Information Processing and Management 60 (2023) 103144

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

A deep learning-based expert finding method to retrieve agile software teams from CQAs

Peyman Rostami [a], Azadeh Shakery [a,b,*]

[a] School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran
[b] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

ARTICLE INFO

Dataset link: https://doi.org/10.1016/j.eswa.2018.10.015

Keywords:
Information retrieval
Expert finding
Deep learning
Agile team formation
T-shaped expert
Community question answering

ABSTRACT

Currently, many software companies are looking to assemble a team of experts who can collaboratively carry out an assigned project in an agile manner. The most ideal members for an agile team are T-shaped experts, who not only have expertise in one skill-area but also have general knowledge in a number of related skill-areas. Existing related methods have only used some heuristic non-machine learning models to form an agile team from candidates, while machine learning has been successful in similar tasks. In addition, they have only used the number of candidates' documents in various skill-areas as a resource to estimate the candidates' T-shaped knowledge to work in an agile team, while the content of their documents is also very important. To this end, we propose a multi-step method that rectifies the drawbacks mentioned. In this method, we first pick out the best possible candidates using a state-of-the-art model, then we re-estimate their relevant knowledge for working in the team with the help of a deep learning model, which uses the content of the candidates' posts on StackOverflow. Finally, we select the best possible members for the given agile team from among these candidates using an integer linear programming model. We perform our experiments on two large datasets C# and Java, which comprise 2,217,366 and 2,320,883 posts from StackOverflow, respectively. On datasets C# and Java, our method selects, respectively, 68.6% and 55.2% of the agile team members from among T-shaped experts, while the best baseline method only selects, respectively, 49.1% and 40.2% of the agile team members from among T-shaped experts. In addition, the results show that our method outperforms the best baseline method by 8.1% and 11.4% in terms of F-measure on datasets C# and Java, respectively.

1. Introduction

Expert finding is a challenging task that has attracted the attention of many researchers in the field of *Information Retrieval*. It generally seeks to generate a ranking of candidates who have expertise in a given query (Balog et al., 2012). This task has many applications, such as reviewer assignment (Mimno & McCallum, 2007), supervisor finding (Alarfaj et al., 2012), doctor recommendation (Hu et al., 2020), and question routing (Zhang et al., 2020), but one of its most important applications is *job routing* (Liang, 2019; Nobari et al., 2020). In this issue, in general, a ranking of related candidates who are experts in one or more skill-areas required for a given job is generated and then the job is routed to one or more of the top candidates.

Job routing using expert finding methods can greatly affect the recruitment of specialized people because so many companies are seeking reliable experts to improve the quality of their projects. However, traditional expert finding methods, which only seek

* Corresponding author at: School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran.
E-mail addresses: pe.rostami@ut.ac.ir (P. Rostami), shakery@ut.ac.ir (A. Shakery).

https://doi.org/10.1016/j.ipm.2022.103144
Received 13 July 2022; Received in revised form 8 October 2022; Accepted 27 October 2022

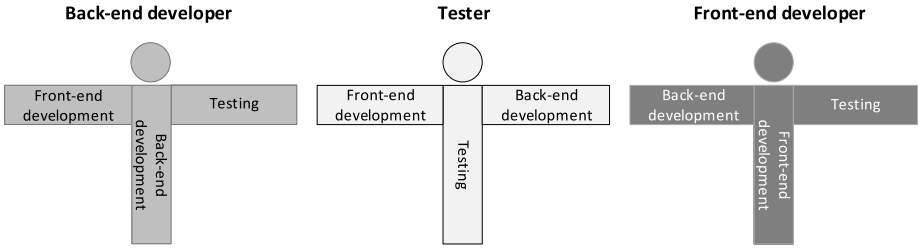

Fig. 1. A toy example of an agile team with three T-shaped experts.

to retrieve experts, are not able to properly meet the needs of many software companies, because these companies usually need experts who can work in a team to carry out an assigned project according to a given software development methodology.

One of the most popular software development methodologies used by many companies is *agile methodology*. In agile teams, i.e., teams that work on an assigned project in an agile manner, because of the flat structure that should exist between team members (Danait, 2005), the most ideal members are the ones who have T-shaped expertise, or in other words, are generalizing specialists (Ambler, 2012). As the horizontal part of the letter *T* indicates, the breadth of knowledge of these people is vast, so they are familiar enough with several skill-areas. Also, as the vertical part of the letter *T* indicates, they have deepened their knowledge in one of their skill-areas. Therefore, given a set of skill-areas required to carry out a project, if we form an ideal agile team with a group of T-shaped experts, each of whom possesses expertise in one distinct skill-area required for the team and possesses general knowledge in the other required skill-areas, then this team will have three important characteristics (Rostami & Neshati, 2019):

- **High coverage**: All the required skill-areas are covered by people who specialize in those skill-areas.
- **Strong communication**: Since each team member has at least general knowledge in all the skill-areas required for the team, he/she is familiar enough with the main skill-areas of the other team members and can therefore communicate effectively with them to advance the project.
- **Cost-effective**: Forming a team, each of whom has expertise in only one skill-area due to his/her T-shaped expertise, is more cost-effective than forming a team whose members specialize in more skill-areas (i.e., the members are Comb-shaped experts).[1]

Fig. 1 shows a toy example of an agile team with three T-shaped experts. The first one is a Back-end developer who has general knowledge in Front-end development and Testing. The second one is a Tester who has general knowledge in Front-end and Back-end development. Finally, the third one is a Front-end developer who has general knowledge in Back-end development and Testing. This team is an ideal agile team whose members cover all the required skill-areas in parallel and meanwhile interact effectively with each other.

One of the most well-known environments for retrieving T-shaped experts is StackOverflow,[2] which contains users' questions and answers in various fields of software engineering. Since users can demonstrate their knowledge in the fields of software engineering by answering questions, this site can be a valuable environment for finding experts and especially T-shaped experts.

In recent years, several studies have been conducted on T-shaped expert finding in StackOverflow (Dehghan et al., 2019, 2020; Gharebagh et al., 2018). In these studies, the problem was to generate a ranking of T-shaped candidates who have expert knowledge in a given skill-area. More precisely, for the sake of simplicity, they paid attention only to T-shaped candidates' expertise and ignored their general knowledge. Also, they only ranked T-shaped experts and did not get into the problem of forming an agile team with T-shaped experts. In contrast, Rostami and Neshati (2019) addressed the problem of agile team formation by retrieving the best group of T-shaped experts who can work together. To solve the problem, they presented two heuristic models that take a greedy approach to select the best suitable candidates for a given agile team. The first model called XEBM estimates candidates' required knowledge using the entropy resulting from the distribution of the number of candidates' documents in various skill-areas. On the other hand, the second model called RDM estimates candidates' required knowledge using the relative deepness of their knowledge, which is obtained from the distance between the number of their documents in different skill-areas. In these models, only the number of documents is used as a resource for estimating the knowledge, while the number of documents alone is not a sufficient resource to show the knowledge. In addition to the number of candidates' documents, the content of their documents is also very important, but in these models, no attention has been paid to the content. The second drawback is that they have not tried to use any machine learning algorithm to retrieve more relevant candidates, while machine learning has been successful in similar tasks such as T-shaped expert ranking (Dehghan et al., 2019). As a result, although the problem addressed in this study does not have the strong simplification assumptions of Dehghan et al. (2019, 2020), Gharebagh et al. (2018), the method of solving it has some main drawbacks.

In this paper, we intend to form better agile teams by rectifying the drawbacks of the previous research. In our problem, the input is a number of skill-areas required for an agile team, and the ideal output is a team of T-shaped experts, each of whom has

[1] Instead of forming an agile team with a group of T-shaped experts, if we use only a Comb-shaped person who specializes in all the skill-areas required for the team, then the ability to advance the project in parallel is lost and consequently, the progress of the project is greatly slowed down.

[2] https://stackoverflow.com

expert knowledge in one distinct skill-area required for the team as well as general knowledge in all the other required skill-areas. In other words, each position of an ideal agile team should be filled by a T-shaped candidate who is an expert in the main skill-area related to the position and has general knowledge in the minor skill-areas related to the position.

To solve the problem, first, we pick out the top candidates for each position of the given agile team using state-of-the-art model RDM, which was introduced before. Then, with the help of a proposed deep learning model called DLM, we re-estimate the candidates' relevant knowledge to fill each position. Finally, we select the best team members from among these candidates using an integer linear programming model in such a way that each position is filled by the best possible candidate.

It should be noted that DLM is a deep learning-based method that estimates a candidate's knowledge to fill a given agile team position. More precisely, it estimates in parallel a candidate's shape of expertise, his/her expertise in the main skill-area of the given position, and his/her general knowledge in the other required skill-areas by taking into account changing the distribution of topics of documents associated with the candidate and the required skill-areas over time. Since candidates' profile and the level of knowledge required to gain expert/general knowledge in skill-areas are constantly changing over time, using the concept of time can intuitively lead to a more accurate estimate of the knowledge required to fill a position.

To evaluate the performance of our method in comparison with baseline methods, we use two large datasets C# and Java extracted from StackOverflow (Rostami & Neshati, 2019). We show that we can improve the performance of the best baseline method on datasets C# and Java by up to 8.1% and 11.4% in terms of F-measure, respectively.

The major contributions of our paper are as follows:

1. Using the content of candidates' posts on StackOverflow to estimate their knowledge to work in an agile team.
2. Proposing a deep learning model that estimates the probability that a candidate is relevant for a given position of an agile team.
3. Presenting an integer linear programming model to select the best possible members for an agile team from among the top candidates.

The rest of the paper is organized as follows. In Section 2, we review the recent research related to our work. In Section 3, we formally define the problem and then propose our method to solve it. In Section 4, we first describe how to set up our experiments, then we analyze and discuss the results. In Section 5, we highlight the theoretical and practical implications of our research. Finally, in Section 6, we conclude and then point out future work directions.

2. Related work

One of the main aspects of our work is finding experts by exploring their posts on community question answering (CQA) sites. Many studies have been conducted to retrieve experts from such environments. We will briefly review these studies in Section 2.1.

Another main aspect of our work is finding experts based on the needs of companies. In Section 2.2, we will examine how related studies have addressed this issue.

The last main aspect of our work is the use of deep learning techniques to find experts. Therefore, in Section 2.3, we will review deep learning techniques used for expert finding.

2.1. Expert finding in CQA

The problem of expert finding has been addressed in various environments such as enterprise corpora (Liang & de Rijke, 2016; Liu et al., 2017a; Xu & Ramanathan, 2016), academia (Cifariello et al., 2019; Torkzadeh Mahani et al., 2018; Zheng et al., 2021), social networks (Lahoti et al., 2017; Li et al., 2020; Wei et al., 2016), social coding platforms (Etemadi et al., 2021; Montandon et al., 2019; Wan et al., 2018), and CQAs Ghasemi et al. (2021), Rostami and Neshati (2021), Sun et al. (2021).

Among the environments mentioned, CQAs have received a lot of attention, because such environments are valuable places to discover the users' true knowledge in various fields. By posting high-quality answers, in addition to helping others for free, users show off their expertise in the topic(s) of the questions asked (Neshati, 2017).

One of the major applications of expert finding in CQAs is question routing (Fu et al., 2020; Huang et al., 2017; Liu et al., 2017b; Song et al., 2021). Here, the goal is to generate a ranked list of users who have sufficient expertise to answer a given question. This issue reduces the waiting time, increases user contribution rates, and produces high quality and more reliable answers (Neshati et al., 2017).

Another major application of expert finding in CQAs is job routing (Dehghan et al., 2019; Huang et al., 2018; Nobari et al., 2020), which generates a ranked list of users who are experts in one or more skill-areas/topics required for a given job position. Our problem is more related to this category. It is worth noting that this task has also been addressed in other environments such as enterprise corpora (Liang, 2019; Liu et al., 2017a) and social coding platforms (Lopes et al., 2021; Wan et al., 2018); hence, we will review it in a separate sub-section.

2.2. Job routing using expert finding

The main purpose of creating a job position is to hire people who have true expertise or general knowledge in one or more given skill-areas. Therefore, expert finding can be used to rank candidates for the job position based on their level of knowledge. A lot of research has been done on this issue. Van Gysel et al. (2016) presented an unsupervised discriminative log-linear model to find experts in enterprise corpora. Neshati et al. (2017) proposed a learning framework that considers the evolution of candidates' knowledge over time to find potential experts in CQAs. Liu et al. (2017a) presented a TopicRank-based document priors model to rank expert candidates who possess great knowledge about a given topic in enterprise corpora. Huang et al. (2018) used tree-guided tensor decomposition and matrix factorization to identify expert users across different collaborative networks. Wan et al. (2018) extended a generative probabilistic ranking model by graph regularization to retrieve software developers with the right coding skills from GitHub. Liang (2019) proposed an unsupervised semantic two-player min–max game to find experts in enterprise corpora. Montandon et al. (2019) used traditional unsupervised and supervised machine learning algorithms to identify candidates who have expertise in popular software libraries on GitHub. Nobari et al. (2020) and Fallahnejad and Beigy (2022) presented some skill translation models to improve the performance of expert finding in StackOverflow. Lopes et al. (2021) used syntactic and semantic analysis techniques to find experts in GitHub.

The mentioned papers can find expert candidates, but they are not yet able to properly meet the other key requirements asked by job positions in the field of software engineering, such as the ability to work in a team to carry out a project according to a given software development methodology. To bridge the gap between expert finding and industry needs, a number of studies has been conducted in recent years, mainly to select the right experts (i.e., T-shaped experts) for agile software development teams. Gharebagh et al. (2018) proposed an entropy-based model to generate a ranked list of T-shaped candidates who have expertise in a given skill-area. Dehghan et al. (2019) addressed exactly the problem introduced in Gharebagh et al. (2018). They used a stacked LSTM neural network to estimate whether candidates specialize in various skill-areas based on temporal changes in the number of their documents in those skill-areas. For a given skill-area, they then ranked the candidates based on their T-shaped expertise, which was manually determined based on their expertise estimated by the stacked LSTM and regardless of their general knowledge. Dehghan et al. (2020) improved the methods proposed by Dehghan et al. (2019). They ranked candidates based on their T-shaped expertise in a given skill-area by learning common patterns between documents associated with the candidates and that skill-area using two parallel CNNs. Rostami and Neshati (2019) expanded the problem and tried to find a group of T-shaped experts who can form a given agile software development team. To this end, as fully explained in Section 1, they proposed two non-machine learning models for finding T-shaped experts who can work together in an agile team. The first one was an entropy-based model and the second one was a model based on the relative deepness of knowledge estimated by the number of documents. Our problem is also retrieving a group of T-shaped experts who can form the best agile team possible, but our proposed method, unlike the methods presented by Rostami and Neshati (2019), is a deep learning-based method that uses the temporal changes in the topics of documents associated with candidates and skill-areas required for the agile team to reconsider candidates selected by state-of-the-art method RDM presented by Rostami and Neshati (2019). In Section 4.5, we show that our proposed method can form better agile teams.

2.3. Deep learning methods for expert finding

In the past years, the methods of expert finding had been mainly based on probabilistic language models (Balog et al., 2009; Liang & de Rijke, 2016; Petkova & Croft, 2008), link analysis (Bouguessa et al., 2008; Zhu et al., 2011, 2014), latent topic modeling (Daud et al., 2010; Momtazi & Naumann, 2013; Yang et al., 2013), etc., but currently, with the advent of deep learning, the methods for expert finding are mainly based on deep learning (He et al., 2021; Nikzad-Khasmakhi et al., 2021; Zhang et al., 2020).

Many studies have recently been conducted on the use of deep learning in expert finding. Zhao et al. (2016) and He et al. (2021) developed frameworks based on random-walk and LSTM neural network to rank candidates to answer a given question. Wang et al. (2017) proposed a CNN-based model to find experts in StackOverflow. Azzam et al. (2017) generated a ranked list of candidates to answer a given question by calculating the cosine similarity between the latent semantic vectors associated with each candidate and the question. They used fully connected neural networks to learn the latent semantic vectors. Dehghan et al. (2019) fed an LSTM neural network with breadth-first and depth-first traversal of candidates' expertise tree to find T-shaped experts who specialize in a given skill-area. Li et al. (2019) proposed a model called NeRank. This model first generates the embedding representations of answerers and a given question using an LSTM-based model, then computes the answerers' rank using a convolutional recommender system. Tang et al. (2020) proposed an attention-based factorization machine to generate a ranked list of experts in CQAs. Dehghan et al. (2020) presented a CNN-based model to generate a ranked list of T-shaped experts who have expertise in a given skill-area. (Nikzad-Khasmakhi et al., 2021) presented BERTERS, a model that uses transformers and graph embedding techniques to identify expert candidates. Similar to the reviewed studies, we intend to propose a deep learning-based method. However, our problem is different from the objectives of the methods discussed. They sought to rank experts/T-shaped experts specialized in a given query, but we intend to use deep learning techniques to find T-shaped experts who can form an agile team. Therefore, the discussed methods are not suitable for our problem.

3. Proposed method (DLM + RDM)

In this section, we formally define our problem, then we describe our approach to solve the problem, and finally, we present the architecture of our model.

Table 1
List of notations and their definitions.

Notation	Definition
S	set of all skill-areas
sa	skill-area
C	pool of candidates
ca	candidate
ϱ	position number in an agile team
C_t	group of candidates selected to form an agile team
S_t	set of all skill-areas required for an agile team
$sa_t^{(\varrho)}$	ϱ-th skill-area of S_t (i.e., expertise skill-area required for the ϱ-th position in an agile team)
$S_t - \{sa_t^{(\varrho)}\}$	set of all general skill-areas required for the ϱ-th position in an agile team
$ca_t^{(\varrho)}$	ϱ-th member of C_t (i.e., candidate selected for the ϱ-th position)
n	team size (i.e. size of S_t and C_t)
k	number of top candidates selected from model RDM (Rostami & Neshati, 2019)
λ	parameter of Eq. (1)
$X_{\varrho,i}$	binary decision variable indicating whether candidate ca_i is selected for the ϱ-th position of an agile team
$D_.$	collection of documents
m	maximum number of documents selected randomly
$D_.^{[m]}$	set of at most m documents selected randomly from $D_.$
$d_.^{(j')}$	j'-th document (in terms of creation date) in $D_.^{[m]}$
$v_{d_.^{(j')}}$	document representation of $d_.^{(j')}$
v_{sa_i}	skill-area representation of sa_i
v_{ca_i}	candidate representation of ca_i
$v_{ca_i}^s$	representation of *shape of expertise* of candidate ca_i
$v_{ca_i}^e$	expert knowledge representation of ca_i given skill-area $sa_t^{(\varrho)}$
$v_{ca_i}^{g_j}$	general knowledge representation of ca_i given jth skill-area of $S_t - \{sa_t^{(\varrho)}\}$
n_d	number of LDA topics (i.e., size of $v_{d_.^{(j')}}$)
n_t	number of output neurons of TFE (i.e., size of v_{sa_i} and v_{ca_i})
n_k	number of output neurons of KRE (i.e., size of $v_{ca_i}^e$ and $v_{ca_i}^{g_j}$)
n_s	size of $v_{ca_i}^s$

3.1. Problem formulation

In this work, we assume that we have a pool of candidates C, a set of skill-areas S, and a collection of documents $D = \bigcup_{ca \in C, sa \in S} D_{ca,sa}$. Here, $D_{ca,sa}$ denotes the set of documents associated with candidate ca that is related to skill-area sa. In the following, we use $D_{ca} = \bigcup_{sa \in S} D_{ca,sa}$ and $D_{sa} = \bigcup_{ca \in C} D_{ca,sa}$ to indicate the set of documents associated with candidate ca and skill-area sa, respectively.

Given a set of skill-areas $S_t \subseteq S$, $|S_t| = n$, required for an agile team, the problem of agile team formation is to retrieve a group of candidates $C_t \subseteq C$, $|C_t| = n$, in such a way that ϱ-th member of C_t, $ca_t^{(\varrho)}$, $1 \leq \varrho \leq n$, is a T-shaped expert who has expertise in ϱ-th skill-area of S_t, $sa_t^{(\varrho)}$, and general knowledge in all skill-areas of $S_t - \{sa_t^{(\varrho)}\}$.

All notations used in this paper are summarized in Table 1.

3.2. Modeling approach

In order to form an ideal agile team given a set of skill-areas S_t, we go through two major steps. In the first step, we use an iterative approach to prepare n lists of top candidates for joining the team. In the second step, we use integer linear programming to form the best possible agile team of size n from the listed candidates. The details of these two steps are as follows:

Step 1. Preparing lists of top candidates.
In this step, in the ϱ-th iteration of an iterative approach, we intend to prepare a list of top candidates for the ϱ-th position in the team, which requires T-shaped expertise in $sa_t^{(\varrho)}$ and general knowledge in all skill-areas of $S_t - \{sa_t^{(\varrho)}\}$. To prepare this list for the ϱ-th position, we take 4 steps as follows:

Step 1.1. We pick out top k candidates for the ϱ-th position, $C_k^{(\varrho)}$, by the state-of-the-art model RDM (Rostami & Neshati, 2019). Among the existing related methods, we have chosen RDM because it has the best overall performance.

Step 1.2. For each candidate $ca_i \in C_k^{(\varrho)}$, $1 \leq i \leq k$, we estimate the probability of $P^{(\varrho)}(ca_i|S_t)$, which indicates the probability that candidate ca_i is a relevant candidate for the ϱ-th position in an agile team with the set of skill-areas S_t, or in other words, the probability that he/she is a T-shaped expert with expertise in $sa_t^{(\varrho)}$ and general knowledge in all skill-areas of $S_t - \{sa_t^{(\varrho)}\}$. To estimate the mentioned probability, we have:

$$P^{(\varrho)}(ca_i|S_t) = \lambda \cdot P_{DLM}^{(\varrho)}(ca_i|S_t) + (1-\lambda) \cdot P_{RDM}^{(\varrho)}(ca_i|S_t), \tag{1}$$

where $P_{DLM}^{(\varrho)}(ca_i|S_t)$ is the probability estimated by the proposed model DLM presented in Section 3.3, $P_{RDM}^{(\varrho)}(ca_i|S_t)$ is the probability estimated by the state-of-the-art model RDM (Rostami & Neshati, 2019), and $\lambda \in [0, 1]$ is a parameter.

As we explained in Section 1, RDM uses the number of a candidate's documents in the required skill-areas as a resource for estimating his/her knowledge, but intuitively it can be said that the number of the candidate's documents alone cannot be a sufficient resource for estimating his/her knowledge to perform tasks related to the ϱ-th position in the agile team. Therefore, in addition to RDM, we use the proposed model DLM, which uses the content of the candidate's documents in the form of a deep learning method. The combination of these two models allows the number and content of the candidate's documents to be considered simultaneously as resources for estimating his/her knowledge to work in the agile team. Therefore, we expect that the combination of these two models performs better than each one alone.

Step 1.3. We store all the top k candidates along with their probabilities estimated by Eq. (1) for the ϱ-th position.

After the nth iteration, we have come up with n lists of top k candidates along with their probabilities for the n positions.

Step 2. Forming the best agile team.

In this step, we want to form the best agile team from the candidates selected in the previous step. To this end, we represent the agile team formation problem as an integer linear program as follows:

$$\begin{aligned} \text{Maximize} \quad & \sum_{\varrho=1}^{n}\sum_{i=1}^{k} P^{(\varrho)}(ca_i|S_t)\cdot X_{\varrho,i} \\ \text{Subject to} \quad & \forall \varrho: \ \sum_{i=1}^{k} X_{\varrho,i} = 1 \\ & \forall i: \ \sum_{\varrho=1}^{n} X_{\varrho,i} \le 1 \\ & \forall \varrho, i: \ X_{\varrho,i} \in \{0,1\}. \end{aligned} \tag{2}$$

In this program, the objective is to maximize the sum of the probabilities that the selected candidates are qualified to take the positions of an agile team with the set of skill areas S_t. Here, $X_{\varrho,i}$ is a binary decision variable (according to the third constraint) indicating whether candidate ca_i is selected for the ϱ-th position of the agile team or not. The first constraint states that exactly one candidate must be selected for each position, and the second constraint states that each candidate must be selected for at most one position. In other words, these two constraints state that each position must be filled with one distinct candidate. A group of candidates who can maximize the value of the objective function given the mentioned constraints forms the best agile team possible.

In the next sub-section, we present our proposed deep learning model (DLM) to estimate the probability of $P^{(\varrho)}_{DLM}(ca_i|S_t)$ in Eq. (1).

3.3. DLM architecture

In this section, we propose the deep learning model DLM, which is intended to estimate the probability of $P^{(\varrho)}_{DLM}(ca_i|S_t)$, which indicates the probability that candidate ca_i is relevant to the ϱ-th position in an agile team with the set of skill-areas S_t. To be relevant to the ϱ-th position, ca_i should be a T-shaped expert who has expertise in skill-area $sa_t^{(\varrho)}$ and general knowledge in all skill-areas of $S_t - \{sa_t^{(\varrho)}\}$. Therefore, our deep learning model should be able to simultaneously acquire the shape of expertise, expertise in skill-area $sa_t^{(\varrho)}$, and general knowledge in all skill-areas of $S_t - \{sa_t^{(\varrho)}\}$. By knowing this information, DLM can estimate the probability of $P^{(\varrho)}_{DLM}(ca_i|S_t)$ with the help of a fully connected (FC) network.

To estimate the shape of expertise, expertise, and general knowledge, DLM uses neural networks SHE, EKN, and GKN., respectively. In all of these networks, we use the concept of time to discover the relevant knowledge because the shape of expertise of candidates and the level of knowledge required to gain general/expert knowledge in each skill-area are constantly changing over time. To consider the concept of time, we use a recurrent neural network that takes into account changing the distribution of topics of documents associated with candidates/skill-areas over time.

The DLM architecture is fully illustrated in Fig. 2. In the following, we will explain the architecture of each of its parts in full detail:

SHE: Learning a vector representation for the *shape of expertise* of ca_i.

Here, first, a set of at most m randomly selected documents associated with candidate ca_i, $D^{[m]}_{ca_i} \subseteq D_{ca_i}$, is fed to a temporal feature extractor (TFE). TFE is a recurrent neural network that takes a set of documents as input and outputs a representation for this set by examining the distribution of topics of its documents over time. The TFE architecture is fully described in Section 3.3.1.

After TFE gets the set of documents $D^{[m]}_{ca_i}$, it generates candidate representation of ca_i, $v_{ca_i} \in \mathbb{R}^{n_t}$, which represents ca_i's profile based on the distribution of topics of his/her documents over time. A fully connected network is then fed with vector v_{ca_i} to learn the representation of the *shape of expertise* of candidate ca_i, $v^s_{ca_i} \in \mathbb{R}^{n_s}$, based on his/her profile. Given $v^s_{ca_i}$, DLM can get closer to accurately estimating the probability of $P^{(\varrho)}_{DLM}(ca_i|S_t)$ by estimating how T-shaped ca_i is.

EKN: Learning expert knowledge representation of ca_i given skill-area $sa_t^{(\varrho)}$.

Here, first, a set of at most m randomly selected documents associated with skill-area $sa_t^{(\varrho)}$, $D^{[m]}_{sa_t^{(\varrho)}} \subseteq D_{sa_t^{(\varrho)}}$, is passed to TFE to generate skill-area representation of $sa_t^{(\varrho)}$, $v_{sa_t^{(\varrho)}} \in \mathbb{R}^{n_t}$. Then, vector $v_{sa_t^{(\varrho)}}$ and vector v_{ca_i} generated by SHE are fed together to a knowledge representation extractor (KRE). KRE is a neural network that takes both a candidate representation and a skill-area

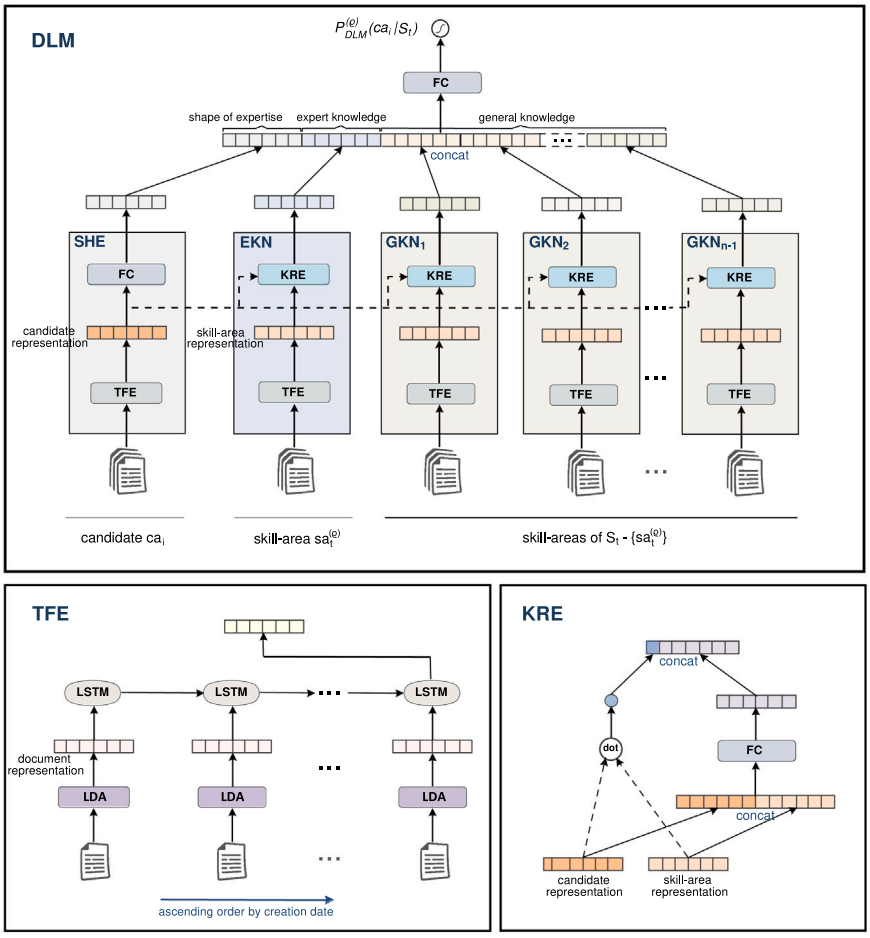

Fig. 2. The architecture of our proposed model DLM.

representation as input and outputs a vector that represents the candidate's knowledge in the required skill-area by learning the common patterns between the two representations. The KRE architecture is fully described in Section 3.3.2.

After KRE gets both vector representations v_{ca_i} and $v_{sa_t^{(o)}}$, it learns expert representation of candidate ca_i given skill-area $sa_t^{(o)}$, $v^e_{ca_i} \in \mathbb{R}^{n_k}$. Vector representation $v^e_{ca_i}$ can facilitate the estimation of the probability of $P^{(o)}_{DLM}(ca_i|S_t)$ for DLM.

GKN$_j$, $1 \leq j \leq$ n-1: Learning general knowledge representation of ca_i given jth skill-area of $S_t - \{sa_t^{(o)}\}$.

The architecture of GKN$_j$ is similar to EKN; however, training of GKN$_j$ is for learning general knowledge representation of candidate ca_i given jth skill-area of $S_t - \{sa_t^{(o)}\}$, $v^{g_j}_{ca_i} \in \mathbb{R}^{n_k}$. Each of the vector representations $v^{g_j}_{ca_i}$, $1 \leq j \leq n-1$, can help DLM to estimate the probability of $P^{(o)}_{DLM}(ca_i|S_t)$ more accurately.

The concatenation of all the output vectors of the mentioned neural networks is passed to a fully connected network to estimate the probability of $P^{(o)}_{DLM}(ca_i|S_t)$.

In the following, we describe the architecture of TFE and KRE in Sections 3.3.1 and 3.3.2, respectively.

3.3.1. TFE: temporal feature extractor

TFE gets a set of textual documents and then generates a vector representation that represents the knowledge contained in this set. To generate the vector representation, TFE examines the process of changing the distribution of topics of the documents over time using a recurrent neural network (LSTM). To acquire the distribution of topics of each document, TFE uses LDA (Blei et al., 2003), which is one of the most well-known methods for topic modeling.

To be more precise, TFE is used to learn candidate representation v_{ca_i} or skill-area representation v_{sa_t} over time. We describe TFE for the first purpose. The other can be described in the same way. The network takes documents of set $D^{[m]}_{ca_i}$ as input and then sorts them in ascending order by their creation date. Next, it uses LDA topic modeling to extract document representation $v_{d^{(j')}_{ca_i}} \in \mathbb{R}^{n_d}$

for every document $d_{ca_i}^{(j')} \in D_{ca_i}^{[m]}$, $1 \le j' \le m$. Finally, it feeds an LSTM neural network with $\{v_{d_{ca_i}^{(1)}}, v_{d_{ca_i}^{(2)}}, \dots, v_{d_{ca_i}^{(m)}}\}$ to learn candidate representation of ca_i, v_{ca_i}, over time.

3.3.2. KRE: knowledge representation extractor

KRE gets both a candidate representation and a skill-area representation as input and then generates the candidate's knowledge in the required skill-area as a vector representation. More precisely, by learning common patterns between candidate representation of v_{ca_i} and skill-area representation of $v_{sa_t^{\cdot}}$, KRE estimates the expert/general knowledge of candidate ca_i given skill-area $sa_t^{\cdot}$ in the form of a vector representation. To this end, in parallel, (1) concatenation of the given two vectors is fed into a fully connected network to learn the expert/general knowledge representation, and (2) dot product of the given two vectors is calculated and is then appended to the mentioned representation.

If the goal of the network is to learn expert knowledge representation (i.e., use of KRE in neural network EKN), there should be a great deal of commonality between the given two vectors so that the expert knowledge representation can represent ca_i's expertise in the given skill-area. Otherwise, i.e., when the goal of the network is to learn general knowledge representation (i.e., use of KRE in neural network GKN_j, $1 \le j \le n\text{-}1$), there should be a relative commonality between the given two vectors so that the general knowledge representation can represent ca_i's general knowledge in the given skill-area.

To sum up, this section presented an approach based on deep learning techniques and integer linear programming to solve the agile team formation problem. In the next section, we will conduct the experiments.

4. Experiments

4.1. Evaluation measures

Let $S_t = \{sa_t^{(1)}, sa_t^{(2)}, \dots, sa_t^{(n)}\}$ be the set of skill-areas required for an agile team t, and $C_t = \{ca_t^{(1)}, ca_t^{(2)}, \dots, ca_t^{(n)}\}$ be the group of candidates selected to build the team. To evaluate this team, we use the following measures introduced in Rostami and Neshati (2019). Here, $G^e_{sa_t^{\cdot}}$ and $G^g_{sa_t^{\cdot}}$ indicate the set of candidates who have expert and general knowledge in skill-area $sa_t^{\cdot}$ according to the golden set, respectively. Besides, $G^e_{ca_t^{\cdot}}$ denotes the set of skill-areas in which candidate $ca_t^{\cdot}$ has expertise according to the golden set.

Coverage. This measure indicates what percentage of the skill-areas required for the team is covered by its members (Rostami & Neshati, 2019). It is defined as follows:

$$TeamCov = \frac{1}{n}\sum_{\varrho=1}^{n} PosCov(\varrho), \tag{3}$$

in which $PosCov(\varrho)$ determines whether the candidate selected for the ϱ-th position, $ca_t^{(\varrho)}$, has expert knowledge in expertise skill-area related to that position, $sa_t^{(\varrho)}$:

$$PosCov(\varrho) = \begin{cases} 1, & \text{if } ca_t^{(\varrho)} \in G^e_{sa_t^{(\varrho)}}, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Communication. This measure indicates how well team members are able to communicate with each other based on their knowledge (Rostami & Neshati, 2019). It is defined as follows:

$$TeamCom = \frac{1}{n}\sum_{\varrho=1}^{n} PosCom(\varrho), \tag{5}$$

where $PosCom(\varrho)$ is a function that indicates how familiar the candidate selected for the ϱ-th position, $ca_t^{(\varrho)}$, is with the expertise skill-areas related to other positions, $\{sa_t^{(\varrho')} | \varrho' : 1, ..., n,\ \varrho' \neq \varrho\}$. It is defined as follows:

$$PosCom(\varrho) = \frac{1}{n-1}\sum_{\varrho'=1, \varrho' \neq \varrho}^{n} HasKnow(\varrho, \varrho'), \tag{6}$$

in which $HasKnow(\varrho, \varrho')$ determines whether candidate $ca_t^{(\varrho)}$ is familiar enough with skill-area $sa_t^{(\varrho')}$, or in other words, whether he/she has at least general knowledge in skill-area $sa_t^{(\varrho')}$:

$$HasKnow(\varrho, \varrho') = \begin{cases} 1, & \text{if } ca_t^{(\varrho)} \in G^g_{sa_t^{(\varrho')}} \cup G^e_{sa_t^{(\varrho')}}, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Optimality. This measure indicates what percentage of team members are cost-effective based on their knowledge, which probably estimates the income they expect to earn (Rostami & Neshati, 2019). The measure is defined as follows:

$$TeamOpt = \frac{1}{n}\sum_{\varrho=1}^{n} PosOpt(\varrho), \tag{8}$$

Table 2
Summary of datasets statistics.

Dataset	#Skill-areas	#Posts		#Users		
		#Questions	#Answers	#Experts		#Non-experts
				#T-shaped	#Non-T-shaped	
C#	20	763,717	1,453,649	1,976	2,514	79,605
Java	26	810,071	1,510,812	1,744	2,394	79,419

where $PosOpt(\varrho)$ determines how cost-effective candidate $ca_t^{(\varrho)}$ is based on the number of skill-areas in the collection in which he/she has expert knowledge:

$$PosOpt(\varrho) = \begin{cases} \frac{1}{|G^e_{ca_t^{(\varrho)}}|^2}, & \text{if } \exists sa \in S, ca_t^{(\varrho)} \in G^e_{sa}, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

F-measure. This measure indicates a trade-off between the mentioned measures by calculating their harmonic mean as follows:

$$TeamF = \frac{3}{\frac{1}{TeamCov} + \frac{1}{TeamCom} + \frac{1}{TeamOpt}}. \tag{10}$$

We evaluate the overall performance of agile team formation methods using this measure.

4.2. Datasets

In this paper, we use two datasets C# and Java generated by Rostami and Neshati (2019), which are augmented versions of two of the three datasets generated by Gharebagh et al. (2018). Both datasets contain a large number of documents posted on StackOverflow from August 2008 to March 2015. A summary of general statistics of both datasets is presented in Table 2.

Dataset C# comprises 763,717 questions that have been tagged with *c#* and 1,453,649 answers to those questions. Each of these documents is associated with one or more of the 20 skill-areas related to c# language programming that have been extracted by Gharebagh et al. (2018).[3] In each skill-area, each user is classified into three categories of *expert*, *possessor of general knowledge*, and *possessor of beginner knowledge* based on the number of his/her accepted answers posted in response to questions related to this skill-area. Then, according to his/her level of knowledge in various skill-areas, his/her shape of expertise is determined. In total, there are 84,095 users, 1,976 of whom are T-shaped experts and 2,514 of whom are non-T-shaped experts. We use 5-fold cross-validation to make sure that every user will be tested. The description of dataset Java is similar to the description of dataset C#.

It is worth noting that, as can be perceived from the description of the datasets, in the golden set, we only know what shape of expertise each candidate has and what level of knowledge he/she has in each skill-area. Therefore, we cannot extract the actual output of model DLM directly from the golden set. In order to calculate the actual output, we use the harmonic mean of the functions $PosCov(\varrho)$, $PosCom(\varrho)$, and $PosOpt(\varrho)$ presented in Section 4.1 because it can determine how relevant a given candidate is to the ϱ-th position of an agile team according to the golden set.

4.3. Baseline methods

We use two state-of-the-art methods XEBM and RDM proposed in Rostami and Neshati (2019) as the baseline methods. XEBM is an entropy-based method that employs two main approaches simultaneously to select the right candidates for an agile team: (1) calculating the diversity among candidates' level of knowledge in various skill-areas and (2) comparing candidates' level of knowledge in the required skill-areas with the average level of knowledge in those skill-areas per candidate. RDM, similar to XEBM, is a non-machine learning method that picks out the right candidates to build an agile team by employing two main approaches simultaneously: (1) estimating the relative deepness of candidates' knowledge in their strongest skill-area and (2) an approach similar to the second approach employed in XEBM. In both XEBM and RDM, the number of documents is used as the resource for estimating the knowledge.

4.4. Implementation details and parameter settings

We have used Keras[4] library to implement our proposed model DLM. For training, we have used RMSprop optimizer with a learning rate of 0.005 and a batch size of 512. In addition, we have used mean absolute error as the loss function. Gensim[5] library

[3] Each extracted skill-area represents a set of related tags.
[4] https://keras.io/
[5] https://radimrehurek.com/gensim/

has been used to extract LDA topic distribution of documents, and PULP_CBC_CMD solver in PuLP[6] library has been used for solving the integer linear programming problem.

In this work, we have used answers posted by candidates as their documents. We have considered size of an agile team, n, to be 3 and we have set number of LDA topics, n_d, to 100. In addition, number of units in each LSTM cell has been set to 32; hence, number of output neurons of TFE, n_t, is also 32. Moreover, number of output neurons of KRE, n_k, and size of vector representation of *shape of expertise* of candidate ca, n_s, have been set to 8.

The LSTM layer uses both a dropout rate and a recurrent dropout rate of 0.6. The fully connected networks used in SHE and KRE contain a hidden layer with 16 neurons, followed by a batch normalization layer, and then a dropout layer with a dropout rate of 0.6. Moreover, we have performed batch normalization and dropout at a rate of 0.3 after the output layers of these networks. Also, the last fully connected network used in DLM contains two hidden layers with sizes 16 (first hidden layer) and 4 (second hidden layer), each of which is followed by a batch normalization layer.

In Section 4.5, we will examine how effectiveness of our proposed method varies by changing (1) maximum number of documents selected for candidate ca_i and each of the skill-areas of set S_t, m, (2) number of top candidates selected from model RDM, k, and (3) parameter λ in Eq. (1).

4.5. Results and discussion

In this section, we intend to answer the following research questions by designing several experiments:

RQ1: In DLM, we used two ways to estimate candidates' knowledge in the required skill-areas: (1) dot product and (2) concatenation. How do these two variants affect the performance of our proposed method?
RQ2: How does the performance of our proposed method vary by changing the parameters m, k, and λ?
RQ3: Which measures can our proposed method influence the most compared to other methods? Does it have the best overall performance?
RQ4: In terms of the shape of expertise of candidates selected for agile teams, how does our proposed method perform compared to other methods? Does it use T-shaped experts more than others to form agile teams?

We will answer research questions RQ1 to RQ4 in Sections 4.5.1 to 4.5.4, respectively.

It is worth mentioning that in this section, by the proposed method, we mean method DLM + RDM. In addition, whenever we mention the overall performance, we mean the performance in terms of F-measure.

4.5.1. Analysis of approaches used in KRE

Fig. 3 shows the effects of using dot product and concatenation in KRE on the overall performance of our proposed method on two datasets C# and Java. In this figure, by F-measure, we mean the value of the best F-measure obtained by testing all values of the parameter k from 3 to 50 and all possible values with an accuracy of 2 decimal places for parameter λ. The following observations can be made from Fig. 3:

- In general, using the dot product of vector representations of candidates and skill-areas in KRE has a greater impact on the performance of our proposed method than using a fully connected network fed by the concatenation of these two representations.
- Although the performance of the concatenation variant is generally weaker than the dot product, the simultaneous use of these two variants can perform better than each one alone. This shows that each of these variants can complement the other.

It should be noted that since the performance changes in these three variants (i.e., dot product, concatenation, and simultaneous use of the two variants) with respect to m are almost similar and we want to analyze the parameter m completely in the next sub-section, we will skip this analysis for now.

4.5.2. Parameter sensitivity

Fig. 4 shows the effects of changes in parameters m and k on the overall performance of our method on two datasets C# and Java in the form of two heatmap diagrams. In each diagram, the value (i.e., color) of each cell represents the value of the best F-measure for a particular k and m. The best F-measure is obtained by testing all possible values with an accuracy of 2 decimal places for parameter λ. The following observations can be made from Fig. 4:

- For any value of m, despite the fluctuations observed in the performance of our method at small k, from one point onward, in general, the higher the value of k, the weaker the performance of our method. Therefore, the performance of our method at relatively small k is better than the performance when k is large. This shows that if the number of top candidates selected from model RDM is relatively small, our method has more power to select the best of them to form an agile team.

[6] https://coin-or.github.io/pulp/

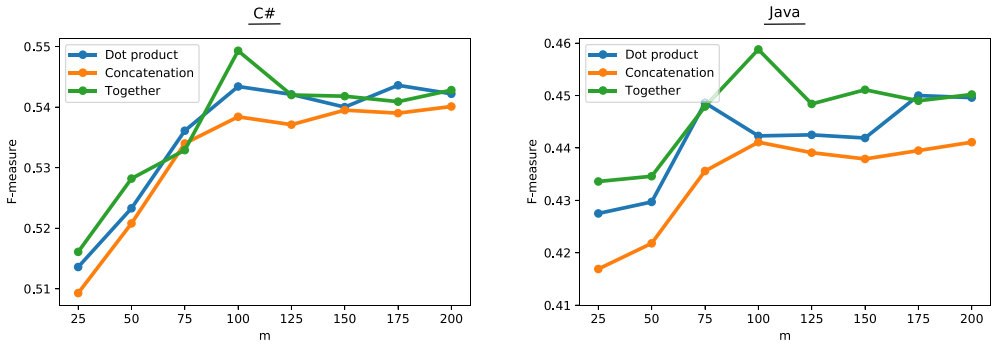

Fig. 3. The effects of using dot product and concatenation in KRE on the performance of our proposed method.

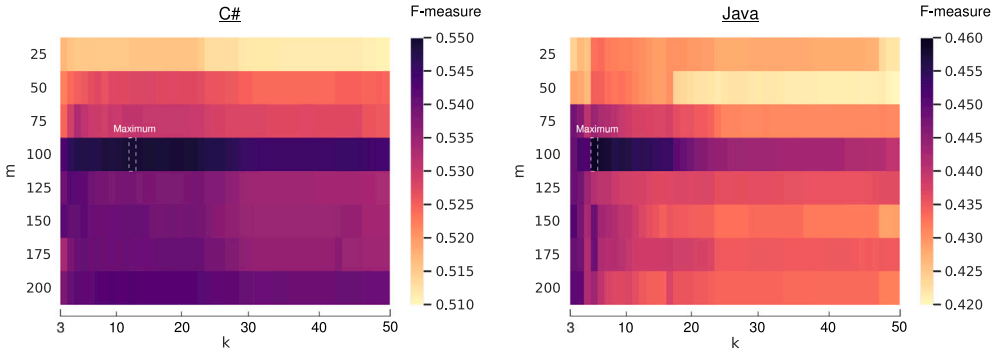

Fig. 4. Sensitivity analysis of our proposed method to changes in parameters m and k.

- Up to $m = 100$, the higher the value of m, the better the performance of our method. For values of m greater than 100, the performance is worse than when $m = 100$ but it is generally better than when m is less than 100. This shows that if we feed a maximum of 100 documents associated with each candidate and skill-area into DLM, we can find more qualified candidates to form an agile team. If the maximum number of selected documents is more than 100, because of the large number of learnable parameters, DLM is prone to overfit and is therefore unable to generalize better than when the maximum number of selected documents is 100.

It should be noted that as shown in Fig. 4, in dataset C#, our method performs best when $m = 100$ and $k = 13$, and in dataset Java, it performs best when $m = 100$ and $k = 6$. In the rest of this section, we will perform our experiments based on these settings.

Fig. 5 shows the effects of changes in parameter λ on the performance of our method in terms of Coverage, Communication, Optimality, and F-measure on both datasets C# and Java. Based on this figure, we can make the following observations:

- In general, the smaller the value of λ, the higher the performance of our method in terms of Coverage and Communication; hence, since according to Eq. (1), the smaller the value of λ, the larger the effect of RDM, RDM performs better in terms Coverage and Communication.
- In general, the larger the value of λ, the higher the performance of our method in terms of Optimality; hence, since according to Eq. (1), the larger the value of λ, the larger the effect of DLM, DLM performs better in terms of Optimality.
- With respect to the above observations, finding the best trade-off point that simultaneously uses RDM to increase Coverage and Communications and DLM to increase Optimality is crucial because at this point, the best overall performance of our method (i.e., the best performance in terms of F-measure) will be achieved.
  On dataset Java, $\lambda = 0.41$ is the point that leads to the best possible trade-off between Coverage, Communication, and Optimality. Here, the closer the value of λ is to 0.41, the better the overall performance of our method. In addition, the farther away the value of λ is from 0.41, the weaker the overall performance of our method, because if $\lambda \to 0$, F-measure decreases as Optimality decreases, and if $\lambda \to 1$, it decreases as Coverage and Communication decrease. Moreover, on dataset C#, $\lambda = 0.36$ is the best possible trade-off point. The analysis of the changes in F-measure on dataset C# is similar to the changes on dataset Java, with the difference that from the best trade-off point (i.e., $\lambda = 0.36$) to $\lambda = 0.8$, the value of F-measure fluctuates slightly,[7] but as expected it decreases when λ is greater than 0.8.

[7] Due to the enlarged scale, the fluctuations seem large despite their small size.

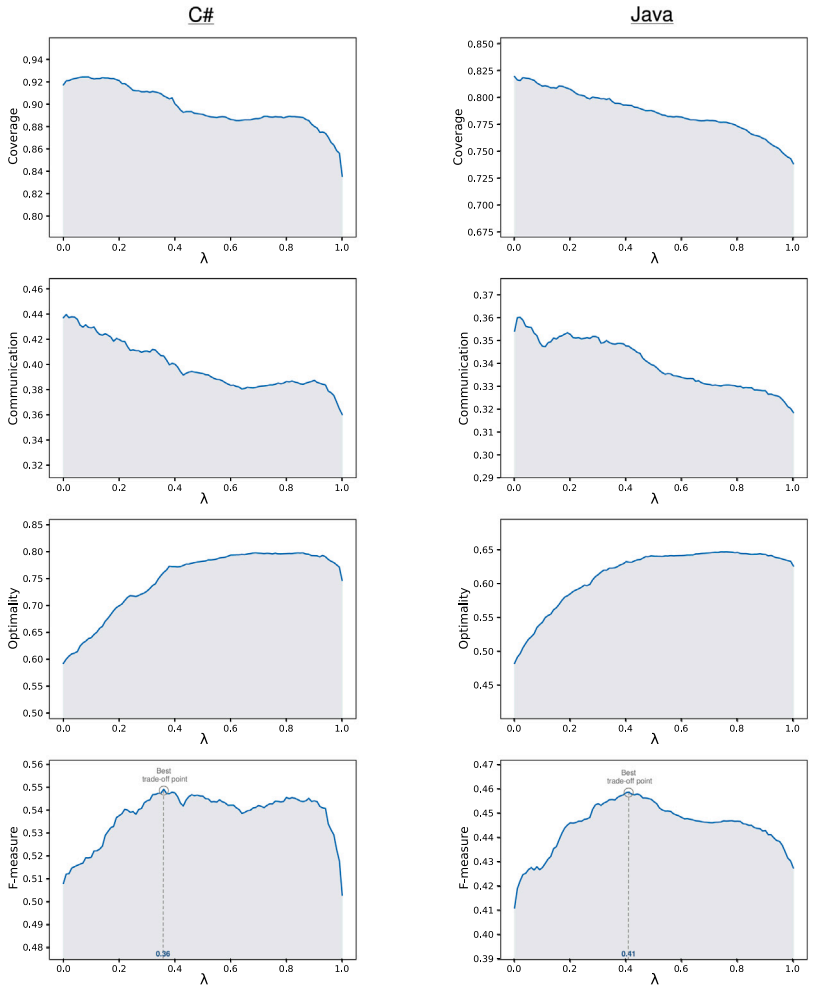

Fig. 5. Sensitivity analysis of our proposed method to changes in parameter λ.

4.5.3. Performance comparison

Table 3 shows the performance of methods XEBM, RDM, DLM, and DLM + RDM on two datasets C# and Java in terms of Coverage, Communication, Optimality, and F-measure. For DLM and DLM + RDM, we use the parameters set in Section 4.5.2. The following observations can be made from Table 3:

- **RDM vs. XEBM**: On datasets C# and Java, the performance of RDM in terms of F-measure is superior to XEBM by 21.2% and 9%, respectively. More precisely, on dataset C#, although RDM performs worse than XEBM by 3.8% in terms of Coverage, it outperforms XEBM by 9.5% and 25.1% in terms of Communication and Optimality, respectively; hence, its overall performance is better than XEBM. In addition, on dataset Java, RDM performs worse than XEBM by 10% and 4% in terms of Coverage and Communication, respectively, but it outperforms XEBM by 18.4% in terms of Optimality; hence, its overall performance is also better than XEBM.
- **DLM vs. RDM**: DLM and RDM are generally not superior to each other. In terms of F-measure, RDM outperforms DLM by 1% on dataset C#, whereas on dataset Java, DLM outperforms RDM by 3.9%. RDM has achieved this performance with a focus on Coverage and Communication, while DLM has achieved this performance with a focus on Optimality. These show that the content of candidates' documents can be very important in estimating the candidates' shape of expertise. In contrast, using the content to estimate candidates' level of general/expert knowledge in the required skill-areas is more challenging than estimating based only on the number of documents in those skill-areas.
- **DLM + RDM vs. DLM and RDM**: DLM + RDM has the best overall performance among the methods presented in Table 3. Due to the simultaneous use of DLM and RDM, DLM + RDM is able to cover the weaknesses of DLM in terms of Coverage and Communication by utilizing the strengths of RDM in terms of Coverage and Communication, and it is able to cover the weaknesses of RDM in terms of Optimality by utilizing the strengths of DLM in terms of Optimality. More precisely, on dataset C#, DLM + RDM outperforms DLM by 8.6% and 13.1% in terms of Coverage and Communication, respectively, and outperforms RDM by 28%

Table 3
Performances of the baselines methods (i.e., XEBM and RDM) and the proposed methods (i.e., DLM and DLM+RDM) on two datasets C# and Java in terms of Coverage, Communication, Optimality, and F-measure.

Dataset	Method	Evaluation measures			
		Coverage	Communication	Optimality	**F-measure**
C#	XEBM	0.952	0.399	0.474	0.419
	RDM	0.917	0.437	0.593	0.508
	DLM	0.835	0.360	0.747	0.503
	DLM + RDM	0.907	0.407	0.762	**0.549**
Java	XEBM	0.902	0.368	0.408	0.378
	RDM	0.820	0.354	0.483	0.412
	DLM	0.738	0.318	0.626	0.428
	DLM + RDM	0.792	0.347	0.632	**0.459**

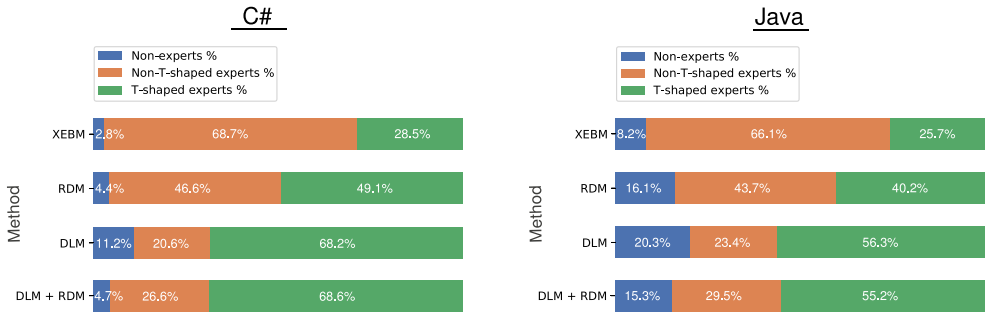

Fig. 6. The distribution of candidates selected for agile teams based on their shapes of expertise and the methods used. Here, by the shapes of expertise, we mean Non-experts, Non-T-shaped experts, and T-shaped experts.

in terms of Optimality; hence, in terms of F-measure, it is superior to RDM and DLM by 8.1% and 9.1%, respectively. In addition, on dataset Java, DLM + RDM outperforms DLM by 7.3% and 9.1% in terms of Coverage and Communication, respectively, and outperforms RDM by 30.8% in terms of Optimality; hence, in terms of F-measure, it is superior to RDM and DLM by 11.4% and 7.2%, respectively. As a result, it can be said that the simultaneous use of the number and content of candidates' documents can play an important role in accurately estimating the relevant knowledge of candidates seeking to join an agile team.

4.5.4. Analysis of shapes of expertise

Fig. 6 shows the distribution of candidates selected to form agile teams based on their shapes of expertise and the methods used. In general, the smaller the number of selected Non-experts (i.e., the blue part) and the larger the number of selected T-shaped experts (i.e., the green part), the better. Based on Fig. 6 and referring to Table 3, we can make the following observations:

- Although XEBM has used more experts than other methods to form agile teams, it has used fewer T-shaped experts than other methods. This is because XEBM has the best performance in terms of Coverage, but it has the worst performance in terms of Optimality.
- Compared to XEBM, RDM has used more T-shaped experts for agile teams. This is because RDM outperforms XEBM in terms of Optimality.
- Although DLM has used fewer experts than XEBM and RDM, it has used more T-shaped experts than them. The reason is that DLM is inferior to XEBM and RDM in terms of Coverage, but instead, it is superior to them in terms of Optimality.
- Similar to RDM, DLM + RDM has used experts more than DLM because it outperforms DLM in terms of Coverage. Also, similar to DLM, it has used T-shaped experts more than RDM because it outperforms RDM in terms of Optimality.

5. Implications

Here, we clearly describe the theoretical and practical implications of our research and discuss how our research differs from existing work.

Existing studies on the agile team formation problem can be criticized from two perspectives: (1) They have used some heuristic non-machine learning methods to solve the problem, while machine learning has been successful in similar tasks such as expert/T-shaped expert ranking, and (2) Their main resource for estimating candidates' relevant knowledge was only the number of the candidates' documents in the required skill-areas, while the content of the candidates' documents is also a valuable resource to estimate the required knowledge from the candidates' profile. Our main proposed method (i.e., DLM + RDM) not only remedies the deficiencies outlined but also makes use of their potential strengths to form a better agile team. In our method, we used a deep

learning model called DLM, which examines the distribution of topics extracted from the content of candidates' documents over time to estimate how relevant the candidates are to the positions of a given agile team. We then combined DLM with the best existing method called RDM to make use of both models that can be complementary.

We practically compared our method with two existing related methods, namely XEBM and RDM, using two datasets extracted from StackOverflow. We showed that the baseline methods perform better in terms of Coverage and Communication, but on the other hand, DLM performs much better in terms of Optimality. For this reason, the proposed method DLM + RDM, which is a combination of DLM with the best baseline method (i.e., RDM), is able to take advantage of the strengths of each to cover the weaknesses of the other and thus lead to forming better agile teams. According to the results, we also found that using the content of candidates' documents can be very important for estimating their shape of expertise that requires a deep understanding of their profile. For this reason, DLM and DLM + RDM use more T-shaped experts than baseline methods to form agile teams.

In the future, researchers can develop new methods using our framework. In our opinion, they can take at least three approaches to better form agile teams:

1. Providing a new model instead of RDM:
   We have shown that the number of documents is a valuable resource for estimating the knowledge of candidates who want to work in an agile team. Therefore, if researchers can propose a new model instead of RDM to better estimate candidates' knowledge based on the number of their documents, they can possibly improve the performance of the main proposed method (Eq. (1)).
2. Providing a new model instead of DLM:
   We have shown that the content of documents is also a valuable resource for estimating candidates' knowledge for agile teams. Therefore, if researchers can propose a new model instead of DLM to better estimate candidates' knowledge based on the content of their documents, they can possibly improve the performance of the main proposed method (Eq. (1)).
3. Extending Eq. (1):
   Researchers can extend Eq. (1) by adding a new model that uses a different valuable resource of knowledge to estimate candidates' knowledge in agile teams. For example, the network of candidates, which can be created through questions and answers exchanged among candidates on CQA sites, can possibly be a valuable resource of knowledge to find the best candidates for agile teams.

6. Conclusion and future work

In this paper, we addressed the agile team formation problem, where the input is a set of skill-areas required for a project and the ideal output is the best agile team possible for carrying out the project. To solve the problem, we presented an integer linear programming model to select the best group of T-shaped experts who can form the given agile team. In addition, a multi-step iterative method, which simultaneously applies baseline model RDM and proposed model DLM, was used to determine whether a candidate is relevant to a given position in the agile team. RDM is a heuristic non-machine learning model that estimates candidates' relevant knowledge by focusing on the number of their documents in various skill-areas. In contrast, DLM is a proposed deep learning model that uses the content of candidates' documents to estimate the knowledge they need to work in a given agile team. More precisely, DLM estimates whether a candidate has the relevant knowledge to work in an agile team by estimating his/her shape of expertise, level of expertise, and level of general knowledge in parallel. To estimate the shape of expertise, it uses a recurrent neural network that learns the candidate's temporal profile by analyzing the process of changing the distribution of topics of his/her documents over time. In addition, to estimate the general/expert knowledge, DLM uses a deep neural network that compares the candidate's temporal profile with the expected temporal profile of a candidate having general/expert knowledge in the required skill-areas. RDM was able to improve the performance of our method in terms of Coverage and Communication, while DLM improved its performance in terms of Optimality. For this reason, the simultaneous use of these two models could improve the overall performance of our method. On datasets C# and Java, our method outperforms RDM, which is considered the best baseline method, by 8.1% and 11.4% in terms of F-measure, respectively.

For future work, we intend to use more resources to better estimate candidates' knowledge to work in an agile team. In addition to the number and content of candidates' documents, using the network of candidates may also be useful. To create this network, we can use the questions and answers exchanged among candidates on StackOverflow.

CRediT authorship contribution statement

Peyman Rostami: Conceptualization, Methodology, Software, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Visualization. **Azadeh Shakery:** Conceptualization, Methodology, Validation, Writing – review & editing, Supervision, Project administration.

Data availability

We have used the dataset introduced in (Rostami & Neshati, 2019) in this research. https://doi.org/10.1016/j.eswa.2018.10.015.

Acknowledgments

This research was in part supported by a grant from the School of Computer Science, Institute for Research in Fundamental Sciences, IPM, Iran (No. CS1400-4-237).

References

Alarfaj, F., Kruschwitz, U., Hunter, D., & Fox, C. (2012). Finding the right supervisor: expert-finding in a university domain. In *Proceedings of the NAACL HLT 2012 student research workshop* (pp. 1–6).

Ambler, S. (2012). *Agile database techniques: Effective strategies for the agile software developer*. John Wiley & Sons.

Azzam, A., Tazi, N., & Hossny, A. (2017). A question routing technique using deep neural network for communities of question answering. In *International conference on database systems for advanced applications* (pp. 35–49). Springer.

Balog, K., Azzopardi, L., & de Rijke, M. (2009). A language modeling framework for expert finding. *Information Processing & Management*, *45*(1), 1–19.

Balog, K., Fang, Y., De Rijke, M., Serdyukov, P., & Si, L. (2012). Expertise retrieval. *Foundations and Trends in Information Retrieval*, *6*(2–3), 127–256.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Bouguessa, M., Dumoulin, B., & Wang, S. (2008). Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 866–874).

Cifariello, P., Ferragina, P., & Ponza, M. (2019). Wiser: A semantic approach for expert finding in academia based on entity linking. *Information Systems*, *82*, 1–16.

Danait, A. (2005). Agile offshore techniques-a case study. In *Agile development conference (ADC'05)* (pp. 214–217). IEEE.

Daud, A., Li, J., Zhou, L., & Muhammad, F. (2010). Temporal expert finding through generalized time topic modeling. *Knowledge-Based Systems*, *23*(6), 615–625.

Dehghan, M., Biabani, M., & Abin, A. A. (2019). Temporal expert profiling: With an application to T-shaped expert finding. *Information Processing & Management*, *56*(3), 1067–1079.

Dehghan, M., Rahmani, H. A., Abin, A. A., & Vu, V.-V. (2020). Mining shape of expertise: A novel approach based on convolutional neural network. *Information Processing & Management*, *57*(4), Article 102239.

Etemadi, R., Zihayat, M., Feng, K., Adelman, J., & Bagheri, E. (2021). Collaborative experts discovery in social coding platforms. In *Proceedings of the 30th ACM international conference on information & knowledge management* (pp. 3009–3013).

Fallahnejad, Z., & Beigy, H. (2022). Attention-based skill translation models for expert finding. *Expert Systems with Applications*, *193*, Article 116433.

Fu, J., Li, Y., Zhang, Q., Wu, Q., Ma, R., Huang, X., & Jiang, Y.-G. (2020). Recurrent memory reasoning network for expert finding in community question answering. In *Proceedings of the 13th international conference on web search and data mining* (pp. 187–195).

Gharebagh, S. S., Rostami, P., & Neshati, M. (2018). T-shaped mining: A novel approach to talent finding for agile software teams. In *European conference on information retrieval* (pp. 411–423). Springer.

Ghasemi, N., Fatourechi, R., & Momtazi, S. (2021). User embedding for expert finding in community question answering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *15*(4), 1–16.

He, T., Guo, C., & Chu, Y. (2021). Enhanced user interest and expertise modeling for expert recommendation. In *2020 25th international conference on pattern recognition* (pp. 556–562). IEEE.

Hu, J., Zhang, X., Yang, Y., Liu, Y., & Chen, X. (2020). New doctors ranking system based on VIKOR method. *International Transactions in Operational Research*, *27*(2), 1236–1261.

Huang, C., Yao, L., Wang, X., Benatallah, B., & Sheng, Q. Z. (2017). Expert as a service: Software expert recommendation via knowledge domain embeddings in stack overflow. In *2017 IEEE international conference on web services* (pp. 317–324). IEEE.

Huang, C., Yao, L., Wang, X., Benatallah, B., Zhang, S., & Dong, M. (2018). Expert recommendation via tensor factorization with regularizing hierarchical topical relationships. In *International conference on service-oriented computing* (pp. 373–387). Springer.

Lahoti, P., De Francisci Morales, G., & Gionis, A. (2017). Finding topical experts in Twitter via query-dependent personalized PageRank. In *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017* (pp. 155–162).

Li, G., Dong, M., Yang, F., Zeng, J., Yuan, J., Jin, C., Hung, N. Q. V., Cong, P. T., & Zheng, B. (2020). Misinformation-oriented expert finding in social networks. *World Wide Web*, *23*(2), 693–714.

Li, Z., Jiang, J.-Y., Sun, Y., & Wang, W. (2019). Personalized question routing via heterogeneous network embedding. In *Proceedings of the AAAI conference on artificial intelligence* (01), (pp. 192–199).

Liang, S. (2019). Unsupervised semantic generative adversarial networks for expert retrieval. In *The World Wide Web conference* (pp. 1039–1050).

Liang, S., & de Rijke, M. (2016). Formal language models for finding groups of experts. *Information Processing & Management*, *52*(4), 529–549.

Liu, J., Jia, B., Xu, H., Liu, B., Gao, D., & Li, B. (2017). A topicrank based document priors model for expert finding. In *Advanced computational methods in life system modeling and simulation* (pp. 334–341). Springer.

Liu, Z., Li, K., & Qu, D. (2017). Knowledge graph based question routing for community question answering. In *International conference on neural information processing* (pp. 721–730). Springer.

Lopes, T., Ströele, V., Braga, R., David, J. M. N., & Bauer, M. (2021). Identifying and recommending experts using a syntactic-semantic analysis approach. In *2021 IEEE 24th international conference on computer supported cooperative work in design* (pp. 739–744). IEEE.

Mimno, D., & McCallum, A. (2007). Expertise modeling for matching papers with reviewers. In *Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 500–509).

Momtazi, S., & Naumann, F. (2013). Topic modeling for expert finding using latent Dirichlet allocation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *3*(5), 346–353.

Montandon, J. E., Silva, L. L., & Valente, M. T. (2019). Identifying experts in software libraries and frameworks among github users. In *2019 IEEE/ACM 16th international conference on mining software repositories* (pp. 276–287). IEEE.

Neshati, M. (2017). On early detection of high voted q&a on stack overflow. *Information Processing & Management*, *53*(4), 780–798.

Neshati, M., Fallahnejad, Z., & Beigy, H. (2017). On dynamicity of expert finding in community question answering. *Information Processing & Management*, *53*(5), 1026–1042.

Nikzad-Khasmakhi, N., Balafar, M., Feizi-Derakhshi, M. R., & Motamed, C. (2021). BERTERS: Multimodal representation learning for expert recommendation system with transformers and graph embeddings. *Chaos, Solitons & Fractals*, *151*, Article 111260.

Nobari, A. D., Neshati, M., & Gharebagh, S. S. (2020). Quality-aware skill translation models for expert finding on StackOverflow. *Information Systems*, *87*, Article 101413.

Petkova, D., & Croft, W. B. (2008). Hierarchical language models for expert finding in enterprise corpora. *International Journal on Artificial Intelligence Tools*, *17*(01), 5–18.

Rostami, P., & Neshati, M. (2019). T-shaped grouping: Expert finding models to agile software teams retrieval. *Expert Systems with Applications*, *118*, 231–245.

Rostami, P., & Neshati, M. (2021). Intern retrieval from community question answering websites: A new variation of expert finding problem. *Expert Systems with Applications*, *181*, Article 115044.

Song, J., Xu, X., & Wang, X. (2021). TSAR-based expert recommendation mechanism for community question answering. In *2021 IEEE 24th international conference on computer supported cooperative work in design* (pp. 162–167). IEEE.

Sun, J., Zhao, J., Sun, H., & Parthasarathy, S. (2021). EndCold: an end-to-end framework for cold question routing in community question answering services. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence* (pp. 3244–3250).

Tang, W., Lu, T., Li, D., Gu, H., & Gu, N. (2020). Hierarchical attentional factorization machines for expert recommendation in community question answering. *IEEE Access*, *8*, 35331–35343.

Torkzadeh Mahani, N., Dehghani, M., Mirian, M. S., Shakery, A., & Taheri, K. (2018). Expert finding by the Dempster-Shafer theory for evidence combination. *Expert Systems*, *35*(1), Article e12231.

Van Gysel, C., de Rijke, M., & Worring, M. (2016). Unsupervised, efficient and semantic expertise retrieval. In *Proceedings of the 25th international conference on World Wide Web* (pp. 1069–1079).

Wan, Y., Chen, L., Xu, G., Zhao, Z., Tang, J., & Wu, J. (2018). Scsminer: mining social coding sites for software developer recommendation with relevance propagation. *World Wide Web*, *21*(6), 1523–1543.

Wang, J., Sun, J., Lin, H., Dong, H., & Zhang, S. (2017). Convolutional neural networks for expert recommendation in community question answering. *Science China. Information Sciences*, *60*(11), 1–9.

Wei, W., Cong, G., Miao, C., Zhu, F., & Li, G. (2016). Learning to find topic experts in Twitter via different relations. *IEEE Transactions on Knowledge and Data Engineering*, *28*(7), 1764–1778.

Xu, Z., & Ramanathan, J. (2016). Thread-based probabilistic models for expert finding in enterprise microblogs. *Expert Systems with Applications*, *43*, 286–297.

Yang, L., Qiu, M., Gottipati, S., Zhu, F., Jiang, J., Sun, H., & Chen, Z. (2013). Cqarank: jointly model topics and expertise in community question answering. In *Proceedings of the 22nd ACM international conference on information & knowledge management* (pp. 99–108).

Zhang, X., Cheng, W., Zong, B., Chen, Y., Xu, J., Li, D., & Chen, H. (2020). Temporal context-aware representation learning for question routing. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (pp. 753–761).

Zhao, Z., Yang, Q., Cai, D., He, X., & Zhuang, Y. (2016). Expert finding for community-based question answering via ranking metric network learning. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence* (pp. 3000–3006).

Zheng, W., Hou, H., Wu, N., & Sun, S. (2021). Bayesian belief network model using sematic concept for expert finding. In *International conference on knowledge science, engineering and management* (pp. 114–125). Springer.

Zhu, H., Cao, H., Xiong, H., Chen, E., & Tian, J. (2011). Towards expert finding by leveraging relevant categories in authority ranking. In *Proceedings of the 20th ACM international conference on information and knowledge management* (pp. 2221–2224).

Zhu, H., Chen, E., Xiong, H., Cao, H., & Tian, J. (2014). Ranking user authority with relevant knowledge categories for expert finding. *World Wide Web*, *17*(5), 1081–1107.