2023

Temporal Neural Team Formation with Negative Sampling

Seyed Sobhan Dashti
University of Windsor

Temporal Team Formation with Negative Sampling

By

Seyed Sobhan Vagheh Dashti

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2023

Temporal Team Formation with Negative Sampling

by

Seyed Sobhan Vagheh Dashti

APPROVED BY:

M. Hassanzadeh
Department of Electrical and Computer Engineering

R. Gras
School of Computer Science

S. Samet, Co-Advisor
School of Computer Science

H. Fani, Co-Advisor
School of Computer Science

January 13, 2023

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Predicting *future* successful teams of experts who can synergistically work in concert with each other and en masse cover a set of required skills of a degree necessary for the achievement of the desired outcome is challenging due to several reasons, including 1) the magnitude of the pool of plausible expert candidates with diverse backgrounds and skills, and 2) the drift and variability of collaborative ties of experts and their level of expertise in each area in time. Prior works in team formation have neglected the fact that experts' skill, interests, and collaborative ties change over time. We can categorize previous works in team formation based on their method of optimization: 1) search-based, where the search for the optimum team is carried over all the subgraphs of expert networks or via integer programming, however, these works overlooked the temporal nature of human collaborations. 2) learning-based, where machine learning approaches are used to learn the distributions of experts and skills in the context of successful teams in the history to predict almost surely successful teams in the future. However, they also fail to recognize the possible drift and variability of experts' skills, interest, and collaborative ties in time and its impact on the prediction of future successful teams. Moreover, neural models are prone to overfitting when training data suffers from the long-tail phenomenon, i.e., few experts have a lot of successful collaborations and the majority have participated sparingly. To overcome the aforementioned problems, i) we propose a streaming scenario training strategy for neural models to help the model in the prediction of *future* successful teams of experts, where instead of shuffling our datasets, we train the models in an orderly manner, to grasp the changes in experts' skills, interests, and collaborations, and ii) we propose an optimization objective that leverages both successful and virtually unsuccessful teams via various negative sampling heuristics, and iii) we conduct experiments on four large-scale benchmark datasets with varying distribution of skills and members namely, `dblp`, `imdb`, `uspt`, and `github`. Finally, we empirically demonstrate how our proposed objective functions and training method, outperform the state-of-the-art approaches in terms of effectiveness and efficiency.

DEDICATION

I would like to dedicate this thesis to my mom for her incredible love and support. Because I believe that she is the real backbone of our family, this is to appreciate her selfless hard work and efforts towards the family.

Furthermore, I dedicate it to my dad to raise me like a son and give me wings to fly. To my grandfather, for always trusting me and supporting me in my hard times, without his encouragement, nothing would have been easy. And to my entire family for their unconditional affection towards me.

AKNOWLEDGEMENTS

I would like to sincerely express my most profound gratitude towards my supervisor, Dr. Hossein Fani, whose input helped me immensely. With his input, I was able to look at my research with a different perspective and a more critical eye. I also want to thank my co-supervisor, Dr. Saeed Samet, who patiently helped me grow as a researcher.

Secondly, I would like to express my gratitude to my thesis committee members for their beneficial advice and suggestions for my thesis.

I humbly extend my thanks to the School of Computer Science and all concerned people who helped me in this regard.

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY **III**

ABSTRACT **IV**

DEDICATION **V**

AKNOWLEDGEMENTS **VI**

LIST OF TABLES **IX**

LIST OF FIGURES **XI**

LIST OF ABBREVIATIONS **XII**

1 Introduction **1**
1.1 Importance of Team Formation 1
1.2 How teams have been formed 2
1.3 How we want to form teams 3
1.3.1 Considering unsuccessful teams to prevent failure 4
1.3.2 Predicting *future* successful teams 6

2 Related Works **9**
2.1 Non-temporal Methods 10
2.1.1 Search-based Methods 10
2.1.2 Learning-based Methods 13
2.2 Time as a constraint 14

3 Problem Definition **16**
3.1 Team Formation 16
3.2 Temporal Team Formation 16

4 Methodology **18**
4.1 Neural Team Formation with Negative Sampling 18
4.1.1 Negative Sampling Heuristics 19
4.2 Streaming Learning for Future Team Prediction 20
4.2.1 Streaming Learning 21

5 Experiments and Results **23**
5.1 Setup 25
5.1.1 Dataset 25
5.1.2 Baselines 31
5.1.2.1 Neural Team Formation with Negative Sampling . . 31
5.1.3 Evaluation Strategy and Metrics 32

5.2 Results 33
5.2.1 Impact of Negative Sampling 33
5.2.2 Impact of Streaming Training Strategy and Temporal Skills . 37
5.2.3 Hyperparameter Study 38
5.2.3.1 Model Size 39
5.2.3.2 #Bayesian Samples 40
5.2.3.3 Embedding Size 40
5.2.3.4 #Negative Samples 41

6 Conclusion and Future Work **43**

REFERENCES **45**

VITA AUCTORIS **53**

LIST OF TABLES

1.3.1 Sample research collaborations. 4

5.1.1 Statistics of the raw and preprocessed `dblp.v12` dataset: #teams with ≥`75` members, #members with ≥`3` teams. 29

5.1.2 Statistics of the raw and preprocessed `imdb` dataset: #teams with ≥`75` members, #members with ≥`3` teams. 29

5.1.3 Statistics of the raw and preprocessed `uspt` dataset: #teams with ≥`75` members, #members with ≥`3` teams. 30

5.1.4 Statistics of the raw and preprocessed `gith` dataset: #teams with ≥`10` members, #members with ≥`3` teams. 30

5.2.1 Average performance of `5-fold` (non-)Bayesian neural models on test set with(out) negative sampling in computer science publicaiton (`dblp`). 36

5.2.2 Average performance of `5-fold` (non-)Bayesian neural models on test set with(out) negative sampling in movie (`imdb`). 36

5.2.3 Average performance of 5-fold neural models with(out) streaming learning and temporal skills on test set in dblp. 38

5.2.4 Average performance of 5-fold neural models with(out) streaming learning and temporal skills on test set in imdb. 38

5.2.5 Average performance of 5-fold neural models with(out) streaming learning and temporal skills on test set in uspt. 39

5.2.6 Average performance of 5-fold neural models with(out) streaming learning and temporal skills on test set in gith. 39

5.2.7 Average performance of 5-fold neural models with different model sizes on test set in dblp. 40

5.2.8 Average performance of 5-fold neural models with different model sizes on test set in imdb. 40

5.2.9 Average performance of 5-fold neural models with different number of Bayesian samples on test set in dblp. 41

5.2.10 Average performance of 5-fold neural models with different number of Bayesian samples on test set in imdb. 41
5.2.11 Average performance of 5-fold neural models with different input embedding sizes on test set in dblp. 41
5.2.12 Average performance of 5-fold neural models with different input embedding sizes on test set in imdb. 42
5.2.13 Average performance of 5-fold neural models with different number of negative samples on test set in dblp. 42
5.2.14 Average performance of 5-fold neural models with different number of negative samples on test set in imdb. 42

LIST OF FIGURES

1.3.1 Streaming Training Strategy. 6

5.1.1 Distribution of teams over skills and members in computer science publications (`dblp`). 26

5.1.2 Distribution of teams over skills and members in computer science publications (`dblp`) after filtering. 26

5.1.3 Distribution of teams over skills and members in movies (`imdb`). . . . 27

5.1.4 Distribution of teams over skills and members in movies (`imdb`) after filtering. 27

5.1.5 Distribution of teams over skills and members in U.S. patents (`uspt`). 27

5.1.6 Distribution of teams over skills and members in U.S. patents (`uspt`) after filtering. 28

5.1.7 Distribution of teams over skills and members in GitHub repositories (`gith`). 28

5.1.8 Distribution of teams over skills and members in GitHub repositories (`gith`) after filtering. 28

5.2.1 Effect of negative sampling on training time vs. inference accuracy on `dblp.v12` dataset. 35

5.2.2 Effect of negative sampling on training time vs. inference accuracy on `imdb` dataset. 35

LIST OF ABBREVIATIONS

fnn	feed-forward neural network
bnn	bayesian neural network
vbnn	variational bayesian neural network
MST	Minimum Spanning Tree
emb	embedding
rnn	recurrent neural network
uspt	US patents
gith	GitHub
aucroc	area under curve of receiver characteristic operator
pr	precision
rec	recall
ndcg	normalized discounted cumulative gain
map	mean average prevision
#bs	number of bayesian samples
#nns	number of negative samples
OR	Operation Research
SNA	Social Network Analysis

CHAPTER 1

Introduction

1.1 Importance of Team Formation

Being a master of one or a jack-of-all-trades in today's world does not bring about success if you cannot work as part of a team. Teamwork was less vital during the industrial age than now, when most vocations were represented by individuals working on a manufacturing line all day. On the other hand, the value of teamwork has shown to be crucial in today's interdisciplinary environment, like in academia [63], manufacturing [13], law [55], freelancing [10], and the healthcare system [16]. Research groups whose success can be measured by scientific publications and citations in the scientific community, a group of inventors who create a product that meets market demands, or crew members for the upcoming blockbuster sci-fi movie with a touch of drama are a few instances of team formation.

Due to the sheer number of candidates with temporal backgrounds, cultural bonds, skills, and personality traits, as well as an unknown synergistic balance among them, it can be challenging to compose a successful team whose members can work together effectively and deliver the results within the designated constraints, such as the planned budget and timeline. Not all teams with the best experts are necessarily successful [56]. Furthermore, we all experience changes in our interests and abilities during the course of our lives, sometimes out of our own volition and other times due to circumstances. For instance, more and more individuals are required to learn new skills like programming due to the development of technology and automation. Consequently, the skills of any individual may or may not be relevant at any given

time.

1.2 How teams have been formed

In the past, teams have been put together based on intuition and experience, which occasionally leads to an inferior team composition due to insufficient candidate knowledge and unacknowledged cognitive biases, among other things. Recently, novel team formation strategies have been put forth to form expert teams that can work together on a particular job while considering both human and non-human aspects, including scheduling preferences, skill coverage, expert availability, budget, and team size.

The foremost algorithmic methods for team formation were developed in operations research (OR) [9, 60], where it is required to maximize several objective functions while considering constraints for human and non-human aspects and scheduling preferences. The goal is to identify the best composition for a team, or a team whose success is virtually positively assured. In addition to budget limitations and team size, other constraints include skill coverage and expert availability. However, such work ignored the organizational and social links among experts and was predicated on the mutual independence of expert selection.

In order to overcome this gap, researchers used social network analysis (SNA), which incorporates interpersonal collaboration and social relationships aspects utilizing metrics like density, degree centrality, and closeness centrality. To this end, most existing techniques characterize team member cooperation using a graph representation of the expert network, where there are direct and indirect relationships based on previous expert collaboration. Wherein, to find an optimum team optimization happens over all possible subgraphs based on, e.g., the diameter of the subgraph or the sum of the distances between every pair of nodes within subgraphs of the network. Wherein finding an optimum team optimization happens over all possible subgraphs based on, e.g., the diameter of the subgraph or the sum of the distances between every pair of nodes within subgraphs of the network.

The optimal synergistic integration of social network analysis for team formation is

severely hampered by the enormous difficulty faced by socially driven team formation approaches when efficiency is the primary consideration. Although heuristics have been suggested as a means of reducing the search space, for example, Akiba et al. [5] developed enhanced shortest path indexing approaches that speed up the computation of proximity functions, the output teams are still sub-optimal, and lack accuracy since the heuristics are often based on domain-specific or ill-posed assumptions and rendered irrelevant in general circumstances.

Recently, researchers approached the team formation problem using a learning-based methodology instead of search-based techniques. They have specifically proposed machine learning methods that utilize neural architectures to discover correlations between experts and their collaborative ties. Due to the fundamentally iterative and online learning procedure in neural networks, they take into account all previous successful team compositions as training samples to anticipate the best teams for a given set of required skills to increase efficiency while preserving efficacy.

Among the first, Sapienza et al. [54] utilized a neural autoencoder to form an ideal team with members who can collaborate effectively, promote one another's development, and enhance one another's skills. However, autoencoders are prone to overfitting and cannot recognize the uncertainty level in sparse data [12]. Moreover, according to Rad et al. [50], training datasets for team formation suffer from the long-tail phenomenon when a small number of experts successfully collaborate on a limited set of skills while the remainder only rarely participates. As a result, popular experts receive higher scores to the given skills than their ideal values and are recommended more frequently leading to the popularity bias. To overcome this problem, Rad et al. [50] employed a variational Bayesian neural model to relieve the long-tail problem through introducing uncertainty on the weights of the neural model.

1.3 How we want to form teams

Even though they work successfully as they do, current neural models only take into account teams who succeed and ignore those that don't. Table 1.3.1 displays a

Table 1.3.1: Sample research collaborations.

#	**research topics (skills)**	**members (experts)**	**published (success)**
1	natural language processing language modelling	Pennington Manning Socher	yes
2	natural language processing machine translation	Pennington Manning	yes
3	natural language processing machine translation	John Mary	no
4	evolutionary algorithms genetic programming	Mary John	yes
new	language modelling machine translation	?	?

fictitious sample group of research teams for various research topics (skills) and their researchers (experts), as well as whether or not their findings have been accepted for publication in a peer-reviewed journal. As can be seen, if we choose to research on {*language modelling, machine translation*} (last row), we may suggest {*Pennington, Manning, Socher*} as the ideal team because their collaborations have so far been the most effective on such research topics. Contrarily, proposing the team {*John, Mary*}, whose members failed the given research topics would not be a good idea. Proposed neural models overlook the latter case and give a non-zero chance to {*John, Mary*} when recommending a team for the given research topics.

1.3.1 Considering unsuccessful teams to prevent failure

It has been demonstrated in the literature that using both positive and negative samples, such as connections in social networks or distrust, conveys complementary signals to neural models and increases accuracy in various tasks related to social network analysis, natural language processing, and recommender systems [34, 37,

57, 49, 43, 53, 64]. Indeed, team formation can also be subsumed into recommender systems where experts are recommended for the required skills. However, the majority of real-world training datasets in the team formation domain lack explicit failure teams (e.g., collections of rejected papers). Based on the closed-world assumption and in the absence of unsuccessful teams, we assume that a group of experts is an unsuccessful team if they have not already worked together for the required subset of skills.

To this end, we develop three negative sampling heuristics: 1) *uniform*: where subsets of experts are randomly selected with the same probability as samples of unsuccessful teams, 2) *unigram*: where subsets of experts are chosen based on their frequencies in the training set, and 3) *smoothed unigram in training minibatches*: where we employed Laplace smoothing to calculate the unigram probability of subsets of experts in each training minibatch. In unigram and smoothed unigram heuristics, experts that have collaborated more often on skills different from the given input skills will be chosen more frequently to diminish popularity bias.

We use negative samples during training to prompt neural models to learn vector representations (embeddings) for experts and skills in the same vector space so that vectors of experts who have already collaborated for the required skills (have been in the same teams) end up closer to each other whereas vectors of experts who have not yet collaborated (virtually unsuccessful teams) end up farther apart. We replicate the proposed Bayesian and non-Bayesian neural models under the use of negative sampling heuristics and a lack thereof on two large-scale datasets from various domains with varied distributions of experts and skills in teams, namely academic papers (dblp)[1] and movies (imdb)[2]. The empirical findings demonstrate that, for the best team, Bayesian neural models constantly offer a greater prediction power when integrating negative samples. Non-Bayesian neural models, on the other hand, are sensitive to how teams are distributed, and negative samples have a discounting effect as in imdb.

[1] aminer.org/citation
[2] imdb.com/interfaces/

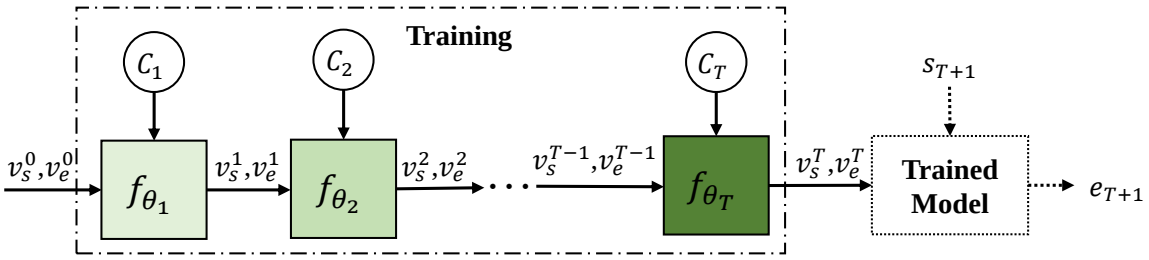

Fig. 1.3.1: Streaming Training Strategy.

1.3.2 Predicting *future* successful teams

To the best of our knowledge, there has not been any research on the impact of temporality on team formation, that is, how expert skills and collaborative ties change over time or how societal demands change. The transient character of experts' interests, skill, and collaborative ties as well as the constantly shifting demands of society were not taken into consideration in any of the earlier studies. Additionally, while some studies in the operation research literature utilize time as a constraint, in our study we concentrate on the horizontal component of time where change and progress occurs in experts' activities and society's demands. We propose a temporal neural team formation to predict the ideal expert teams of the *future*. First, our methodology involves temporality in contrast to non-temporal methods. Second, we examine the horizontal character of time to take into account variation in expert behaviour, in contrast to operation research literature that considers time as a constraint. Last but not least, in contrast to earlier neural models for team formation, which aim to predict expert teams that will work successfully and efficiently regardless of the time of their collaborations, we aim to predict expert teams that will work optimally and effectively for a specific set of required skills in the *future*.

Our objective is to predict *future* expert teams with a new set of required skills in time interval T + 1, given a sequence of expert collaborations with the required set of skills from time intervals 1 to T. In order to achieve this aim, we train our neural models starting from the earliest expert collaborations with their required skills at time interval 1 rather than using the traditional method of shuffling the dataset. We then utilize the developed model to kick-start training on the following time interval,

as shown in Fig. 1.3.1. We train our models in this way to account for the evolution of experts' interests, skills, and collaborative ties over time, ensuring that the model has absorbed the most recent and up-to-date interests, skills, and collaborative ties when training for time interval T is complete.

Our suggested training technique allows experts to adjust their vector locations in latent space as their knowledge and relationships with others change over time. It then records the change trajectories up to time interval T to correctly forecast experts' vector positions in the *future* time interval T+1. In contrast to non-temporal approaches that use a bag of teams to train instances of teams and presume i.i.d [35, 28, 54, 50], our approach integrates temporality by streaming the teams over time intervals throughout the training phase. Second, rather of approaching time as a constraint, we investigate the horizontal aspect of time to understand the development of expert skills and linkages between teams. In order to show the domain-free efficacy of our suggested strategy, we conduct tests on four major datasets: scholarly papers in computer science (dblp)[3], movies (imdb)[4], US patents (uspt)[5], and github repositories (github)[6]. Our findings demonstrate that taking into account the temporal evolution of expert skills and cooperation links outperforms state-of-the-art approaches in terms of identifying *future* effective expert teams. Concretely, our contributions are as follows:

1. We propose and develop three negative sampling heuristics to incorporate virtually unsuccessful teams in neural team formation.

2. We reproduce the state-of-the-art Bayesian neural models as well as non-Bayesian neural models under the negative sampling heuristics and lack thereof to study the effects of incorporating negative samples during neural model training.

3. We investigate the effect of negative sampling heuristics on training time speedup vs. inference accuracy.

[3]aminer.org/citation
[4]imdb.com/interfaces/
[5]https://www.uspto.gov/ip-policy/economic-research/research-datasets
[6]github

4. We critically assess neural models with and without negative sampling heuristics on two large-scale datasets of scholarly papers (`dblp`) and movies (`imdb`), that are from different domains with distinct statistical distributions of skills in teams.

5. We demonstrate the effect of negative sampling heuristics for neural team formation on a host of information retrieval and classification metrics such as map, ndcg, as well as precision, recall, and rocauc.

6. We propose a streaming scenario training strategy to utilize the evolution of experts' interests, skills, and collaborative ties over time.

7. We examine the impact of our proposed streaming training strategy on the state-of-the-art neural models' abillity to predict *future* successful teams on four large-scale datasets of (`dblp`), movies (`imdb`), US patents (`uspt`), and github repositories (`gith`), that are from different domains with distinct statistical distributions of skills in teams.

8. We showcase the impact of our streaming training strategy on prediction of *future* successful teams using a host of information retrieval and classification metrics such as map, ndcg, as well as precision, recall, and rocauc.

CHAPTER 2

Related Works

Since Zakarian and Kusiak's work [66], there has been an influx of literature in the team formation domain that can be distinguished based on their optimization method: 1) search-based, where the search for the optimum team is carried over all the subgraphs of expert networks or via integer programming, and 2) learning-based, where machine learning approaches are used to learn the distributions of experts and skills in the context of experts' past successful collaborations to form successful teams. Even though literature in natural language processing [49, 43] and graph neural network [34, 37, 57] has already shown that using negative samples can improve the model's efficiency during training and effectiveness during inference, no work has utilized negative sampling in team formation. Moreover, literature related to the team formation problem has ignored the impact of time and the temporal nature of experts' interests, skills, and collaborative ties by and large despite widespread successful incorporation of temporality in other domains such as temporal information retrieval, temporal knowledge graphs, and temporal recommender systems [20, 39]. There is little work [9, 52] that studied time but as a constraint like the projects' deadlines in the optimization function. In this section, we review some of the remarkable works in the team formation literature:

2.1 Non-temporal Methods

2.1.1 Search-based Methods

The foremost methods of team formation was conceived in the Operations Research (OR) where multiple objectives must be optimized simultaneously via integer or real programming to find the optimum team, given constraints for human and non-human factors as well as scheduling preferences. Based on the engineering characteristics of the product and the importance of customer requirements, Zakarian et al.[66] used the integer linear programming approach to form multi-functional teams. They imposed integer constraints on applicants, such as a cap on the amount of projects each team member may take on, the required number of teams, and when to join which team. Wi et al. [60] presented a framework to analyze candidates' ability for the manager position and team members. They proposed a genetic algorithm and social network measures to pick the team manager and the team members. More recently, Neshati et al. [44] proposed an optimization framework based on the Facility Location Analysis, a well-known branch in Operation Research, to form groups of experts to perform a multi-aspected task. Even though the importance of effective cooperation has been studied rigorously before [22, 15], these works, however, were premised on the mutually independent selection of experts and overlooked the organizational and social ties among experts.

Although Chen and Lin [15] were among the first to consider candidates' interpersonal relationships for team formation, they were Lappas et al. [35] who employed social network analysis to fill the gap by incorporating social ties and interpersonal collaboration features. They represented the experts' social network with a graph where nodes are experts with their set of skills, and edges represent the previous collaboration between them. The optimum team hence can be found by a search on all possible subgraphs. They proposed two algorithms based on the diameter of the graph and the cost of the minimum spanning tree (MST) to find a subgraph in which experts collectively hold the set of required skills and can collaborate effectively with minimum communication cost. Finding a subgraph with a minimum diameter

in MST, however, falls short to estimate the true cost of all the required communications. Kargar and An [28] claim that neither the diameter of the sub-graph nor the cost of the minimum spanning tree measure the communication cost accurately. The diameter function only calculates the communication cost between the two experts furthest away from each other. The MST function does not measure the cost of all the required communication either. The other disadvantage of these functions is their instability to slight changes in the graph, which results in a radical change in the solution. To overcome these issues, they proposed two novel communication cost functions that minimize the sum of distances function for teams with/without a leader and an approximation algorithm to minimize the cost.

Later, Kargar et al. [30] further proposed to consider additional budget constraints (expert salary) on top of communication costs as in real-world scenarios. Therefore, they proposed an approximation algorithm to optimize two objectives: communication cost and salary. Li et al. [38] included additional factors by specifying the number of required experts for each skill. They devised the generalized Enhanced Steiner algorithm considering the number of experts, subsequently, they condense the expertise information to a compact representation based on the required skills. Also, they propose a density-based measure and embed it into their method, which improves the effectiveness of the final teams.

Rahman et al. [51] adapted two criteria from organizational sciences and social theories: affinity and upper critical mass. They defined affinity based on age and geographical location and conducted experiments to reveal the importance of a limit on the size of teams for successful collaboration. They proved that their problem formulation is computationally expensive and unacceptable for a real-time crowdsourcing platform. Even after breaking down the problem into two stages, it is still computationally intractable in the worst case, but it allows them to design optimal exact algorithms for the average case and efficient approximation algorithms with provable bounds. In the first stage, they form a group of workers with maximal intra-affinity that also satisfies the required skills and cost constraint. In the second stage, they break down the group into smaller cliques that satisfy the upper critical size

constraint with maximal inter-affinity between smaller groups.

Khan et al. [32] introduced the problem of compact attributed group (AG) discovery, which is another generalized version of the Team Formation problem, where given a set of query keywords and the desired solution size, they aim to find closely connected sub-graphs with the specific number of nodes where each node contains as many query keywords as possible. They also proved that their objective is NP-hard and therefore propose an approximation algorithm with a guaranteed ratio of two. Since the total number of answers is exponential in the number of query keywords and the size of the group, they proposed a method to find the approximate top-k groups with polynomial delay. Kou et al. [33] disclose that the previous literature did not take into account that team members with different skills/roles have different degrees of communication. Therefore, they proposed a novel method that considers both the structure and communication constraints based on the Constrained Pattern Graph (CPG). First, they present a preprocessing method to normalize a CPG. Second, they construct a Communication Cost Index (CCI) to accelerate the matching between a CPG and its corresponding social network. Lastly, they proposed a CCI-based node matching algorithm to minimize the total number of intermediate results.

Methods of efficient keyword search on attributed graphs have also been employed for team formation [33, 32]. For instance, given a set of query keywords as skills and the desired size of the subgraph as the team size, Khan et al. [32] aimed to find closely connected subgraphs with the specific number of nodes wherein nodes contain as many query keywords as possible. Since the total number of answers is exponential in the number of query keywords and the size of the group, they proposed a method to find the approximate top-k groups with polynomial delay.

Nonetheless, the above proposed optimization models for the task of team formation were all computationally intractable and had to be followed by polynomial heuristic solutions such as Multichoice [7] for subgraph identification with shortest diameter in [35] or simulated annealing [9], branch-and-cut, genetic algorithms [60], and balanced placement [22] for those based on integer programming (IP). Indeed, IP is NP-hard and subgraph optimization can be reduced to the decision version of the

Steiner-tree problem which is proved to be NP-Hard [31]. Moreover, network changes due to the constant emergence of new collaborations lead to frequent changes in the shortest paths, which requires the expensive recalculation of indexed shortest paths for all pairs of experts. Inevitably, all such methods undergo heuristics to efficiently find the optimum team most of which are based on ill-posed assumptions such as minimizing communication costs necessarily yields in successful teams. Indeed, to the best of our knowledge, no search-based methods have evaluated their heuristics (assumptions) intrinsically on a labelled dataset of successful teams.

2.1.2 Learning-based Methods

Contrary to search-based methods, little work has been proposed recently to use machine learning to overcome these issues. Wherein, all past successful team compositions are considered as training samples to machine learning algorithms to automatically form optimum teams from a large pool of experts to accomplish required tasks successfully. Sapienza et al. [54] employed a deep neural autoencoder to form teams and proposed a computational framework to capture which teammates foster growth of their peers. However, when data is sparse, such as in the case of Team Formation, where a few teams have successful collaboration for a specific set of skills, autoencoder neural networks are prone to overfitting and cannot assess the uncertainty in data effectively [12].

Rad et al. [50] proposed a Variational Bayesian neural architecture to overcome these shortcomings. However, their model was trained on published scholarly papers in computer science and lacks observing unsuccessful research (rejected papers). Literature in natural language processing [49, 43] and graph neural network [34, 37, 57] has already shown that utilizing positive and negative instances in tandem carries complementary signals to the model in order to deviate from the popularity bias and can improve the model's efficiency during training (i.e., the model converges sooner to the minimum loss) and effectiveness during inference. To this end, we aim at exploiting not only successful teams but also unsuccessful ones and studying how they would fare in the context of neural team formation. We demonstrate that neural models

that take advantage of unsuccessful instances (negative samples) are more efficient in training, also more effective in inference. we aim at exploiting not only successful teams but also unsuccessful ones. We show that learning-based models that utilize unsuccessful instances of teams (take advantage of negative samples during training) are able to excel at both effectiveness during inference and efficiency by discriminating successful teams from unsuccessful ones during training.

2.2 Time as a constraint

There has been little work on the impact of time as a constraint. Given hard-crisp constraints such as the team candidates' availability and salaries (human factors), as well as time constraints for start and due dates (non-human factors), Baykasoglu et al. [9] introduced a fuzzy bi-objective optimization model, namely team size, and suitability, which maximizes members' fit to the team based on their degrees of competence (skill). They employed a simulated annealing (SA) approach to solve the suggested fuzzy team formation model utilizing a max-min operator.

When forming real-time, on-demand, ad hoc teams of experts from various sources, Durfee et al. [19] take into account scheduling constraints or preferences in a two-step team formation process. Teams are built first and foremost in the matchmaking optimization stage using integer linear programming, taking into account not only the required skills and knowledge but also the ability to be more readily (re)scheduled in accordance with the timing requirement. In the scheduling optimization stage, time slots are allotted to the team for completing the work using integer nonlinear programming optimization in a way that minimizes the total of the starting times of all the members while satisfying sequential and concurrent ordering constraints.

Rahmanniyay et al. [52] studied various factors that can change the duration of a project like weather conditions can delay the delivery of material to a manufacturing company. They benchmark different scenarios for the required amount of time that is needed to accomplish an activity in a project. Yang et al. [62] apply integer programming to determine the optimum group of experts available at a certain point

in time. Their temporal scheduling technique considers the social distance between group members to avoid lacking too many direct links.

Contrary to these works, we investigate the temporality of experts' skills and collaboration ties and utilize how they evolved through time to predict their *future* teams. Further, such work is again based on exhaustive search in multidimensional integer space as in integer programming and hence, is computationally prohibitive.

CHAPTER 3

Problem Definition

In this chapter, we will define the problem of team formation in two parts: *i*) Team Formation and *ii*) Temporal Team Formation.

3.1 Team Formation

Our goal is to assemble the best possible team of experts who can successfully collaborate on a task that requires a set of skills. Let $\mathcal{S}$ and $\mathcal{E}$ be the set of skills and experts, respectively. $\mathcal{C} = \{((s, e), y); s \subseteq \mathcal{S}, e \subseteq \mathcal{E}, s, e \neq \emptyset, y \in \{0, 1\}\}$ be the set of all collaborations where (s, e) is a team whose members are a subset of experts e that collectively hold the subset of skills s and has been either successful $y : 1$ or a failure $y : 0$. Given a subset of skills s and all the previous collaborations $\mathcal{C}$, we aim at identifying an optimal subset of experts e such that their collaboration in the predicted team (s, e) will be successful, that is $((s, e), y = 1)$, and avoiding subset of experts e' that $((s, e'), y = 0)$. More concretely, we aim to estimate a mapping function f of parameters θ from a subset of skills and experts to a boolean set; $f_\theta : P(\mathcal{S}) \times P(\mathcal{E}) \rightarrow \{0, 1\}$.

3.2 Temporal Team Formation

We aim to incorporate the evolution of experts' skills and collaborative ties over time in order to predict *future* successful teams of experts who collectively hold a set of required skills and can effectively cooperate toward a shared goal based on their

gained experience through time. Let $\mathcal{S}$ and $\mathcal{E}$ be the sets of skills and experts and $\mathcal{C}_t = \{(s, e, y)_t | s \subseteq \mathcal{S}, e \subseteq \mathcal{E}, 1 \leq t \leq T, y \in \{0, 1\}\}$ be the set of collaborations at time t where (s, e) is a team whose members are a subset of experts e that collectively hold the subset of skills s and has been either successful $y = 1$ or a failure $y = 0$, and t is a discrete entity showing the time intervals. Intuitively, $\mathcal{C}_t$ is a snapshot of all teams of experts over skills during the time interval t and $[\mathcal{C}_1..\mathcal{C}_t..\mathcal{C}_{\mathrm{T}}]$ streams the dynamic distribution of experts over skills within T consecutive time intervals in the context of teams. Given the stream of collaboration sets $[\mathcal{C}_t]_{1 \leq t \leq \mathrm{T}}$ in the past, we aim to recommend a new team of experts e' for a given subset of skills s' at a yet-to-be-seen time interval T+1 whose collaboration has a high chance of success, i.e., $(s', e', 1)_{\mathrm{T}+1}$. More formally, we aim to estimate a mapping function f of parameters θ from the stream of collaboration sets and a subset of skills to a subset of experts whose collaboration in a team is almost surely successful for the one-step-ahead *future* time interval T+1; that is, $f([\mathcal{C}_t]_{t \leq \mathrm{T}}, s'; \theta) = e'$ such that $(s', e', 1)_{\mathrm{T}+1}$.

CHAPTER 4

Methodology

In this chapter, we will explain the methods we used to tackle the problems introduced in the Problem Definition section.

4.1 Neural Team Formation with Negative Sampling

Given all previous collaborations $\mathcal{C}$, we maximize the average log probability of teams' success or failure:

$$\frac{1}{|\mathcal{C}|} \sum_{((s,e),y)\in\mathcal{C}} \log \mathrm{P}(y|(s,e) \tag{1}$$

where (s, e) is a team of experts e who collectively hold the set of skills s and can either work successfully together or fail otherwise. We propose to learn vector representations (embeddings) for experts and skills in the same vector space with the expectation that vectors of experts whose teams have been successful for the required skills will end up closer to each other in the vector space while vectors of experts whose teams for the required skills have been unsuccessful will end up farther from each other. We estimate the $\mathrm{P}(y|(s,e)$ through pairwise cosine similarities of vector representations for the skills $\forall i \in s$ and experts $\forall j \in e$. Specifically, for a successful team $((s,e), y = 1)$, we estimate $\mathrm{P}(y = 1|(s,e)$ by learning $v_s = \sum_{i\in s} v_i$ and $v_e = \sum_{j\in e} v_j$ that are close in the vector space and have high cosine similarity while for an unsuccessful team $((s,e), y = 0)$, we estimate $\mathrm{P}(y = 0|(s,e))$ by learning v_s and

v_e that are far from each other and have low cosine similarity. Formally, $\mathrm{P}(y|(s,e))$ can be formulated using the sigmoid function σ:

$$P(y|(s,e)) = \sigma(v_e^\top \cdot v_s) \tag{2}$$

where v_s and v_e are the vector representations of the skill and expert subsets, respectively.

4.1.1 Negative Sampling Heuristics

We have only data for successful teams for the team formation problem. The `dblp` dataset only contains published research papers in computer science and does not have rejected submissions. In the `imdb` dataset of movies, it remains controversial what constitutes a failure for a movie; its reception by the people (box office) or critical reviews. In the absence of unsuccessful training instances, we follow the closed-world assumption that no currently known successful team for the required skills is considered unsuccessful. We presume that teams of experts e who have little or no collaborative history, i.e., few or no (s,e), have a low chance at yielding success. For instance, given $s = \{$*natural language processing*, *machine translation*$\}$, an accurate estimator f would recommend $e = \{$*Pennington*, *Manning*, *Socher*$\}$ who collectively hold the skills in s and have already collaborated in successful publications. In contrast, f would not recommend team of researchers $e = \{$*Banzhaf*, *Nordin*$\}$ who have not had a shared publication related to skill subset s. Inspired by [43, 34, 37, 57, 49], we propose an optimization function that discriminates successful from unsuccessful teams through negative sampling from a distribution over the subsets of experts:

$$\sum_{((s,e),1)\in\mathcal{C}} [\log \sigma(v_e^\top \cdot v_s) + \sum_{(s,e')\sim\mathbb{P}:(s,e')\notin\mathcal{C}}^{k} \log \sigma(-v_{e'}^\top \cdot v_s)] \tag{3}$$

where $\mathbb{P}$ is the probability distribution from which we draw k subsets of experts e' as negative samples for a given subset of skills s where $(s,e) \in \mathcal{C}$ but $(s,e') \notin \mathcal{C}$. We present three different negative sampling distributions, two static negative sampling

distributions [43] and an adaptive noise distribution [17, 4, 49], and study their effects on neural models:

1. **uniform distribution**, where each subset of experts e' is chosen with the same probability from the uniform distribution over all subsets of experts $\mathcal{P}(\mathcal{E})$, i.e. $\mathrm{P}(e') = \frac{1}{|\mathcal{P}(\mathcal{E})|}$

2. **unigram distribution**, where each subset of experts e' is chosen regarding their frequency in all previous teams, i.e. $\mathrm{P}(e') = \frac{|(s',e')|}{|\mathcal{C}|}$ and (s', e') is a team with skill subset $s' \neq s$. Intuitively, subsets of experts that have been in previous teams for other subsets of skills will be given a higher probability and chosen more frequently as negative samples to dampen the effect of popularity bias.

3. **smoothed unigram distribution *in training minibatch***, where we employed the add-1 or Laplace smoothing when computing the unigram distribution of the experts in each training minibatch, i.e. $\mathrm{P}(e') = \frac{1+|(s',e')|}{|b|+|\mathcal{E}|}$, where b is a minibatch subset of $\mathcal{C}$, and (s', e') is a successful team including expert e' in each training minibatch. Minibatch stochastic gradient descent is the *de facto* method for neural models where the data is split into batches of data, each of which is sent to the model for partial calculation in order to speed up training while maintaining high accuracy. Since only a few teams of experts exist in each minibatch, we employ the Laplace smoothing so that no subsets of experts have zero probability.

4.2 Streaming Learning for Future Team Prediction

Let $[\mathcal{C}_t]_{1\leq t\leq \mathrm{T}}$ be the ordered list of all previous collaborations at each time interval t until T in which experts' collaborations over skills in teams are evolving over time. We aim to estimate f using a neural model that maximizes the average log probability of successful subsets of experts:

$$\frac{1}{|\mathcal{C}_{\mathrm{T}+1}|} \sum_{((s,e),y)\in\mathcal{C}_{\mathrm{T}+1}} \log \mathrm{p}(y|(s,e):\mathrm{T}+1) \tag{4}$$

where $\mathcal{C}_{\mathrm{T}+1}$ is the collection of yet-to-be-formed unseen (un)successful teams $((s,e),y)$ in the *future* time interval T+1. Since $C_{\mathrm{T}+1}$ is unseen, we optimized eq. 4 through observed teams of $((s,e),y)$ in the past:

$$\frac{1}{|[\mathcal{C}_t]_{1\leq t\leq \mathrm{T}}|} \sum_{t=1}^{\mathrm{T}} \sum_{((s,e),y)\in\mathcal{C}_t} \log \mathrm{p}(y|(s,e):t) \tag{5}$$

The same team (s,e) may experience instances of success and/or failure in different time intervals. Therefore, $p(y|(s,e):t)$ depends on the time interval information. To maximize eq. 5, we map each subset of skills s and each subset of expert e to a low-rank d-dimensional vector in the same latent space, denoted by v_s and v_e, whose positions up until time interval T depend on the preceding movements in the latent space since the first time interval via observation of $[\mathcal{C}_1..\mathcal{C}_t..\mathcal{C}_{\mathrm{T}}]$ while imposing the following assumptions: (i) skills and experts change their latent representations over time, (ii) subsets of experts who collaborated in teams over similar subsets of skills within $[\mathcal{C}_t]_{1\leq t\leq \mathrm{T}}$ remain close in latent space, (iii) subsets of experts and skills who are close in latent space at their final positions in the latent space are presumably the optimum teams whose successes are almost surely guaranteed in the *future* time interval T+1.

4.2.1 Streaming Learning

Previous works in team formation assume i.i.d property among teams and followed the bag of teams approach during model training on a shuffled dataset [9, 52, 35, 28, 29, 54, 50]. In this work, however, we train a neural model incrementally through an ordered collection of teams in $[\mathcal{C}_1,..\mathcal{C}_t,..\mathcal{C}_{\mathrm{T}}]$. As seen in Fig. 1.3.1, after random initialization of skills' and experts' embeddings, we start training the model on the teams in the first time interval $\mathcal{C}_1$ for a number of epochs, then we continue with training on the second time interval $\mathcal{C}_2$ using the learned embeddings from the first

time interval and so forth until we finish the training on the last training time interval $\mathcal{C}_{\mathrm{T}}$. We believe that using this approach, will help the model understand how experts' skills and collaborative ties evolve through time and the final embeddings are their optimum representation in the latent space to predict *future* successful teams.

At each time interval t, we estimate $p(y|(s,e):t)$ through pairwise cosine similarities of embeddings for the subset of experts e and subset of skills s through all successful and unsuccessful teams at time interval t in $\mathcal{C}_t$. More specifically, we estimate $p(y=1|(s,e):t)$ by learning v_e and v_s that are close (high cosine similarity) in the latent space if the subset of experts e has successful collaborations in $\mathcal{C}_t$ with the subset of skills s during the time interval t and estimate $p(y=0|(s,e):t)$ by learning v_e and v_s that are distant (low cosine similarity) otherwise. Hence, $p(y|(s,e):t)$ can be formulated with the sigmoid function σ:

$$p(y|(s,e):t) = \sigma(v_e^\top \cdot v_s) \tag{6}$$

When no unsuccessful team is available in the training set, we again follow the closed-world assumption to generate *virtually* unsuccessful teams (negative samples), that is, if no successful team for the subset of skills s is known for a randomly selected subset of experts e' at time interval t, i.e., $(s,e') \notin \mathcal{C}_t$, the team is considered to be unsuccessful $(s,e',0)$. To this end, we employ an optimization function that discriminates successful and unsuccessful teams through negative sampling from a distribution over the subsets of experts:

$$\sum_{1\leq t\leq T}[\sum_{((s,e),1)\in\mathcal{C}_t}[\log\sigma(v_e^\top \cdot v_s) + \sum_{(s,e')\sim\mathbb{P}:(s,e')\notin\mathcal{C}_t}^{k}\log\sigma(-v_{e'}^\top \cdot v_s)]] \tag{7}$$

where $\mathbb{P}$ is the probability distribution from which we randomly draw k subsets of experts e' as negative samples for a given subset of skills s.

CHAPTER 5

Experiments and Results

In this chapter, we lay out the details of our experiments and expound on how we examined our proposed methods for the team formation problem. More concretely, we want to address the following research questions:

RQ1: Does negative sampling improve the effectiveness of neural models for the task of team formation? To this end, we benchmark the state-of-the-art Bayesian neural model [50] as well as non-Bayesian neural baselines with our proposed negative sampling heuristics compared to lack thereof.

RQ2: Are the impacts of negative sampling heuristics robust across different training datasets with diverse statistical characteristics? We benchmark baselines for the proposed negative sampling heuristics on computer science publications (`dblp`) and movies (`imdb`).

RQ3: How does negative sampling help efficiency of neural models during training while improving inference effectiveness?

RQ4: Does moving embeddings of experts and skill through time improve the performance of neural models for the prediction of *future* successful teams? To this end, we benchmark state-of-the-art variational Bayesian neural network [50] that utilizes negative sampling heuristics with our proposed streaming scenario training approach and lack thereof.

RQ5: Does adding time explicitly to the input embeddings of skills boost neural models performance? We compare the performance of neural models with and without utilizing temporal skills in the input.

RQ6: Is the impact of our proposed training strategy consistent across different train-

ing data with distinct statistical distributions? We benchmark our proposed training approach on computer science publications (`dblp`), movies (`imdb`), US patents (`uspt`), and github repositories (`gith`).

5.1 Setup

5.1.1 Dataset

Our testbed includes four datasets, namely, `dblp`, `imdb`[1], `uspt`[2], and `gith`. Compared to Rad et al.'s [50] work, we used the more recent and large-scale version of `dblp.v12`[3], where each instance is a publication in computer science consisting of authors, fields of study (fos), and the year it was published, including published papers from 1979 to 2018. We map each publication to a team whose authors are the experts and fields of study are the set of skills. In `imdb`, each instance is a movie consisting of its cast and crew such as actors, director, and producers, as well as the movie's genres and the year it was published spanning from 1914 to 2020. We consider each movie as a team whose members are the cast and crew, and the movie's genres are the teams' required skills. In uspt, each instance is a patent invention in the United States Patents and Trademarks consisting of inventors (experts) and subcategories (skills) and the time the patent is issued, consisting of patents from 1976 to 2019. In github (`gith`)[4], each instance is a repository consisting of the contributors of the repository (experts), the title and programming languages of the project (skills), and the time of the project's release, consisting of repositories from 2008 to 2022.

Like Rad et al. [50], we filter out members who participated in less than `75` teams and teams with less than `3` members for `dblp`, `imdb`, and `uspt`, and filter out members who particiate in less than `10` teams and teams with less than `3` members for `gith` due to its smaller size compared to the other three datasets. In all datasets, we can observe long tails in the distributions of teams over experts. As shown in the left side of Figures 5.1.1, 5.1.3, 5.1.5, 5.1.7 before filtering and in Figures 5.1.2, 5.1.4, 5.1.6, 5.1.8 after filtering, many experts (researchers in `dblp`, cast and crew in `imdb`, inventors in `uspt`, and developers in `gith`) have participated in very few teams (papers in `dblp`, movies in `imdb`, inventions in `uspt`, and repositories in `gith`). For instance,

[1] imdb.com/interfaces/
[2] uspto.gov/ip-policy/economic-research/research-datasets
[3] aminer.org/citation
[4] https://console.cloud.google.com/marketplace/details/github/github-repos

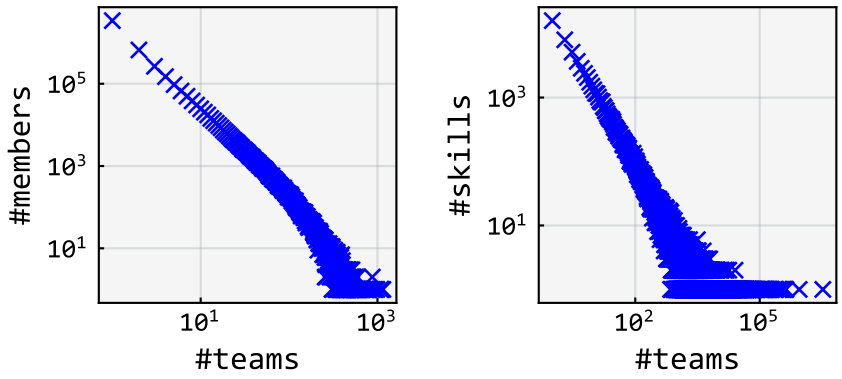

Fig. 5.1.1: Distribution of teams over skills and members in computer science publications (`dblp`).

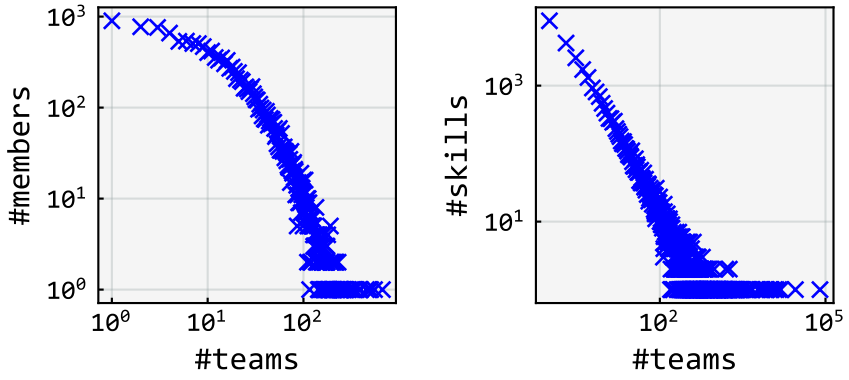

Fig. 5.1.2: Distribution of teams over skills and members in computer science publications (`dblp`) after filtering.

the left side of Figure 5.1.1 shows that close to 10^6 researchers have participated in 1 team only (top-left corner) while few researchers have co-authored more than 10^3 papers (bottom-right corner). With respect to the set of skills, `dblp`/`uspt` and `imdb`/`gith` are clearly following different distributions. While `dblp` and `uspt` suffer further from the long-tailed distribution of skills in teams as shown in the right sides of Figures 5.1.1 and 5.1.5(before filtering), and Figures 5.1.2 and 5.1.6(after filtering), `imdb` and `gith` follow a more fair distribution as shown in the right sides of Figures 5.1.3 and 5.1.7(before filtering), and Figures 5.1.4 and 5.1.8. Specifically, `imdb` and `gith` have a limited variety of skills (genres and programming languages) which are, by and large, employed by many movies and repositories.

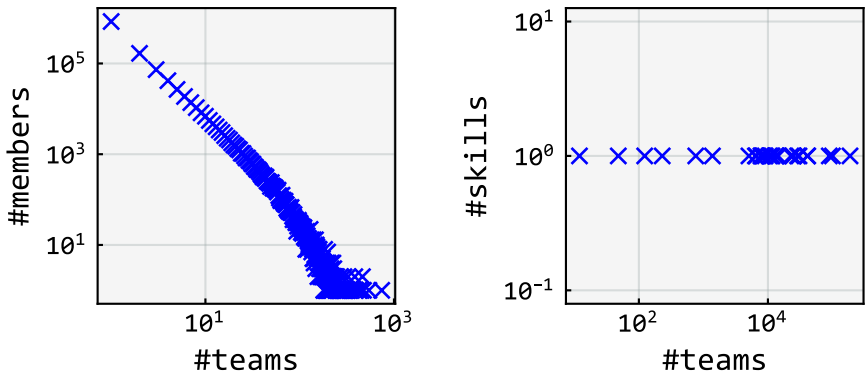

Fig. 5.1.3: Distribution of teams over skills and members in movies (`imdb`).

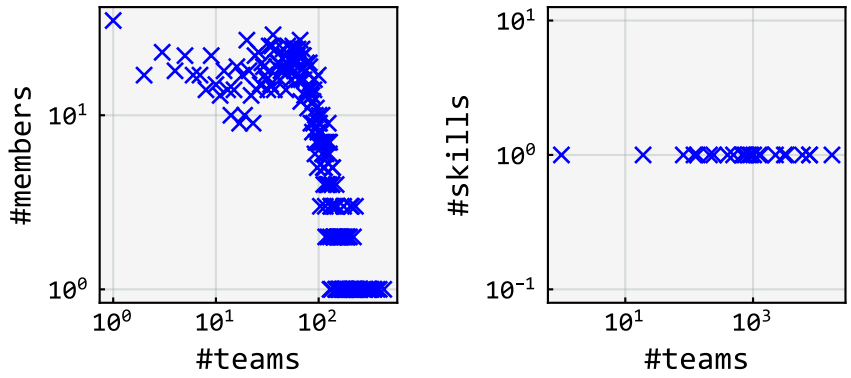

Fig. 5.1.4: Distribution of teams over skills and members in movies (`imdb`) after filtering.

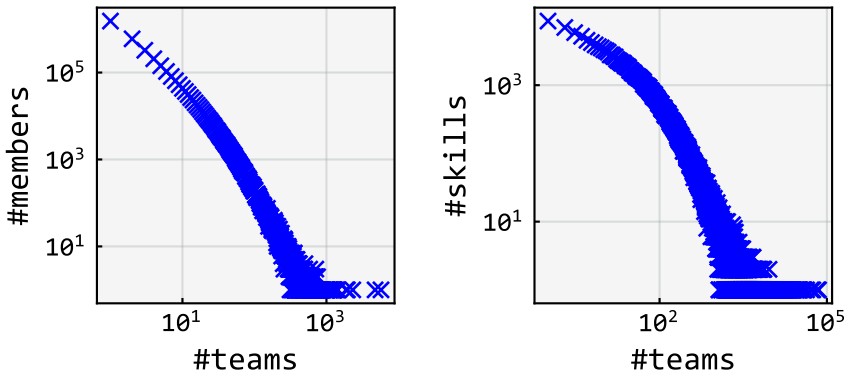

Fig. 5.1.5: Distribution of teams over skills and members in U.S. patents (`uspt`).

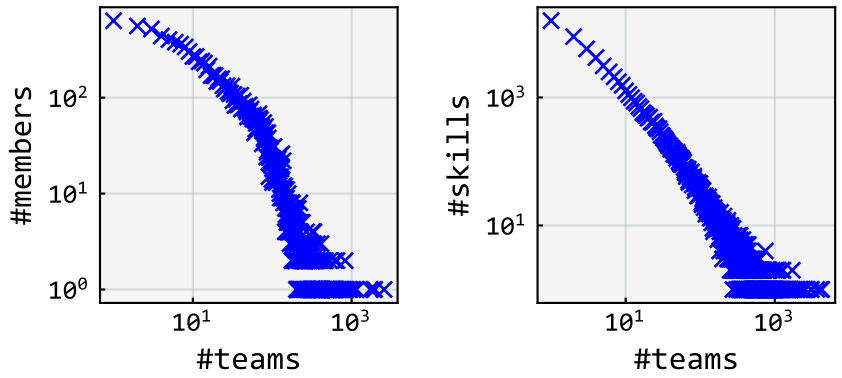

Fig. 5.1.6: Distribution of teams over skills and members in U.S. patents (`uspt`) after filtering.

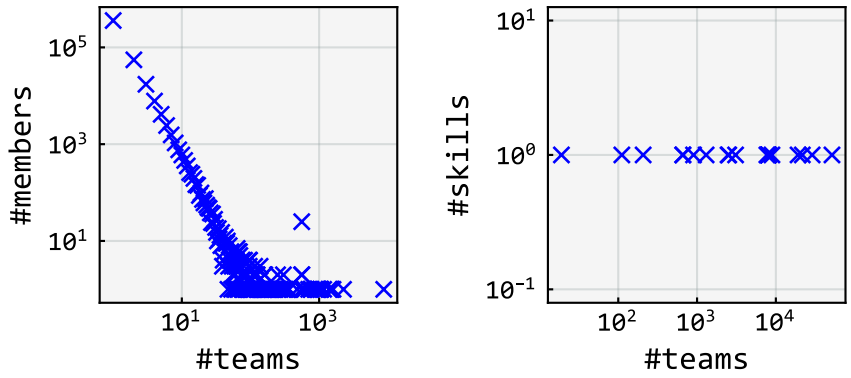

Fig. 5.1.7: Distribution of teams over skills and members in GitHub repositories (`gith`).

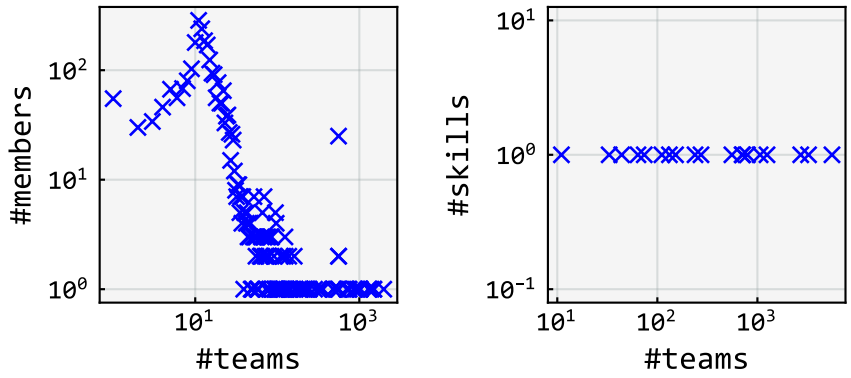

Fig. 5.1.8: Distribution of teams over skills and members in GitHub repositories (`gith`) after filtering.

Table 5.1.1: Statistics of the raw and preprocessed `dblp.v12` dataset: #teams with ≥`75` members, #members with ≥`3` teams.

dblp.v12	**raw**	**preprocessed**
#publications (teams)	4,877,383	99,375
#unique authors (experts)	5,022,955	14,214
#unique field of study (skills)	89,504	29,661
average #author per publication	3.06	3.29
average #fos per publication	8.57	9.71
average #publication per author	2.97	23.02
average #fos per author	16.73	96.72
#publication w/ single author	768,956	0
#publication w/ single fos	5,569	56

Table 5.1.2: Statistics of the raw and preprocessed `imdb` dataset: #teams with ≥`75` members, #members with ≥`3` teams.

imdb	**raw**	**preprocessed**
#movies (teams)	507,034	32,059
#unique castncrew (experts)	876,981	2,011
#unique genre (skills)	28	23
average #castncrew per movie	1.88	3.98
average #genre per movie	1.54	1.76
average #movies per castncrew	1.09	62.45
average #genre per castncrew	1.59	10.85
#movie w/ single castncrew	322,918	0
#movie w/ single genre	315,503	15,180

Table 5.1.3: Statistics of the raw and preprocessed `uspt` dataset: #teams with ≥`75` members, #members with ≥`3` teams.

uspt	**raw**	**preprocessed**
#patents (teams)	7,068,508	152,317
#unique inventors (experts)	3,508,807	12,914
#unique subgroups (skills)	241,961	67,315
average #inventors per patent	2.51	3.79
average #subgroup per patent	6.29	9.97
average #patent per inventor	5.05	44.69
average #subgroup per inventor	19.49	102.53
#patent w/ single inventor	2,578,898	0
#patent w/ single subgroup	939,955	8,110

Table 5.1.4: Statistics of the raw and preprocessed `gith` dataset: #teams with ≥`10` members, #members with ≥`3` teams.

gith	**raw**	**preprocessed**
#repositories (teams)	132,851	11,312
#unique developer (experts)	452,606	2,686
#unique programming language (skills)	20	19
average #developer per repository	5.52	7.53
average #pl per repository	1.37	1.57
average #repository per developer	1.62	31.72
average #pl per developer	2.03	5.18
#repository w/ single developer	0	0
#repository w/ single pl	69,131	6014

5.1.2 Baselines

5.1.2.1 Neural Team Formation with Negative Sampling

Our testbed includes two neural architectures that have previously been used for the team formation problem and one temporal recommender system baseline due to the lack of a temporal neural team formation baseline: *i*) feed-forward non-Bayesian (non-variational) neural network (`fnn`), *ii*) (variational) Bayesian neural network [50] (`bnn`), and iii) recurrent recommender networks [61] (**rrn**), where we recommend experts for input skills. All models include a single hidden layer of size `d=128`, `ReLU` and `sigmoid` are the activation functions for the hidden and the output layers, respectively, and Adam is the optimizer. The input layer of the neural models are either (i) sparse occurrence vector representations for skills of size $|\mathcal{S}|$, (ii) pre-trained dense vector representations (`emb`) for the subsets of skills as suggested by Rad et al. [50], where we consider each team as a document and the skills as the document's words, or (iii) temporal dense skill vector representations (`dt2v`) using temporal word embedding method using Doc2Vec, where we consider each team as a document and the skills and the year as the document's words to directly incorporate temporal evolution of skills into the underlying neural model in addition to our proposed streaming strategy. We used the distributed memory model to generate the real-valued embeddings of the subset of skills with(out) year for both (ii) and (iii) with dimension of `d=100`. The output layer of the model is sparse occurrence vector representations for experts of size $|\mathcal{E}|$. We train the models with a learning rate of `0.1` over `20` epochs including minibatches of size `128`. To evaluate the impact of our proposed objective function (negative sampling) on effectiveness and efficiency of both (non-)Bayesian neural models, we conducted our experiments on them with and without our proposed negative sampling heuristics (`-uniform`, `-unigram`, `-unigram-b`) on `dblp` and `imdb`. To examine the effect of our proposed streaming training strategy and temporal skill embeddings, we used the optimum model based on our experiments regarding the impact of negative sampling on neural models, we conducted our experiments with and without our proposed training strategy and temporal embeddings.

5.1.3 Evaluation Strategy and Metrics

To demonstrate the synergy of negative sampling heuristics in prediction effectiveness, we randomly select `15%` of teams in the datasets for the test set and perform `5`-fold cross-validation on the remaining teams for model training and validation that results in one trained model per each fold. Given a team (s, e) from the test set, we compare the ranked list of experts e', predicted by the model of each fold, with the observed subset of experts e and report the average performance of models on all folds in terms of information retrieval metrics including normalized discounted cumulative gain (`ndcg`), and mean average precision (`map`) at top-`{2,5,10}` as well as classification metrics including precision (`pr`), recall (`rec`), and area under the receiver operating characteristic (`rocauc`) using pytrec_eval[5] and scikit-learn[6]. To evaluate how negative sampling helps with the efficiency of neural models during the training phase while maintaining inference efficacy, we train the baselines on an increasing number of epochs `{1,..,20}` and evaluate them on the test set at each epoch.

To test the impact of the streaming training strategy and incorporation of time information to the input embeddings in the prediction of *future* successful teams, we took the last year of each dataset for the test set. To ensure the effectiveness of our approach, we perform 5-fold cross-validation on the teams in each year for model training and validation. Given a team $(s, e)_{T+1}$ from the test set, we compared the ranked list of predicted experts e' by the model of each fold with the observed subset of experts e and report the average performance of models in all folds by the same metrics mentioned earlier.

[5] github.com/cvangysel/pytrec_eval
[6] scikit-learn.org

5.2 Results

In this section, we explain our findings in response to our research questions. First, we will address **RQ1-3** and then we will explain the results for **RQ4-6**:

5.2.1 Impact of Negative Sampling

In response to **RQ1**, i.e., whether negative sampling improves the effectiveness of neural models, from Table 5.2.1 and 5.2.2, we can observe that **(1)** all negative sampling heuristics improve Bayesian neural baselines on `dblp` and `imdb` in terms of all our evaluation metrics. In comparison, Bayesian baselines with no negative sampling (`bnn` and `bnn-emb`) are the weakest neural models. Specifically, smoothed unigram negative sampling in minibatches (`bnn-unigram-b` and `bnn-emb-unigram-b`) consistently outperforms all other neural baselines on `dblp` and `imdb` in terms of `ndcg` for top-{`2,5,10`} and `rocauc`.

Contrary to Bayesian baselines, non-Bayesian baselines (`fnn-*`) do not show a consistent similar trend across datasets which bring us to our second research question **RQ2**, i.e., whether the impact of negative sampling heuristics is consistent across training data from diverse statistical distributions. From Table 5.2.1, we can see that negative sampling heuristics improve non-Bayesian baselines in `dblp` in terms of all evaluation metrics. However, in `imdb`, we cannot observe a consistent synergistic trend by using negative sampling heuristics. Indeed, non-Bayesian baseline *without* negative samplings (`fnn`) is the strongest baseline in terms of `map`, `ndcg`, precision (`pr`) and recall (`rec`) for top-{`2,5`}.

We attribute the inefficiency of negative sampling heuristics for neural models on `imdb` to the small size of skill set (genres) and uniform distribution of teams (movies) over skills (almost all the skills are fairly adopted by many movies), as seen in Figure 5.1.3. The fact that dense vector representations for skills are not effective for non-Bayesian baselines in `imdb` is further cementing this view. Overall, **(2)** we conclude that the effect of considering unsuccessful teams via negative sampling in non-Bayesian neural models depends on the underlying distribution of teams over

skills in the training set (`dblp` vs. `imdb`). Moreover, **(3)** in our experiments, that Bayesian neural models outperform non-Bayesian ones, reported earlier by Rad et al. on `dblp`, could *not* be generalized to and reproduced on `imdb`.

In response to **RQ3**, i.e., whether negative sampling increases the efficiency of neural models during training while improving inference effectiveness, from Figure 5.2.1 and 5.2.2, we can observe that **(4)** Bayesian neural models that benefit from negative samples outperforms other models in less number of training epochs for sparse and dense vector representation across all datasets in terms of `ndcg10`.[7]

With respect to the non-Bayesian neural models, we can observe similar synergistic effects of negative sampling heuristics on obtaining the *best* inference effectiveness with a fewer training epochs over `dblp`. However, we cannot observer similar trend over `imdb`. In fact, **(5)** non-Bayesian neural models *without* negative sampling (`fnn` and `fnn-emb`) could gradually gain the momentum and achieve the stellar performance at epoch `7` and after over `imdb`. This observation further explains that when teams are *well*-distributed over a limited set of skills (e.g., movies over genres), overly usage of negative samples in many epochs of training decouples the vectors of experts and skills that should have been stayed close for their participation in successful teams, and consequently degrades the inference performance.

[7]Similar trend has been observed for other metrics. Full results are available at our codebase.

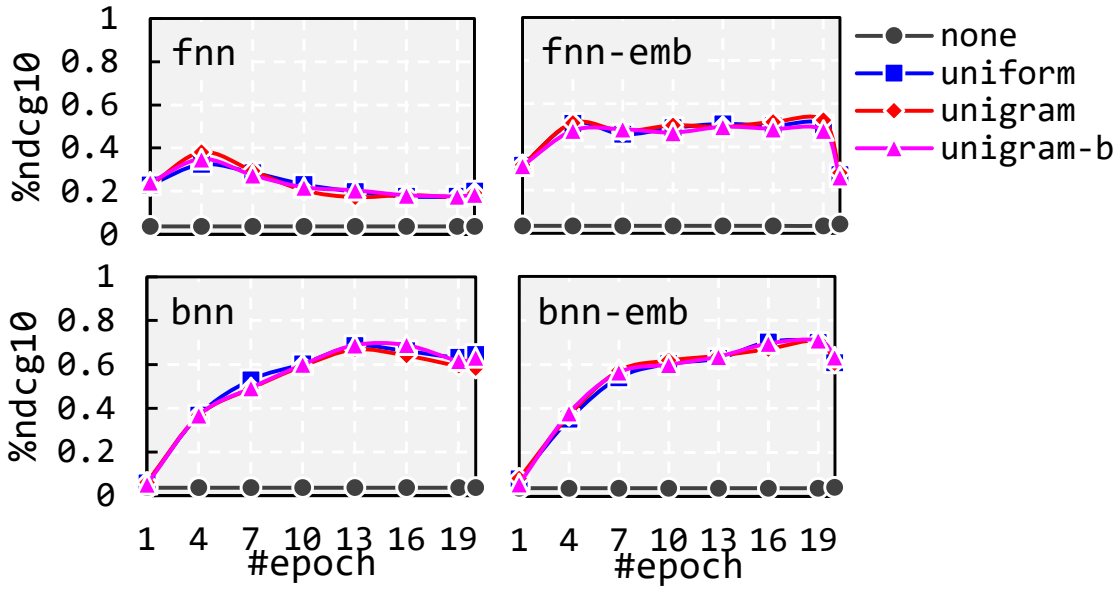

Fig. 5.2.1: Effect of negative sampling on training time vs. inference accuracy on `dblp.v12` dataset.

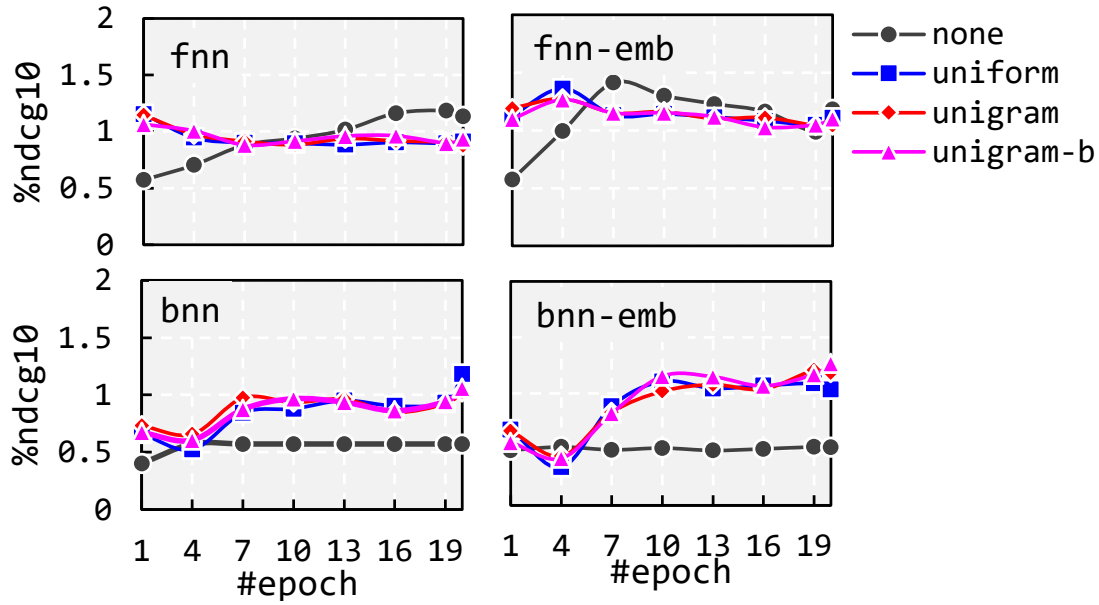

Fig. 5.2.2: Effect of negative sampling on training time vs. inference accuracy on `imdb` dataset.

Table 5.2.1: Average performance of `5-fold` (non-)Bayesian neural models on test set with(out) negative sampling in computer science publicaiton (`dblp`).

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	%aucroc
fnn-nons	0.0067	0.0188	0.0188	0.0045	0.0307	0.0612	0.0082	0.0227	0.0369	0.0045	0.0113	0.0155	50.0000
fnn-uniform	0.1020	0.1030	0.0986	0.0597	0.1522	0.2913	0.1074	0.1350	0.1993	0.0487	0.0741	0.0943	65.1200
fnn-unigram	0.0932	0.0985	0.0971	0.0552	0.1447	0.2854	0.0952	0.1249	0.1907	0.0437	0.0677	0.0880	65.0500
fnn-unigram_b	0.0993	0.0979	0.0932	0.0569	0.1429	0.2702	0.1005	0.1249	0.1846	0.0436	0.0665	0.0847	65.0000
fnn-emb-nons	0.0134	0.0255	0.0215	0.0084	0.0402	0.0688	0.0134	0.0302	0.0428	0.0073	0.0174	0.0209	50.0000
fnn-emb-uniform	0.1543	0.1505	0.1346	0.0870	0.2179	0.3925	0.1537	0.1901	0.2716	0.0668	0.1043	0.1305	63.1300
fnn-emb-unigram	0.1523	0.1500	0.1378	0.0884	0.2194	0.4038	0.1564	0.1942	0.2803	0.0700	0.1084	0.1356	63.3100
fnn-emb-unigram_b	0.1415	0.1374	0.1291	0.0830	0.2011	0.3770	0.1444	0.1782	0.2607	0.0656	0.1015	0.1277	63.2200
bnn-nons	0.0101	0.0161	0.0195	0.0061	0.0254	0.0569	0.0123	0.0204	0.0365	0.0061	0.0107	0.0155	50.0000
bnn-uniform	0.4005	0.3555	0.3102	0.2297	0.5124	0.8974	0.3997	0.4627	0.6434	0.1746	0.2554	0.3110	71.5000
bnn-unigram	0.3401	0.3067	0.2816	0.1971	0.4469	0.8194	0.3488	0.4070	0.5823	0.1539	0.2249	0.2782	71.3200
bnn-unigram_b	0.3904	0.3402	0.3017	0.2256	0.4929	0.8774	0.3983	0.4505	0.6312	0.1757	0.2499	0.3039	71.6800
bnn-emb-nons	0.0168	0.0201	0.0201	0.0112	0.0326	0.0634	0.0175	0.0267	0.0413	0.0089	0.0145	0.0186	50.0000
bnn-emb-uniform	0.3656	0.3405	0.2909	0.2121	0.4934	0.8463	0.3715	0.4426	0.6076	0.1622	0.2397	0.2894	71.2300
bnn-emb-unigram	0.3783	0.3298	0.2858	0.2213	0.4802	0.8313	0.3862	0.4400	0.6049	0.1720	0.2448	0.2966	70.7700
bnn-emb-unigram_b	0.3938	0.3518	0.3033	0.2252	0.5065	0.8775	0.3976	0.4593	0.6340	0.1730	0.2517	0.3062	70.9000

Table 5.2.2: Average performance of `5-fold` (non-)Bayesian neural models on test set with(out) negative sampling in movie (`imdb`).

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	%aucroc
fnn-nons	1.0252	0.7927	0.5556	0.4681	0.9187	1.2931	1.0322	0.9760	1.1331	0.3639	0.5009	0.5663	50.6100
fnn-uniform	0.5760	0.5664	0.5219	0.2634	0.6593	1.2292	0.5802	0.6470	0.9116	0.2038	0.3272	0.4191	59.3000
fnn-unigram	0.5386	0.5190	0.5078	0.2492	0.6054	1.1884	0.5522	0.6035	0.8773	0.1977	0.3085	0.4003	59.2400
fnn-unigram_b	0.5864	0.5598	0.5377	0.2767	0.6551	1.2788	0.5911	0.6448	0.9363	0.2141	0.3329	0.4329	59.4100
fnn-emb-nons	0.7028	0.6970	0.7024	0.3243	0.8082	1.6275	0.7057	0.7898	1.1805	0.2487	0.3784	0.4977	55.8500
fnn-emb-uniform	0.7049	0.6979	0.6467	0.3278	0.8029	1.4972	0.6979	0.7817	1.1083	0.2476	0.3872	0.4961	60.6300
fnn-emb-unigram	0.6654	0.6338	0.6026	0.3110	0.7381	1.3961	0.6758	0.7344	1.0442	0.2433	0.3703	0.4740	60.7500
fnn-emb-unigram_b	0.7507	0.6654	0.6267	0.3443	0.7776	1.4650	0.7559	0.7807	1.1027	0.2678	0.3929	0.4974	60.6400
bnn-nons	0.1560	0.3327	0.4200	0.0842	0.3775	0.9536	0.1301	0.3027	0.5725	0.0490	0.1227	0.2231	50.0000
bnn-uniform	0.8193	0.7261	0.6484	0.3863	0.8431	1.5104	0.8560	0.8735	1.1833	0.3167	0.4441	0.5397	63.0500
bnn-unigram	0.7320	0.6563	0.5673	0.3347	0.7711	1.3280	0.7423	0.7651	1.0246	0.2568	0.3749	0.4550	63.0400
bnn-unigram_b	0.6883	0.6654	0.6001	0.3226	0.7795	1.3931	0.6954	0.7673	1.0580	0.2483	0.3810	0.4670	63.1300
bnn-emb-nons	0.1643	0.3194	0.3531	0.0834	0.3696	0.8102	0.1695	0.3124	0.5180	0.0642	0.1334	0.2024	50.0000
bnn-emb-uniform	0.7424	0.6355	0.5548	0.3544	0.7575	1.3128	0.7739	0.7766	1.0370	0.2855	0.3970	0.4730	62.6200
bnn-emb-unigram	0.7819	0.7569	0.6675	0.3620	0.8767	1.5490	0.7969	0.8727	1.1852	0.2820	0.4243	0.5256	63.1300
bnn-emb-unigram_b	0.8858	0.7802	0.7053	0.3982	0.9002	1.6341	0.9207	0.9272	1.2662	0.3267	0.4692	0.5728	64.7000

5.2.2 Impact of Streaming Training Strategy and Temporal Skills

We should note that we have initiated our experiments for evaluation of our streaming training strategy and temporal skills using Bayesian neural network with *smoothed unigram distribution in each training minibatch* negative sampling heuristic, hence `*bnn*` is actually `*bnn*-unigram_b`. Models starting with `tbnn*` have utilized our streaming scenario training strategy and `tbnn_dt2v_emb` uses temporal skills embeddings as input. Here we have used two more datasets, namely `uspt` and `gith`, compared to our experiments on negative sampling.

In response to **RQ4**, i.e., whether the streaming training strategy improves the predictive power of state-of-the-art neural models, from Tables 5.2.3, 5.2.4, 5.2.5, and 5.2.6, comparing `bnn` and `bnn_emb` with `tbnn` and `tbnn_emb` respectively, we can observe that streaming training strategy increases neural models' relative performance between 10 to 20 percent on `dblp` and `uspt` in terms of the classification metrics (`aucroc`). Even though, we do not see the same performance improvement from our training strategy in terms of `aucroc` on `imdb`, we can see that it increases the performance of neural models in terms of the information retrieval metrics (up to 300 percent). Moreover, our training strategy increases neural models' relative performance on the information retrieval metrics between 100 and 200 percent on dblp and uspt as well.

In response to **RQ5**, i.e., whether adding time explicitly to the input of the neural model improves its performance while utilizing the streaming training strategy, from Tables 5.2.3, 5.2.4, 5.2.5, and 5.2.6, comparing `tbnn_emb` with `tbnn_dt2v_emb`, we can see that the models that utilize temporal skills in the input gain relative performance of between 20 to 50 percent in terms of `map`, `ndcg`, precision (`pr`) and recall (`rec`) for top-{2,5,10} on `dblp`, `uspt`, and `gith` and up to 100 percent on `imdb` on some metrics.

Regarding **RQ6**, i.e., whether the impact of our proposed streaming training strategy is consistent across different datasets with distinct distributions of skills and

Table 5.2.3: Average performance of 5-fold neural models with(out) streaming learning and temporal skills on test set in dblp.

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	aucroc
bnn	0.0570	0.0663	0.0710	0.0351	0.0993	0.2118	0.0538	0.0806	0.1330	0.0242	0.0411	0.0558	0.6352
bnn_emb	0.1124	0.1290	0.1251	0.0668	0.1909	0.3699	0.1083	0.1555	0.2397	0.0474	0.0792	0.1033	0.6681
rrn	0.0570	0.0391	0.0472	0.0380	0.0630	0.1552	0.0478	0.0523	0.0959	0.0217	0.0281	0.0446	0.5073
tbnn	0.1189	0.1413	0.1664	0.0706	0.2090	0.4984	0.1126	0.1689	0.3031	0.0484	0.0845	0.1223	0.7308
tbnn_emb	0.2996	0.2938	0.2811	0.1816	0.4433	0.8431	0.3048	0.3860	0.5721	0.1411	0.2095	0.2635	0.7483
tbnn_dt2v_emb	0.4299	0.3973	0.3612	0.2601	0.5963	1.0801	0.4284	0.5221	0.7465	0.1947	0.2864	0.3520	0.7701

Table 5.2.4: Average performance of 5-fold neural models with(out) streaming learning and temporal skills on test set in imdb.

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	aucroc
bnn	0.7994	0.8164	0.7533	0.3541	0.9167	1.7064	0.8046	0.9022	1.2736	0.2746	0.4386	0.5500	0.6429
bnn_emb	0.4255	0.5106	0.6383	0.2837	0.8511	1.9574	0.3292	0.5923	1.1358	0.1418	0.2813	0.4389	0.5182
rrn	0.0000	0.8511	0.8511	0.0000	1.4184	2.8369	0.0000	0.8163	1.4606	0.0000	0.3191	0.6265	0.5222
tbnn	0.8511	1.5319	1.4043	0.5319	2.4610	4.4965	0.7548	1.7381	2.6829	0.3369	0.8215	1.1674	0.6346
tbnn_emb	0.8511	1.1064	1.0638	0.5674	1.7518	1.3262	0.9474	1.4848	2.2007	0.4965	0.8138	1.0099	0.6687
tbnn_dt2v_emb	1.9149	1.1915	1.4468	1.2411	1.9504	4.5532	1.8667	1.8703	3.0303	0.9043	1.1099	1.4293	0.6656

experts, from Tables 5.2.3, 5.2.4, 5.2.5, and 5.2.6, we can see a positive impact on the model's predictive power in terms of information retrieval metrics for `dblp`, `imdb`, and `uspt`. However, for `gith`, we can see that Bayesian neural networks using our proposed streaming training strategy have a positive impact when using sparse vector representations as input and a small decrease in performance when using embeddings (`*emb*`) generated by Doc2Vec. More specifically, in terms of `aucroc`, the improvement is more significant on `dblp` and `uspt` compared to `imdb` and `gith`.

Finally, from Tables 5.2.3, 5.2.4, 5.2.5, and 5.2.6, we can see that the results of our proposed training strategy and incorporation of temporal skills are superior compared to the temporal recommender system baseline [61] (`rrn`) on all four datasets for all of the classification and information retrieval metrics.

5.2.3 Hyperparameter Study

We also conducted a hyperparameter study on the optimum neural model based on our experiments in Section 5.2.1. To this end, we used the `bnn_emb-unigram_b`

Table 5.2.5: Average performance of 5-fold neural models with(out) streaming learning and temporal skills on test set in uspt.

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	aucroc
bnn	0.0657	0.0769	0.0910	0.0353	0.0976	0.2212	0.0655	0.0883	0.1481	0.0266	0.0433	0.0592	0.6454
bnn_emb	0.3663	0.4123	0.3748	0.1608	0.4509	0.8141	0.3652	0.4531	0.6094	0.1212	0.2027	0.2583	0.6985
rrn	0.0239	0.0383	0.0654	0.0140	0.0500	0.1370	0.0221	0.0408	0.0868	0.0096	0.0186	0.0340	0.5160
tbnn	0.1843	0.1841	0.2029	0.0933	0.2321	0.5158	0.1794	0.2152	0.3481	0.0681	0.1056	0.1429	0.7544
tbnn_emb	0.8272	0.7539	0.7042	0.3970	0.9021	1.6933	0.8457	0.9057	1.2657	0.3104	0.4533	0.5679	0.8359
tbnn_dt2v_emb	1.2268	1.0583	0.9324	0.6037	1.2928	2.2518	1.2322	1.2960	1.7348	0.4626	0.6659	0.8118	0.8534

Table 5.2.6: Average performance of 5-fold neural models with(out) streaming learning and temporal skills on test set in gith.

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	aucroc
bnn	3.0693	2.8515	2.6931	1.2164	2.8846	5.1174	3.1365	3.2893	4.2340	1.0104	1.5706	2.1633	0.5618
bnn_emb	7.3267	4.7129	3.3861	3.5441	5.1580	6.1885	6.4753	5.8418	6.2665	2.3424	3.0822	3.3837	0.6265
rrn	0.0000	0.1980	0.0990	0.0000	0.0619	0.0619	0.0000	0.1679	0.1090	0.0000	0.0206	0.0206	0.5226
tbnn	3.8614	2.8515	2.3564	1.8801	3.1525	4.5754	4.3319	3.9721	4.5031	1.8025	2.3978	2.8768	0.5665
tbnn_emb	4.9505	3.5248	3.1287	1.9434	3.0770	4.3718	5.0849	4.4715	4.9844	1.6957	2.1431	2.5949	0.6220
tbnn_dt2v_emb	5.7426	4.5941	3.8020	2.1874	3.8474	4.7855	5.6081	5.3287	5.6670	1.7131	2.4258	2.7858	0.6489

with a single hidden layer of size `d=128` as the baseline, with `ReLU` and `sigmoid` as the activation functions for the hidden and the output layers, respectively, and Adam as the optimizer with learning rate of 0.1 with 20 epochs. We conducted our experiments on `dblp` and `imdb` datasets only because `uspt` and `gith` datasets have similar distribution of teams over experts and skills to `dblp` and `imdb` datasets, respectively, and models perform similarly on them.

5.2.3.1 Model Size

We experimented with size of the model first, i.e., we increased the number of layers and the number of nodes in the model. To this end, we tried models with layers/#nodes of [128,64,128], i.e., first hidden layer has 128 nodes, second hidden layer has 64 nodes, and the third hidden layer has 128 nodes, and with layers/#nodes of [256,128,64,128,64]. As can be seen, in Tables 5.2.7, 5.2.8, we can see increasing the size of the model decreases its performance on both datasets.

Table 5.2.7: Average performance of 5-fold neural models with different model sizes on test set in dblp.

Layers/Nodes	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	%aucroc
[128]	0.1124	0.1290	0.1251	0.0668	0.1909	0.3699	0.1083	0.1555	0.2397	0.0474	0.0792	0.1033	0.6681
[128,64,128]	0.0309	0.0373	0.0325	0.0190	0.0599	0.1034	0.0312	0.0482	0.0679	0.0145	0.0269	0.0337	0.5015
[256,128,64,128,256]	0.0456	0.0502	0.0482	0.0288	0.0802	0.1564	0.0485	0.0672	0.1017	0.0236	0.0364	0.0495	0.5003

Table 5.2.8: Average performance of 5-fold neural models with different model sizes on test set in imdb.

Layers/Nodes	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	%aucroc
[128]	0.4255	0.5106	0.6383	0.2837	0.8511	1.9574	0.3292	0.5923	1.1358	0.1418	0.2813	0.4389	0.5182
[128,64,128]	0.1890	0.2292	0.1852	0.0887	0.2825	0.4587	0.1772	0.0459	0.3233	0.0593	0.1113	0.1335	0.5098
[256,128,64,128,256]	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.4888

5.2.3.2 #Bayesian Samples

Second, we conducted experiments on the number of samples for Bayesian neural networks. Bayesian neural networks learn a distribution for parameters through training instead of learning a single parameter for each edge between two nodes like feed-forward neural networks. To do this, they learn a mean and a standard deviation for each edge, and for each train/validation/test instance, they sample a set of parameters based on the learned distribution of parameters and use that for inference. For instance, if we set the number of samples to 3, it means that the model samples the parameters 3 times and infers 3 probabilities for each expert, we then take the average probability of each expert and use that for evaluation. As can be seen in Tables 5.2.9 and 5.2.10, the impact of #bs depends on the distribution of the dataset. On `dblp`, increasing the number of Bayesian samples to $bs = \{3\}$ have a positive impact of the performance of the model, but with higher number of Bayesian sampling, the model's performance start to diminish, but on `imdb`, even increasing the number of Bayesian samples up to $bs = \{3\}$ will degrade the model's performance.

5.2.3.3 Embedding Size

Third, we aim to see if increasing the size of real-valued learned embeddings using Doc2Vec for the input layer will help improve model's performance. To this end,

Table 5.2.9: Average performance of 5-fold neural models with different number of Bayesian samples on test set in dblp.

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	%aucroc
#bs = 1	0.1124	0.1290	0.1251	0.0668	0.1909	0.3699	0.1083	0.1555	0.2397	0.0474	0.0792	0.1033	0.6681
#bs = 3	0.2003	0.1856	0.1814	0.1185	0.2737	0.5403	0.2058	0.2473	0.3706	0.0924	0.1346	0.1702	0.6656
#bs = 5	0.1352	0.1088	0.1055	0.0780	0.1597	0.3150	0.1318	0.1451	0.2164	0.0565	0.0782	0.0988	0.6078
#bs = 10	0.1205	0.1120	0.1104	0.0726	0.1681	0.3296	0.1168	0.1441	0.2188	0.0532	0.0780	0.0996	0.5948

Table 5.2.10: Average performance of 5-fold neural models with different number of Bayesian samples on test set in imdb.

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	%aucroc
#bs = 1	0.4255	0.5106	0.6383	0.2837	0.8511	1.9574	0.3292	0.5923	1.1358	0.1418	0.2813	0.4389	0.5182
#bs = 3	0.2128	0.2553	0.3830	0.1418	0.4255	1.0496	0.2609	0.3768	0.6884	0.1418	0.2175	0.3080	0.5264
#bs = 5	0.0000	0.0851	0.2979	0.0000	0.0851	0.8794	0.0000	0.0622	0.4064	0.0000	0.0213	0.1095	0.5256
#bs = 10	0.2128	0.1702	0.4255	0.1418	0.2270	1.2482	0.2609	0.2555	0.7249	0.1418	0.1589	0.2982	0.5371

we experiment with input embedding sizes of $d = \{100, 200, 300, 500\}$. Note that $d = \{100\}$ is the baseline. As can be seen from Tables 5.2.11 and 5.2.12, increasing the dimension of Doc2Vec embeddings for input skills, will generally result in a lower performance in terms of both information retrieval and classification metrics.

5.2.3.4 #Negative Samples

Finally, we want to examine the impact of increasing the number of negative samples on the performance of Bayesian neural networks. The default number of negative samples (nns) is 3, which is the minimum size of teams after filtering. We have conducted experiments on $nns = \{5, 10, 20, 50\}$ and have observed that increasing nns up to 20 can lead to a higher performance in terms of all metrics, however, $nns = \{50\}$ resulted in poorer performance compared to $nns = \{20\}$, which shows that if we

Table 5.2.11: Average performance of 5-fold neural models with different input embedding sizes on test set in dblp.

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	%aucroc
d = 100	0.1124	0.1290	0.1251	0.0668	0.1909	0.3699	0.1083	0.1555	0.2397	0.0474	0.0792	0.1033	0.6681
d = 200	0.1352	0.1094	0.1120	0.0792	0.1612	0.3308	0.1377	0.1499	0.2284	0.0610	0.0824	0.1045	0.6588
d = 300	0.1205	0.0997	0.0964	0.0727	0.1499	0.2893	0.1168	0.1332	0.1978	0.0517	0.0720	0.0904	0.6520
d = 500	0.0651	0.0840	0.0918	0.0400	0.1289	0.2807	0.0666	0.1044	0.1748	0.0306	0.0537	0.0737	0.6465

Table 5.2.12: Average performance of 5-fold neural models with different input embedding sizes on test set in imdb.

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	%aucroc
d = 100	0.4255	0.5106	0.6383	0.2837	0.8511	1.9574	0.3292	0.5923	1.1358	0.1418	0.2813	0.4389	0.5182
d = 200	0.0000	0.3404	0.3404	0.0000	0.5106	0.9645	0.0000	0.3214	0.5585	0.0000	0.1277	0.1986	0.5087
d = 300	0.4255	0.3404	0.3830	0.2270	0.4539	1.1631	0.4255	0.4097	0.7318	0.1560	0.2057	0.2998	0.5086
d = 500	0.0000	0.1702	0.2979	0.0000	0.1915	0.8440	0.0000	0.1389	0.4569	0.0000	0.0525	0.1484	0.5075

Table 5.2.13: Average performance of 5-fold neural models with different number of negative samples on test set in dblp.

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	%aucroc
#nns = 3	0.1124	0.1290	0.1251	0.0668	0.1909	0.3699	0.1083	0.1555	0.2397	0.0474	0.0792	0.1033	0.6681
#nns = 5	0.1677	0.1557	0.1593	0.0994	0.2313	0.4718	0.1629	0.2010	0.3145	0.0719	0.1067	0.1389	0.6755
#nns = 10	0.2215	0.2247	0.2000	0.1318	0.3379	0.6009	0.2185	0.2858	0.4075	0.0972	0.1506	0.1852	0.6789
#nns = 20	0.2638	0.2495	0.2189	0.1556	0.3716	0.6547	0.2704	0.3301	0.4608	0.1209	0.1783	0.2163	0.6793
#nns = 50	0.2003	0.1915	0.1788	0.1207	0.2865	0.5364	0.2036	0.2513	0.3672	0.0922	0.1364	0.1692	0.6748

use too many negative samples for each training instance, instead of giving complimentary signals to the model, we are confounding it, i.e., the model will take vector representations of negative sampled experts farther from possibly relevant skills. You can see the results of our experiments on `dblp` and `imdb` in Tables 5.2.13 and 5.2.14.

Table 5.2.14: Average performance of 5-fold neural models with different number of negative samples on test set in imdb.

	%pr 2	%pr 5	%pr10	%rec2	%rec5	%rec10	%ndcg2	%ndcg5	%ndcg10	%map2	%map5	%map10	%aucroc
#nns = 3	0.4255	0.5106	0.6383	0.2837	0.8511	1.9574	0.3292	0.5923	1.1358	0.1418	0.2813	0.4389	0.5182
#nns = 5	0.0000	0.0851	0.1277	0.0000	0.1418	0.3688	0.0000	0.0860	0.2006	0.0000	0.0355	0.0686	0.5238
#nns = 10	0.6383	0.3404	0.3404	0.3688	0.5106	1.0213	0.5902	0.4940	0.7454	0.2553	0.2837	0.3593	0.5195
#nns = 20	0.6383	0.6809	0.5106	0.4255	0.9645	1.4184	0.5902	0.8515	1.0719	0.2837	0.4520	0.5331	0.5408
#nns = 50	0.2128	0.1702	0.5532	0.1418	0.2837	1.6738	0.1646	0.2032	0.8637	0.0709	0.0993	0.3105	0.5470

CHAPTER 6

Conclusion and Future Work

In this paper, we proposed *i*) a streaming training strategy for neural models to learn the evolution of experts' skills and collaborative ties to predict *future* successful teams, *ii*) an objective function that utilizes negative sampling heuristics to bring embeddings of experts and skills that have been part of successful collaborations closer to each other while taking embeddings of experts and skills that have not been part of previous successful collaborations farther from each other. We produce negative samples using three negative sampling heuristics based on the closed-world assumption where we consider no currently known team of experts for the required skills as an unsuccessful team. We performed extensive experiments on four large-scale datasets with distinct distributions of skills and experts over teams within time. Our experiments show that (1) our proposed streaming training strategy improves the predictive power of neural models for *future* successful teams, (2) neural models that leverage temporal information in the input obtain better performance compared to lack thereof, (3) neural models utilizing our proposed training strategy with or without incorporation of temporal information outperform the temporal recommender system baseline, (4) negative sampling improves the effectiveness of Bayesian neural models for the task of team formation, (5) depending on the distribution of teams over skills, while improving the performance of non-Bayesian neural baselines in datasets with a large variety of skills (e.g., `dblp`), negative sampling may discount the efficacy of neural models in datasets with limited skill set (e.g., `imdb`), and (6) negative sampling helps with efficiency during training while improving inference effectiveness for Bayesian neural models. In regard to non-Bayesian neural models, the contribution

of negative sampling heuristics depends on the statistical distribution of skills in the underlying dataset. For future work, we aim to compare our models with other temporal recommender system baselines and identify real unsuccessful teams, e.g., based on their sleeping time in arXiv[1] for research publications or budget-box office ratio for movies.

[1] arxiv.org

REFERENCES

[dbl] Dblp dataset. `https://aminer.org/citation`. Accessed: 2022-05-14.

[imd] Imdb dataset. `https://www.imdb.com/interfaces/`. Accessed: 2022-05-14.

[usp] Uspt dataset. `https://patentsview.org/download/data-download-tables`. Accessed: 2022-05-14.

[4] Ahrabian, K., Feizi, A., Salehi, Y., Hamilton, W. L., and Bose, A. J. (2020). Structure aware negative sampling in knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6093–6101, Online. Association for Computational Linguistics.

[5] Akiba, T., Iwata, Y., and Yoshida, Y. (2013). Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In Ross, K. A., Srivastava, D., and Papadias, D., editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 349–360. ACM.

[6] An, A., Kargar, M., and Zihayat, M. (2013). Finding affordable and collaborative teams from a network of experts. In *SIAM 2013*, pages 587–595. SIAM.

[7] Arkin, E. and Hassin, R. (2000). Minimum diameter covering problems. *Networks*, 36:147–155.

[8] Barnabò, G., Fazzone, A., Leonardi, S., and Schwiegelshohn, C. (2019). Algorithms for fair team formation in online labour marketplaces10033. In Amer-Yahia, S., Mahdian, M., Goel, A., Houben, G., Lerman, K., McAuley, J. J., Baeza-Yates,

R., and Zia, L., editors, *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 484–490. ACM.

[9] Baykasoglu, A., Dereli, T., and Das, S. (2007). Project team selection using fuzzy optimization approach. *Cybernetics and Systems: An International Journal*, 38(2):155–185.

[10] Bernabé, R. B., Navia, I. A., and García-Peñalvo, F. J. (2015). Faat: Freelance as a team. In *Proceedings of the 3rd International Conference on Technological Ecosystems for Enhancing Multiculturality*, TEEM '15, page 687–694, New York, NY, USA. Association for Computing Machinery.

[11] Blackwell, G. W. (1954). Multidisciplinary team research. *Soc. F.*, 33:367.

[12] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France. PMLR.

[13] Bursic, K. (1992). Strategies and benefits of the successful use of teams in manufacturing organizations. *IEEE Transactions on Engineering Management*, 39(3):277–289.

[14] Campion, M. A., Medsker, G. J., and Higgs, A. C. (1993). Relations between work group characteristics and effectiveness: Implications for designing effective work groups. *Personnel psychology*, 46(4):823–847.

[15] Chen, S. G. and Lin, L. (2004). Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. *IEEE Trans. Engineering Management*, 51(2):111–124.

[16] Craig, M. and McKeown, D. (2015). How to build effective teams in healthcare. *Nursing times*, 111(14):16—18.

[17] Daghaghi, S., Medini, T., Meisburger, N., Chen, B., Zhao, M., and Shrivastava, A. (2021). A tale of two efficient and informative negative sampling distributions. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2319–2329. PMLR.

[18] Dashti, A., Samet, S., and Fani, H. (2022). Effective neural team formation via negative samples. In *Proceedings of the 31st ACM International Conference on Information amp; Knowledge Management*, CIKM '22, page 3908–3912, New York, NY, USA. Association for Computing Machinery.

[19] Durfee, E. H., Boerkoel, J. C., and Sleight, J. (2014). Using hybrid scheduling for the semi-autonomous formation of expert teams. *Future Generation Computer Systems*, 31:200–212. Special Section: Advances in Computer Supported Collaboration: Systems and Technologies.

[20] Fani, H., Bagheri, E., and Du, W. (2020). Temporal latent space modeling for community prediction. pages 745–759, Cham. Springer International Publishing.

[21] Fathian, M., Saei-Shahi, M., and Makui, A. (2017). A new optimization model for reliable team formation problem considering experts' collaboration network. *IEEE Trans. Engineering Management*, 64(4):586–593.

[22] Fitzpatrick, E. and Askin, R. G. (2005). Forming effective worker teams with multi-functional skill requirements. *Comput. Ind. Eng.*, 48(3):593–608.

[23] Goel, R., Kazemi, S. M., Brubaker, M., and Poupart, P. (2020). Diachronic embedding for temporal knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3988–3995.

[24] Hamilton, W. L., Leskovec, J., and Jurafsky, D. (2016). Diachronic word embeddings reveal statistical laws of semantic change. In *ACL 2016*.

[25] Haque, B., Pawar, K. S., and Barson, R. J. (2000). Analysing organisational

issues in concurrent new product development. *International Journal of Production Economics*, 67(2):169–182.

[26] Horowitz, D. and Kamvar, S. D. (2010). The anatomy of a large-scale social search engine. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 431–440, New York, NY, USA. Association for Computing Machinery.

[27] Horowitz, D. and Kamvar, S. D. (2012). Searching the village: Models and methods for social search. *Commun. ACM*, 55(4):111–118.

[28] Kargar, M. and An, A. (2011). Discovering top-k teams of experts with/without a leader in social networks. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 985–994.

[29] Kargar, M. and An, A. (2012). Efficient top-k keyword search in graphs with polynomial delay. In *2012 IEEE 28th International Conference on Data Engineering*, pages 1269–1272.

[30] Kargar, M., Zihayat, M., and An, A. (2013). Finding affordable and collaborative teams from a network of experts. In *Proceedings of the 2013 SIAM international conference on data mining*, pages 587–595. SIAM.

[31] Karp, R. M. (1972). *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA.

[32] Khan, A., Golab, L., Kargar, M., Szlichta, J., and Zihayat, M. (2020). Compact group discovery in attributed graphs and social networks. *Inf. Process. Manag.*, 57(2):102054.

[33] Kou, Y., Shen, D., Snell, Q., Li, D., Nie, T., Yu, G., and Ma, S. (2020). Efficient team formation in social networks based on constrained pattern graph. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, pages 889–900. IEEE.

[34] Kunegis, J., Preusse, J., and Schwagereit, F. (2013). What is the added value of negative links in online social networks? In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 727–736, New York, NY, USA. Association for Computing Machinery.

[35] Lappas, T., Liu, K., and Terzi, E. (2009). Finding a team of experts in social networks. In IV, J. F. E., Fogelman-Soulié, F., Flach, P. A., and Zaki, M. J., editors, *SIGKDD 2009*, pages 467–476. ACM.

[36] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1188–II–1196. JMLR.org.

[37] Leskovec, J., Huttenlocher, D., and Kleinberg, J. (2010). Predicting positive and negative links in online social networks. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 641–650, New York, NY, USA. Association for Computing Machinery.

[38] Li, C., Shan, M., and Lin, S. (2015). On team formation with expertise query in collaborative social networks. *Knowl. Inf. Syst.*, 42(2):441–463.

[39] Liao, S., Liang, S., Meng, Z., and Zhang, Q. (2021). Learning dynamic embeddings for temporal knowledge graphs. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 535–543, New York, NY, USA. Association for Computing Machinery.

[40] Macdonald, W. R. et al. (1986). Characteristics of interdisciplinary research teams. *Interdisciplinary Analysis and Research*, pages 395–406.

[41] Magill-Evans, J., Hodge, M., and Darrah, J. (2002). Establishing a transdisciplinary research team in academia [electronic version]. *Journal of allied health*, 31:222–6.

[42] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

[43] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

[44] Neshati, M., Beigy, H., and Hiemstra, D. (2014). Expert group formation using facility location analysis. *Inf. Process. Manag.*, 50(2):361–383.

[45] Pawar, K. S. and Sharifi, S. (1997). Physical or virtual team collocation: Does it matter? *International Journal of Production Economics*, 52(3):283–290.

[46] Petrie, H. G. (1976). Do you see what i see? the epistemology of interdisciplinary inquiry. *Educational Researcher*, 5(2):9–15.

[47] Prasad, B. (1998). Decentralized cooperation: a distributed approach to team design in a concurrent engineering organization. *Team Performance Management: An International Journal*.

[48] Prince, A., Brannick, M. T., Prince, C., and Salas, E. (1992). Team process measurement and implications for training. In *Proceedings of the Human Factors Society Annual Meeting*, volume 36, pages 1351–1355. SAGE Publications Sage CA: Los Angeles, CA.

[49] Qin, P., Xu, W., and Guo, J. (2016). A novel negative sampling based on tfidf for learning word representation. *Neurocomput.*, 177(C):257–265.

[50] Rad, R. H., Fani, H., Kargar, M., Szlichta, J., and Bagheri, E. (2020). Learning to form skill-based teams of experts. In *CIKM '20*, pages 2049–2052. ACM.

[51] Rahman, H., Roy, S. B., Thirumuruganathan, S., Amer-Yahia, S., and Das, G. (2019). Optimized group formation for solving collaborative tasks. *VLDB J.*, 28(1):1–23.

[52] Rahmanniyay, F., Yu, A. J., and Seif, J. (2019). A multi-objective multi-stage stochastic model for project team formation under uncertainty in time requirements. *Computers Industrial Engineering*, 132:153–165.

[53] Rendle, S. and Freudenthaler, C. (2014). Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, page 273–282, New York, NY, USA. Association for Computing Machinery.

[54] Sapienza, A., Goyal, P., and Ferrara, E. (2019). Deep neural networks for optimal team composition. *Frontiers Big Data*, 2:14.

[55] Sherer, P. D. (1995). Leveraging human assets in law firms: Human capital structures and organizational capabilities. *ILR Review*, 48(4):671–691.

[56] Swaab, R. I., Schaerer, M., Anicich, E. M., Ronay, R., and Galinsky, A. D. (2014). The too-much-talent effect: Team interdependence determines when more talent is too much or not enough. *Psychological Science*, 25(8):1581–1591.

[57] Tang, J., Chang, Y., Aggarwal, C., and Liu, H. (2016). A survey of signed network mining in social media. *ACM Comput. Surv.*, 49(3).

[58] Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. (2008). Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, pages 990–998.

[59] Taylor, J. B. et al. (1986). Building an interdisciplinary team. *Interdisciplinary Analysis and Research*, pages 141–154.

[60] Wi, H., Oh, S., Mun, J., and Jung, M. (2009). A team formation model based on knowledge and collaboration. *Expert Systems with Applications*, 36(5):9121–9134.

[61] Wu, C.-Y., Ahmed, A., Beutel, A., Smola, A. J., and Jing, H. (2017). Recurrent recommender networks. WSDM '17, page 495–503, New York, NY, USA. Association for Computing Machinery.

[62] Yang, D.-N., Chen, Y.-L., Lee, W.-C., and Chen, M.-S. (2011). On social-temporal group query with acquaintance constraint. *Proc. VLDB Endow.*, 4(6):397–408.

[63] Younglove-Webb, J., Gray, B., Abdalla, C. W., and Thurow, A. P. (1999). The dynamics of multidisciplinary research teams in academia. *The Review of Higher Education*, 22(4):425–440.

[64] Zhang, W., Chen, T., Wang, J., and Yu, Y. (2013). Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, page 785–788, New York, NY, USA. Association for Computing Machinery.

[65] Zihayat, M., An, A., Golab, L., Kargar, M., and Szlichta, J. (2017). Authority-based team discovery in social networks. In Markl, V., Orlando, S., Mitschang, B., Andritsos, P., Sattler, K., and Breß, S., editors, *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*, pages 498–501. OpenProceedings.org.

[66] Zzkarian, A. and Kusiak, A. (1999). Forming teams: an analytical approach. *IIE transactions*, 31(1):85–97.

VITA AUCTORIS

NAME:	Seyed Sobhan Vagheh Dashti
PLACE OF BIRTH:	Rasht, Guilan, Iran
YEAR OF BIRTH:	1997
EDUCATION:	University of Tehran, B.Sc. in Computer Science, Tehran, Tehran, 2015-2020
	University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2021-2023