Jointly Learning Content-Network Representations for Collaborative Expert Discovery

Abstract

The success of community question answering (CQA) platforms is in part dependent on how well questions are matched with the best experts that can effectively answer them in a timely manner. As the complexity of questions on CQA platforms increases, necessitating the collaboration of multiple experts, existing expert-finding methods are inadequate because they are designed to evaluate the suitability of individual experts for specific questions rather than collaborative expertise. This paper introduces a novel approach to identify collaborative teams of experts by employing graph neural networks and kernel pooling techniques, trained end-to-end. Our approach not only predicts potential individual experts but also forms teams that collectively enhance the answer quality. Through extensive experiments on real-life datasets, we show our proposed model is able to (1) find more qualified experts for new questions by at least 4.4% and 6.7% superior ranks in terms of ranking metrics `NDCG` and `MAP`, respectively, compared to the best expert finding baselines; (2) form a collaborative team of experts with 4.7% higher skill coverage than collaborative expert finding baselines.

1 Introduction

Community-based question answering (CQA) platforms are active communities of users who provide high-quality answers to questions posted by other users. The success of such communities can be significantly influenced by their ability to match questions with users who possess the appropriate expertise to provide answers. The research literature offers rich approaches to effectively pair experts and questions, and as such, *expert finding* or *question routing* has become a popular topic in the past few years [1–4]. Existing expert finding approaches typically rely on the textual similarity between newly posted questions and the content previously generated by experts. Additionally, these approaches may leverage the social network connections of experts to enhance the quality and relevance of the identified experts [5–7].

However, despite these advances, a substantial number of questions on CQA platforms remain unanswered or without satisfactory answers. For example, as of November 2023, Stack Overflow had approximately 3.4 million unanswered questions

and over 11.7 million questions without an accepted answer[1]. This gap underscores the limitations of current expert finding systems, particularly when it comes to addressing complex inquiries that often require the input of multiple experts.

The complexity of many questions on platforms like Stack Overflow and Yahoo! is reflected in extended discussion threads, where the average length is 6.7 interactions [8]. Research indicates that extended discussions are more likely to lead to an accepted answer when answerers collaborate effectively [9]. Collaboration is most effective when experts with complementary skills work together, allowing them to tackle different aspects of complex questions and deliver higher-quality answers. [9]. These insights suggest a new approach to expert finding, emphasizing collaboration as a means to increase the likelihood of resolving complex questions.

In this paper, we tackle the problem of *collaborative expert finding for question answering*. We define 'collaboration' as a process where each expert's contribution to a discussion thread introduces new insights or perspectives that enrich the conversation. These contributions enable subsequent experts to refine, expand, or build upon the discussion, ultimately leading to a more comprehensive answer. Our objective is to assemble a group of experts who, through their complementary contributions, collectively address the various aspects of a complex question.

The collaborative expert finding task is defined by two key objectives: (1) *Skill Completeness*: The assembled team must collectively encompass all the skills required to effectively answer the question. (2) *Effective Collaboration*: The team members must contribute in a way that their insights enable other experts to refine and enhance the overall answer, leading to a comprehensive and well-rounded solution.

Two potential directions for addressing the *collaborative experts finding task* include: **1. Top-k Expert Selection:** A straightforward approach is to use the top-k experts identified by traditional expert-finding methods [6, 10]. These methods leverage both textual information (e.g., question titles, bodies, tags) and topological data (e.g., relationships between questions, tags, and answerers) to retrieve experts relevant to newly posted questions [1, 2, 6, 10]. However, this approach does not account for the quality of past collaborations among experts. It primarily ranks experts based on individual suitability for the question, without considering how well these experts might work together as a team (*Challenge 1*). **2. Graph-Based Team Formation:** Another approach is to adopt existing graph-based team formation techniques, which identify teams from expert networks by maximizing skill coverage and minimizing collaboration costs [11–14]. However, they are not well-suited to our task for two main reasons. First, most graph-based methods require a pre-defined set of skills as input, which is impractical in the dynamic environment of CQA platforms where skills are latent and continuously evolving (*Challenge 2*). Second, these methods are computationally expensive and often only explore limited portions of the graph, leading to sub-optimal teams (*Challenge 3*). As a result, they are unlikely to effectively address the collaborative expert finding task in a scalable and efficient manner.

To address these challenges, in this paper, we propose a neural embedding-based method for *collaborative experts finding* in the context of community question answering. Our method aspires to find a team of experts to answer a given question such

[1] https://data.stackexchange.com/stackoverflow/queries

that the experts have a high likelihood to collaborate effectively and collectively cover all of the required skills for a new question. Our main contributions are as follows:

- **Joint Embedding of Content and Network for Collaborative Quality:** We propose a novel neural embedding-based framework that jointly learns representations of both content (questions, answers) and network topology (expert interactions) in an end-to-end manner. This joint learning captures the quality of past collaborations among experts, ensuring that the recommended teams have a higher likelihood of effective collaboration, directly addressing *Challenge 1*.
- **Dynamic Skill Representation without Predefined Skill Sets:** Our framework dynamically learns latent skill representations from the continuous interactions and content on the CQA platform. By eliminating the need for pre-defined skills, our method remains effective in the constantly evolving environment of CQA platforms, directly addressing *Challenge 2*.
- **Efficient and Scalable Team Formation:** The proposed framework maps the CQA network and its textual data into an efficient embedding space, significantly reducing computational complexity. This enables scalable identification of expert teams, overcoming the limitations of existing graph-based methods in terms of efficiency and effectiveness, thus addressing *Challenge 3*.
- **Enhanced Expert Team Recommendation:** Our approach extends traditional expert finding by recommending teams rather than individuals, with a focus on maximizing skill coverage and collaboration potential. This ensures that the assembled teams are knowledgeable and capable of working well together.

2 Related Work

Expert Finding Techniques. Expert finding studies can be classified into three categories according to the types of information sources they utilize from a CQA platform, i.e., content-based, network-based, and hybrid methods. *Content-based methods* utilize the textual content of each expert generated in the past to recommend the best answerers for a newly posted question. The studies usually model the task of expert finding as a document ranking problem and solve it [15, 16]. *Network-based methods* utilize the topological information extracted from the relationships among entities in the environment [17, 18]. *Hybrid methods* combine content-based and network-based methods to improve the performance of the expert finding systems [19, 20]. Traditional expert-finding methods rely on hand-crafted features, which limit their understanding of question semantics. Recent advancements in deep learning have improved these systems by enabling a better grasp of question meaning. For example, Peng et al. [4] proposed a multi-view matching method for expert finding, called `PMEF`, that learns question features from their title, body, and tag views, instead of relying only on one view; then it integrates different view information by a personalized attention network. Liu et al. [21] proposed an efficient non-sampling learning model that works based on whole data instead of negative sampling for the task of expert finding. Peng et al., [22] introduced the Hierarchical Matching network (EFHM), which features word and question-level match encoders to capture fine-grained semantic matching between historical answers and target questions, along with an expert-level match encoder to learn an overall expert feature for matching the target question. Despite these studies

that only utilizes textual content of questions, Sun et al. [3] proposed a method, called `EnC` here, which constructs a heterogeneous network using questions, tags, askers, and answerers. Then, a graph convolutional network is employed to learn the embeddings of the four types of entities in an end-to-end fashion by utilizing only the topological information. Li et al. [2] proposed a method, called `NeR` here, which applies a heterogeneous information network (HIN) embedding model to embed question content, their askers, answerers, and their relationships into the same latent space. Then, such latent representations of the three entities are fed into a convolutional scoring function to rank existing answerers for a new question. The main differences between our method and previous deep-learning studies are: (1) Our method jointly learns latent representations of question and answer content and the relationships of entities on the CQA platform, while others typically separate topological and textual embeddings. This joint learning enriches embedding vectors, enhancing the model's understanding of expert knowledge and community connectivity. (2) Our end-to-end framework continuously optimizes latent representations based on predicted expert ranks, directly aligning with real-world recommendation needs, unlike other methods with separate optimization stages.

Recently, researchers explore pre-training's potential for expert finding [6, 23]. Peng et all. [23] proposed an expert-level pre-training paradigm that integrates expert interest and expertise simultaneously. This approach incorporated historical answered question titles and vote score information to capture expert representations comprehensively. They extended their work in [24] by introducing personalized information integration and a more fine-grained expert pre-training architecture, enhancing the model's ability to capture unique expert representations and interests. As another work, Peng et al. [25] proposed a title-body contrastive learning task during pre-training to improve question representations by leveraging the semantic relationship between question titles and bodies. These approaches for expert finding in CQA platforms overlook expert collaboration, making it challenging to provide qualified answers, especially for multi-disciplinary questions. The research done in [9, 26], are among the few studies that aims to route a new question to a small team of experts who collaboratively work with each other to answer the question. Our work stands out by estimating expert collaboration and skill alignment using embeddings for experts, questions, and tags. Experts with more interactions or common tags receive higher similarity scores due to closely aligned embeddings.

Team Formation Techniques. Another line of research related to our work is the problem of finding a team of experts from expert networks using graph-based search techniques which has received a lot of attention in recent years [12, 27]. For example, Lappas et al. [11] proposed a method, called `CC` here, to find a team while maximizing the collaboration level among the members. Khan et al. [12] proposed approximation algorithms, called `CS`, to form compact groups in a way that members are closely connected and each one owns as many required skills as possible. Kargar et al. [27] designed a method, named `CO`, to find a team while maximizing the collaboration level among team members and their expertise level by considering the problem of group discovery over weighted node-labeled network graphs. Recently, there has been an interest in the problem of finding a team of experts from expert networks using

neural-based approaches. Sapienza et al.'s work [28] pioneers the use of neural architectures for team formation, employing an autoencoder design for faster computation. However, this approach is susceptible to overfitting, leading to less-than-optimal performance, especially given the sparse nature of collaboration networks. Etemadi et al. [29] proposed team2box (`t2b`), a system that creates expert teams to answer specific questions by using neural embeddings. These embeddings consider question content, expert engagement, and past collaboration history. Rad et al. [14] proposed a neural network architecture, going beyond simple mappings between skill and expert nodes. Their variational Bayesian neural network mines teams with past collaborative history, ensuring coverage of the required skills.

Despite the great success in team formation methods, these methods are computationally expensive (is NP-hard in practice), and requires heuristic-based approximations that lead to sub-optimal teams due to how subgraphs are explored locally. Our method overcomes such limitations by mapping the CQA network graph and its textual data into an efficient embedding space.

3 Problem Formulation

Let $\mathcal{Q} = \{q_1, q_2, ..., q_n\}$ be a set of n questions, and $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ be a set of m users in a CQA platform. For each question q_i, let $ak_i \in U$ be its asker. Assume that there exist n_i answers as $A_i = \{a_1, a_2, .., a_k, .., a_{n_i}\}$ where answer a_k is provided by expert $e_k \in U$ with quality (voting) scores of s_k. Score s_k is a value calculated based on the difference between answer a_k's up-votes and down-votes which is assigned by users who viewed the answer. Further, let $Tg_i = \{t_1, t_2, .., t_z\}$ be a set of tags of question q_i assigned by its asker. Terms *expert* and *answerer* are used interchangeably in the rest of the paper. Given new question q, our task is to retrieve a ranked list of k potential experts from U such that top-ranked expert(s) satisfy three Objectives (O):

O1) Each highly ranked expert has a high probability of giving a qualified answer. Suppose $\mathcal{R} : \mathcal{U} \rightarrow \mathbb{N}_1$ be a rank function learned to return the ranks of the answerers for question q. Given two answerers e_i and e_j with $s_i \geq s_j$, we have: $\mathcal{R}(e_i) \geq \mathcal{R}(e_j)$, where $\mathcal{R}(e)$ is a positive integer indicates the rank of e;

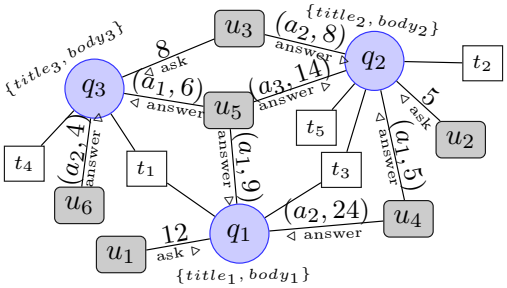
O2) The top-k experts as a team have all the required complementary skills to answer the question. That is, the content similarity between new question q and the answers given by the experts in the team set T to past questions should be maximized. Given question q, and two teams T_m and T_n, T_m is preferred iff:

$$\sum_{a_i \in A^{\mathrm{T}_m}} sim(q, a_i) \geq \sum_{a_j \in A^{\mathrm{T}_n}} sim(q, a_j), \tag{1}$$

where function $sim(q, a)$ is the content similarity between question q and answer a, and A^{T} is a set of past answers written by members of team T.

O3) The top-k experts as a team have exhibited a high likelihood to collaborate with each other. In other words, there is a high tendency for each pair of experts in the team to collaborate with each other:

$$\sum_{e_i, e_j \in \mathrm{T}_m} \mathtt{P}_{\mathtt{cl}}(e_i, e_j) \geq \sum_{e_k, e_l \in \mathrm{T}_n} \mathtt{P}_{\mathtt{cl}}(e_k, e_l), \tag{2}$$



Teams	Collaboration	Teams	Collaboration
$\{u_3, u_4\}$	1	$\{u_3, u_5\}$	1
$\{u_3, u_6\}$	0	$\{u_4, u_5\}$	2
$\{u_4, u_6\}$	0	$\{u_5, u_6\}$	1

Fig. 1 A CQA heterogeneous network comprises three node types: questions (q), tags (t), and users (u), where users can be either askers or answerers. The edge weight $q-e$ indicates an answer's score, for instance, $(a_3, 14)$ means the answer a_3 by user u_5 received 14 votes. Consider forming a team of two; past collaboration data shows that experts u_3 and u_4 jointly answered one question, indicating a collaboration score of 1. The optimal team, based on highest collaboration, is $\{u_4, u_5\}$.

where function $\mathtt{P}_{\mathtt{cl}}$ quantifies the past collaboration level between a pair of experts. We will present an example of $\mathtt{P}_{\mathtt{cl}}$ in Section 5.4.

4 Proposed Framework

In this section, we first provide an overview of our framework for collaborative expert finding. Then, we explore its various components.

4.1 Overview

To retrieve a ranked list of k potential experts for a qiven question q, we model a CQA platform as an undirected heterogeneous network in which questions, tags, askers, and answerers are nodes of the network, and their relationships are considered as the edges of the network. Note that, as we will use this network for learning the embedding representations, the network is modelled as an undirected graph. This allows the proposed framework to capture deeper relationships among entities in the network while a directed network would limit the learning model to only one directional relationship (e.g., from expert to question or vice versa) among the entities. Formally, the CQA network is defined as follows:

CQA Heterogeneous Network is denoted as $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{T})$ where $\mathcal{V}$ is a set of nodes, $\mathcal{E}$ a set of edges, and $\mathcal{T}$ is the set of node and edge types. Furthermore, let $\mathcal{T}_{\mathcal{V}}(\subset \mathcal{T})$ denote node types, which can be *questions (q)*, *tags* (t), and *users* (u) which can be *askers* (ak), or *answerers* (e). Similarly, $\mathcal{T}_{\mathcal{E}}(\subset \mathcal{T})$ indicates a set of edge types which can be *question-tag (q—t)*, *question-asker (q—ak)*, or *question-answerer (q—e)* relationships. A toy example of a CQA network with three questions, five tags, and six users (three askers and four answerers) is depicted in Fig. 1. Note that a user can be both an asker and an answerer. Each question node has two attributes, i.e., title and body. Furthermore, each question-answerer edge has two attributes, namely the answer and the answer score. As an example in Fig. 1, answer a_2 with a voting score of 24 is provided by answerer u_4 for question q_1.

Our proposed framework is illustrated in Fig. 2. Given a question, we employ the topological and textual information of the entities in the CQA network to predict the ranking scores of the answerers. Our framework encodes topological information through node embeddings and textual content of nodes' attributes through soft term

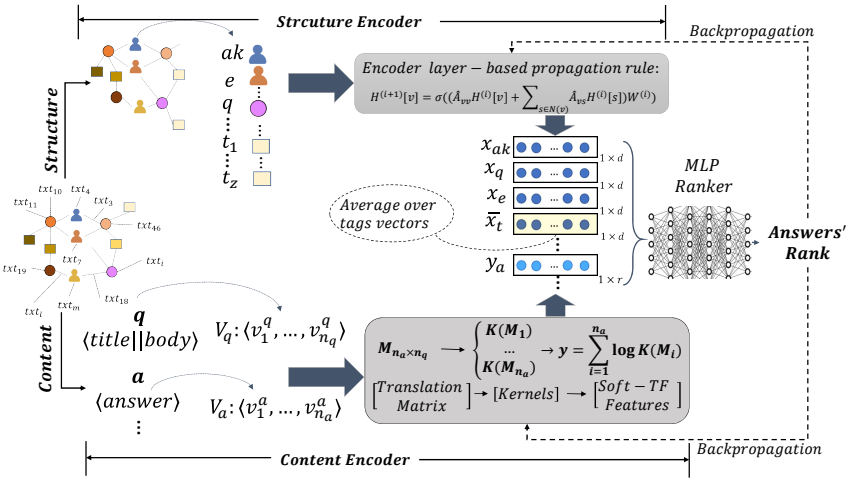


Fig. 2 The proposed framework overview

frequency (TF) features. Then, such latent representations are fed into a multilayer perceptron (MLP) based ranker to predict the ranking scores of the answerers.

4.2 Key Components of the Framework

Our framework consists of three main components: (1) **Structure encoder** that maps a sub-graph of the CQA network into vector space. It receives a sampled node of type question and all its neighbours as subgraph g and uses a graph neural network to produce the embeddings of nodes in g named $\mathbf{X} = (\mathbf{x}_1, .., \mathbf{x}_{n_g}) \in \mathbb{R}^{n_g \times d}$ where n_g is the number of nodes in g and d is the domination of embedding vectors $\mathbf{x}$; (2) **Content encoder** that captures the content similarity between a question and its answers. To do so, a kernel pooling function is employed to capture such text similarities and generate Soft-TF feature vectors denoted by $\mathbf{Y} = (\mathbf{y}_1, .., \mathbf{y}_z) \in \mathbb{R}^{z \times r}$, where z can be the number of edges of question-expert type in g and r is the dimension of Soft-TF feature vectors $\mathbf{y}$; (3) **MLP Ranker**, which receives the topological features, i.e. embedding vectors $\mathbf{X}$, and textual information, i.e. embedding vectors $\mathbf{Y}$, and predicts the ranking scores of the answerers. The model is trained end-to-end. For a new question, its asker, and tags, a subgraph g is constructed with potential answerers. The edges in g are attributed with the content of the experts' past answers. Finally, the predicted scores for all answerers are employed to rank the experts.

4.2.1 Structure Encoder

The structure encoder captures the relationships between the entities in the CQA platform as the nodes of the CQA network. It preserves the higher-order proximity among the entities, i.e. questions, their tags, askers and answerers, in the embedding space. This latent representation is utilized to discover potential answerers to new questions. The encoder works as follows. Suppose $\mathcal{A}$ is the adjacency matrix of undirected CQA graph $\mathcal{G}$, and $g \subseteq \mathcal{G}$ is a subgraph of the CQA network $\mathcal{G}$ with a node of question type and all of its neighbours. In general, the latent representation of node $v \in g$ is obtained

by the encoder based on the representations of its neighbouring nodes as follows:

$$H^{(i+1)}[v] = \underset{\forall s \in N(v),(s,v)\in\mathcal{E}}{\texttt{Aggregate}} \left(\texttt{Extract}(H^{(i)}[s], H^{(i)}[v])\right), \tag{3}$$

where $H^{(i)}[v]$ is the latent representation of v in the ith layer of the structure encoder ($i = 0, 1, .., h$), $H^{(0)}$ is initialized by the identity matrix, and $N(v)$ is a set of neighbor nodes of v. The function `Extract`(.) gathers useful information from the latent representation of each neighbor of v, and `Aggregate` is an operator that combines the extracted information from the neighborhood of v. We define the encoder as:

$$H^{(i+1)}[v] = \sigma\left(\left(\sum_{s\in N(v)} H^{(i)}[s]\right) W^{(i)}\right), \tag{4}$$

where $W^{(i)}$ is the matrix of trainable weights in layer i and σ is a non-linear activation function. Eq. 4 extracts the latent representation of neighbours of v and utilizes the summation operator as an aggregate function to obtain embedding of node v in layer $i+1$ of the network. It also applies convolution operators $W^{(i)}$ to transform features into a specific space with a given dimension and adds non-linearity into the layer using σ. However, Eq. 4 suffers from two problems. First, for each node, it only considers the feature vectors of all its neighbours and overlooks the features of the node itself. Thus, the feature vector of node v should be added to the summation of its neighbours' feature vectors. In other words, a self-loop should be added into the adjacency matrix A to consider each node in its neighbour set, i.e., $\tilde{\mathcal{A}} = \mathcal{A} + I_N$, where I_N is the identity matrix. Applying such remedy in Eq. 4, we obtain:

$$H^{(i+1)}[v] = \sigma\left(\left(H^{(i)}[v] + \sum_{s\in N(v)} H^{(i)}[s]\right) W^{(i)}\right). \tag{5}$$

Second, the summation operator, as an aggregate function, will completely change the scale of embedding vectors in layer $i+1$, i.e., $H^{(i+1)}$. To resolve the problem, the feature vectors in layer i can be normalized based on the importance of the neighbour nodes which can be inferred using node degrees. Thus, matrix $\tilde{\mathcal{A}}$ should be normalized based on node degrees as: $\hat{\mathcal{A}} = \tilde{D}^{-0.5}\tilde{\mathcal{A}}\tilde{D}^{-0.5}$, where $\tilde{D}$ is the degree matrix of $\tilde{\mathcal{A}}$ and its diagonal elements is obtained as:

$$\tilde{D}_{ij} = \begin{cases} \sum_{k=1}^{N} \tilde{A}_{ik} & i == j, \\ 0 & otherwise \end{cases} \tag{6}$$

where $\tilde{A}_{ij}$ is the element of the i^{th} row and j^{th} column in matrix $\tilde{\mathcal{A}}$. Finally, the latent representation of node $v \in g$ is obtained by:

$$H^{(i+1)}[v] = \sigma\left(\left(\hat{\mathcal{A}}_{vv}H^{(i)}[v] + \sum_{s \in N(v)} \hat{\mathcal{A}}_{vs}H^{(i)}[s]\right) W^{(i)}\right). \tag{7}$$

Suppose n_g be the number of nodes in subgraph g and their labels as v_1, .., v_{n_g}. We obtain a list of embedding vectors of nodes in g as $\mathbf{X} = (H^{(h)}[v_1], H^{(h)}[v_2], .., H^{(h)}[v_{n_g}])$ where $\mathbf{X} \in \mathbb{R}^{n_g \times d}$ and d is the embedding dimension. The trainable parameters in the structure encoder, i.e. $W^{(0)}, W^{(1)}, ..., W^{(h)}$, are randomly initialized and learned in an end-to-end way in parallel with the parameters of the content encoder. The training details will be explained later in Section 4.3. Other parameters, i.e. $\tilde{\mathcal{A}}$, $\tilde{\mathcal{D}}$, and $\hat{\mathcal{A}}$, are computed based on the adjacency matrix $\mathcal{A}$ of the CQA network graph $\mathcal{G}$.

4.2.2 Content Encoder

In parallel with the structure encoder, the textual information extracted from g is used to learn soft TF features. Let l be textual contents as txt_1, txt_2, .., txt_l extracted from the attributes of nodes and edges of g. Given txt_i and txt_j as a pair of such texts, the similarity between them is translated into soft-TF features. Given txt_i and txt_j with n_i and n_j words, respectively, let $\mathbf{V}_i = (\mathbf{v}_1^i, \mathbf{v}_2^i, ..., \mathbf{v}_{n_i}^i)$ and $\mathbf{V}_j = (\mathbf{v}_1^j, \mathbf{v}_2^j, ..., \mathbf{v}_{n_j}^j)$ be lists of embedding vectors of words in txt_i and txt_j, where $\mathbf{V}_i \in \mathbb{R}^{n_i \times d'}$, $\mathbf{V}_j \in \mathbb{R}^{n_j \times d'}$ and d' be the embedding dimension of words. We let differentiable function Ψ be used to obtain the soft-TF feature vector of pair (txt_i, txt_j) as:

$$\mathbf{y}_k = \Psi(\mathbf{V}_i, \mathbf{V}_j). \tag{8}$$

Let $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, .., \mathbf{y}_z)$ be a list of z soft-TF vectors extracted from the textual similarities among l contexts as txt_1, txt_2, .., txt_l and $\mathbf{Y} \in \mathbb{R}^{z \times r}$ where r is the dimension of Soft-TF features vectors. Note that only the pair of related texts (e.g. question title or body with one of its answers) are encoded in list $\mathbf{Y}$. We utilize a neural learn-to-ranking method based on *kernel pooling* as function Ψ to capture the textual similarity between questions and their answers as follows. First, embeddings of words in q and its answer a are utilized to compute the translation matrix $\mathbf{M}$ as:

$$M_{ij} = \frac{\mathbf{v}_i^a \mathbf{v}_j^q}{||\mathbf{v}_i^a||.||\mathbf{v}_j^q||} \tag{9}$$

where vectors $\mathbf{v}_i^a$ and $\mathbf{v}_j^q$ are the embedding vector of the i^{th} word in answer a and the j^{th} word in question q, respectively. Note that such word embedding vectors are learned in an end-to-end fashion during the training phase. Element M_{ij} demonstrates the similarity between the embedding vector of the i^{th} word in answer a and the j^{th} word in question q. We note that row i in matrix $\mathbf{M} \in \mathbb{R}^{n_a \times n_q}$ denoted by $\mathbf{M}_i$ captures the similarity between the i^{th} word of answer a and all the words in q. To reduce the size of $\mathbf{M}$, r kernels denoted by K are applied on $\mathbf{M}_i$ $(i = 1, 2, ..., n_a)$ to obtain an

r-dimensional feature vector. The log-sum of such n_a feature vectors is computed to obtain the similarity between answer a and question q as the soft-TF feature vector $\mathbf{Y}$. The effectiveness of the soft-TF features depends on the kernels used in the model. We adopt the RBF kernel due to its performance reported in the literature [30], and for being differentiable. Assume a kernel vector $\mathbf{K}$ with r kernels as $\{K_1, K_2, ..., K_r\}$, the RBF kernel K_k $(k = 1, 2, ..., r)$ is applied on the i^{th} row of the translation matrix $\mathbf{M}$, i.e., $\mathbf{M}_i$, as follows:

$$K_k(\mathbf{M}_i) = \sum_{j=1}^{n_q} exp\left(-\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2}\right). \tag{10}$$

where μ_k and σ_k are the parameters of kernel K_k. By applying r kernels on the i^{th} row of the translation matrix $\mathbf{M}$, we get a vector with r values as:

$$\mathbf{K}(\mathbf{M}_i) = \{K_1(\mathbf{M}_i), K_2(\mathbf{M}_i), ..., K_r(\mathbf{M}_i)\}.$$

Given matrix $\mathbf{M}$ with n_a rows, we end up with n_a vectors as $\mathbf{K}(\mathbf{M}_1), \mathbf{K}(\mathbf{M}_2), ..., \mathbf{K}(\mathbf{M}_{n_a})$. The soft-TF feature vector $\mathbf{y}_a$ is computed as:

$$\mathbf{y}_a = \sum_{i=1}^{n_a} log\ \mathbf{K}(\mathbf{M}_i). \tag{11}$$

where $\mathbf{y}_a \in \mathbb{R}^r$. Such soft-TF vectors are computed for pairs of questions and answers.

4.2.3 The MLP Ranker

The MLP ranker predicts the ranking scores of answerers extracted from question-expert relations in g. We construct the input vector $\mathbf{X}_{in}$ with fixed size by concatenating the vectors in $\mathbf{X}$ and $\mathbf{Y}$. Such feature vector $\mathbf{X}_{in}$ is fed into an MLP network to predict the ranking scores of answerers given the question sampled in subgraph g as:

$$\mathbf{R}(\mathbf{X}_{in}) = \sigma^{(l)}\left(\sigma^{(l-1)}(...\sigma^{(0)}(\mathbf{X}_{in}\theta^{(0)} + b^{(0)})...)\theta^{(l)} + b^{(l)}\right). \tag{12}$$

where $\sigma^{(l)}$, $\theta^{(l)}$, and $b^{(l)}$ are the non-linear activation function, the trainable weights matrix, and the bias vector of hidden layer l, respectively.

4.3 Model Training

The main steps of each epoch of training in our framework are summarized in Alg. 1.

Algorithm 1: Training the model

Input: $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{T})$
Output: $\theta, b, \mathbf{V}, \mathbf{W}^{(0)}, \ldots, \mathbf{W}^{(h)}$
begin
  foreach *node* q *of type question in* $\mathcal{G}$ **do**
    Construct sub-graph g using all neighbors of q
    foreach *node* v *in* g **do**
      for $i \leftarrow 0$ **to** $h-1$ **do**
        $H^{(i+1)}[v] = \sigma\left(\left(\hat{\mathcal{A}}_{vv} H^{(i)}[v] + \sum_{s \in N(v)} \hat{\mathcal{A}}_{vs} H^{(i)}[s]\right) W^{(i)}\right)$
      end
    end
    $\overline{\mathbf{x}}_t = \frac{1}{|tg|} \sum_{t \in tg} H^{(h)}[t], tg = \{t \in g \mid \mathcal{T}_{\mathcal{V}}(t) == tags\}$
    $A_q = \{u \in g \mid \mathcal{T}_{\mathcal{V}}(u) == experts\}$
    $ak = u \in g$ where $\mathcal{T}_{\mathcal{V}}(u) == asker$
    foreach $u \in A_q$ **do**
      a_u = answer of u extracted from edge (q, u)
      s_u = score of answer a_u
      $\mathbf{y}_{a_u} = \Psi(\mathbf{V}_q, \mathbf{V}_{a_u})$ `// Content encoder`
      $\hat{s}_u = \mathbf{R}(H^{(h)}[q] \parallel H^{(h)}[ak] \parallel \overline{\mathbf{x}}_t \parallel H^{(h)}[u] \parallel \mathbf{y}_{a_u})$
    end
    Minimize $\mathcal{L}(\theta, b, \mathbf{V}, \mathbf{W}^{(0)}, \ldots, \mathbf{W}^{(h)}) = \frac{1}{|A_q|} \sum_{u \in A_q} (\hat{s}_u - s_u)^2$
  end
end

First, for each node q of type question in $\mathcal{G}$, sub-graph g is constructed by randomly selecting q, and all its neighbour nodes and edges among them (Line 3). Then, the proposed structure encoder embeds topological properties of g (Lines 4-8). Then, topological information encoded in the embeddings of nodes is utilized along with textual information to predict the ranking score of each answerer of q (Lines 12-18). To do so, for each answerer u of question q, content encoder Ψ extracts the soft-TF features between the content of q and answer a_u written by u, i.e. $\mathbf{y}_{a_u}$ (Line 15). Then, input vector $\mathbf{X}_{in}$ is constructed by concatenating embeddings of question q, its asker, tags, answerer u, and soft-TF feature vector $\mathbf{y}_{a_u}$, and fed into the model (Line 16). Let tg be a set of nodes of type tag in g. Since the size of tg is not the same for all questions and given the fact that the MLP ranker accepts a fixed-sized input $\mathbf{X}_{in}$, we obtain the average of embeddings of nodes in tg (Line 9) and utilize it in the input vector. Finally, the MLP ranker predicts the ranking scores of answerer u of question q given the topological and textual information encoded in $\mathbf{X}_{in}$. Then, the mean square error of predicted ranking scores of answerers and their answers' voting scores are employed as a loss function to learn the trainable parameters of the model (Line 18). The loss function for all questions in the training set is computed as:

$$\mathcal{L}(\theta, b, \mathbf{V}, \mathbf{W}^{(0)}, .., \mathbf{W}^{(h)}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{n_i} \sum_{u=1}^{n_i} (\hat{s}_u - s_u)^2. \tag{13}$$

where $\hat{s}_u$ and s_u are the predicted ranking score of answerer u and the true voting score of her answer a_u given question q_i with n_i answers, and n is the total number of questions. The training parameters are the ranking parameters θ, and b, the word embedding vectors $\mathbf{V}$ and the weight matrices $\mathbf{W}^{(0)}$,..,$\mathbf{W}^{(h)}$ of the structure encoder. The trainable parameters are learned using back-propagation through the network by optimizing Eq. 13. The details of parameter training is discussed in Appendix A.

Table 1 Statistics of the datasets after cleaning.

Datasets	$\|\mathcal{Q}\|$	Answers		Words		Tags	Experts	Users	Teams
		All	Score >=4	Original	After Cleaning				
android	882	3,728	2,136	6,862,586	205,154	510	1,018	1,511	857
history	1,697	6,839	4,702	5,447,670	916,589	567	1,164	1,838	1,575
dba	2,906	9,357	6,976	27,783,124	1,171,967	578	1,794	3,475	2,554
physics	4,912	19,475	12,559	31,327,261	2,142,405	717	2,938	5,636	4,613

4.4 Predicting Expert Ranks

The trained model recommends potential answerers given new question q. Assume ak and t_1,..., t_z to be the asker and tags of question q, respectively. Our framework predicts the relevance score of each user u in the CQA platform as follows. First, the structure encoder produces the embedding vectors of nodes q, ak, u, t_1, ..., t_z. Note that the encoder returns zeros for the embeddings of nodes that are not in $\mathcal{G}$ (e.g., node q). Concurrently, the content encoder captures the textual similarity between the past content generated by user u and new question q, and generates soft-TF features. Let $\mathbf{X}_u$ and $\mathbf{Y}_u$ be the encoded topological and textual information given question q and user u. Then, the MLP ranker predicts the relevance score of user u given the topological and textual data encoded by the structure and content encoders: $\hat{s}_u = \mathbf{R}(\mathbf{X}_u||\mathbf{Y}_u)$, where $\hat{s}_u$ is the ranking score of user u given question q. This process is repeated for all experts to obtain their scores, after which they are ranked.

5 Experiments

We carried out extensive experiments on several real-world datasets from a variety of domains to investigate the performance of the proposed method. Our code, pre-processed datasets, and results are publicly available[2].

Datasets. We conduct experiments using several real-world datasets from *Stack Exchange*, released in September 2019. In the pre-processing step, we removed all stop words and special characters and kept only questions with at least two answers. Additionally, answers with voting scores lower than four were eliminated. Table 1 summarizes the properties of the datasets before and after performing preprocessing. The statistics show that more than 80% of answers in each dataset have voting scores in the range of $[4, 20]$.

Baselines. We adopt two sets of baselines: **(1) Expert Finding Techniques**: We evaluate the proposed method against state-of-the-art expert finding techniques, including `PMEF`[4] that incorporates different view textual information for expert and question learning in a personalized way, and `NeRank` (`NeR`) [2] and `EndCold` (`EnC`) [3] which use network topology for node representation and then the textual information of questions are leveraged (not in an end-to-end fashion) to predict the best answerer for a new question. Additionally, we apply neural-based learn-to-rank methods including DUET [31], CKNRM [32], KNRM [30], DSSM [33], ArcII [34], and ANMM [35] for expert finding. Such methods treat questions and their answers as queries and corresponding documents, employing neural ranking algorithms to prioritize the most relevant information. To apply these methods, we employ the default

[2]Anonymized due to the submission policy. Link will be added here.

parameters implemented in MatchZoo [36]. The methods were repeated five times on the datasets, and the best results were reported for each one. **(2) Team Formation Techniques**: We also compare the proposed method with four state-of-the-art team formation techniques: `CC` [11], `CO` [27], `CS` [12]. These methods consider different combinations of collaboration, expertise levels, or both, among team members and within their network connections to form teams of experts. It's worth mentioning that all these methods are heuristics-driven. We further consider recent neural based team discovery method, `t2b` [29]. This method uses neural embeddings to match experts and questions effectively, focusing on their compatibility and previous successful collaborations.

Experimental Setup. For the *neural-based methods*, 90% of each dataset is used as training and validation data, and the rest is used for testing. For the *expert finding task*, the methods are used to predict the relevance score of the answerers of each test question along with some random experts as noise. The number of such random experts is equal to the number of actual answerers to that question. Based on our datasets, the minimum number of potential answerers for a test question is four. Furthermore, for the *team formation methods*, all experts in each dataset are considered as potential answerers given a test question. Please See Appendix B.1 for further details.

Evaluation Metrics. Given the different nature of methods in expert finding and team formation, different sets of metrics are employed. **(1) Expert Finding**: Given a test question and a list of potential answerers, expert finding methods produce ranked lists of answerers in which the top-ranked answerer is the most suitable one to answer the question. Thus, popular ranking measures, such as *normalized discounted cumulative gain* (`NDCG`) and *mean average precision* (`MAP`), are utilized. Intuitively, the `NDCG`@k metric indicates how well the predicted ranked list of size k matches with the true ranks of the answerers. Similarly, `MAP`@k reveals the average portion of the top-k ranked experts among the actual answerers. In our experiments, the maximum value of search depth k is determined based on the maximum number of potential answerers of the test questions. Take android data as an example, we set $k = 1, 2, .., 5$ because the maximum number of answerers of test questions in this dataset is five (see details in Appendix B1). **(2) Team Formation**: Inspired by [29], we employ the following metrics in our evaluations: **(a)** The **Skill Coverage** (`SC`) metric is used to gauge how well the background knowledge of experts in a retrieved team aligns with the subject matter of a new question. **(b) Collaboration Level** (`CL`) reveals the willingness of the team members to collaborate as a team. Such a metric plays an important role in team success. **(c) Gold Standard Team Match** (`GM`) assesses how closely a discovered team matches the actual answerers of a question, who are considered the gold standard team. In the context of the Stack Exchange datasets, a gold standard team consists of users who have successfully answered a question marked as 'Answered'. Please see Appendix B.2 for further details on the metrics.

5.1 Comparison with Baselines

In this section, we compare our proposed framework with two sets of baselines: Expert Finding Techniques and Team Formation Techniques.

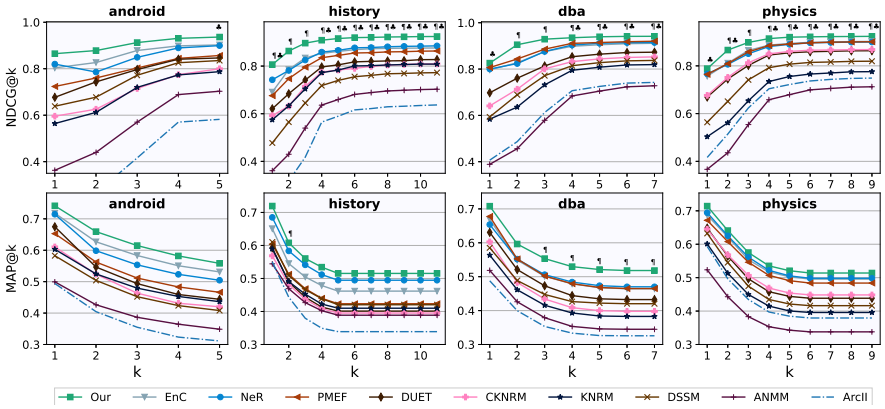


Fig. 3 Performance of the our method compared to expert finding and learn-to-rank techniques using `NDCG` and `MAP` at search depths k from one to the maximum number of answerers. Statistical significance is indicated by ¶ and ♣ when our method is significantly better than `EnC` and `NeR`, respectively, based on a paired t-test with a 95% confidence interval. Very small evaluation metric values for some k are omitted for clarity.

5.1.1 Comparison with Expert Finding Techniques

In this section, we evaluate our method's performance in expert finding by predicting relevance scores for answerers of each test question, including some random experts as noise. The gold standard is the answerer with the highest voting score, with other answerers ranked accordingly. Random noise experts receive a voting score of zero. We use common ranking measures, `NDCG` and `MAP`, to assess the results. The results achieved by the methods are reported in Fig. 3. Our observations are as follows:

- The expert ranks recommended by our approach are better aligned with their actual ranks compared to all the other baseline methods. The experiments reported in the first row of Fig. 3 indicate that the experts' ranking produced by our method achieves consistently higher ranking scores in terms of `NDCG` compared to those retrieved by the expert finding and learn-to-rank baselines. The empirical results reveal that our method generates on average 4.4% superior ranks in terms of `NDCG` compared to our best baselines, i.e. `EnC` and `NeR`, on all datasets;
- Our method effectively differentiates between relevant and irrelevant potential answerers for a new question. The results depicted in the second row of Fig. 3 demonstrate that our method stably outperforms the baselines in terms of `MAP`. `MAP` measures how well a method identifies true answerers among experts for a test question. Our experiments show that, on average, our method achieves 7.7% and 6.7% higher `MAP` scores across all datasets compared to `EnC` and `NeR`, our top baselines.
- The results depicted in Fig. 3 shows that our method consistently obtains similar `NDCG` and `MAP` regardless of the dataset. This shows the stability of our proposed approach compared to the other baselines. Furthermore, the superiority of our method does not degrade by employing different values for the search depth k.

Experiments show that our model's end-to-end learning of latent representations from both topological and textual data in the CQA environment significantly outperforms state-of-the-art expert-finding methods.

Table 2 Comparative Performance Analysis of Top Expert Finding Methods (NeR, Enc), Team Formation Methods (t2b, CC), and Our Method in Terms of Team Size as Determined by CC.

Datasets	$\overline{\tau}$	SC%					CL					GM%				
		CC	NeR	EnC	t2b	Our	CC	NeR	EnC	t2b	Our	CC	NeR	EnC	t2b	Our
android	4	78.7	53.0	78.8	85.6	**86.7**	3.5	0.9	7.1	6.2	**8.5**	15.6	4.2	7.6	14.2	**27.4**
history	4	89.4	83.9	91.2	93.7	**94.4**	9.6	9.1	**53.3**	28.9	14.4	14.4	7.7	13.5	17.0	**31.3**
dba	3	88.9	81.4	86.3	**92.1**	90.4	6.8	5.4	11.1	**21.0**	6.0	13.4	6.9	10.4	13.8	**38.1**
physics	3	92.7	87.2	91.3	**96.2**	95.1	6.2	16.9	**40.2**	22.4	36.0	9.5	5.7	8.8	12.1	**17.5**

Table 3 Comparative Performance Analysis of Top Expert Finding Methods (NeR, Enc), Team Formation Methods (t2b, CO), and Our Method in Terms of Team Size as Determined by CO.

Datasets	$\overline{\tau}$	SC%					CL					GM%				
		CO	NeR	EnC	t2b	Our	CO	NeR	EnC	t2b	Our	CO	NeR	EnC	t2b	Our
android	6	86.3	59.0	80.4	85.6	**90.5**	2.6	1.8	9.4	12.4	**16.8**	17.8	5.2	8.2	14.8	**29.2**
history	5	90.2	87.0	95.1	95.0	**96.6**	18.5	13.2	**88.0**	50.7	26.2	15.4	9.8	18.2	18.8	**39.8**
dba	6	92.8	88.8	92.9	93.5	**96.0**	6.4	8.7	28.9	**55.6**	19.3	14.3	10.4	14.7	17.4	**50.6**
physics	7	94.3	96.2	97.3	97.1	**98.0**	7.0	75.0	119.9	85.5	**125.6**	11.0	13.3	15.7	14.7	**26.8**

Table 4 Comparative Performance Analysis of Top Expert Finding Methods (NeR, Enc), Team Formation Methods (t2b, CS), and Our Method in Terms of Team Size as Determined by CS.

Datasets	$\overline{\tau}$	SC%					CL					GM%				
		CS	NeR	EnC	t2b	Our	CS	NeR	EnC	t2b	Our	CS	NeR	EnC	t2b	Our
android	8	85.0	68.0	82.9	86.4	**92.4**	2.9	3.4	10.3	21.0	**33.0**	20.3	8.5	10.6	16.0	**34.8**
history	8	95.2	91.5	97.6	95.1	**97.8**	23.7	16.2	**157.7**	84.1	44.8	19.8	12.1	23.9	21.6	**48.1**
dba	6	91.2	89.6	93.2	93.8	**95.8**	10.1	8.6	27.0	**55.0**	19.1	13.9	10.6	14.4	17.4	**51.8**
physics	10	91.5	97.5	98.0	97.3	**98.6**	3.7	144.0	**219.0**	122.0	212.6	9.7	17.6	20.3	15.6	**30.4**

5.1.2 Comparison with Team Formation Techniques

We now investigate the performance of our proposed method against team formation techniques. We compare our proposed method and the top two best performing expert finding baselines, i.e. EnC and NeR, against the team formation baselines. The results are reported in Tables 2, 3, and 4. The team size for CC, CO, CS, i.e. τ, is not the input parameter. Thus, they build teams with different sizes for test questions. As such, the sizes of teams discovered by the team formation methods are used as the number of experts that will be retrieved by EnC, NeR, t2b and our method. The average size of teams ($\overline{\tau}$) for each dataset is depicted in the second column of the tables. Note that the t2b method is a neural-based approach and takes the team size as the input. Therefore, it is run using the team sizes determined by CC, CO, and CS separately, with results reported in the tables. The results reported in the tables indicate that:

- The experts retrieved by our proposed method possess broader skills. As inferred from the table, the skill coverage, i.e., SC, of teams discovered by our method is on average 4.22%, 4.37%, 5.42%, and 2.5% higher than those constructed by team formation baselines CC, CO, CS, and t2b, respectively;
- The teams' members formed by our method enjoy a high willingness to cooperate with each other. The experiments indicate that the past collaboration level, i.e. CL, between the members of teams constructed by our approach is on average 2.4, 5.4, 7.6 times higher than the experts in teams created by CC, CO, and CS, respectively.

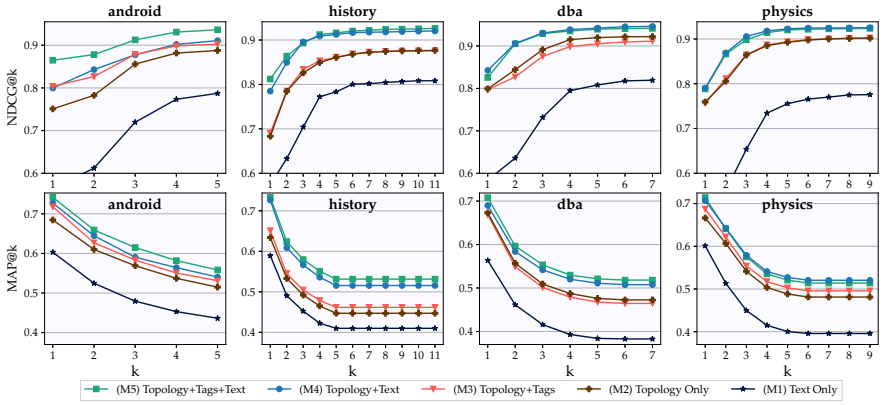


Fig. 4 Performance of the proposed method variations in terms of `NDCG` and `MAP` at different k.

- Our approach constructs more realistic teams. The `GM` metric reveals that our method involves the actual answerers of test questions in its discovered teams on average 2.4 times higher than the team formation baselines.
- Although `EnC` and `t2b` sometimes achieve higher `CL` scores, this does not equate to better overall team effectiveness, as shown by their lower `GM` scores. Higher `CL` suggests frequent past interactions, but the goal is to provide the best answers, thus higher `GM` score. Our method's significantly higher `GM` scores indicate that it forms teams that are both theoretically ideal and practically competent in addressing questions, even if `CL` scores are occasionally lower.

5.2 Ablation Study

Our framework depicted in Fig. 2 can have different variations. As such, in this section, we investigate the impact of using the textual and topological information extracted from the CQA platform in the performance of such variations. To do so, we consider different variations of the proposed framework as follows.

(M1) **Text Only**: This variation employs only the content of the questions (tags, title, and body) and their answers in the content encoder.

(M2) **Topology Only**: In this model, the CQA network is built by removing tags from the network and only the topological data are considered in the structure encoder.

(M3) **Topology+Tags**: This variation employs the topological network data and tag information of the CQA network.

(M4) **Topology+Text**: In this variation, we remove the tag nodes from the CQA network. As such, the network topological data is used in the structure encoder and the questions, their askers and answerers are used in the content encoder but tag information are not considered.

(M5) **Topology+Tags+Text**: This variation incorporates all the proposed elements of our proposed framework as described in Fig 1. The results of our ablation study are reported in Fig. 4. We summarize our observations as follows.

- Employing both textual and topological data extracted from a CQA system leads to more effective rankings and hence to the retrieval of more relevant experts. As depicted, the models that use both textual and topological data, i.e., M4 and M5, obtain superior results in terms of ranking measures, i.e., `NDCG` and `MAP`, in all

datasets. The results demonstrate that M5 achieves on average 4.4% and 21.6% higher NDCG, and 8.6% and 29.8% higher MAP compared to M3 (tpology+tags) and M1 (textual content only), respectively.

- Utilizing only the content of questions and their answers, i.e., M1, results in a model with the lowest performance. The reason is that such a model only relies on the textual similarity between a new question and past textual content generated by the experts and overlooks the information extracted from the relationships among the entities in the CQA network.
- We also find that the topological data extracted from a CQA platform is more informative compared to the textual information, i.e., the content of questions and their answers. The model that employs only the topology of the CQA network, i.e., M3, obtains 16.4% and 19.7% higher NDCG and MAP compared to the model that only employs textual content (M1).
- Removing tags from the CQA network has a negative yet small impact on performance. The impact becomes negligible as the size of the network increases. Note that the number of tags are almost the same in all datasets (see Appendix B1). When the dataset is small the chance of having unique tags among questions will be high compared to larger datasets. Therefore, using tags in smaller datasets will be more informative to discover relevant experts.

5.3 Discussion

Our method outperformed state-of-the-art expert finding baselines, achieving on average 4.4% and 6.7% superior expert rankings in terms of NDCG and MAP, respectively. It also formed teams with higher skill coverage and comparable past collaboration levels, meeting **objectives O2** and **O3**. Specifically, our method provided 4.7% broader knowledge and 14.6% higher expertise levels, with collaboration levels 1.7 times higher than the best team formation baseline. Additionally, using both topological and textual data led to 4.4% and 8.61% better NDCG and MAP scores compared to using only topological data. Topological data alone proved more effective than textual data, with 16.44% and 19.7% better NDCG and MAP performance. Overall, our method effectively balances team formation metrics, producing teams with comprehensive skills, substantial expertise, and strong past collaboration. Additional experimental results are available in Appendix B.

While our framework shows notable improvements in collaborative expert finding, it is important to recognize its limitations and potential drawbacks for a balanced view. (1) **Dynamic Nature of CQA Platforms.** CQA platforms are dynamic, with constant additions of questions, answers, and users. Our current framework, which requires retraining to update embeddings and model parameters, may struggle with these changes. Future work could explore incremental or online learning approaches for real-time updates without extensive retraining. (2) **Network Sparsity.** In sparse CQA networks, where some users rarely answer questions or certain questions involve only a small group of users, the available topological information may be limited. This can impact the quality of the learned embeddings and the final results. Although our framework uses textual information to mitigate this issue, network sparsity remains a challenge. Future research could investigate data augmentation techniques or the integration of external knowledge sources to improve model robustness in such conditions.

(3) **Diversity of Expertise.** While our method emphasizes forming collaborative teams with complementary skills, it does not explicitly ensure diversity in expertise or perspectives. Incorporating diversity metrics in future work could enhance team quality by providing more comprehensive and innovative solutions to complex questions. (4) **Bias and Fairness.** Like many machine learning models, our framework may unintentionally learn and propagate biases from the training data, potentially leading to unfair recommendations that favor certain users or groups. Future work should focus on incorporating bias detection and mitigation strategies to ensure fairness and equity in expert finding and team formation. By addressing these limitations in future research, we aim to enhance the robustness and fairness of the proposed framework, making it more suitable for practical deployment in diverse CQA environments.

6 Conclusion

We introduced a framework for finding collaborative experts in CQA platforms by integrating topological and textual data. Our method jointly learns latent representations of these data, resulting in superior performance compared to existing techniques. Extensive experiments on four real-world datasets demonstrated that our approach achieves higher ranking metrics (NDCG and MAP) and better skill coverage in forming expert teams. In the future, we plan to focus on recommending experts with specific roles, such as reviewers and editors, to create more realistic and functional teams. Additionally, we plan to consider users' willingness to participate and study the characteristics of effective teams to enhance our team formation strategies.

References

[1] Yuan, S., Zhang, Y., Hall, W., Cabotà, J.B.: Expert finding in community question answering: a review. AI Review **53**(2), 843–874 (2020)

[2] Li, Z., Jiang, J.-Y., Sun, Y., Wang, W.: Personalized question routing via heterogeneous network embedding. In: AAAI, vol. 33, pp. 192–199 (2019)

[3] Sun, J., Zhao, J., Sun, H., Parthasarathy, S.: Endcold: An end-to-end framework for cold question routing in community question answering services. In: IJCAI, pp. 3244–3250 (2020)

[4] Peng, Q., Liu, H., Wang, Y., Xu, H., Jiao, P., Shao, M., Wang, W.: Towards a multi-view attentive matching for personalized expert finding. In: WWW (2022)

[5] Qian, L., Wang, J., Lin, H., Xu, B., Yang, L.: Heterogeneous information network embedding based on multiperspective metapath for question routing. Knowl. Based Syst. **240**, 107842 (2022)

[6] Liu, H., Lv, Z., Yang, Q., Xu, D., Peng, Q.: Expertbert: Pretraining expert finding. In: CIKM, pp. 4244–4248 (2022)

[7] Kundu, D., Mandal, D.P.: Formulation of a hybrid expertise retrieval system in community question answering services. Appl. Intell. **49**(2), 463–477 (2019)

[8] Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G., Hartmann, B.: Design lessons from the fastest q&a site in the west. In: SIGCHI, pp. 2857–2866 (2011)

[9] Chang, S., Pal, A.: Routing questions for collaborative answering in community question answering. In: ASONAM, pp. 494–501 (2013)

[10] Liu, H., Lv, Z., Yang, Q., Xu, D., Peng, Q.: Efficient non-sampling expert finding. In: CIKM, pp. 4239–4243 (2022)

[11] Lappas, T., Liu, K., Terzi, E.: Finding a team of experts in social networks. In: SIGKDD, pp. 467–476 (2009)

[12] Khan, A., Golab, L., Kargar, M., Szlichta, J., Zihayat, M.: Compact group discovery in attributed graphs and social networks. Information Processing & Management **57**(2), 102054 (2020)

[13] Kou, Y., Shen, D., Snell, Q., Li, D., Nie, T., Yu, G., Ma, S.: Efficient team formation in social networks based on constrained pattern graph. In: ICDE, pp. 889–900 (2020)

[14] Hamidi Rad, R., Fani, H., Bagheri, E., Kargar, M., Srivastava, D., Szlichta, J.: A variational neural architecture for skill-based team formation. TOIS, 1–28 (2023)

[15] Li, M., Li, Y., Chen, Y., Xu, Y.: Batch recommendation of experts to questions in community-based question-answering with a sailfish optimizer. Expert Syst. Appl. **169**, 114484 (2021)

[16] Krishna, V., Vasiliauskaite, V., Antulov-Fantulin, N.: Question routing via activity-weighted modularity-enhanced factorization. SNAM **12**(1), 155 (2022)

[17] Zhu, H., Chen, E., Xiong, H., Cao, H., Tian, J.: Ranking user authority with relevant knowledge categories for expert finding. WWW **17**(5), 1081–1107 (2014)

[18] Shahriari, M., Parekodi, S., Klamma, R.: Community ranking algorithms for expert identification in question-answer forums. In: I-KNOW, pp. 8–188 (2015)

[19] Sorkhani, S., Etemadi, R., Bigdeli, A., Zihayat, M., Bagheri, E.: Feature-based question routing in community question answering platforms. INS **608**, 696–717 (2022)

[20] Amendola, M., Passarella, A., Perego, R.: Towards robust expert finding in community question answering platforms, 152–168 (2024)

[21] Liu, H., Lv, Z., Yang, Q., Xu, D., Peng, Q.: Efficient non-sampling expert finding. In: CIKM, pp. 4239–4243 (2022)

[22] Peng, Q., Wang, W., Liu, H., Wang, Y., Xu, H., Shao, M.: Towards comprehensive expert finding with a hierarchical matching network. Knowl. Based Syst. (2022)

[23] Peng, Q., Liu, H.: Expertplm: Pre-training expert representation for expert finding. In: EMNLP, pp. 1043–1052 (2022)

[24] Peng, Q., Liu, H., Xu, H., Wang, Y., Wang, W.: PEPT: expert finding meets personalized pre-training. CoRR **abs/2312.12162** (2023)

[25] Peng, Q., Liu, H., Lv, Z., Yang, Q., Wang, W.: Contrastive pre-training for personalized expert finding. In: EMNLP, pp. 15797–15806 (2023)

[26] Etemadi, R., Zihayat, M., Feng, K., Adelman, J., Bagheri, E.: Collaborative experts discovery in social coding platforms. In: CIKM, pp. 3009–3013 (2021)

[27] Kargar, M., Golab, L., Srivastava, D., Szlichta, J., Zihayat, M.: Effective keyword search over weighted graphs. IEEE Transactions on Knowledge and Data Engineering (2020)

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

[29] Etemadi, R., Zihayat, M., Feng, K., Adelman, J., Bagheri, E.: Embedding-based team formation for community question answering. INS, 671–692 (2023)

[30] Xiong, C., Dai, Z., Callan, J., Liu, Z., Power, R.: End-to-end neural ad-hoc ranking with kernel pooling. In: SIGIR, pp. 55–64 (2017)

[31] Mitra, B., Diaz, F., Craswell, N.: Learning to match using local and distributed representations of text for web search. In: WWW, pp. 1291–1299 (2017)

[32] Dai, Z., Xiong, C., Callan, J., Liu, Z.: Convolutional neural networks for soft-matching n-grams in ad-hoc search. In: WSDM, pp. 126–134 (2018)

[33] Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.: Learning deep structured semantic models for web search using clickthrough data. In: CIKM, pp. 2333–2338 (2013)

[34] Hu, B., Lu, Z., Li, H., Chen, Q.: Convolutional neural network architectures for matching natural language sentences. In: NIPS, pp. 2042–2050 (2014)

[35] Yang, L., Ai, Q., Guo, J., Croft, W.B.: anmm: Ranking short answer texts with attention-based neural matching model. In: CIKM, pp. 287–296 (2016)

[36] Guo, J., Fan, Y., Ji, X., Cheng, X.: Matchzoo: A learning, practicing, and developing system for neural text matching. In: SIGIR, pp. 1297–1300 (2019)

[37] Liu, S., Chen, L., Dong, H., Wang, Z., Wu, D., Huang, Z.: Higher-order weighted graph convolutional networks. arXiv preprint arXiv:1911.04129 (2019)

7 Appendix A

Appendix A Parameter training of MLP Ranker

Starting from the loss in Eq. 13, the gradients are first propagated to the last layer of the MLP ranker and update the parameters $\theta^{(l)}$ and $b^{(l)}$ as:

$$\theta^{(l)} = \theta^{(l)} - \mu \frac{\partial \mathcal{L}(\theta, b, \mathbf{V}, \mathbf{W}^{(0)}, .., \mathbf{W}^{(h)})}{\partial \theta^{(l)}}, \tag{A1}$$

$$b^{(l)} = b^{(l)} - \mu \frac{\partial \mathcal{L}(\theta, b, \mathbf{V}, \mathbf{W}^{(0)}, .., \mathbf{W}^{(h)})}{\partial b^{(l)}}, \tag{A2}$$

where μ is the learning rate. After updating trainable parameters through l layers of the MLP ranker, the gradients are propagated to the weight matrices of the structure encoder through Eq. 7 as:

$$\mathbf{W}^{(h)} = \mathbf{W}^{(h)} - \mu \frac{\partial \sigma^{(0)} \left(\mathbf{X}_{in} \theta^{(0)} + b^{(0)} \right)}{\partial \mathbf{W}^{(h)}}. \tag{A3}$$

Finally, the trainable weights $\mathbf{W}^{(i)}$ for i=h-1,..,0 in the remaining layers of the structure encoder are updated as:

$$\mathbf{W}^{(i)} = \mathbf{W}^{(i)} - \mu \frac{\partial \sigma \left(\hat{\mathcal{A}} H^{(i+1)} W^{(i+1)} \right)}{\partial \mathbf{W}^{(i)}}. \tag{A4}$$

In parallel, the gradients are passed from the first layer of the MLP ranker to the word embedding vectors starting from the soft-TF vectors obtained by Eq. 11. Then, the kernels (Eq. 10) pass the gradients to the word similarities in the translation matrix $\mathbf{M}$. Finally, the embedding vectors of words in questions and answers, i.e., $\mathbf{V}$, are updated as:

$$\mathbf{V} = \mathbf{V} - \mu \frac{\partial \mathbf{M}}{\partial \mathbf{V}}. \tag{A5}$$

The gradient received by each element of the translation matrix $\mathbf{M}$ is obtained based on Eq. 10 as:

$$\nabla M_{ij} = \sum_{k=1}^{r} \frac{\nabla K_k(M_i) \times \sigma_k^2}{(\mu_k - M_{ij}) exp\left(\frac{(M_{ij} - \mu_k)^2}{-2\sigma_k^2} \right)}. \tag{A6}$$

The gradients of each kernel K_k for $k = 1, .., r$ are received from the first layer of the MLP ranker based on Eq. 11.

Appendix B Supplementary Experiments

B.1 Experimental details on Network Parameters and Baselines

The number of hidden layers of the GCN in our model and the `EnC` method is two. It has been empirically proved that GCNs with two layers are capable of capturing higher-order topological proximities [37]. The embedding dimension, i.e., d, is set to 32 for nodes and 300 for words. Akin to [30], the number of RBF kernels is set as $r = 11$. Parameters of the kernels are set as $\mu_1 = 1.0$, and $\mu_2 = 0.9, \mu_3 = 0.7, .., \mu_{11} = -0.9$, and $\sigma_1 = 10^{-3}$, and for the rest $\sigma = 0.1$. The first kernel, i.e. $\mu_1 = 1.0$ and $\sigma_1 = 10^{-3}$, captures the exact matches. While the others are evenly spaced in [0.9,-0.9] and can be viewed as ten soft-TF bins. We employed

`Adam` as the optimization method with an exponential decay learning rate initialized from $\{5 \times 10^{-5}, 10^{-4}\}$ and other parameters with default values. For the other methods, we used the implementations suggested by the authors with default parameters (the best values used by their authors). For each method, the best results from five independent repetitions on the datasets are reported.

B.2 Team Formation Performance Metrics

Below we present how we compute each performance metrics **(1)** The **Skill Coverage (`SC`)** metric is used to gauge how well the background knowledge of experts in a retrieved team aligns with the subject matter of a new question. To calculate `SC` for a new question, we count the number of common tags between the new question and existing questions answered by members of a discovered team. Ideally, a team should have complete coverage, indicating that the tags of existing questions answered by team members are fully matched with the tags of the new question. Given $tg^{(i)}$ as a set of tags for question q_i, *skill coverage*, denoted as `SC`, of retrieved team T_k given new question q_k is computed as:

$$\texttt{SC}(q_k,\mathrm{T}_k) = |tg^{(k)} \cap \{ \bigcup_{q_i \in Q^{\mathrm{T}_k}} tg^{(i)}\}|/|tg^{(k)}|, \tag{B7}$$

where Q^{T_k} is the set of past questions answered by experts in T_k, and $|x|$ is the size of set x. We compute the average of `SC` over n test questions as: $\frac{100}{n}\sum_{k=1}^{n}\texttt{SC}(q_k,\mathrm{T}_k)$. The range of `SC` values is in $[0, 100]$ where larger values are more desirable. Given two teams T_n and T_m and test question q_k, team T_m is preferred based on Eq. 1 iff $\texttt{SC}(q_k,\mathrm{T}_m) \geq \texttt{SC}(q_k,\mathrm{T}_n)$. **(2) Collaboration Level (`CL`)** metric evaluates team collaboration by counting shared questions answered by team members. It normalizes this count based on the number of potential pairs in the team to ensure fair comparisons. Essentially, a higher `CL` score indicates more frequent collaboration among team members, suggesting a stronger team dynamic and is computed as:

$$\texttt{CL}(\mathrm{T}_k) = \alpha \sum_{\substack{v,u \in \mathrm{T}_k, \\ v \neq u}} |Q^{(v)} \cap Q^{(u)}|, \tag{B8}$$

where $Q^{(v)}$ is a set of existing questions answered by expert v, and α is the fraction of the number of non-empty sets of $Q^{(v)} \cap Q^{(u)}$ over the number of unique pairs of experts in team T_k, i.e., $|\mathrm{T}_k|\left(|\mathrm{T}_k|-1\right)/2$. We compute the average of the `CL` metric over n new test questions as: $\sum_{k=1}^{n}\texttt{CL}(\mathrm{T}_k)$, where T_k is the new team assigned to the new test question q_k. The value of `CL` is between zero and the number of question nodes in the CQA network. Given two teams T_n and T_m, team T_m is preferred, based on Eq. 2, iff $\texttt{CL}(\mathrm{T}_m) \geq \texttt{CL}(\mathrm{T}_n)$. **(3) Gold Standard Team Match (`GM`)** score is calculated by determining the percentage of the discovered team (T) that overlaps with the gold standard team (T_g). Specifically, it measures the proportion of the discovered team members who are also part of the gold standard team, then multiplies this by 100 to express it as a percentage. A higher `GM` score indicates that the discovered team closely resembles the actual experts who answered the question, suggesting effectiveness in the team discovery process.

The gold standard team match metric, denoted by `GM`, is calculated as $\texttt{GM} = \left(|\mathrm{T} \cap \mathrm{T}_g|/|\mathrm{T}_g|\right) \times 100$. The `GM` metric reflects how likely it is that the discovered experts worked together in reality to answer a given question.

The computation of the evaluation metrics is elaborated in the following example.

Example 1 *Suppose the CQA system depicted in Fig. 1 and a test question ($n = 1$) as follows:*
$q_t = \{body_t, title_t\}, tg^{(t)} = \{t_2, t_4, t_5\}$

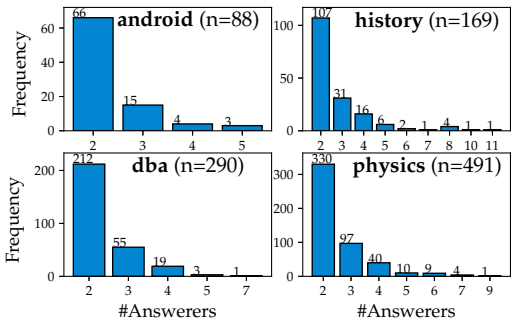


Fig. B1 The statistics of test questions. Here n is the number of test questions.

With actual answerers $\mathrm{T}_g = \{u_3, u_4\}$.
Let the retrieved team with size three be:
$\mathrm{T}_t = \{u_4, u_5, u_6\}$
The evaluation metrics are computed as:

$$\begin{aligned} \texttt{SC} &= \frac{100}{n}\texttt{SC}(q_t, \mathrm{T}_t) \\ &= \frac{100\ |tg^{(t)} \cap (\{t_1, t_2, t_3, t_5\} \cup \{t_1, .., t_5\} \cup \{t_1, t_4\})|}{|tg^{(t)}|} \\ &= 100. \end{aligned}$$

SC=100 means that there is a complete match between test question tags, i.e. $tg^{(t)}$*, and the tags of past questions answered by members of* T_t*, i.e.* $\{t_1, t_2, t_3, t_5\} \cup \{t_1, .., t_5\} \cup \{t_1, t_4\}$.
Collaboration level is computed as:

$$\begin{aligned} \texttt{CL} &= \frac{1}{n}\texttt{CL}(\mathrm{T}_t) = \alpha \sum_{\substack{v,u \in \mathrm{T}_t, \\ v \neq u}} |Q^{(v)} \cap Q^{(u)}| \\ &= \alpha\big[|Q^{(u_4)} \cap Q^{(u_5)}| + |Q^{(u_4)} \cap Q^{(u_6)}| \\ &\qquad + |Q^{(u_5)} \cap Q^{(u_6)}|\big] \\ &= \frac{2}{3}\big[|\{q_1, q_2\}| + |\{\}| + |\{q_3\}|\big] = 2. \end{aligned}$$

Finally, GM *is computed as:*

$$\texttt{GM} = \frac{|\mathrm{T}_t \cap \mathrm{T}_g|}{|\mathrm{T}_g|} \times 100 = \frac{100}{2} = 50.$$

It means that 50% of the actual answerers of the test question are among the members of the retrieved team.

B.3 Test Dataset Statistics

The statistics of test questions are summarized in Fig. B1.

B.4 Performance Gap Analysis on Collaborative Team Discovery

We demonstrate the power of our method compared to team formation approaches through performance gap analysis. The top two expert finding methods `EnC` and `NeR` are selected based

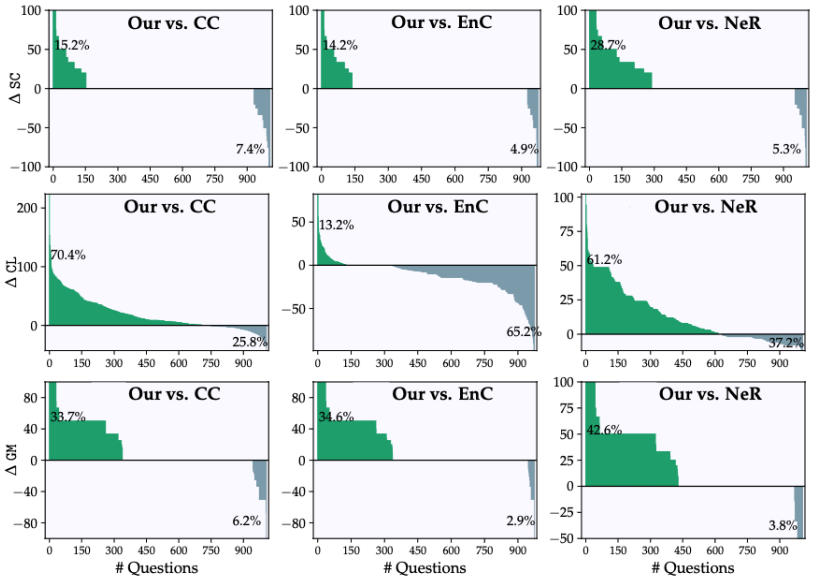


Fig. B2 Differential Performance Analysis Across Test Questions — Our method versus `CC` and top two expert finding performers, `EnC` and `NeR`, as detailed in Table 2.

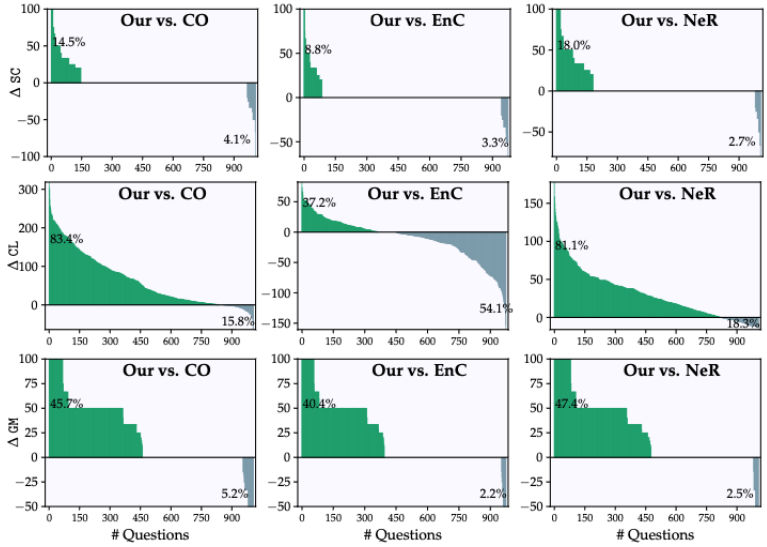


Fig. B3 Differential Performance Analysis Across Test Questions — Our method versus `CO` and top two expert finding performers, `EnC` and `NeR`, as detailed in Table 3.

on the results reported in Tables 2, 3, and 4. Furthermore, we focus on three main team formation techniques `CC`, `CO`, and `CS`. Note that although t2b shares similar algorithmic features with our model, particularly in leveraging network-based information for expert discovery, the performance gap analysis was specifically designed to examine aspects of expert finding that are distinctively influenced by methods like skill diversity and collaboration level. These aspects are more prominently addressed by the `CC`, `CS`, and `CO` approaches. This focus

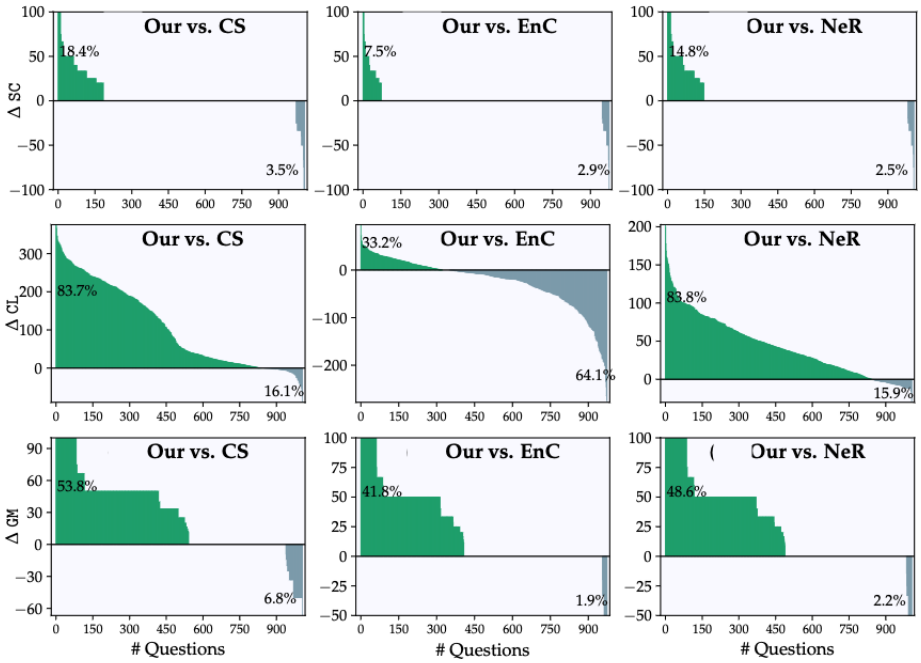


Fig. B4 Differential Performance Analysis Across Test Questions — Our method versus `CS` and top two expert finding performers, `EnC` and `NeR`, as detailed in Table 4

allows us to better highlight and assess the strengths of our model in terms of these particular metrics. We compare them side by side with our method based on their performance on each test question and demonstrate the results in Figs. B2, B3, and B4. A positive value in the figures indicates that the proposed method obtains a superior result compared the corresponding baseline. In contrast, negative values demonstrate that the corresponding baseline outperforms our method. Zero values demonstrate the methods achieve the same results. We summarize our observations from the results depicted in the figures as follows:

- The proposed solution outperforms the baselines in terms of skill coverage, i.e. `SC` demonstrated in the first row of the figures. In comparison with team formation baselines, the experiments indicate that our method achieves superior results in 15.2%, 14.5%, and 18.4% of the test questions compared to `CC`, `CO`, and `CS`. In contrast, baselines `CC`, `CO`, and `CS` outperform our method only in 7.4%, 4.1%, and 3.5% of the test questions, respectively. Our method also outperforms expert finding baselines `EnC` and `NeR` in 14.2% and 28.7% of the test questions, and falls short only in 4.9% and 5.3% of the cases, respectively;

- Our method is the second-best method in terms of past collaborations among team members. The results indicate that `EnC` outperforms the other methods in terms of `CL`. It achieves superior results in 61.9% of questions compared our method. The reason for such an observation is that `EnC` retrieves expert nodes that have a high degree in the CQA network graph. Such high-degree nodes are likely to have common neighbours and hence lead to a higher `CL`;

- Our method consistently achieves superior results in terms of retrieving actual answerers of test questions, i.e., `GM`. Compared to the second best baseline, i.e., `CC`, our method obtains 4.18 times superior results.

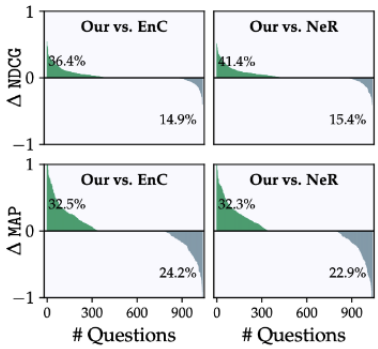


Fig. B5 The difference of results in terms of ranking metrics NDCG and MAP obtained by the proposed method and the baselines on each test question on all datasets. Positive values indicate the superiority of our method compared to baselines.

We argue that while based on above points, EnC has a higher performance on CL, this comes at the cost of sacrificing skill coverage, SC and GM.

B.5 Performance Gap Analysis on Expert Finding Methods

To better demonstrate why the existing expert finding approaches are not the best solution for the problem of collaborative expert finding, we analyze the performance gaps in terms of different performance metrics. We report the performance difference over NDCG, denoted ΔNDCG, and MAP, denoted ΔMAP, obtained from differencing the performance of our approach and top two performer expert finding methods, EnC and NeR, on each test question in the help-hurt plots in Fig. B5. In the figure, positive values indicate the superiority of our method. In contrast, negative values demonstrate the better performance of the corresponding baseline. The summary of our observations is as follows: **(a)** Our method outperforms the best baseline, EnC, on 36.4% of the questions in terms of NDCG, demonstrating significant improvement in ranking quality where our method is effectively more aligned with the ideal answer rankings; **(b)** our proposed approach improves the results over our best baseline, EnC, in terms of MAP by 32.5%, indicating our method's robustness in identifying relevant experts regardless of their order in the ranking list; **(c)** Interestingly, all methods tend to perform better on MAP compared to NDCG. This can be attributed to the fact that MAP focuses solely on the presence of relevant experts within the results, disregarding their specific positions, which often results in higher performance scores under this metric.

B.6 Trade-off between the Team Formation Metrics

Each team formation metric measures a distinct aspect of the team formation task. Obviously, these metrics possess different levels of importance given the task at hand. For example, having the background knowledge required to answer a question, i.e., skill coverage SC, is the most critical when compared to the other metrics. In other words, forming a highly collaborative team that does not have the right skill set to answer a question would be basically pointless. As such, we investigate the trade-off between SC and the other metrics, CL. We also employ GM to demonstrate how well each method has been able to discover the actual set of experts who answered the question in reality. To this end, the parameters of the methods are tuned to form teams of the same size equivalent to the number of experts

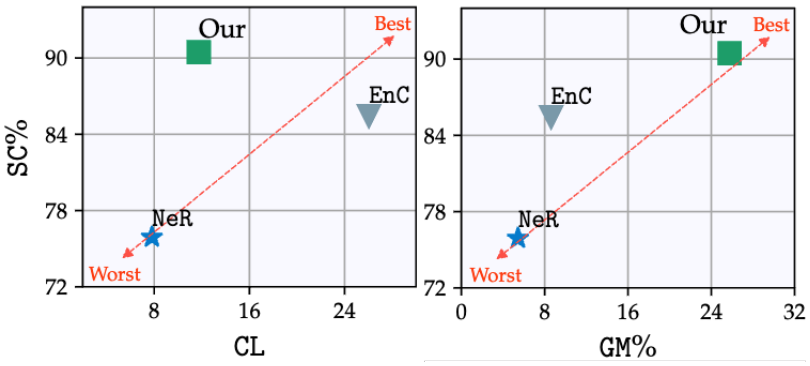


Fig. B6 Comparison of the methods based on the trade-off between the team formation metrics. The arrows in the panels indicate the best and worst performance levels in which each method can be obtained.

who answered the same question in reality. Then, the metrics are computed for each dataset, and their average on all four datasets is reported in Fig. B6. In each sub-figure, the arrows demonstrate the best and worst trade-off points. We make several observations as follows: **(a)** Our approach is the best choice in terms of the trade-off between skill coverage, i.e. `SC`. `NeR` behaves very poorly in terms of `SC` which is critical for a team. Our method achieves on average 19.18% superior results in terms of `SC` compared to `NeR`. As depicted in the figure (the 1st sub-figure from the left), such achievements make our method the best option while considering both metrics; **(b)** Although our method forms teams with on average 54.88% lower past collaboration level, i.e. `CL`, compared to our best baseline `EnC`, the skill coverage of such teams formed by `EnC` are on average 5.91% lower than those discovered by our method. This means that having high `CL` does not guarantee to have the full coverage of the skills required to answer new questions; **(b)** Our method not only forms teams with high-skill coverage but also retrieves more experts from the actual answerers of the test questions. The results reveal that our model employs the actual answerers in the discovered teams on average 1.99 times more than the best baseline, i.e. `EnC`.

As demonstrated in Fig. B6, our proposed method is able to discover teams with superior characteristics in terms of the trade-off between different team formation metrics. This shows that our method not only satisfies the critical constraint of team formation metric, i.e., skill coverage `SC`, but also shows superior or comparable performance on the other soft constraint, i.e., `CL`; hence, reporting the best tradeoff between the team formation metrics.

B.7 Computational Time Analysis and Scalability

In this section, we provide an analysis of the computational time and scalability of our proposed framework, combining both theoretical analysis and empirical evidence.

The computational time of our framework is primarily determined by the graph neural network (GNN) and multi-layer perceptron (MLP) components. Let n be the number of nodes, m the number of edges in the community question answering (CQA) network, and d the embedding dimension. The GNN's time complexity for each layer is $O(m \cdot d)$, and for L layers, it becomes $O(L \cdot m \cdot d)$. The MLP component, which processes the node embeddings, has a complexity of $O(n \cdot d^2)$ for each layer. Assuming H hidden layers, the total complexity for the MLP is $O(H \cdot n \cdot d^2)$.

Given the typical sparsity of CQA networks, where m is often proportional to n, the overall training complexity can be approximated as $O(L \cdot n \cdot d + H \cdot n \cdot d^2)$. This complexity is

well within the range of other state-of-the-art methods. In fact, our design focuses on leveraging efficient computations without over-complicating the process. Compared to methods like NeR and EnC, which have similar complexities due to their reliance on node and edge embeddings, our framework maintains efficiency while providing the added benefit of learning richer embeddings through joint optimization. Unlike some heuristic-based methods, such as CC, CO, and CS, which involve iterative processes, our method's end-to-end learning approach simplifies the computational flow, ensuring scalability even as the dataset size increases.

We also conducted experiments across four datasets: *android*, *history*, *dba*, and *physics*. The experiments measured the number of samples processed per second during both training and inference phases, providing empirical evidence of the model's efficiency. The performance results are illustrated in Figure B7, which shows the number of samples processed per second during training and inference across different datasets.

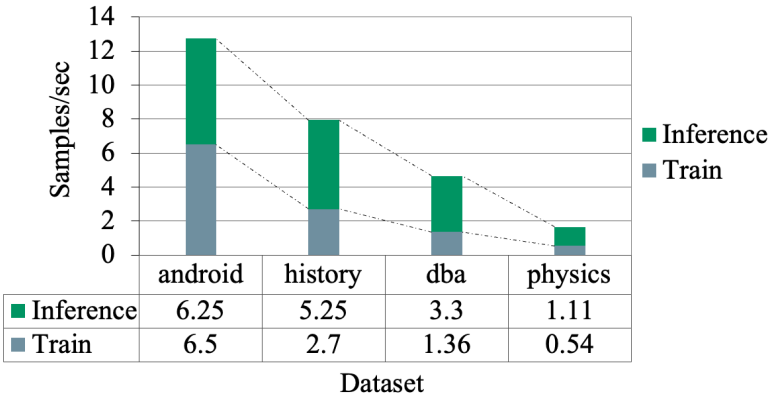


Fig. B7 Performance of the proposed framework (samples/second) across datasets during training and inference.

The data from Figure B7 shows that as the dataset size increases, the throughput decreases. For example, the *android* dataset, being relatively smaller, shows higher throughput compared to the larger *physics* dataset. This is consistent with the theoretical expectations, as larger datasets introduce more complexity and require more computational resources.

Despite this, the model's scalability is confirmed to be practical and efficient. Even with the *physics* dataset, the model processes over half a sample per second during training and more than one sample per second during inference. This indicates that the model can handle large datasets effectively, especially in batch processing environments. The high inference speed on smaller datasets also supports the model's suitability for real-time applications.

B.8 Details of Our Findings

In our experiments, we have explored the performance of our proposed method along with our baselines from two perspectives, i.e., *expert finding* and *team formation*. We summarize our findings as follows:

- To satisfy **objectives O1**, our proposed method produces superior rankings for experts compared to the state-of-the-art expert finding baselines. Our experimental study on four

diverse real-world datasets indicates that our method achieves on average 4.4%, and 6.7% superior expert ranks in terms of NDCG and MAP compared to our best baseline;

- Our proposed method is able to discover teams with high skill coverage and comparable past collaboration level among team members; therefore, satisfying **objective O2** and **O3**. The experiments reveal that our method obtains on average at least 4.7% broader background knowledge required to answer new questions with on average at least 14.6 higher expertise level compared to existing team formation methods. Furthermore, the collaboration level of members of teams constructed by our model is on average 1.7 times higher than our best team formation baseline;
- We also found that employing both topological and textual data extracted from the CQA network leads to on average 4.4% and 8.61% superior results compared to the model which only uses topological data in terms of NDCG and MAP, respectively. Furthermore, the experiments indicate that topological data are more effective compared to textual data. Experimental results demonstrate that the model which uses only topological data shows on average 16.44% and 19.7 % better performance on NDCG and MAP, respectively;
- Our method discovers efficient teams by hitting the right trade-off between team formation metrics. The experiments demonstrate that teams discovered by our method possess not only broader required skills to answer the new question but also possess comparable expertise and past collaboration levels.