefficients to a skill node's neighbours to signify their importance in updating the skill node's vector.
- **Graph isomorphism** methods [19,55], wherein, next to the skill vectors, the `agg` and `comb` functions are also parameterized and learnable based on Weisfeiler-Lehman graph isomorphism test [10].

2.4 Neural Team Recommendation

Neural team recommendation estimates $f_{\boldsymbol{\theta}}(\mathbf{s}) \approx f(v_{\mathbf{s}})$ using a multi-layer neural network that learns, from $\mathcal{T}$, to map a vector representation of subset of skills

$\mathbf{s}$, referred to as $v_{\mathbf{s}}$, to a vector representation of subset of experts $\mathbf{e}$, referred to as $v_{\mathbf{e}}$, by maximizing a posterior (`map`) probability of $\boldsymbol{\theta}$ in $f_{\boldsymbol{\theta}}$ over $\mathcal{T}$, that is, $\texttt{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{T})$. Using a neural model of one hidden layer $\mathbf{h}$, without loss of generality to multiple hidden layers, with the input layer $v_{\mathbf{s}}$ and output layer $v_{\mathbf{e}}$, a neural team recommendation method can be formalized as [7,12,38,41,43]:

$$\mathbf{h} = \texttt{act}([\boldsymbol{\theta}_1]v_{\mathbf{s}} + \mathbf{b}_1) \tag{2}$$

$$\textit{logits} \rightarrow \mathbf{z} = [\boldsymbol{\theta}_2]\mathbf{h} + \mathbf{b}_2 \tag{3}$$

$$v_{\mathbf{e}} = \texttt{sig}(\mathbf{z}) \tag{4}$$

where `act` is a nonlinear activation function, `sig` is the `sigmoid` function, and $\boldsymbol{\theta} = \boldsymbol{\theta}_1 \cup \boldsymbol{\theta}_2 \cup \mathbf{b}_1 \cup \mathbf{b}_2$ are learnable parameters for the mapping function f. While the vector representation of a subset of skills $v_{\mathbf{s}}$ can be learnt using different methods as explained above, *for* the output layer *as* vector representation of subset of experts $v_{\mathbf{e}}$, existing neural team recommendation methods frame the problem as a *multilabel* Boolean classification task and used occurrence vector representation for $\mathbf{e}$, that is, $v_{\mathbf{e}} \in \{0,1\}^{|\mathcal{E}|}$ where $v_{\mathbf{e}}[j] = 1$ if $e_j \in \mathbf{e}$, and 0 otherwise. Therefore, `sigmoid` has been the common activation function to generate the predictions for *each* expert.

Furthermore, from Sect. 2.1, team recommendation aims at estimating f from samples of teams that are labelled with success or failure. Yet most available training data only consists of successful teams. The `dblp` dataset of published research papers in computer science lacks unsuccessful submissions. In the `imdb` dataset of movies, it remains a question of what defines a movie's failure, its immediate reception by audiences, as indicated by the budget-to-box-office ratio, or its critical reviews reflected in ratings over a prolonged period. Hence, existing methods presume samples of teams in the training dataset are all successful (positive samples), that is, $\mathcal{T} = \mathcal{T}^+ = \{t_{\mathbf{s},\mathbf{e},y=1}\}$. During training, given a team $t_{\mathbf{s},\mathbf{e}} \in \mathcal{T}$, neural models tune the parameters $\boldsymbol{\theta}$ by maximizing the posterior probability of $\boldsymbol{\theta}$ in $f_{\boldsymbol{\theta}}$ over $\mathcal{T}$. By Bayes theorem:

$$\texttt{argmax}_{\boldsymbol{\theta}}\ p(\boldsymbol{\theta}|\mathcal{T}) = \frac{p(\boldsymbol{\theta})p(\mathcal{T}|\boldsymbol{\theta})}{p(\mathcal{T})} \ \propto\ p(\boldsymbol{\theta})p(\mathcal{T}|\boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{t_{\mathbf{s},\mathbf{e}} \in \mathcal{T}} p(\mathbf{e}|\mathbf{s},\boldsymbol{\theta}) \tag{5}$$

$$p(\mathbf{e}|\mathbf{s},\boldsymbol{\theta}) = \prod_{j \in e} \sigma(\mathbf{z}[j]) \propto \sum_{j \in e} \texttt{log}\ \sigma(\mathbf{z}[j]) \tag{6}$$

where $p(\mathcal{T})$ is independent of optimizing the parameters $\boldsymbol{\theta}$, $p(\mathcal{T}|\boldsymbol{\theta})$ is the likelihood, and $p(\boldsymbol{\theta})$ is the prior joint probability of weights, which is unknown. The *true* prior probability of weights $p(\boldsymbol{\theta})$ cannot be calculated analytically or efficiently sampled [14], and as such, we can assume *uniform* probability distribution over all possible real-values of $\boldsymbol{\theta}$ and proceed with maximum likelihood estimation $p(\mathcal{T}|\boldsymbol{\theta})$ [7] by minimizing cross-entropy loss, or estimate $p(\boldsymbol{\theta})$ by Gaussian distribution and calculate the maximum a posteriori via a variational Bayesian neural architecture [16,25,38,43] by minimizing Kullback-Leibler divergence [29].

During inference, given a required subset of skills $\mathbf{s}$, the output of a neural team recommendation method is a ranked list of *all* experts where each expert

$e \in \mathcal{E}$ is assigned a probability of her membership in the final recommended team $t_{\mathbf{s},\mathbf{e}}$ where $\mathbf{e} \subseteq \mathcal{E}$ is either the top-k highest probabilities, or those whose probabilities meet a minimum threshold τ.

3 Experiments

In this section, we seek to answer the following research questions:

- **RQ1:** Does dense vector representation of skills perform better than sparse vectors in neural team recommendation across different neural team recommenders, and training datasets with diverse statistical characteristics?

If **RQ1**'s answer is positive, we further investigate:

- **RQ2:** Which dense vector representation is of the best quality across neural team recommenders and datasets?
- **RQ3:** Do neural team recommenders of different architectures benefit from dense vectors equally?
- **RQ4:** Does collaboration graph structure impact the efficacy of graph-based vector representations?
- **RQ5:** Does the vector dimension impact the efficacy of dense vector representations?

Table 1. Statistics of the raw and filtered datasets.

	dblp		imdb	
	raw	filtered	raw	filtered
#teams	4,877,383	99,375	507,034	32,059
#unique experts	5,022,955	14,214	876,981	2,011
#unique skills	**89,504**	**29,661**	**28**	**23**
avg #expert per team	3.06	3.29	1.88	3.98
avg #skill per team	**8.57**	**9.71**	**1.54**	**1.76**
avg #team per expert	2.97	23.02	1.09	62.45
avg #skill per expert	**16.73**	**96.72**	**1.59**	**10.85**
#team w/ single expert	**768,956**	**0**	**322,918**	**0**
#team w/ single skill	**5,569**	**56**	**315,503**	**15,180**

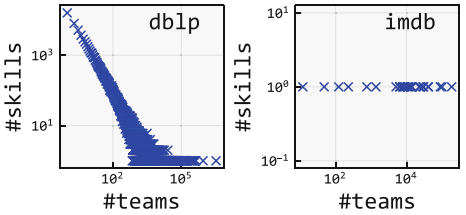

Fig. 2. Distribution of teams over skills for `dblp` and `imdb`.

3.1 Setup

Datasets. We include datasets from varying domains:

- `dblp` collection of computer science publications [1] where a *published* paper is considered as a successful team whose authors are the subset of experts, and the keywords are the required skills [7,16,26,43]; and,
- `imdb` collection of moving pictures [21], where a *produced* movie is considered as a successful team whose experts are the cast and crew, and the movie's genres are the required skills. The choice of `imdb` in team recommendation literature is *not* to be confused with its use cases in movie recommender systems or movie review analysis; herein, the goal is to form a team of casts and crews for a movie *production* as opposed to a movie recommendation [7, 23,24].

We exclude teams with less than `3` experts [7,12,43] while ensuring no major change to the statistical distributions of the datasets, as summarized in Table 1. From Fig. 2, the distribution of teams over skills is highly skewed in `dblp`, whereas `imdb` shows a fairer distribution for a limited set of skills (genres), which are uniformly employed by many movies.

Baselines. Our skill vector representation methods include:

- `m-hot` as the naive $|\mathcal{S}|$-dimensional sparse occurrence vector for a subset of skills `s`.
- `d2v` [16,43] learns dense vectors by `doc2vec` [31] with context window of `10`.
- `m2v` [38] is graph-based and learns dense vectors by Dong et al. [9]'s `metapath2vec`. The length and number of random walks are set to `10` and `20`, respectively.
- `graphsage` [18] is a graph neural network that learns dense vectors via an inductive message passing.
- `gat` [51] is an attentive graph neural network that learns dense vectors via node neighbours with learnable attention scores.
- `han` [53] is a hierarchical variant of `gat` that learns the attention scores at node level and semantic-level.
- `gatv2` [5] extends `gat` with a strictly more expressive attention mechanism by fixing the standard `gat`'s scoring function.
- `gin` [55] is a graph neural network that learns dense vectors via learnable aggregation and combination functions.
- `gine` [19] extends `gin` to incorporate global graph features based on a *context subgraph* defined as neighbours located between k_1-hop and k_2-hop of a skill node where $k_1 < k_2$.

For graph neural networks, we use four sequential convolutional layers for all the methods except `han`, which requires a final linear transformation layer instead of the fourth convolutional layer. Our search for an optimum aggregation function results in `mean` in all methods except `gin` and `gine`, which learn the aggregation function. Our neural team recommenders include:

– `fnn` as the nonvariational feedforward model with the cross-entropy loss.
– `bnn` as the variational Bayesian feedforward with Kullback-Leibler loss of the hidden layer summed with the cross-entropy loss of the final layer.

Both models include a single hidden layer of size `128`, `leaky relu` as the activation function for the hidden layer, and `Adam` as the optimizer. We searched for the optimum learning rates from `0.00001` to `0.1`, which yield `0.01` for `bnn` and `0.0001` for `fnn`. We used `opentf` [8] for dataset preprocessing, `d2v` vectors, graphs construction, and `fnn`. We used `Bayesian-torch` [27] for `bnn`.

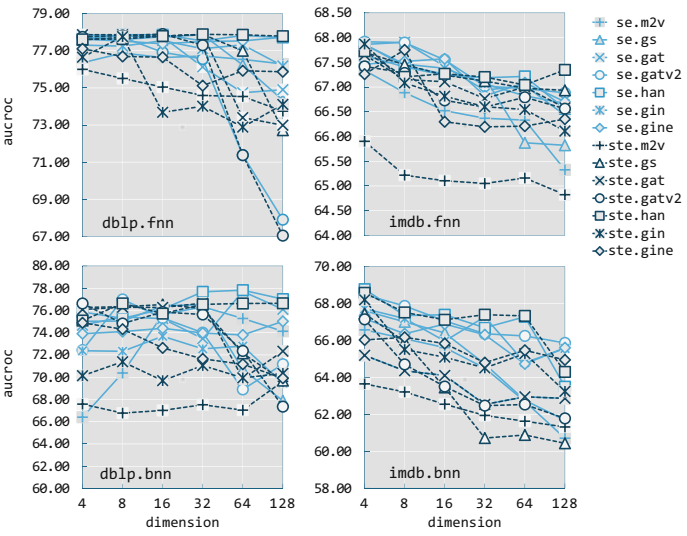

Fig. 3. Performance across increasing vector *dimension*.

3.2 Evaluation Strategy and Metrics

We randomly select `15%` of teams for the test set and perform `3`-fold cross-validation on the remaining teams for model training using each of embedding generated models over `25` epochs that results in one trained model per each fold. Given a team from the test set, we compare the top-k experts predicted by the model of each fold with the observed subset of experts $\mathbf{e}$ and report the average performance of the model on all folds in terms of information retrieval metrics including normalized discounted cumulative gain (`ndcg`) and mean average precision (`map`) as well as classification metrics including `precision` and `recall` at top-$\{2, 5, 10\}$, and area under the receiver operating characteristic (`aucroc`).

3.3 Results

To answer **RQ1**, i.e., dense vs. sparse vectors of skills, from Tables 2, 3, 4, 5, not unexpectedly all dense vector representations outperform the sparse one (`m-hot`) across `dblp` and `imdb` datasets, and `fnn` and `bnn` neural team recommenders in terms of all metrics, which is consistent with previous research [7, 12, 16, 43].

Table 2. Average performance of 3-fold `fnn` on test set in `dblp`.

	%aucroc	%precision			%recall			%ndcg			%map		
		@2	@5	@10	@2	@5	@10	@2	@5	@10	@2	@5	@10
	skill-expert												
m-hot [43]	77.54	0.78	0.70	0.65	0.49	1.07	1.96	0.79	0.96	1.37	0.37	0.53	0.66
d2v [16,31,43]	76.36	1.05	0.83	0.72	0.64	1.27	2.21	1.09	1.18	1.62	0.51	0.72	0.88
m2v [38]	76.88	1.21	1.01	0.80	0.73	1.54	2.44	1.25	1.42	1.83	0.58	0.84	0.99
gs [18]	77.81	1.01	0.92	0.83	0.60	1.38	2.51	1.05	1.24	1.76	0.51	0.72	0.90
gat [51]	77.62	1.30	**1.10**	**0.85**	0.79	**1.66**	**2.57**	1.29	1.49	1.92	0.59	0.84	0.98
gatv2 [5]	77.57	1.28	0.96	0.81	0.78	1.44	2.45	1.33	1.39	1.85	0.62	0.82	0.97
han [53]	**77.84**	**1.39**	1.09	0.83	**0.84**	1.64	2.49	**1.46**	**1.56**	**1.96**	**0.67**	**0.93**	**1.05**
gin [55]	77.25	0.44	0.63	0.69	0.27	0.93	2.07	0.41	0.72	1.25	0.18	0.37	0.53
gine [19]	77.53	1.15	0.92	0.79	0.68	1.37	2.35	1.15	1.26	1.72	0.51	0.71	0.87
	skill-team-expert												
m-hot [43]	77.54	0.78	0.70	0.65	0.49	1.07	1.96	0.79	0.96	1.37	0.37	0.53	0.66
d2v [16,31,43]	75.51	0.97	0.82	0.70	0.59	1.24	2.13	1.00	1.13	1.55	0.46	0.66	0.82
m2v [38]	74.65	0.81	0.74	0.68	0.50	1.14	2.07	0.83	1.01	1.44	0.39	0.59	0.75
gs [18]	77.73	1.12	0.87	0.74	0.67	1.31	2.24	1.19	1.26	1.69	0.55	0.73	0.88
gat [51]	**77.87**	**1.35**	**1.19**	**0.92**	**0.81**	**1.78**	**2.76**	1.39	**1.62**	**2.08**	0.64	**0.92**	**1.09**
gatv2 [5]	77.75	1.27	0.92	0.80	0.77	1.39	2.40	1.29	1.32	1.80	0.59	0.77	0.94
han [53]	77.86	1.34	1.07	0.89	**0.81**	1.61	2.66	**1.42**	1.53	2.02	**0.66**	0.90	1.08
gin [55]	77.77	1.31	0.95	0.78	0.78	1.43	2.35	1.35	1.37	1.80	0.61	0.78	0.92
gine [19]	76.70	1.18	1.02	0.83	0.69	1.52	2.49	1.16	1.35	1.80	0.51	0.76	0.92

In response to **RQ2** about the best dense vector representation across datasets, our results show that attentive methods yield the best skill vectors across `dblp` and `imdb` datasets and `fnn` and `bnn` neural team recommenders, with `gat` being the best and `han` the runner-up. Our results, however, contrast with existing benchmarks for graph neural networks [55] where `gin` and `gine` with learnable aggregation and combination functions are the state-of-the-art in node representation learning. We attribute the inefficiency of such methods compared to attentive methods to a higher number of parameters for learnable functions and, hence, the need for more training epochs. Moreover, the meta-path-based method (`m2v`) is the poorest graph-based method, on par with `d2v`, for team recommendation across datasets and neural recommenders.

Regarding **RQ3**, comparing Tables 2 and 3 for `dblp` and Tables 4 and 5 for `imdb`, we observe that both neural team recommenders benefit from dense vectors generally equally and the trend from the best to the poorest skill vector representations are similar in both models, with `fnn` being marginally the better recommender.

In response to **RQ4**, i.e., whether the graph structure impacts the efficacy of graph-based vectors, comparing `skill-expert` with `skill-team-expert` parts

Table 3. Average performance of 3-fold `bnn` on test set in `dblp`.

	%aucroc	%precision			%recall			%ndcg			%map		
		@2	@5	@10	@2	@5	@10	@2	@5	@10	@2	@5	@10
	skill-expert												
m-hot [43]	71.42	0.60	0.53	0.49	0.37	0.82	1.51	0.61	0.73	1.05	0.30	0.43	0.54
d2v [16,31,43]	72.86	0.82	0.68	0.61	0.49	1.02	1.86	0.84	0.95	1.33	0.39	0.56	0.69
m2v [38]	70.38	0.46	0.41	0.39	0.29	0.63	1.20	0.47	0.57	0.83	0.22	0.32	0.40
gs [18]	75.05	0.70	0.63	0.58	0.42	0.95	1.75	0.71	0.85	1.22	0.32	0.47	0.59
gat [51]	75.41	0.54	0.54	0.53	0.32	0.82	1.60	0.54	0.70	1.06	0.24	0.38	0.49
gatv2 [5]	**76.99**	**1.00**	**0.87**	**0.74**	**0.59**	**1.32**	**2.23**	**1.03**	**1.20**	**1.62**	**0.47**	**0.68**	**0.83**
han [53]	75.29	0.59	0.52	0.51	0.35	0.79	1.54	0.60	0.71	1.05	0.28	0.40	0.51
gin [55]	72.33	0.80	0.61	0.52	0.48	0.92	1.57	0.83	0.88	1.18	0.40	0.54	0.64
gine [19]	74.10	0.76	0.63	0.56	0.46	0.96	1.71	0.78	0.88	1.23	0.37	0.52	0.63
	skill-team-expert												
m-hot [43]	71.42	0.60	0.53	0.49	0.37	0.82	1.51	0.61	0.73	1.05	0.30	0.43	0.54
d2v [16,31,43]	72.86	0.82	0.68	0.61	0.49	1.02	1.86	0.84	0.95	1.33	0.39	0.56	0.69
m2v [38]	66.77	0.33	0.31	0.32	0.20	0.48	0.98	0.34	0.42	0.65	0.15	0.23	0.30
gs [18]	76.34	0.75	0.71	0.66	0.46	1.08	2.00	0.74	0.94	1.36	0.34	0.52	0.65
gat [51]	76.29	0.72	0.66	0.62	0.44	0.99	1.86	0.72	0.87	1.28	0.33	0.49	0.62
gatv2 [5]	74.82	0.73	0.64	0.54	0.44	0.96	1.64	0.75	0.87	1.19	0.35	0.49	0.58
han [53]	**76.63**	0.92	**0.80**	**0.71**	0.55	**1.20**	**2.15**	0.94	**1.09**	**1.53**	0.43	0.61	0.76
gin [55]	71.39	0.69	0.60	0.52	0.42	0.91	1.59	0.73	0.84	1.15	0.35	0.49	0.60
gine [19]	74.31	**0.95**	0.75	0.63	**0.57**	1.12	1.90	**1.00**	1.08	1.44	**0.48**	**0.66**	**0.79**

of Tables 2, 3, 4, and 5, we generally see a better quality for skill vectors that are learnt from the `skill-team-expert` graph across learning methods, neural team recommenders, and datasets. There are few exceptions for `m2v` and `han` where the `skill-expert` graph structure yields higher quality skill vectors compared to their counterparts in `skill-team-expert` graph.

For **RQ5**, we studied skill vectors in increasing dimensions. From Fig. 3, we observe two distinct trends per dataset across neural team recommenders. In `dblp`, dense vector representations are generally agnostic to the skill vector dimensions for neural recommenders, which can be attributed to the large set of skills $\mathcal{S}$, on the one hand, and relatively small number of required skills in a team, on the other hand. However, in `imdb`, increasing the dimension negatively impacts the quality of skill vectors. From Table 1, `imdb` has a limited set of skills (genres), and the average number of skills in a team (movie) is about `1.76` such that increasing the dimension of skill vectors to large values discounts the efficacy of skill vectors.

Table 4. Average performance of 3-fold `fnn` on test set in `imdb`.

	%aucroc	%precision			%recall			%ndcg			%map		
		@2	@5	@10	@2	@5	@10	@2	@5	@10	@2	@5	@10
	skill-expert												
m-hot [43]	67.05	0.72	0.75	0.70	0.33	0.86	1.59	0.73	0.84	1.19	0.26	0.41	0.52
d2v [16,31,43]	66.17	0.93	0.84	**0.76**	0.42	0.93	**1.72**	0.93	0.95	1.32	**0.31**	0.45	0.57
m2v [38]	66.89	0.88	0.78	0.73	0.39	0.87	1.64	0.91	0.91	1.26	**0.31**	0.43	0.54
gs [18]	67.41	**0.98**	**0.84**	0.75	**0.42**	**0.94**	1.70	**0.98**	0.96	1.31	**0.31**	0.45	0.56
gat [51]	67.42	0.90	0.83	0.75	0.38	0.91	1.71	0.94	**0.96**	**1.32**	**0.31**	**0.46**	**0.57**
gatv2 [5]	**67.81**	0.87	0.82	0.72	0.37	0.90	1.61	0.84	0.89	1.22	0.26	0.40	0.51
han [53]	67.52	0.94	0.81	0.72	0.41	0.89	1.63	0.96	0.94	1.27	**0.31**	0.45	0.56
gin [55]	67.46	0.96	0.78	0.65	0.42	0.86	1.45	0.96	0.91	1.17	**0.31**	0.43	0.51
gine [19]	67.52	0.86	0.74	0.69	0.37	0.81	1.53	0.90	0.87	1.20	0.29	0.41	0.52
	skill-team-expert												
m-hot [43]	67.05	0.72	0.75	0.70	0.33	0.86	1.59	0.73	0.84	1.19	0.26	0.41	0.52
d2v [16,31,43]	66.17	0.93	0.84	0.76	0.42	0.93	1.72	0.93	0.95	1.32	0.31	0.45	0.57
m2v [38]	65.22	0.90	0.84	0.77	0.40	0.95	1.76	0.91	0.95	1.33	0.31	0.46	0.57
gs [18]	67.45	0.90	0.83	0.73	0.38	0.91	1.64	0.93	0.95	1.27	0.30	0.44	0.55
gat [51]	67.15	1.06	0.92	0.82	**0.45**	1.00	1.83	**1.07**	1.06	1.43	**0.34**	0.50	0.62
gatv2 [5]	67.00	**1.07**	**1.00**	**0.92**	0.44	**1.13**	**2.06**	1.06	**1.11**	**1.55**	0.32	**0.50**	**0.64**
han [53]	67.47	0.99	0.83	0.74	0.42	0.93	1.65	1.01	0.97	1.30	0.33	0.46	0.56
gin [55]	67.08	0.86	0.76	0.68	0.37	0.84	1.56	0.86	0.87	1.19	0.28	0.41	0.51
gine [19]	**67.52**	0.86	0.74	0.69	0.37	0.81	1.53	0.90	0.87	1.20	0.29	0.41	0.52

4 Conclusion and Future Work

In this paper, we performed a comparative study of different graph structures, skill vector representations, and neural recommenders across datasets from varied domains, withholding different statistical distributions to seek the optimum team recommendation pipeline. Our results show that (1) dense vector representation of skills perform better than sparse vectors in neural team recommendation across different training datasets with diverse statistical characteristics, (2) attentive methods are of the best quality across both datasets, (3) neural team recommenders of different architectures benefit from dense vectors generally equally, (4) expert collaboration graph structure mostly impact the efficacy of graph-based vector representations, and (5) the vector dimension impact the efficacy of dense vector representations. Moving forward, we focus on proposing a graph neural network-based method to directly learn skill vectors in an end-to-end approach. We also intend to extend our benchmark to other domains, including `uspt` which is the collection of *issued* patents, and `github`, the collection of software *repositories*.

Table 5. Average performance of 3-fold `bnn` on test set in `imdb`.

	%aucroc	%precision			%recall			%ndcg			%map		
		@2	@5	@10	@2	@5	@10	@2	@5	@10	@2	@5	@10
	skill-expert												
m-hot [43]	64.05	0.82	0.83	0.74	0.37	0.95	1.71	0.81	0.92	1.28	0.29	0.45	0.57
d2v [16,31,43]	65.07	0.93	0.87	0.81	0.42	0.99	1.84	0.92	0.98	1.38	0.31	0.47	0.60
m2v [38]	66.09	1.08	1.00	0.89	0.49	1.14	2.04	1.09	1.14	1.56	0.37	0.56	0.69
gs [18]	67.01	1.00	1.03	0.96	0.45	1.16	2.18	1.00	1.13	1.61	0.34	0.53	0.68
gat [51]	**68.06**	**1.14**	**1.08**	**1.01**	**0.51**	**1.23**	**2.31**	**1.15**	**1.23**	**1.74**	**0.40**	**0.61**	**0.76**
gatv2 [5]	67.86	1.11	1.04	1.00	0.50	1.18	2.25	1.12	1.19	1.70	0.39	0.58	0.74
han [53]	67.34	1.08	1.02	0.94	0.49	1.16	2.15	1.11	1.17	1.64	0.39	0.57	0.72
gin [55]	67.19	1.08	1.06	0.96	0.48	1.20	2.19	1.08	1.19	1.64	0.37	0.58	0.72
gine [19]	66.36	1.06	0.98	0.94	0.48	1.12	2.14	1.09	1.14	1.62	0.37	0.56	0.71
	skill-team-expert												
m-hot [43]	64.05	0.82	0.83	0.74	0.37	0.95	1.71	0.81	0.92	1.28	0.29	0.45	0.57
d2v [16,31,43]	65.07	0.93	0.87	0.81	0.42	0.99	1.84	0.92	0.98	1.38	0.31	0.47	0.60
m2v [38]	63.21	0.80	0.76	0.71	0.35	0.83	1.58	0.80	0.85	1.20	0.27	0.41	0.52
gs [18]	65.94	1.00	0.99	0.90	0.44	1.12	2.03	1.02	1.12	1.55	0.35	0.54	0.67
gat [51]	64.36	1.07	0.96	0.86	0.48	1.07	1.94	1.07	1.10	1.50	0.36	0.53	0.66
gatv2 [5]	64.70	0.98	0.93	0.80	0.44	1.05	1.84	0.98	1.05	1.41	0.33	0.51	0.62
han [53]	**67.51**	1.13	**1.06**	**0.99**	0.50	**1.19**	**2.25**	**1.14**	**1.21**	**1.70**	**0.39**	**0.59**	**0.74**
gin [55]	65.51	1.07	0.90	0.83	0.48	1.04	1.92	1.08	1.07	1.47	0.37	0.52	0.65
gine [19]	66.17	**1.15**	1.01	0.93	**0.51**	1.15	2.14	1.14	1.16	1.62	0.39	0.58	0.71

References

1. https://www.aminer.org/citation
2. Amine, O., Mestari, M.: Graph oriented attention networks. IEEE Access **12**, 47057–47067 (2024). https://doi.org/10.1109/ACCESS.2024.3378094
3. Barnabò, G., Fazzone, A., Leonardi, S., Schwiegelshohn, C.: Algorithms for fair team formation in online labour marketplaces. In: Companion Proceedings of the 2019 World Wide Web Conference, pp. 484–490 (2019)
4. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, Vancouver, British Columbia, Canada], 3–8 December 2001, pp. 585–591. MIT Press (2001)
5. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? In: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, 25–29 April 2022. OpenReview.net (2022)
6. Campêlo, M.B., Figueiredo, T.F., Silva, A.: The sociotechnical teams formation problem: a mathematical optimization approach. Ann. Oper. Res. **286**(1), 201–216 (2020). https://doi.org/10.1007/S10479-018-2759-5
7. Dashti, A., Samet, S., Fani, H.: Effective neural team formation via negative samples. In: CIKM (2022)

8. Dashti, A., Saxena, K., Patel, D., Fani, H.: OpenTF: a benchmark library for neural team formation. In: CIKM (2022)
9. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017, pp. 135–144. ACM (2017). https://doi.org/10.1145/3097983.3098036
10. Douglas, B.L.: The Weisfeiler-Lehman method and graph isomorphism testing. arXiv preprint arXiv:1101.5211 (2011)
11. Ducoffe, S.J.S., Tromley, C.L., Tucker, M.: Interdisciplinary, team-taught, undergraduate business courses: the impact of integration. J. Manag. Educ. **30**(2), 276–294 (2006)
12. Fani, H., Barzegar, R., Dashti, A., Saeedi, M.: A streaming approach to neural team formation training. In: Proceedings of the 46th European Conference on Information Retrieval, ECIR 2024 (2024)
13. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 1263–1272. PMLR (2017)
14. Graves, A.: Practical variational inference for neural networks. In: NIPS (2011)
15. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Krishnapuram, B., Shah, M., Smola, A.J., Aggarwal, C.C., Shen, D., Rastogi, R. (eds.) Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016, pp. 855–864. ACM (2016)
16. Hamidi Rad, R., Fani, H., Bagheri, E., Kargar, M., Srivastava, D., Szlichta, J.: A variational neural architecture for skill-based team formation. ACM Trans. Inf. Syst. **42**(1) (2023). https://doi.org/10.1145/3589762
17. Hamilton, W.L.: Graph Representation Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2020). https://doi.org/10.2200/S01045ED1V01Y202009AIM046
18. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017, pp. 1024–1034 (2017)
19. Hu, W., et al.: Strategies for pre-training graph neural networks. In: ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020. OpenReview.net (2020)
20. Huang, J., Lv, Z., Zhou, Y., Li, H., Sun, H., Jia, X.: Forming grouped teams with efficient collaboration in social networks. Comput. J. **60**(11), 1545–1560 (2017). https://doi.org/10.1093/comjnl/bxw088
21. IMDb: IMDb non-commercial datasets - https://developer.imdb.com/non-commercial-datasets/
22. Kalayathankal, S.J., Kureethara, J.V., Narayanamoorthy, S.: A modified fuzzy approach to project team selection. Soft Comput. Lett. **3**, 100012 (2021)
23. Kargar, M., An, A.: Discovering top-k teams of experts with/without a leader in social networks. In: CIKM, pp. 985–994 (2011)
24. Kargar, M., Golab, L., Srivastava, D., Szlichta, J., Zihayat, M.: Effective keyword search over weighted graphs. IEEE Trans. Knowl. Data Eng. **34**(2), 601–616 (2022). https://doi.org/10.1109/TKDE.2020.2985376

25. Kaw, S., Kobti, Z., Selvarajah, K.: Transfer learning with graph attention networks for team recommendation. In: IJCNN 2023, pp. 1–8. IEEE (2023)
26. Kou, Y., et al.: Efficient team formation in social networks based on constrained pattern graph. In: ICDE, pp. 889–900 (2020)
27. Krishnan, R., Esposito, P., Subedar, M.: Bayesian-torch: Bayesian neural network layers for uncertainty estimation, January 2022. https://doi.org/10.5281/zenodo.5908307
28. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika **29**(1), 1–27 (1964)
29. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)
30. Lappas, T., Liu, K., Terzi, E.: Finding a team of experts in social networks. In: SIGKDD (2009)
31. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21–26 June 2014. JMLR Workshop and Conference Proceedings, vol. 32, pp. 1188–1196. JMLR.org (2014). http://proceedings.mlr.press/v32/le14.html
32. Mayer, R.C., Davis, J.H., Schoorman, F.D.: An integrative model of organizational trust. Acad. Manag. Rev. **20**(3), 709–734 (1995)
33. McKnight, D.H., Choudhury, V., Kacmar, C.: The impact of initial consumer trust on intentions to transact with a web site: a trust building model. J. Strateg. Inf. Syst. **11**(3–4), 297–323 (2002)
34. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Bengio, Y., LeCun, Y. (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, 2–4 May 2013, Workshop Track Proceedings (2013)
35. Nguyen, H., et al.: Learning heterogeneous subgraph representations for team discovery. Inf. Retr. J. **26**(1), 8 (2023). https://doi.org/10.1007/S10791-023-09421-6
36. Pasupa, T., Suzuki, S.: Two-stage adaptive large neighbourhood search for team formation and worker assignment problems in cellular manufacturing systems. Appl. Sci. **12**, 8323 (2022)
37. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Macskassy, S.A., Perlich, C., Leskovec, J., Wang, W., Ghani, R. (eds.) The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, New York, NY, USA, 24–27 August 2014, pp. 701–710. ACM (2014)
38. Rad, R.H., Bagheri, E., Kargar, M., Srivastava, D., Szlichta, J.: Retrieving skill-based teams from collaboration networks. In: SIGIR (2021)
39. Rad, R.H., Bagheri, E., Kargar, M., Srivastava, D., Szlichta, J.: Subgraph representation learning for team mining. In: WebSci (2022)
40. Rad, R.H., Mitha, A., Fani, H., Kargar, M., Szlichta, J., Bagheri, E.: PyTFL: a Python-based neural team formation toolkit. In: CIKM (2021)
41. Rad, R.H., Seyedsalehi, S., Kargar, M., Zihayat, M., Bagheri, E.: A neural approach to forming coherent teams in collaboration networks. In: EDBT (2022)
42. Rad, R.H., Fani, H., Bagheri, E., Kargar, M., Srivastava, D., Szlichta, J.: A variational neural architecture for skill-based team formation. ACM Trans. Inf. Syst. **42**(1), 7:1–7:28 (2024). https://doi.org/10.1145/3589762, https://doi.org/10.1145/3589762

43. Rad, R.H., Fani, H., Kargar, M., Szlichta, J., Bagheri, E.: Learning to form skill-based teams of experts. In: CIKM 2020 (2020)
44. Rahman, H., Roy, S.B., Thirumuruganathan, S., Amer-Yahia, S., Das, G.: Optimized group formation for solving collaborative tasks. VLDB J. **28**, 1–23 (2019)
45. Sapienza, A., Goyal, P., Ferrara, E.: Deep neural networks for optimal team composition. Front. Big Data **2** (2019)
46. Selvarajah, K., Zadeh, P.M., Kobti, Z., Palanichamy, Y., Kargar, M.: A unified framework for effective team formation in social networks. Expert Syst. Appl. **177**, 114886 (2021)
47. Sherer, P.D.: Leveraging human assets in law firms: human capital structures and organizational capabilities. ILR Rev. **48**, 671–691 (1995)
48. Sozio, M., Gionis, A.: The community-search problem and how to plan a successful cocktail party. In: SIGKDD (2010)
49. Stoller, J.K.: Building teams in health care. Chest **159**(6), 2392–2398 (2021)
50. Velickovic, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. In: ICLR 2019 (2019)
51. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings
52. Wang, L., Zeng, Y., Chen, B., Pan, Y., Cao, L.: Team recommendation using order-based fuzzy integral and NSGA-II in starcraft. IEEE Access **8**, 59559–59570 (2020)
53. Wang, X., et al.: Heterogeneous graph attention network. In: The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, 13–17 May 2019, pp. 2022–2032. ACM (2019)
54. Xiao, T., Chen, Z., Wang, D., Wang, S.: Learning how to propagate messages in graph neural networks. In: Zhu, F., Ooi, B.C., Miao, C. (eds.) KDD 2021: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, 14–18 August 2021, pp. 1894–1903. ACM (2021). https://doi.org/10.1145/3447548.3467451
55. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019. OpenReview.net (2019)
56. Yu, S., Xia, F., Liu, H.: Academic team formulation based on Liebig's barrel: discovery of anticask effect. IEEE Trans. Comput. Soc. Syst. **6**, 1083–1094 (2019)