Explaining Expert Search and Team Formation Systems with ExES

by

Kiarash Golzadeh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2024

Abstract

Expert search and team formation systems operate on collaboration networks with nodes representing individuals, labeled with their skills, and edges denoting collaboration relationships. Given a query corresponding to a set of desired skills, these systems identify experts or teams that best match the query. However, state-of-the-art solutions to this problem lack transparency and interpretability. To address this issue, we propose ExES, an interactive tool designed to explain black-box expert search systems. Our system leverages saliency and counterfactual methods from the field of explainable artificial intelligence (XAI). ExES enables users to understand why individuals were or were not included in the query results and what individuals could do, in terms of perturbing skills or connections, to be included or excluded in the results. Based on several experiments using real-world datasets, we verify the quality and efficiency of our explanation generation methods. We demonstrate that ExES takes a significant step toward interactivity by achieving an average latency reduction of 50% in comparison to an exhaustive approach while maintaining over 82% precision in producing saliency explanations and over 70% precision in identifying optimal counterfactual explanations.

Acknowledgements

I would like to express my sincere gratitude and thanks to my supervisor, Professor Lukasz Golab, for his invaluable support, patient mentorship, and exceptional guidance during my research endeavors.

I would like to thank Professor Jaroslaw Szlichta, Professor Mehdi Kargar, and Professor Morteza Zihayat Kermani for their thoughtful advice and for guiding me throughout this research.

I would also like to thank Professor Robin Cohen and Professor Jian Zhao who served as the readers of this thesis for their valuable time.

Dedication

I dedicate this thesis to my family.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The expert search problem takes in a collaboration network as input, with nodes representing individuals, node labels corresponding to skills held by these individuals, and edges denoting collaboration relationships. When given a query consisting of a set of desired skills, the output is a ranked list of individuals who best match the query. The expert search process has become a fundamental process, both in academia and industry [33, 29]. These systems enable expertise seekers to find the most knowledgeable people in their domain, ensuring a good fit for their particular requirements. Expert search has attracted considerable attention in various contexts such as:

- **Academic domain** [80]: Academic expert search systems aim to find key scholars in specific research topics. The academic collaboration networks include researchers, faculty members, and students who publish in established venues as experts. The connections formed through previous collaborative research projects are considered as edges. Typical users of these systems would be supervisors, university admission committees, and researchers. They utilize the system to find potential new collaborators or to seek expert advice for their academic endeavors.

- **Medical fields** [74]: Expert search systems are found beneficial in the healthcare system, where patients should be assigned to the best clinician, regarding their disease contexts. Medical collaboration networks contain clinicians and physicians and their skills would be their medical specialties. Cooperation and collaboration in these networks are considered when clinicians provide diagnosis recommendations for each other's patients when requested.

- **Social media** [8]: Expert search in social media refers to identifying potential user profiles with enough expertise on that topic, based on their discussions and shared content. In this domain, expert search systems would be helpful for marketers and social media analysts to find their ideal influencers for content promotion. Social expert search systems consider the users' peers in social media whom they interact with, such as followers, connections, or friends as links in the collaboration network.

- **Organizations** [37]: In the context of organizations, expert search tools locate employees in an organization who know the solution to specific issues in the organization. Here, the experts are employees and workers of the organization. They are represented in a hierarchical graph where the links depict managerial and peer-working relationships, which constitute the collaboration network.

- **Legal domain** [3]: In legal expert search, the objective is to enable citizens to find lawyers based on their expertise to assist them in their legal cases or inquiries. The collaboration network consists of lawyers who have participated in various cases, and collaborations are indicated by lawyers working on the same case or cases with similar circumstances.

- **Question answering** [57, 23]: Expert search plays an important role in question routing in community question-answering (CQA) platforms such as StackOverflow and Quora. By accurately identifying experts, it ensures that questions are directed to technically-equipped individuals who can provide valuable answers, thereby increasing user engagement on these platforms. In CQA-related expert search, the collaboration network contains platform users, having their skill profiles built upon their previous answer contributions, and the past interactions between experts serve as the links within this network.

Typically, a second step ensues, referred to as team formation [62], in which a subgraph is returned, spanning said experts along with potentially other individuals. In cases where single individuals fall short of possessing the required knowledge, expertise, or capacity for a project, forming a team of experts proves to be significantly useful. An optimal team, in addition to covering required skills, can also benefit other aspects such as productivity and good communication among team members.

In terms of the collaboration network, the selected experts for a team should be closely connected to each other, indicating prior collaborations, either directly or through a small number of intermediaries. Such connectivity increases team cohesion and enhances the likelihood of selected members getting along with each other positively. Team formation applications include scenarios such as:

- **Recruiting software developers** [65]: Team formation is particularly valuable in technology companies for locating and hiring talented software developers to collaboratively work on software projects. In such contexts, the collaboration network could include collaborative code hosting networks such as GitHub, or software-related CQA networks such as StackOverflow.
- **Forming academic collaborations** [31]: Using team formation methods could be helpful for gathering a productive research group. This is especially crucial when working on multi-dimensional research topics, where a diverse set of skills and perspectives is necessary for success.

State-of-the-art solutions for expert search combine a variety of factors during expert ranking and team selection in non-obvious ways, including skills and the network structure [62, 31]. As a result, while these solutions are effective, they lack transparency, making it difficult to debug them and limiting their practical uptake. However, since the expert search methods have been broadly applied to real-world contexts, it is crucial to confirm their transparency.

In general, to overcome the opaqueness of AI tools, explainable artificial intelligence (XAI) methods have been widely developed to bring transparency and trust to these *black-box* tools. XAI methods exist for a variety of tasks, including classification and recommendation engines. Particularly, XAI approaches have been commonly developed to create saliency and counterfactual explanations, where saliency explanations reveal the reasons behind a prediction, and counterfactuals reveal minimal input changes needed to alter a decision [27, 7]. However, we are not aware of any specific applications of these methods to the domains of expert search and team formation. This brings us to the first research question: **How can we address the explainability gap in expert search and team formation systems?**

Beyond addressing the explainability issue, another key challenge resides in managing a vast search space for generating explanations. Real-world collaboration networks exhibit complexity due to their large scale and diverse skill sets. Thus, devising a well-structured pruning method to narrow down the search space for the candidate explanations is crucial for maintaining system interactivity. This raises our second research question: **How can we efficiently navigate the exponential search space given the large nature of collaboration networks?**

1.2 Contribution

To fill the gaps described above, we propose ExES: an interactive tool to explain black-box expert search systems. Our contributions and demonstration goals are outlined as follows.

1. **Framework Formulation.** Our explainability framework casts expert search and team formation as binary classification problems, utilizing the social network to derive interpretable features. For expert search explanations, we define the *relevance status* binary function, which is true when an individual is identified as an expert, and false otherwise. For team formation tasks, we define the *membership status* binary function, which is true if and only if an individual is taken into the resulting team. This approach allows us to adapt saliency and counterfactual explanation methods from explainable AI to this novel problem. Specifically, saliency methods identify important skills and network connections that were influential in expert selection, while counterfactual methods serve as career advancement tools, by finding skills and network connections that, when added, can turn non-experts into experts for a given query. ExES is model-agnostic, meaning it does not require access to the system's inner workings and only needs to probe the expert search system with different perturbations of the input and observe how the output changes.

2. **Inference Pruning Strategies.**

   We developed an efficient implementation of the above explanation framework, exploiting the inherent structure of the expert search problem to prune the search space of candidate explanations. We took advantage of a keyword embedding model [79], to learn semantic relationships between the network's skills, and a link prediction model [52] on the network's collaborations, in our pruning strategies. Employing these models allowed us to narrow down the expansive list of potential explanations, thereby reducing computational overhead, ensuring system responsiveness, and allowing real-time user interaction.

3. **Explanation Evaluation.** We report extensive experiments, assessing the effectiveness of ExES on various models and datasets. Results showcase that ExES not only provides concise insights on expert search/team formation model behaviors but also highlights its superior performance compared to exhaustive explanation baselines, proving the effectiveness of its pruning strategies. We demonstrate that ExES achieves an average latency reduction of 50% while maintaining over 82% precision in generating saliency explanations, and over 70% precision in identifying optimal counterfactual explanations.

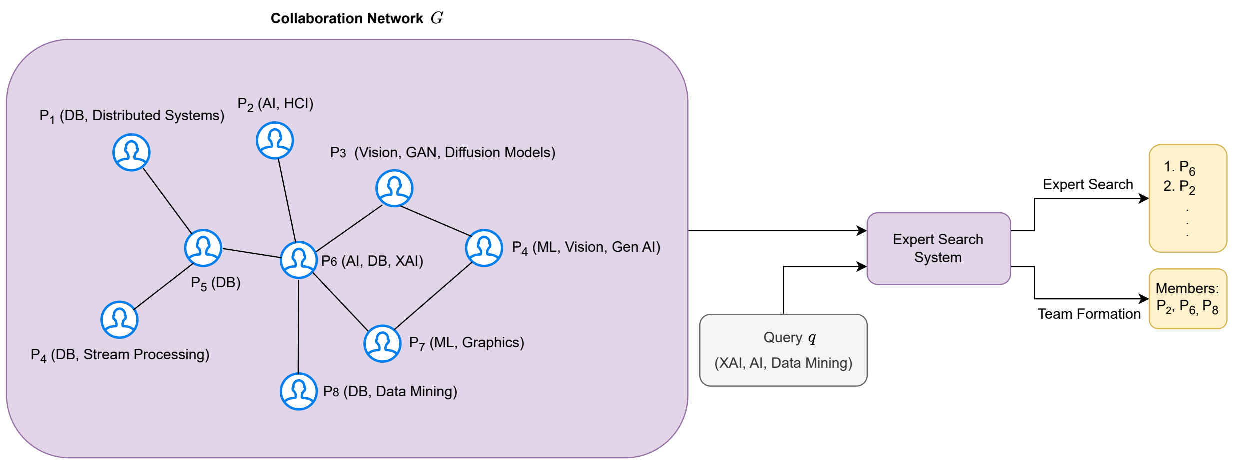
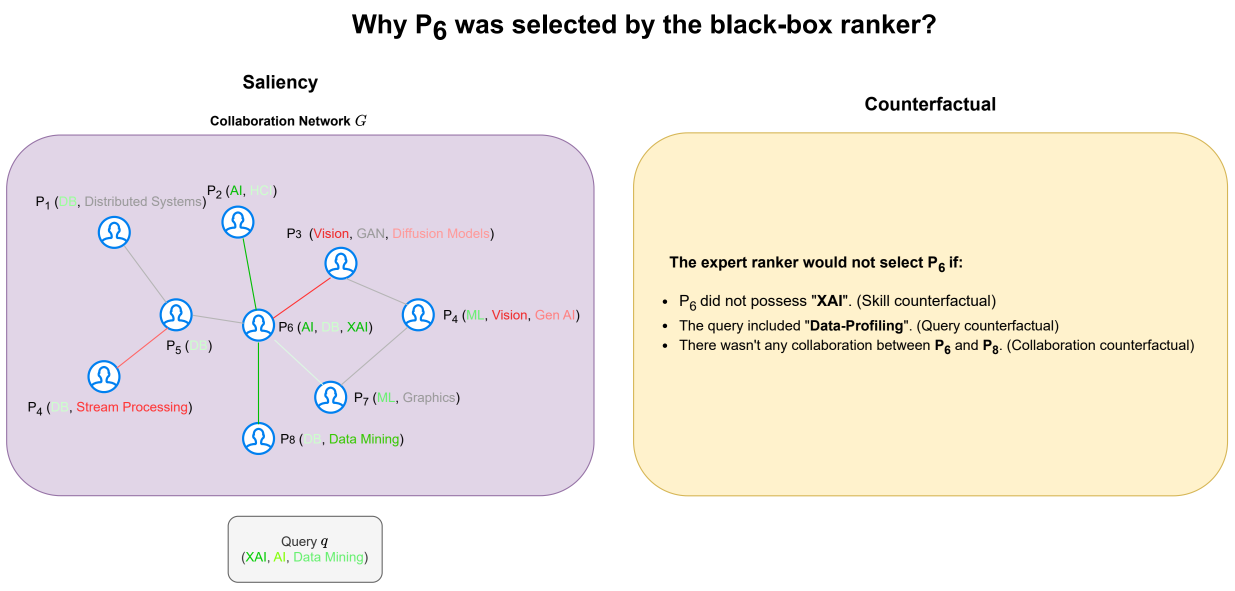


Figure 1.1: An overview of Expert Search workflow.

1.3 Motivating Example

Here, to motivate our approach, we demonstrate a toy example. Figure 1.1 illustrates the typical workflow of an expert search system on a portion of a collaboration network in academia. In this example, nodes represent researchers and edges denote co-authorship links. Node labels are written in parentheses. Given a sample query including keywords "XAI", "AI", and "Data Mining", sample outcomes for expert search and team formation tasks are generated by the black-box model.

Given the collaboration graph in Figure 1.1, ExES can be applied to explain the selection process of experts and team members. For instance, consider a scenario where a user wants to investigate why P_6 was selected as the top expert by the black-box expert ranker. Figure 1.2 shows typical explanations generated by ExES to explain this decision. It underscores the positive (colored in green) and negative (colored in red) impacts of input features, such as skills, network collaborations, and query keywords, through saliency explanations (the opacity of colored features shows their degree of importance). Specifically, skills like AI, XAI, Data Mining, ML, and DB are identified as having positive influences on p_6's selection, while others like Vision, Stream Processing, Gen AI, and Diffusion Models have a negative impact. Notably, ExES considers not only p_6's skills but also the skills of their network neighbors, for the explanations. Additionally, collaborations between p_6 and p_2, as well as between p_6 and p_8, have the strongest positive impacts, whereas connections between p_6 and p_3, and between p_5 and p_4, have the greatest negative impact. Meanwhile, ExES offers counterfactual explanations, demonstrating minimal alterations to these input


Why P$_6$ was selected by the black-box ranker?
Saliency
Collaboration Network G
P$_1$ (DB, Distributed Systems)
P$_2$ (AI, HCI)
P3 (Vision, GAN, Diffusion Models)
P$_4$ (ML, Vision, Gen AI)
P6 (AI, DB, XAI)
P$_5$ (DB)
P$_4$ (DB, Stream Processing)
P$_7$ (ML, Graphics)
P8 (DB, Data Mining)
Query q
(XAI, AI, Data Mining)
Counterfactual

The expert ranker would not select P$_6$ if:

- P$_6$ did not possess "**XAI**". (Skill counterfactual)
- The query included "**Data-Profiling**". (Query counterfactual)
- There wasn't any collaboration between **P$_6$** and **P$_8$**. (Collaboration counterfactual)

Figure 1.2: Typical explanations for selection of P_6

features that would result in P_6 not being selected as a top expert by the black-box ranker.

1.4 Thesis Overview

The structure of this thesis is as follows:

In Chapter 2, we review the literature on proposed solutions for Expert Search and Team Formation problems. Following this, we describe the fundamentals of explainable AI methods, including saliency and counterfactual explanations. After that, we provide the related work in explaining graph-based and ranking models, which have inspired our proposed solution.

In Chapter 3, we begin with defining the core concepts of expert search and team formation tasks, and presenting an outline of the ExES key components. Then, we formulate the expert search problem as a binary classification task, which enables us to generate saliency and counterfactual explanations for the expert search task. Subsequently, we describe the algorithms for producing each type of explanation, declaring the pruning strategies to reduce the search space for explanations. We continue by discussing the time complexity of our explanation generation approaches and comparing pruned and exhaustive searches.

Lastly, we define the binary classification task that we use for explaining team formation systems.

In Chapter 4, we begin with defining the setting of our experiments, followed by the experimental results that establish the efficiency and efficacy of explanations produced by ExES.

In Chapter 5, we conclude by summarizing our findings and outlining future work directions for extending the explainability of expert search and team formation systems.

Chapter 2

Related Work

2.1 Expert Search Solutions

We categorize proposed solutions for expert search into three categories: Document-based, Graph-based, and hybrid solutions. The classification of the related literature is provided in Table 2.1.

The first solutions for Expert Search were document-based approaches. These approaches used language models to capture expertise levels from available textual sources of expertise, such as documents authored by experts. These models then predict the probability of an individual being an expert in a given topic [5]. We can categorize the document-based solutions into document-centric and profile-centric solutions. The document-centric models work by retrieving textual documents that are most relevant to the query and then ranking their authors according to the relevance scores of their matching documents [55, 51, 6]. On the other hand, profile-centric models learn representation vectors for each individual and then directly rank experts according to the query [46, 76, 4]. More recently, [12] suggested combining classic document-centric approaches with a novel profiling

Table 2.1: A taxonomy of Expert Search Solutions

Approach	**References**
Document-based	[5, 55, 51, 6, 13, 46, 76, 4, 17, 12]
Graph Representation Learning	[9, 34, 59]
Hybrid Models	[58, 35]

technique, where entities within documents are linked to a knowledge graph, and their PageRank scores are incorporated into expert profiles.

A critical challenge in document-based solutions is the mismatch between query keywords and terms in the document corpus. For instance, experts in “Artificial Intelligence (AI)” may not use the term “AI” in their authored documents, but use words with related concepts such as “ML”, “neural”, or “loss”. To overcome this issue, [13] presented statistical and word embedding-based methods and [17] proposed attention-based classification networks to obtain skill translations and capture semantic similarity between keyword queries and raw documents.

However, aside from the implicit textual semantics of the expertise sources, the explicit relations between their authors should be taken into account as well. The linking edges among the network components reveal a lot of valuable information about the potential relevance and expertise propagation over the network [15]. Nevertheless, the aforementioned methods overlooked the structure of the collaboration network and the connections and relationships between individuals. Through the introduction of graph analysis methods, many studies have been conducted by including graph-based representations in the search process [9, 34, 59]. These approaches use procedures such as DeepWalk [60], PageRank [10], and HITS [41] scores to reflect local and global structure information from the collaboration network.

Finally, hybrid expert search models have drawn significant attention in recent years. These methods have been developed to combine features extracted from the documents and features obtained from the collaboration network to formulate recommendations [58, 35].

While having a comprehensive list of studies concentrating on enhancing the accuracy of expert search systems, to the best of our knowledge, there is no effort to explain the outcomes of these systems. ExES addresses this gap by providing understandable explanations for the search results.

2.2 Team Formation Solutions

The task of expert search is generally extended to team formation, where the goal is to find a set of candidates, as a subgraph of the collaboration network, who can collectively cover the required skills of the query while being able to collaborate effectively with each other. The effectiveness of team collaborations is majorly measured through node distances in the collaboration network. However, the criteria for measuring effectiveness vary across different approaches, resulting in diverse answers. For instance, some methods focus on

Table 2.2: A taxonomy of Team Formation Solutions

Approach	**References**
Graph Optimization	[43, 36, 83, 10]
Integer Programming	[56, 21]
Neural Methods	[68, 30, 14, 38]

pairwise distances between team members, while others designate a node as the *connection node* around which the team is built, and calculate the sum of distances between members and the connection node. The proposed solutions to the team formation problem can be categorized into three classes: graph optimization approaches, integer programming optimizations, and neural methods. Table 2.2 displays a classification of the references.

In the graph optimization category, an initial solution was introduced by [43] which minimized the communication cost among the team members. They presumed an appropriate team to be optimal in two heuristic functions; specifically in having the shortest diameter or being a minimum spanning tree. Subsequently, several research were conducted to refine the heuristic functions and improve the overall team quality. [36] considered the optimization heuristic to be the sum of distances between the candidates in the subgraph. [83] formulated the team formation task by taking experts' authority into account and introducing the heuristic objective function as a combination of communication cost and authority weights. [10] proposed a new metric based on a random walk with restart, to define the proximity between nodes in the collaboration network. This metric takes the network's structural properties into account and leads to presenting robust answers.

Another way to solve the team formation problem is using integer programming optimization. In this regard, [56] applied facility location analysis, a well-known task in Operation Research, to team formation and leveraged integer programming to optimize the assignment of experts to tasks with varied skill requirements. Additionally, [21] addressed multi-aspect team formation by formulating it as an integer linear programming problem, using a heuristic approach to optimize the allocation of teams to tasks.

However, a major drawback of the optimization solutions is their scalability. The proposed optimization models have been shown to be a reduced version of the Group Steiner tree task [16, 42] which is an NP-hard problem [43]. In addition, the integer-programming-based optimizations are NP-hard as well. In this regard, researchers have considered designing data-driven methods and neural network architectures to identify teams. The goal of these methods is to estimate a mapping function from the space of skills to the space of individuals. [68] trained an autoencoder architecture processes the collaboration network's adjacency matrix to learn latent representations of individuals, thus identifying experts who can learn from other team members. To mitigate the overfitting issue of non-variational autoencoders, [30] improved the neural networks by using variational autoencoders.

Using contrastive learning is common for training team formation neural networks, where given a query, the model learns to prefer ground-truth teams (teams included in the dataset, deemed successful) over unsuccessful ones. To illustrate the concept of success in teams, a collaboration leading to publication in a top-tier venue is considered successful in the academic domain. Similarly, in the software domain, jointly contributing to a high-starred repository demonstrates a successful collaboration. However, since datasets usually only contain successful teams and collaborations, [14] introduced and compared three negative sampling scenarios specifically team formation, to enhance the contrastive learning process. Additionally, [38] employed graph attention networks to address the evolution of collaboration networks through time, while reducing the computation time of team identification.

This review reveals that Graph Optimization methods typically form teams by optimizing node distances. On the other hand, Integer Programming approaches focus on selecting team members to maximize task allocation metrics such as diversity, personality compatibility, commitment, balanced load, and leadership qualities of the members [25]. Ultimately, neural methods integrate both stated aspects into their decision process.

Although there is significant research on developing team formation systems, [22] is the state-of-the-art work on explaining team formation systems. However, their method only supports team formation systems that use integer linear programming (ILP), limiting its generalizability. In addition, their work only covers *contrastive explanations*, which compare the output teams' characteristics (such as diversity, coherence, and member satisfaction) against unformed teams, rather than addressing the impact of system inputs. ExES fills these gaps by reflecting the effect of input features on the resulting team, both factually and counterfactually.

2.3 Explainable AI Solutions

Research on Explainable AI has flourished over the last few years [28]. Explainability approaches are categorized in various taxonomies, according to their characteristics. For instance, based on their applicable scope, they are classified into *intrinsic* vs *post-hoc* approaches. Intrinsic methods, also known as *interpretable-by-design*, focus on designing model architectures in a manner that ensures transparency. Examples of such models are decision trees and rule-based classifiers. These methods incorporate transparency as a fundamental aspect of the model's architecture. On the other hand, *post-hoc* explainability methods concentrate on explaining the outcomes of existing trained ML models.

Another pivotal taxonomy differentiates between *local* and *global* explanation methods. *Local* methods focus on explaining individual instances or decisions made by the model, while *global* methods aim to provide an understanding of the model's overall behavior and logic.

Finally, explanation methods could be classified into factual and counterfactual explanation methods. Factual explanation techniques, which are referred to as feature-based or saliency methods, are primarily concerned with finding the most important input features for individual predictions. Conversely, counterfactual explanations seek minimal perturbations to the input features which causes an alteration in the predictions. The ExES framework is classified into post-hoc and local categories and is capable of producing both factual and counterfactual explanations.

In the remainder of this section, we review the existing literature of three research directions–post-hoc explainability techniques, explaining graph-based machine learning tools, and interpreting ranking models. Since ExES is a post-hoc explainability tool, we found inspiration in saliency and counterfactual explanation methods. We further reviewed related XAI projects in graph neural networks as ExES processes data inputs with similar structures — graph-based data. Furthermore, we investigate explainability in ranking tasks because the core functions of ExES — expert search and team formation — are essentially based on ranking. We derived the idea of adapting our core tasks into binary classification from these related works.

2.3.1 Post-hoc explainability techniques

Post-hoc explainability methods are applied to interpret the predictions made by black-box models, such as ensemble methods and deep neural networks, after they are fully developed. In this regard, a common approach is to construct simpler, more interpretable models that approximate the functionality of the original complex black-boxes.

To this end, a variety of techniques have been explored in the XAI literature. LIME [63] and SHAP [48] are two prominent tools for post-hoc explanation. These perturbation-based methods provide *local* explanations for any black-box classifier, without any assumptions about the internal mechanisms of the model.

LIME [63] generates explanations by learning an interpretable white-box model, referred to as *surrogate model*, locally around each individual prediction and estimating the

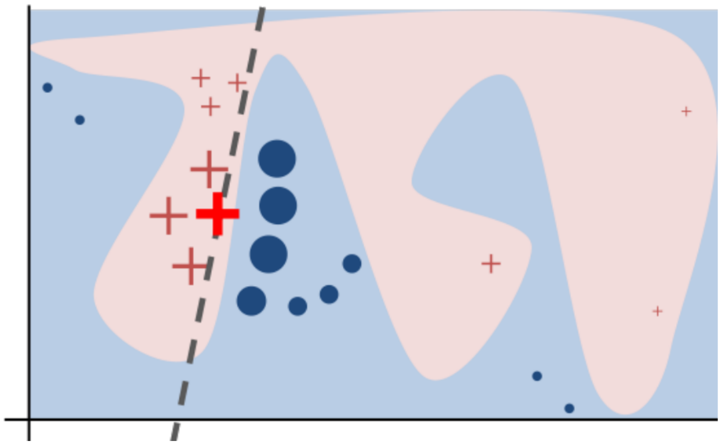


Figure 2.1: A toy example to demonstrate how LIME works [63]

feature attributions using the surrogate model. Typically, a sparse linear model is chosen as the surrogate model and the feature weights are considered as feature attributions. More specifically, given a black-box classifier f, a data point to explain x, and a family of possible explanations G, LIME trains the surrogate model by optimizing the following function:

$$\arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{2.1}$$

where $\pi_x(x')$ is the proximity measure that defines the distance between inputs x and x' (e.g., cosine or l_2 distance), and $\Omega(g)$ is the model complexity of g (e.g., number of non-zero weights in a linear model). the loss function L measures how close the explanation is to the prediction of the original model.

$$L(f, g, \pi_x) = \sum_{x' \in X'} \pi_x(x')(f(x) - f(x'))^2$$

where X' is the sampled set of perturbed data points in the neighborhood of x.

Figure 2.1 shows a toy example to provide intuition on how LIME works. In this example, the black-box f is a binary classifier, and its decisions are shown by blue or pink regions. To explain the instance x shown by the bold red cross, LIME samples other data points in the neighborhood of x, gets their predictions using f, weighs them based on their proximity to x, and then trains the surrogate linear model, which is shown by the black dashed line.

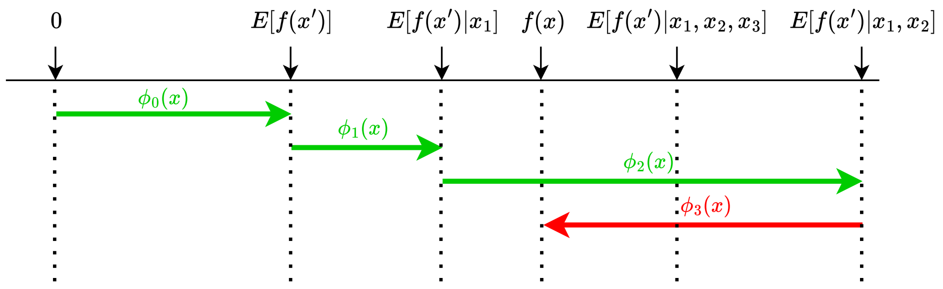


Figure 2.2: A toy example demonstrating SHAP feature attributions [48]

On the other hand, **SHAP** [48] uses intuition from Shapley values [70] in game theory to compute the contribution of each feature to the prediction. In SHAP, $\phi_j(x)$, the contribution of feature j to x's prediction (known as *SHAP values*), is calculated by the expected change in the model prediction when including j in an arbitrary coalition of features. The SHAP method explains the prediction $f(x)$ as the sum of SHAP values, plus the expected outcome if all features were unknown (referred to as the *base value* ϕ_0):

$$f(x) = \phi_0 + \sum_{j=1}^{M} \phi_j(x)$$

Figure 2.2 displays how SHAP values explain the output as a sum of feature attributions ϕ_i of each feature for a single prediction.

However, since forming all coalitions of the input features is not practical, SHAP applies an extension of LIME by introducing its own definition of proximity and sampling method and ignoring the model complexity by setting the Ω to 0. In SHAP, the sampled data points in the neighborhood of x, X', are generated by considering a subset of x's features and masking the other features. The proximity measure $\pi_x(x')$ is computed using the formula:

$$\pi_x(x') = \frac{M-1}{\binom{M}{|x'|}|x'|(M-|x'|)} \tag{2.2}$$

where M is the total number of features and $|x'|$ denotes the number of features in the subset.

Transitioning to another prevalent explanation technique, **Counterfactual Explanations** aim to find minimal changes in the input that lead to getting a different prediction from the model [61]. These explanations allow system users to delve into *what-if* scenarios regarding the model's decisions. Formally, given a black-box classifier f, a data point to explain x, and a set of counterfactual examples X' where for each $x' \in X'$ we have $f(x') \neq f(x)$, a minimal counterfactual example x^* is defined as:

$$x^* = \underset{x' \in X'}{\arg\min}\, \mathcal{D}(x, x')$$

in which $\mathcal{D}$ is the distance function, which shows the dissimilarity between x and x'. X' is also referred to as the *search space* for counterfactuals. Furthermore, the counterfactual explanation $\mathcal{E}$ shows the perturbation needed for turning x into x'. For instance, with numerical vectors, the counterfactual explanation would be $\mathcal{E} = x' - x$.

Counterfactual explanations are categorized into *synthetic* and *instance-based* based on the search space X'. In instance-based explanations, X' is a subset of real-world data points. This ensures high plausibility, but the global minimal explanation might not be found within the limited search space. Conversely, there is no constraint on X' when finding synthetic explanations. They may include synthetic data points — those not present in the original dataset — leading to minimal but potentially less plausible explanations [26].

2.3.2 Explainable AI in Graph-based Models

Explainability approaches for explaining decisions in graph analysis tasks and graph neural networks (GNNs) have been thoroughly investigated [82]. These studies involve explaining decisions for tasks like node classification, link prediction, and graph classification. In graph analysis, the relevant input features can include graph nodes, node attributes, edges, or subgraphs. As a result, explanation methods illuminate the importance and influence of these features in driving the models' predictions and overall outcomes.

In the factual explanation domain, GNNExplainer [81] identifies a subgraph and a subset of node attributes as explanations by learning edge and feature masks such that the mutual information (MI) with GNN's predictions are maximized. PGExplainer [49] embraces the same MI importance concept, but employs the GNN node embeddings to train a model which predicts the probability of edge existence in the mask. On the counterfactual side, [47] find minimal graph adjacency matrix perturbations to change the GNN prediction.

2.3.3 Explainable AI in Ranking Models

Due to the significant advancements in neural ranking models, a variety of solutions have been proposed to shed light on the black-box rankers. For saliency explanations, [71, 11] adapted LIME [63] to explain the relevance of a document to a query. [50] extracted important words from top-k documents to explain the preference pairings in ranked lists.

In the category of counterfactual explanations, [64] demonstrated an efficient framework to explain a document ranking system that provided document and query perturbations, in addition to instance-based counterfactual explanations.

In addition, counterfactual explanations are found beneficial for interpreting top-ranked recommendations by recommender systems. [24, 75, 73] find minimal actionable perturbations on the information network that flip the recommendation decision.

Chapter 3

The ExES Framework

We begin this chapter by defining the preliminaries of expert search and team formation systems and the notations that we use throughout this thesis (Section 3.1). We then present an overview of the architecture of ExES (Section 3.2). Following this, we declare the formulation of the black-box as a binary classification task (Section 3.3, and after that we describe the methodologies for generating saliency (Section 3.4) and counterfactual (Section 3.5) explanations. Later on, we analyze the time complexity of ExES procedures, with and without the pruning strategies (Section 3.6). Finally, we formulate the team formation explanations, denoting the differences from expert search formulations 3.7.

3.1 Preliminaries

Let $S = \{s_1, s_2, \ldots, s_l\}$ be the universe of skills, and $G = (P, E)$ be a node-labeled collaboration network, with individuals $P = \{p_1, p_2, \ldots, p_n\}$ as nodes, and connections $E = \{(p_{i_1}, p_{j_1}), (p_{i_2}, p_{j_2}), \ldots, (p_{i_m}, p_{j_m})\}$ between individuals as edges. Each p_i possesses a set of skills S_i, for $S_i \subset S$. We define the node-label matrix of G as $L^G = \{0,1\}^{n \times l}$, where $L^G_{ij} = 1$ if and only if $s_j \in S_i$, and 0 otherwise. Also we refer to the adjacency matrix of G as $A^G = \{0,1\}^{n \times n}$, in which $A^G_{ij} = 1$ if and only if $(p_i, p_j) \in E$, and 0 otherwise.

Given a query q consisting of a set of skills ($q \subset S$) and a value of k[1], the objective of the expert search problem is to identify the top k experts that best match q. Let $\mathcal{R}_{p_i}(q, G)$

[1]In this work, we assume that k is determined by the user. However, ExES could potentially assist the user in selecting an optimal value for k. For example, if non-experts could become experts by applying a few modifications (see Section 3.5), we could infer that the initial k was set lower than optimal.

Table 3.1: Symbols used in this thesis

Notation	**Description**
S	Universal set of skills
S_i	Skill set of individual p_i
G	Collaboration Network
L^G	Node-label matrix of collaboration network G
A^G	Adjacency matrix of collaboration network G
$\mathcal{R}_{p_i}(q, G)$	Rank of p_i with respect to query q and network G
$\mathcal{C}_{p_i}(q, G)$	Relevance status of p_i with respect to query q and network G
$\mathcal{F}(q, G)$	Team formed for query q and network G
$\mathcal{M}_{p_i}(q, G)$	Membership status of p_i with respect to query q and network G
$\mathcal{N}(p_i)$	Neighborhood of p_i
$S_{\mathcal{N}(p_i)}$	Skills included in the neighborhood of p_i
d	Neighborhood radius
b	Beam size
t	Number of candidate features
γ	Maximum explanation size

be the rank of expert p_i, with respect to query q and the collaboration network G, produced by a solution to the expert search problem.

Furthermore, the goal of the team formation problem is to find a subset of nodes from G, such as $T = \{p_{T_1}, p_{T_2}, \ldots, p_{T_k}\} \subset P$, so that $q \subset \bigcup_{i=1}^{k} S_{T_i}$. We refer to the team formation system as $\mathcal{F}$, where $\mathcal{F}(q, G)$ is the formed team given the query q and the collaboration network G.

A glossary of relevant notations is presented in Table 3.1.

3.2 System Overview

Figure 3.1 shows the architecture of ExES. A user issues a query of keywords, representing the desired skills, to a black-box expert search/team formation system, which outputs a ranked list or a team of experts (recall Figure 1.1). The user selects an individual from the collaboration graph whose presence or absence in the output is to be explained. ExES then repeatedly probes the expert search system using perturbed inputs and applies saliency

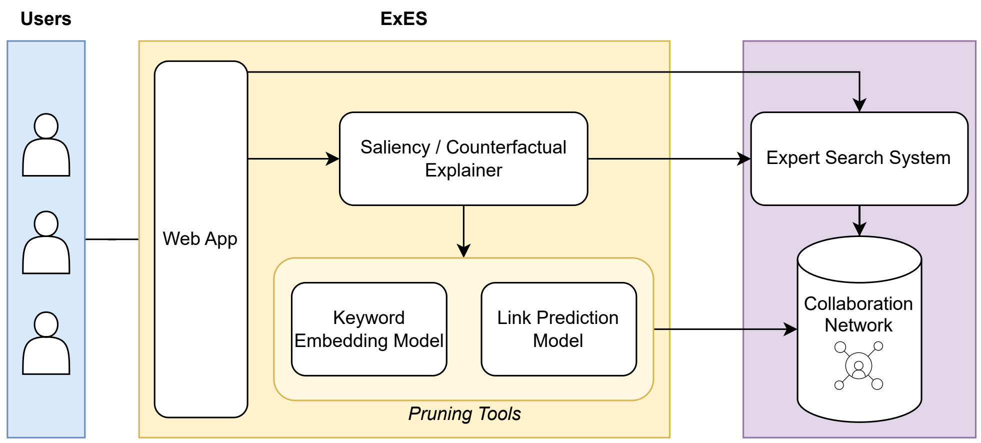


Figure 3.1: ExES Architecture

and counterfactual explanation methods. Along the way, ExES uses pruning strategies—keyword embedding similarity and link prediction—to speed up the explanation search. Finally, various types of explanations are displayed. In Sections 3.4 and 3.5, we discuss the details of the explanation and pruning methods.

3.3 Explaining Expert Search Systems

Inspired by methods such as PageRank [10] and graph neural networks [62, 31], state-of-the-art expert search systems consider a variety of signals from the collaboration network when ranking a given node p_i: its skills, the skills of its collaborators and the network structure around it. This is because expertise "propagates" throughout the collaboration network: even if p_i itself does not possess skill s_i, it may indirectly hold some expertise in, or be able to easily acquire s_i, if its collaborators are experts in s_i [69, 15]. The goal of ExES is to explain the decision-making process of black-box expert search systems, in terms of these network features, given that we can only probe the system and observe its output.

In ExES, we consider the following network features: the skills requested in the query, the skills held by each node, and the edges in the collaboration network. Later in this section, we will discuss pruning strategies that prioritize the network structure around a given node to find the most influential features.

Next, we describe how to assess feature importance in the context of expert search. The first step is to cast this problem as a binary classification problem, inspired by prior work in explainable information retrieval [71, 64]. To do so, given a query q against a collaboration graph G, we ask, for a given node p_i, whether $\mathcal{R}_{p_i}(q, G) \leq k$. Let $\mathcal{C}_{p_i}(q, G)$ be the resulting *relevance status*, true if p_i was deemed to be an expert (i.e., ranked inside the top-k) and false otherwise. This formulation allows us to use approaches for post-hoc classifier explanations through feature perturbations. In particular, ExES implements *saliency* methods, also known as feature attribution methods, that assign an importance score to each feature, as well as *counterfactual* methods that identify minimal perturbations to the input that would flip the model's output.

3.4 Saliency Explanations

For saliency, we use SHAP [48], a popular explanation method that computes an importance score for each feature by probing a model with various perturbations of the inputs. Intuitively, the higher the score, the more likely it is that a change to this feature would change the model's prediction, which in our case is the relevance status $\mathcal{C}_{p_i}(q, G)$.

To explain the relevance status of node p_i (with respect to some query) using SHAP, a trivial approach is to find the SHAP value for all input features, i.e., every query keyword, every skill assigned to every node, and every edge in the collaboration network. When confronted with large real-world collaboration networks, we require pruning methods to deliver explanations to users at interactive speeds. Specifically, we focus on the network structure in the *neighborhood* of p_i, defined as the induced subgraph of nodes located within a distance threshold d from p_i (see Sections 4.3.1 and 4.3.3 for details and analysis on choosing d). We refer to the neighborhood of p_i as $\mathcal{N}(p_i)$, and denote the skills included in the neighborhood as $S_{\mathcal{N}(p_i)}$. Thus, when it comes to skills, the features that will be scored by SHAP are the skills mentioned in the query as well as the skills in $S_{\mathcal{N}(p_i)}$.

Example. Here, we demonstrate an example of the saliency explanations of expert skills, on the DBLP collaboration network. Details about datasets are further provided in Section 4.2.2. In this example on expert search in the academic collaboration network, given the query "social graph" and $k = 10$, the user wants to explain the relevance status of "Jure Leskovec", ranked 10th. Figure 3.2 displays the SHAP values of Leskovec's skills in a plot, named the *force plot*. In the force plot, SHAP values of skills are shown on the X-axis. The *base value* shows the expected value of the relevance status for an expert with p_i's collaborations, and the current relevance status is shown by $f(inputs)$ on the plot. Each arrow's size shows the SHAP value of the corresponding skill, where green arrows show positive impact and red arrows show negative impact on the ultimate relevance status.

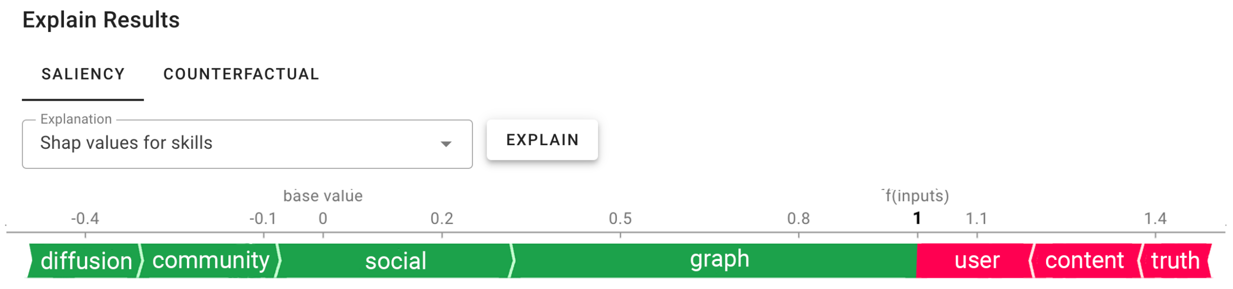


Figure 3.2: SHAP values for skills of selected expert

According to the plot, the skills *graph*, *social* and *community* have the most positive effects, and skills *user* and *content* negatively impact Leskovec's position inside the top-k.

Likewise, we only consider edges within a distance threshold d of p_i to narrow down the search space. Furthermore, ExES uses the following strategy to select influential edges for SHAP scoring. We initialize a queue $\mathcal{Q}$ of *impactful experts* with p_i, and an empty set $\mathcal{I}$ of *impactful links*. Starting from p_i, in each iteration, we expand the first unexpanded impactful expert from $\mathcal{Q}$, denoted as p_x, and calculate the SHAP values of its incident edges. For edges (p_x, p_y) with absolute SHAP values beyond a specified threshold τ, we add them to $\mathcal{I}$, and append p_y to the end of $\mathcal{Q}$. This threshold is defined to limit the branching factor at each stage. In the end, we calculate the SHAP values of only the potentially impactful links in $\mathcal{I}$.

Example. For the same query and expert in the aforementioned example, we demonstrate the saliency explanations for edges in the collaboration network. Figure 3.3 shows the SHAP values of Leskovec's connections (only those selected by the pruning rules) using a node-link diagram, where green edges show a positive effect and red edges show a negative effect toward the relevance status. Edge opacity indicates the degree of importance of each edge, and the size of the nodes shows the rank of the corresponding node with respect to this query. The higher the expert is in the ranking, the larger the node is in the plot. This plot shows that while some of Leskovec's neighbors, like Prem Melville, might not attain high ranks, their presence contributes to Leskovec's relevance for the query, securing a position in the top-k.

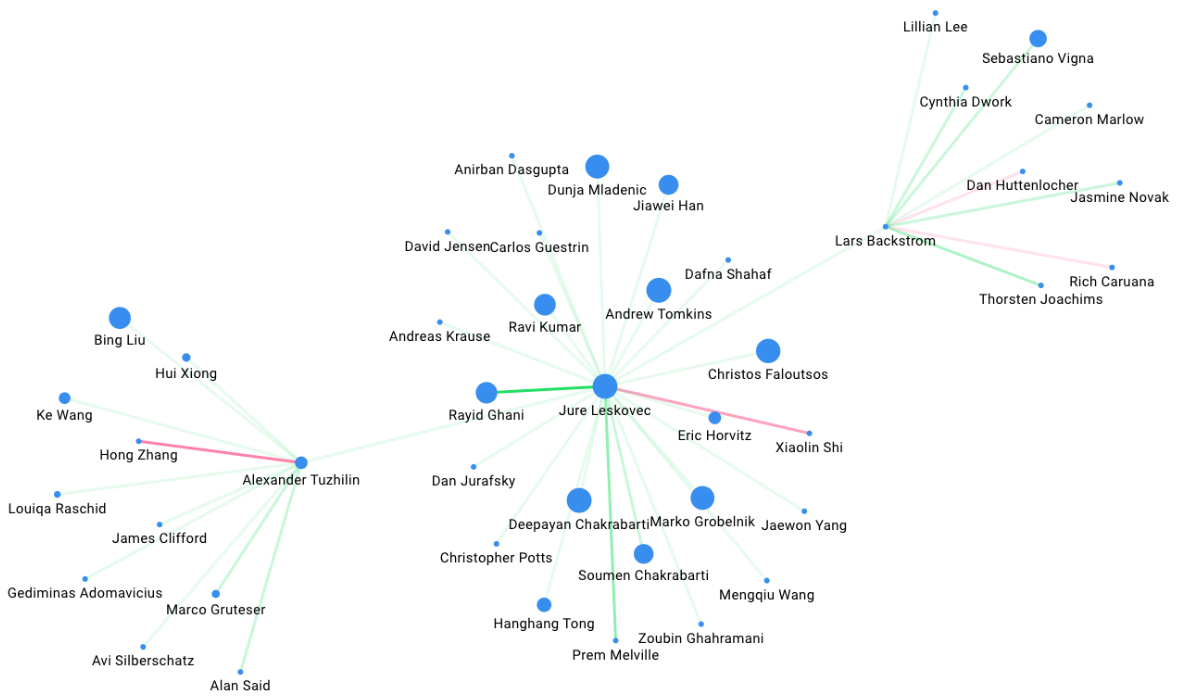


Figure 3.3: SHAP values for collaborations of selected expert

3.5 Counterfactual Explanations

To explain the relevance status of a node p_i counterfactually, we seek small perturbations to the search query q and the collaboration network G that flip $\mathcal{C}_{p_i}(q, G)$. That is, we identify changes that would turn experts (ranked in the top-k) into non-experts (ranked outside the top-k) and vice versa. ExES explores perturbations to skill sets, the search query, and collaborations in the network.

Explanation Minimality. Given the importance of generating human-friendly explanations, a good explanation should include a concise set of features by intuition[2] [78, 54]. Hence, the concept of minimality is a key aspect of ExES, which we formalize in this section.

Given an explanation $\mathcal{E}$, we define the size of $\mathcal{E}$ by the number of changes made to the collaboration network or the query within $\mathcal{E}$. Therefore, the objective is to derive optimal explanations such as $\mathcal{E}^*$ with the minimum number of total perturbations to the input parameters. This could be formulated as:

$$\mathcal{E}^* = \arg\min_{\mathcal{E}} |\mathcal{E}|$$

However, given the expansive sizes of real-world collaboration networks, we need to prune the search space for explanations, by selecting a subset of nodes, skills, or edges for the perturbations. We describe the method for search space pruning and selecting candidate features for every explanation type in detail.

Generating Explanations. Algorithm 1 presents the core framework for generating counterfactual explanations. It employs beam search with a specified beam width, b, and expands potential perturbations up to a maximum size, γ. Here, γ serves as a controlling parameter, which prevents the search algorithm from pursuing expansions with large sizes, thereby ensuring that the search process concludes within a reasonable timeframe. The search process continues until it finds top e minimal explanations within the search space.

In line 1, we select t candidate features to include in the perturbations. These features involve skills added/removed from individuals, query keywords, or edges added/removed from the collaboration network. These feature selections are guided by the word embedding

[2]In this work, we define the notion of minimality based on the literature. In related work on counterfactual explanations, minimality is typically referred to as sparsity (explanation size) [26], minimizing the number of different features between counterfactual and original data points. However, different use cases might prioritize additional metrics, such as robustness or fidelity, at the expense of explanation size. We reserve the consideration of multiple objectives in explanation minimality for future work.

model W or the link prediction model L (see Sections 3.5.1, 3.5.2, and 3.5.3 for details on how these t features are chosen).

In line 2-4, we initialize the result explanations set $\mathcal{E}$, the beam search queue which contains an empty perturbation, and the initial relevance status for the target individual p_i.

During the while loop (line 5), the beam search algorithm iteratively expands the perturbation states in *queue*, until e explanations are found or the queue is empty. Each iteration begins with initializing *expandedQueue*, which contains the expanded states, with an empty set. Then, we expand every state in *queue* by appending each candidate perturbation to it (line 9). After applying the perturbations to G or q (line 10), we compute the new rank and relevance status (lines 11, 12). A perturbation that changes the relevance status is considered successful and added to $\mathcal{E}$ (lines 13, 14). Furthermore, perturbations not exceeding γ are queued in *expandedQueue*, along with their new rank, for further expansion (lines 16 and 17).

After that, the top b states are selected from *expandedQueue*, based on the new rank (line 21), and passed to *queue* for the next iteration (lines 23 and 24). The sorting direction is determined based on the initial relevance; i.e. if the initial relevance is 1, the sorting direction is descending, otherwise it is ascending. After the search has ended, the generated explanations in $\mathcal{E}$ are returned (line 27).

Below, we outline three types of counterfactual explanations supported by the ExES framework.

3.5.1 Counterfactual Skill Explanations

We consider adding or removing skills to the skill set of p_i or collaborators in p_i's neighborhood. Given a query q, adding skills that are directly in q or similar to those in q, to p_i or their neighborhood collaborators, should move p_i up in the ranking, and removing such skills should do the opposite.

However, due to the large space of candidate skills, the search space of the above perturbations needs to be pruned. To do this, we train a word embedding model, denoted W, such as Word2Vec [53], which learns representations for skills to identify similar or dissimilar skills. We train W on the textual expertise corpus from which the collaboration network labels were assigned. We use W to limit the skills that are included in the perturbations, to maintain efficiency. In addition, taking this contextual similarity into account can improve the actionability of our explanations. For instance, we can make sure the skills we suggest for individuals would be relevant to their original skill sets.

Algorithm 1 The core for generating counterfactual explanations

```
Input: Collaboration network $G$, Query $q$, Ranker $\mathcal{R}$, Relevance status function $\mathcal{C}$, Word embedding $W$,
    Link prediction model $L$, expert $p_i$, Number of explanations $e \geq 1$, Number of candidate features
    $t \geq 1$, Beam width $b$, Maximum perturbation size $\gamma$
Output: List of $e$ explanations $\mathcal{E}$
 1: $candidateFeatures \leftarrow \text{GETCANDIDATEFEATURES}(t, G, W, L)$        ▷ Depends on the explanation type
 2: $\mathcal{E} \leftarrow \emptyset$
 3: $queue \leftarrow \{\emptyset\}$
 4: $initialRelevance \leftarrow \mathcal{C}_{p_i}(q, G)$
 5: while $|\mathcal{E}| < e$ and $|queue| > 0$ do
 6:     $expandedQueue \leftarrow \emptyset$
 7:     for all $perturbation \in queue$ do
 8:         for all $feature \in candidateFeatures$ do
 9:             $expandedPerturbation \leftarrow perturbation \cup \{feature\}$
10:             $G', q' \leftarrow \text{APPLY}(perturbation, G, q)$
11:             $newRank \leftarrow \mathcal{R}_{p_i}(q', G')$
12:             $newRelevance \leftarrow \mathcal{C}_{p_i}(q', G')$
13:             if $newRelevance \neq initialRelevance$ then
14:                 $\mathcal{E} \leftarrow \mathcal{E} \cup expandedPerturbation$
15:             end if
16:             if $|expandedPerturbation| < \gamma$ then
17:                 $expandedQueue \leftarrow expandedQueue \cup \{\langle newRank, expandedPerturbation\rangle\}$
18:             end if
19:         end for
20:     end for
21:     $expandedQueue \leftarrow \text{SELECTTOPK}(expandedQueue, b)$
22:     $queue \leftarrow \emptyset$
23:     for $i \in [0, b)$ do
24:         $queue \leftarrow queue \cup \{expandedQueue[i][1]\}$
25:     end for
26: end while
27: return $\mathcal{E}$
```

Let $\mathcal{E}$ be a counterfactual explanation, which contains a perturbation to the node skills of the collaboration network G. Let G' be the perturbed collaboration network and the node-label matrix of G' be $L^{G'}$. We define the difference matrix $\Delta_L = L^{G'} - L^G$.

Skill addition. For a counterfactual explanation that includes skill additions to p_i's neighborhood $\mathcal{N}(p_i)$, the explanation would be in the form of:

$$\mathcal{E} = \{(p_x, s_y) | \Delta_{Lxy} = 1, p_x \in \mathcal{N}(p_i)\}$$

Skill removal. For counterfactual explanations that contain removing skills from p_i's neighborhood $\mathcal{N}(p_i)$, the explanation would be in the form of:

$$\mathcal{E} = \{(p_x, s_y) | \Delta_{Lxy} = -1, p_x \in \mathcal{N}(p_i)\}$$

Therefore, to find the minimal explanations $\mathcal{E}^*$, we solve the following optimization problem:

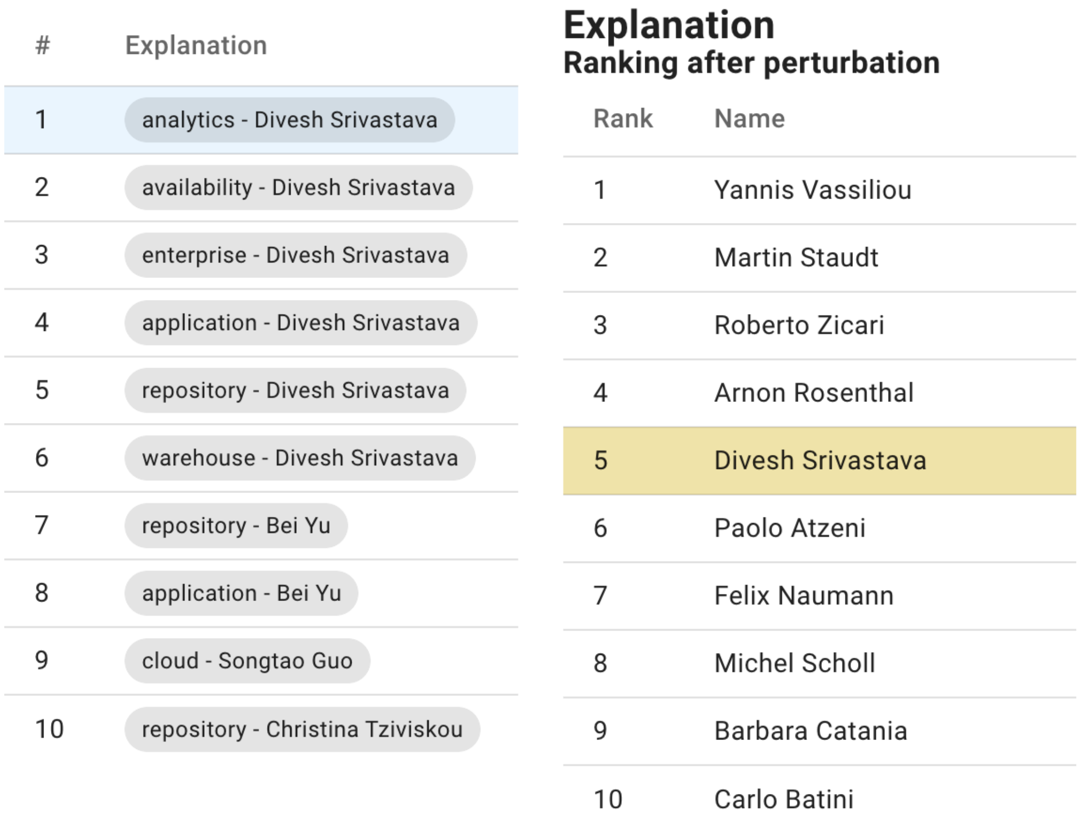
$$\begin{gathered}{\Delta_L}^* = \underset{\Delta_L}{\arg\min} \, ||\Delta_L||_0 \\ \text{s.t } \mathcal{C}_{p_i}(q, G') \neq \mathcal{C}_{p_i}(q, G)\end{gathered}$$

Then, the minimal explanation would be

$$\mathcal{E}^* = \{(p_x, s_y) | \Delta^*_{Lxy} \neq 0\}.$$

To determine which skills to add, we start by selecting the t most similar skills to the query and S_i, based on W. Then, using beam search, we find minimal combinations of skills that improve p_i's ranking. We use a similar approach to find minimal skills to remove from experts, starting from skills in $S_{\mathcal{N}(p_i)}$ that are the most similar to the query, followed by the beam search.

Example. As an example, consider the query "database management quality" on the DBLP collaboration network with $k = 10$, where "Divesh Srivastava" is ranked 11th, i.e., outside the top-k. Suppose the user wants to counterfactually explain why this individual was ranked outside the top-10. Figure 3.4 shows a list of skill addition explanations. These skills are either added to the target individual (Divesh Srivastava), or their neighbors (Bei Yu, Songtao Guo, Christina Tziviskou). Also, as an example, we see that adding the *analytics* skill improves Divesh Srivastava's rank for this query to fifth.



#	Explanation
1	analytics - Divesh Srivastava
2	availability - Divesh Srivastava
3	enterprise - Divesh Srivastava
4	application - Divesh Srivastava
5	repository - Divesh Srivastava
6	warehouse - Divesh Srivastava
7	repository - Bei Yu
8	application - Bei Yu
9	cloud - Songtao Guo
10	repository - Christina Tziviskou

Explanation

Ranking after perturbation

Rank	Name
1	Yannis Vassiliou
2	Martin Staudt
3	Roberto Zicari
4	Arnon Rosenthal
5	Divesh Srivastava
6	Paolo Atzeni
7	Felix Naumann
8	Michel Scholl
9	Barbara Catania
10	Carlo Batini

Figure 3.4: List of counterfactual skill explanations

3.5.2 Counterfactual Query Explanations

To produce counterfactual explanations for the relevance status of a node p_i in terms of the search query, we consider query augmentation. Query augmentation is a commonly employed technique, aimed at improving the recall of search systems [77]. This method bridges the gap between the query and ranked items by adding relevant terms to the query. It effectively addresses issues such as ambiguity, vocabulary mismatch, and absence of specific terms in the initial query. Query augmentation has also proven effective in explaining ranking models [72]. It is worth noting that since the input queries for expert search are typically brief and consist of a few keywords, perturbing the queries through keyword removal is often impractical. Removal of keywords might change the query's intent, resulting in vague and improper queries. Plus, such limited perturbations through keyword removal fail to yield successful counterfactuals in most cases. Therefore, we only consider keyword addition to the query, where adding skills from S_i into q should improve p_i's ranking, and adding skills unrelated to S_i into q should do the opposite.

Let $q' \subset q$ be a perturbed query. In this case, the corresponding explanation $\mathcal{E}$ would be $q' - q$, which is the set of keywords added to q. Hence, we find minimal query perturbations by solving the following optimization problem:

$$q'^* = \arg\min_{q'} |q' - q|$$
$$\text{s.t } \mathcal{C}_{p_i}(q', G) \neq \mathcal{C}_{p_i}(q, G)$$

Then, the minimal explanation would be $\mathcal{E}^* = q'^* - q$.

To find query augmentations that improve the ranking of p_i to bring p_i into the top-k list, we again prune the search space by identifying t most similar skill keywords to the expert skill set S_i and the query q, according to the keyword embedding model W. We then use beam search to find counterfactual perturbations. To find counterfactual query explanations that evict an expert p_i from the top-k list, we run the aforementioned method, starting with skills similar to the query, however, different from S_i.

3.5.3 Counterfactual Collaboration Explanations

Finally, to create counterfactual explanations for $\mathcal{C}_{p_i}(q, G)$ in terms of collaborations, we consider the introduction of new connections to p_i's neighborhood (adding edges to G) or the removal of current connections (deleting edges from G). Intuitively, p_i's ranking can improve if we add an edge for a node in p_i's neighborhood to an expert for the given query q, and vice versa.

Let $\mathcal{E}$ be a counterfactual explanation, which contains a perturbation to the edges of the collaboration network G. Let the perturbed collaboration network be G', and the adjacency matrix of G' be $A^{G'}$. We define the difference matrix $\Delta_A = A^{G'} - A^G$.

Collaboration addition. For counterfactual explanations involving the addition of collaborations, the explanation format would be:

$$\mathcal{E} = \{(p_x, p_y) | \Delta_{Axy} = 1, x \in \mathcal{N}(p_i) \wedge y \notin \mathcal{N}(p_i)\}$$

Collaboration removal. For a counterfactual explanation that includes collaboration removals, the explanation would be in the form of:

$$\mathcal{E} = \{(p_x, p_y) | \Delta_{Axy} = -1, x \in \mathcal{N}(p_i) \wedge y \in \mathcal{N}(p_i)\}$$

Hence, we find minimal collaboration perturbations in the corresponding search space by solving the following optimization problem:

$$\begin{aligned} &\Delta_A^* = \underset{\Delta_A}{\arg\min} \, ||\Delta_A||_0 \\ &\text{s.t } \mathcal{C}_{p_i}(q, G') \neq \mathcal{C}_{p_i}(q, G) \end{aligned}$$

Then, the minimal explanation would be

$$\mathcal{E}^* = \{(p_x, p_y) | \Delta^*_{Axy} \neq 0\}.$$

To prune the search space of collaboration explanations, we leverage a link prediction model, such as Graph Auto-encoder (GAE) [39], denoted L, trained on the collaboration network connections. Surveys on link prediction methods [45] highlight GAE's strong performance in achieving high link prediction accuracy on real-world graphs, alongside its ease of implementation [1]. Moreover, ExES works independently from the internal mechanisms of the L, prioritizing its prediction accuracy. These factors led to our decision to employ GAE as the link prediction model in ExES. We employ L as a recommender for potential future collaborations between experts, to eliminate less promising collaborations from the search space of counterfactuals. This helps us to reduce the size of the search space and maintain efficiency. Further, by considering edges with a high probability of being formed in the perturbations, ExES can hold a strong level of actionability.

We use an iterative strategy based on beam search to construct these explanations efficiently. To improve p_i's ranking, we locate the most likely candidates for future collaborations within $\mathcal{N}(p_i)$, using the link prediction model L. Then, we start forming minimal sets of these new collaborations, by expanding sets with the greatest improvement in p_i's ranking, using beam search. Furthermore, to identify minimal sets for edge removal, we apply a similar beam search method by progressively expanding sets of edges in $\mathcal{N}(p_i)$ whose elimination worsens p_i's ranking the most.

Example. As an example, we revisit the previous scenario from Section 3.5.1 with the query "database management quality" on the DBLP collaboration network, with $k = 10$, and "Divesh Srivastava" as the target individual for explanation. The user can request explanations using collaboration additions. Figure 3.5 shows a list of counterfactual collaboration additions that place Divesh Srivastava inside the top-10. Figure 3.6 visualizes an example counterfactual explanation, having the added edge highlighted.

3.6 Complexity

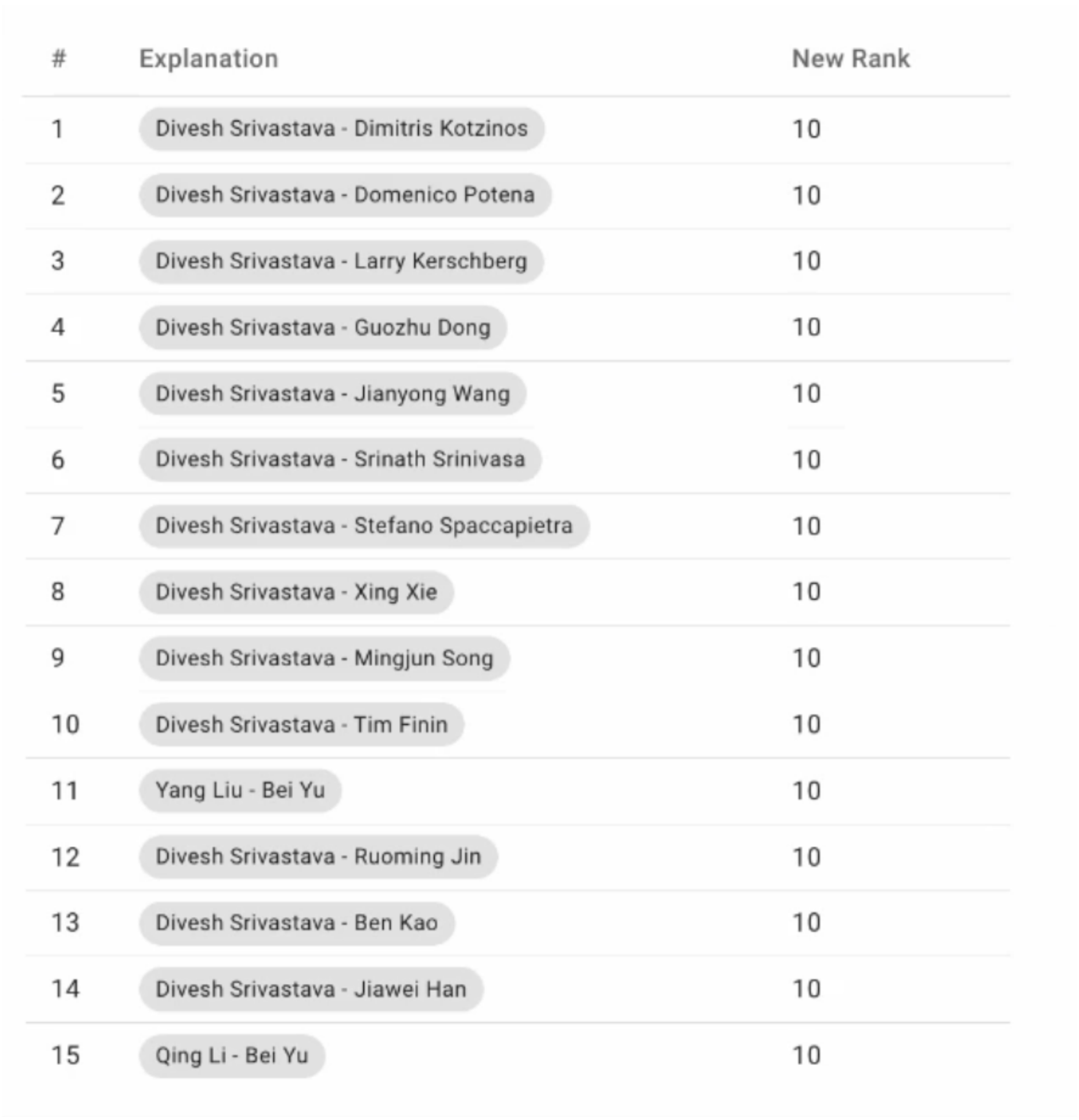
In this section, we analyze the time complexity of ExES for each explanation type.

3.6.1 Saliency Explanations

According to [48], the time complexity of calculating SHAP values of a prediction with M input features is $O(2^M \times T)$, where T is the running time of the black-box for one pass. Since ExES builds saliency explanations on top of the SHAP algorithm, we list the time complexity of each saliency explanation method of ExES in table 3.2. In the provided statements, $T_{ranking}$ is the time complexity of running the black-box ranker for one pass.

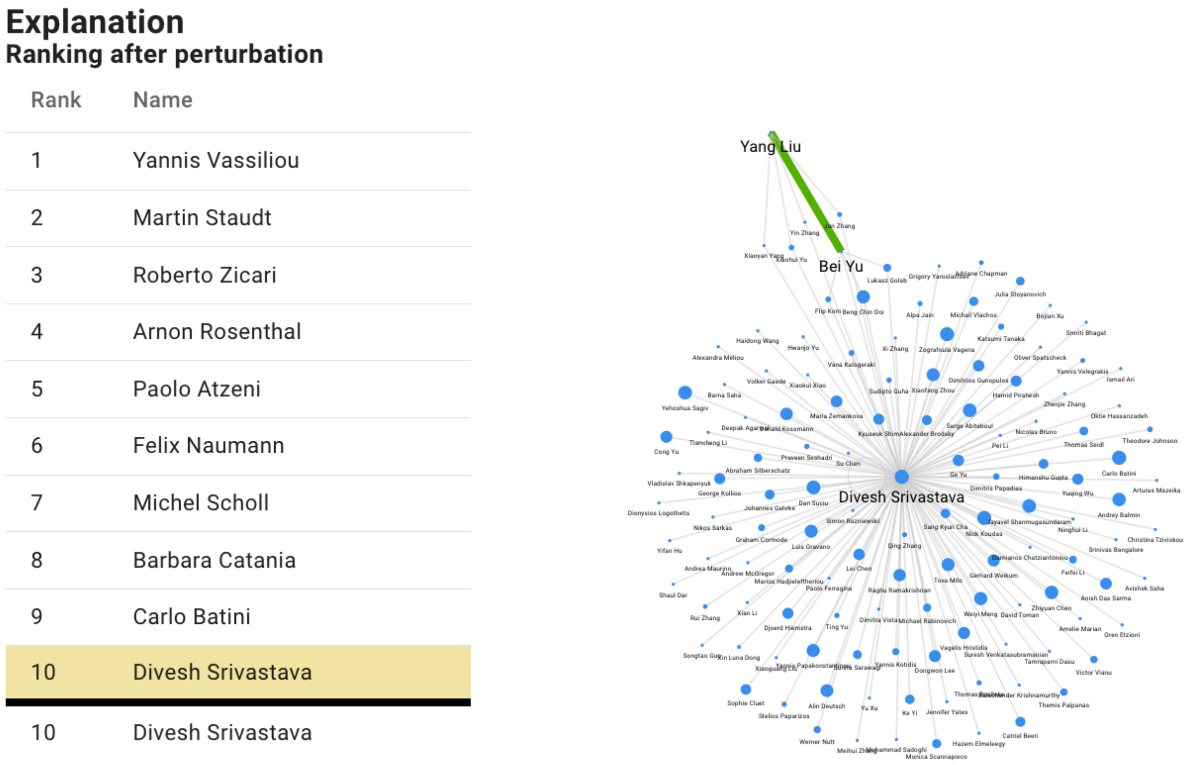
When computing SHAP values for skills to explain the relevance status of p_i, the original input would include all skills possessed by individuals in G, a total of $\sum_{i=1}^{n} |S_i|$ features. In the worst case, this would be equal to $|P| \times |S|$. However, ExES bounds the feature set for SHAP value computation by only considering the skills of individuals within p_i's neighborhood. This reduces the number of input features for SHAP to $\sum_{x|p_x \in \mathcal{N}(p_i)} |S_x| = |S_{\mathcal{N}(p_i)}|$.

Similarly, for calculating SHAP values of collaborations in G, the original features would be every edge in G; therefore, the size of input features for SHAP is $|E|$ which is extremely large in real-world collaboration networks. ExES limits the input features by only including edges within the neighborhood of p_i, $\mathcal{N}(p_i)$. Since the effective edges are



#	Explanation	New Rank
1	Divesh Srivastava - Dimitris Kotzinos	10
2	Divesh Srivastava - Domenico Potena	10
3	Divesh Srivastava - Larry Kerschberg	10
4	Divesh Srivastava - Guozhu Dong	10
5	Divesh Srivastava - Jianyong Wang	10
6	Divesh Srivastava - Srinath Srinivasa	10
7	Divesh Srivastava - Stefano Spaccapietra	10
8	Divesh Srivastava - Xing Xie	10
9	Divesh Srivastava - Mingjun Song	10
10	Divesh Srivastava - Tim Finin	10
11	Yang Liu - Bei Yu	10
12	Divesh Srivastava - Ruoming Jin	10
13	Divesh Srivastava - Ben Kao	10
14	Divesh Srivastava - Jiawei Han	10
15	Qing Li - Bei Yu	10

Figure 3.5: List of counterfactual collaboration explanations



Explanation

Ranking after perturbation

Rank	Name
1	Yannis Vassiliou
2	Martin Staudt
3	Roberto Zicari
4	Arnon Rosenthal
5	Paolo Atzeni
6	Felix Naumann
7	Michel Scholl
8	Barbara Catania
9	Carlo Batini
10	Divesh Srivastava
10	Divesh Srivastava

Figure 3.6: Visualization of Counterfactual collaboration explanation

Table 3.2: Time Complexity of Saliency Explanation methods

Explanation Type	**Time Complexity**	
	with Pruning	**without Pruning**
SHAP values of experts' skills	$O(2^{\lvert S_{\mathcal{N}(p_i)}\rvert} \times T_{ranking})$	$O(2^{\lvert P\rvert \times \lvert S\rvert} \times T_{ranking})$
SHAP values of query keywords	$O(2^{\lvert q\rvert} \times T_{ranking})$	$O(2^{\lvert q\rvert} \times T_{ranking})$
SHAP values of Collaborations	$O(2^{\lvert \mathcal{N}(p_i)\rvert} \times T_{ranking})$	$O(2^{\lvert E\rvert} \times T_{ranking})$

Table 3.3: Time Complexity of Counterfactual Explanation methods

Explanation Type	**Time Complexity**	
	with Pruning	**without Pruning**
Skill Counterfactuals	$O(T_W + b \times t \times \gamma \times \lvert\mathcal{N}(p_i)\rvert \times T_{ranking})$	$O(2^{\lvert S\rvert \times \lvert P\rvert} \times T_{ranking})$
Query Counterfactuals	$O(T_W + b \times t \times \gamma \times T_{ranking})$	$O(2^{\lvert S\rvert} \times T_{ranking})$
Collaboration Counterfactuals	$O(T_L + b \times t \times \gamma \times \lvert\mathcal{N}(p_i)\rvert \times T_{ranking})$	$O(2^{\lvert E\rvert} \times T_{ranking})$

selected using a breadth-first search, the number of effective edges selected for SHAP is at most $|\mathcal{N}(p_i)|$, which is way smaller than $|E|$.

Notice that ExES doesn't apply any pruning method for calculating SHAP values of query keywords. Thus, the time complexity would be the same for ExES and exhaustive searches.

3.6.2 Counterfactual Explanations

We compare the time complexity of generating counterfactual explanations with and without the pruning strategies in Table 3.3.

When generating counterfactuals, the exhaustive strategy involves calculating the new ranking and relevance status for every subset of the search space (ordered by the perturbation size) and then picking the minimal perturbations that flip the relevance status.

For skill addition counterfactuals, the size of the search space is determined by the count of missing skill-individual pairs in G, which is equal to $\sum_{i \in P} |S - S_i|$. Conversely, for skill removal counterfactuals, it corresponds to the number of existing skill-individual pairs, which is $\sum_{i \in P} |S_i|$. These numbers could be equal to $|S| \times |P|$ in the worst case. For query counterfactuals, the search space contains every missing keyword from the query, with a maximum size of $|S - q|$. For collaboration addition counterfactuals, the search space contains every missing edge, which has a size of $\binom{n}{2} - |E|$, and for collaboration removal counterfactuals, it includes all existing edges, which has the size of $|E|$.

On the other hand, as stated in Section 3.5, ExES uses beam search as the base algorithm for generating counterfactual explanations. We recall b as the beam size, and γ as the maximum explanation size.

According to [67], the time complexity for a beam search procedure with beam size b, branching factor w, and maximum depth γ is $O(b \times w \times \gamma)$. For skill counterfactuals, ExES limits the search to the target expert's neighborhood $\mathcal{N}(p_i)$ and selects t keywords as potential skill perturbations, making the search space and branching factor $t \times \mathcal{N}(p_i)$. For query counterfactual, the search space and branching factor is t, corresponding to the t keywords added to the query. In the context of collaboration counterfactuals, ExES narrows down the potential added/or removed edges to manage the search space, selecting t perturbed edges for both additions and removals involving $\mathcal{N}(p_i)$, maintaining the search space and branching factor at t.

We can observe that pruning the search space effectively manages the running time of ExES by reducing the exponential runtime of the exhaustive strategy to a polynomial runtime. In Table 3.3, T_W represents the runtime of the keyword embedding model to identify t potential keywords, and T_L represents the runtime of the link prediction model to find t potential edges for the perturbations.

3.7 Explaining Team Formation Systems

State-of-the-art neural solutions for the team formation problem solve the expert search task as an initial step. In the second step, these solutions then assemble closely connected experts and output a subgraph of the collaboration network.

ExES is capable of providing insights into the team composition. Similar to the expert search explanations (see Section 3.3), we cast the team formation problem as a binary classification task. Given a query q and a collaboration network G, we ask whether node p_i is included in the team or not. We define $\mathcal{M}_{p_i}(q, G)$, referred to as the *membership status*, which is true whenever p_i is selected by the team formation system $\mathcal{F}$ for the team (i.e., $p_i \in \mathcal{F}(q, G)$), and false otherwise. ExES can take advantage of any of the stated saliency and counterfactual explanation methods to explain why an expert p_i was included in a constructed team T, or vice versa, it can explain why an individual was not selected in T. To do so, the membership status is used as the explanation target, instead of relevance status in all explanation methods described in sections 3.4 and 3.5.

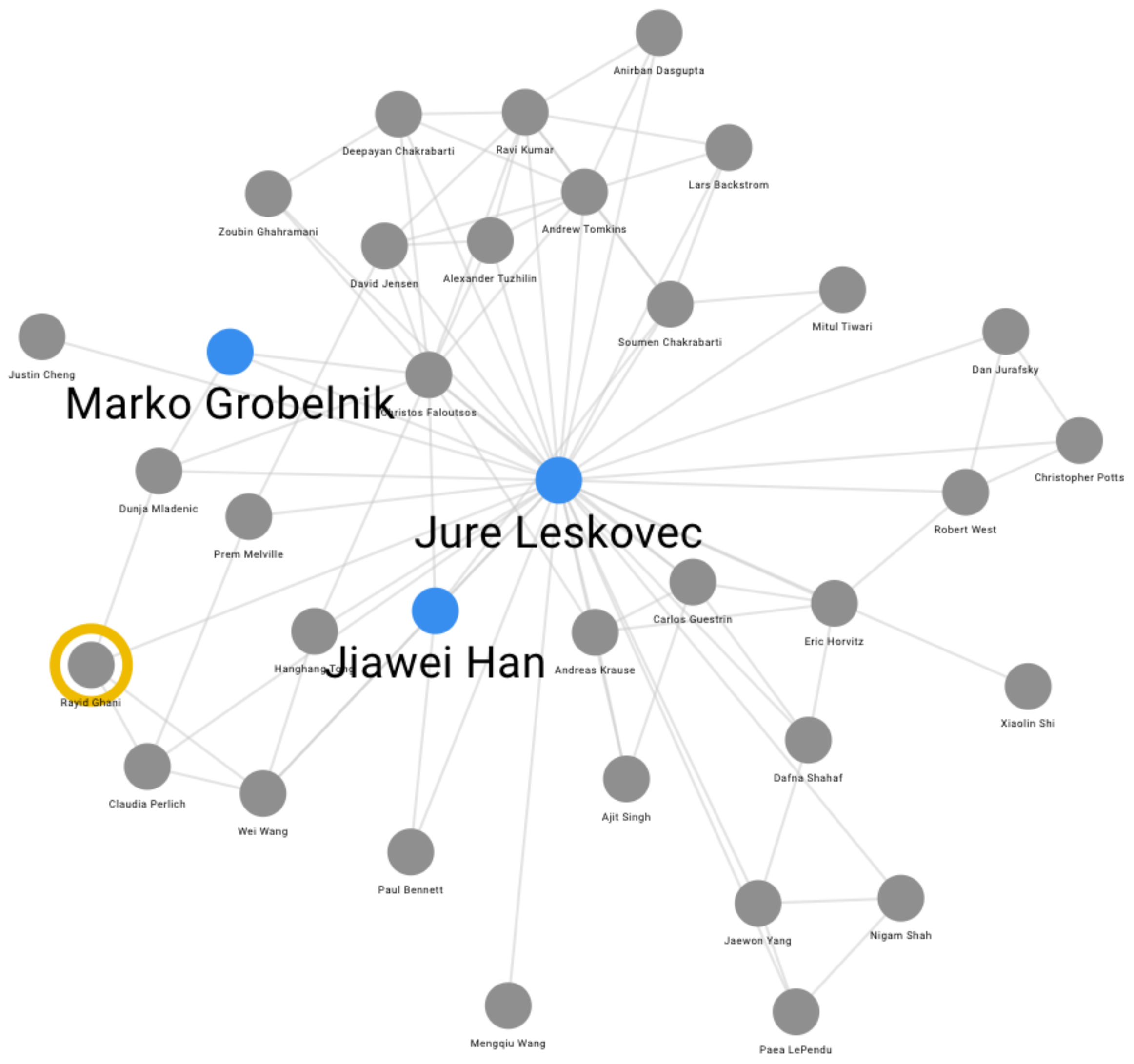
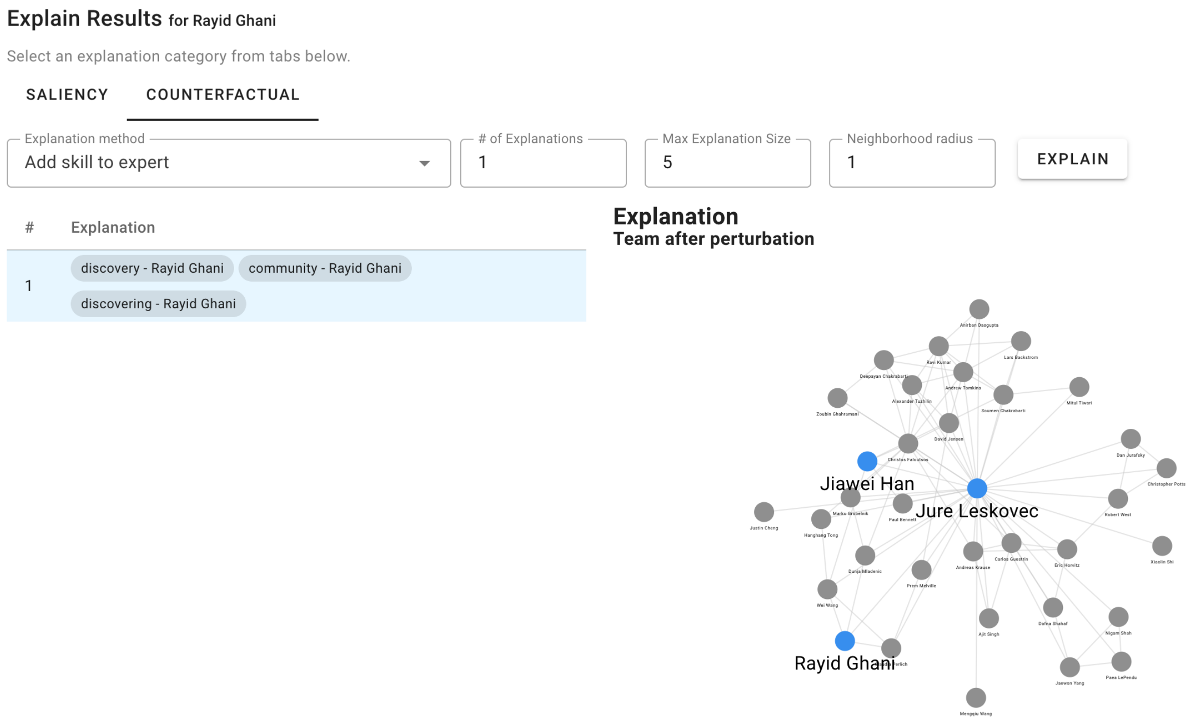


Figure 3.7: Example for team formation output. The user wants to explain why "Rayid Ghani" (indicated with an orange circle) was excluded from the team.



Explain Results for Rayid Ghani

Select an explanation category from tabs below.

SALIENCY COUNTERFACTUAL

Explanation method: Add skill to expert

# of Explanations: 1

Max Explanation Size: 5

Neighborhood radius: 1

EXPLAIN

#	Explanation
1	discovery - Rayid Ghani; community - Rayid Ghani; discovering - Rayid Ghani

Explanation
Team after perturbation

Figure 3.8: Example counterfactual explanation for team formation

Example. As an example, consider the query “social graph pattern mining” on the DBLP collaboration network. The output team includes “Jure Leskovec,” “Jiawei Han,” and “Marko Grobelnik,” as depicted in Figure 3.7, where team members are indicated by blue-colored nodes and non-members by gray nodes. The user can select the team members to explain their selection for the team, or other individuals who were not chosen, to explain their exclusion from the team. For instance, Figure 3.8 illustrates a counterfactual explanation that includes ”Rayid Ghani” as a member of the team.

The time complexity of generating explanations for team formation systems remains consistent with the principles defined in Section 3.6. However, in this context, $T_{ranking}$ is substituted with $T_{teamFormation}$, the time complexity of executing the team formation procedure for a single pass.

Chapter 4

Evaluation

In this section, we provide an overview of our comprehensive experiments designed to showcase the effectiveness and efficiency of ExES in offering easy-to-understand explanations for both the expert search and team formation systems.

4.1 Metrics

To evaluate the performance of ExES, we utilize the following metrics:

- **Explanation Size:** The size of an explanation serves as a crucial factor in assessing its quality. When generating saliency explanations, providing the feature attribution for every input feature would be a valid explanation. However, users prefer shorter explanations due to their ease of comprehension. Moreover, in the case of counterfactual explanations, smaller and simpler counterfactuals are easier to interpret and act upon[2]. For instance, in scenarios such as skill recommendations, it is easier for individuals to learn as few skills as possible.

- **Latency:** Latency refers to the time required to calculate all explanations for a target individual. This encompasses calculating feature attributions for every input feature in saliency explanations (recall Section 3.4), and generating the top e minimal explanations in counterfactual scenarios (recall Section 3.5). ExES is designed to accomplish this within a brief, reasonable timeframe.

- **Precision:** ExES aims to balance its search space reduction with precision. We evaluate ExES's precision by comparing its explanations against those generated by an exhaustive search baseline on the unpruned space. Ideally, the generated explanations should be similar to those produced without pruning.

Saliency explanations. In evaluating saliency explanations, we aim to ensure that important features identified by ExES are similarly deemed important by the exhaustive explanation baseline (i.e. receive non-zero feature attribution values). To measure this, we utilize the **Precision@k** metric. Given a set of saliency explanations generated by ExES, this metric measures the proportion of the top-k important features (ranked by the absolute value of their feature attributions) that also receive non-zero attributions in the corresponding exhaustive explanations.

Counterfactual Explanations. In the scope of counterfactual explanations, an explanation is deemed *optimal* if it is included in ground-truth explanations generated by the exhaustive baseline, or if its size matches the minimal explanation size identified by the baseline. Given a set of counterfactual explanations and their corresponding ground-truth explanations, the **Precision** is defined as:

$$\text{Precision} = \frac{\text{Number of optimal explanations}}{\text{Number of total explanations}}$$

Furthermore, we label explanations as *nearly-optimal* if their size exceeds the optimal size by at most one unit. We calculate the ratio of *nearly-optimal* explanations found by ExES in a variable named **Precision***:

$$\text{Precision*} = \frac{\text{Number of nearly-optimal explanations}}{\text{Number of total explanations}}$$

4.2 Experimental Setting

4.2.1 Environment

ExES consists of a web app frontend, developed with VueJS, and a backend REST API, developed with Flask and Python 3.10.12. We deployed the frontend and backend server and ran all of our evaluation experiments on an Ubuntu virtual machine, with an Intel Core i9-7920X CPU, 128 GB of RAM, and a GeForce RTX 4090 GPU.

4.2.2 Datasets

We evaluated ExES on two well-known datasets: *DBLP* and *GitHub*. However, since these datasets do not provide pre-constructed collaboration networks, we construct the networks

Table 4.1: Dataset statistics

Dataset	# Nodes	# Edges	# Skills
DBLP	17630	128809	1829
GitHub	3278	15502	863

from their raw data. Importantly, ExES is designed to operate on any node-labeled collaboration network (with properties defined in Section 3.1). The processes involved in dataset creation and node labeling are independent of the core functionalities of the expert search/team formation systems, as well as of ExES itself.

In DBLP [44], the collaboration network comprises academic researchers as nodes and paper co-authorship as edges. We extracted expert skills from their paper titles and abstracts. Using TF-IDF on the keywords, we assigned skills to individuals when their TF-IDF values surpassed the threshold θ_{tfidf}. This threshold ensured an average of 15 skills per expert. We established this threshold after testing various levels during dataset creation. Higher thresholds resulted in fewer skills per expert, which limited the potential experts for given queries. In contrast, lower thresholds increased the number of skills assigned to each person, thereby reducing the distinctiveness between individuals and hindering the black-box's ability to distinguish experts from non-experts.

The GitHub dataset includes GitHub users as nodes and project collaborations as edges. The skill sets of these users were deduced by applying the same TF-IDF methodology and threshold to the descriptions and tags of each user's repositories. Detailed statistics for both datasets are provided in Table 4.1.

4.3 Experiments

To evaluate the performance of ExES using the aforementioned metrics, we first prepared a set of queries to run the expert search and team formation models on them. For each dataset, we generated 100 random queries, by sampling between 3 and 5 keywords uniformly from the universe of skills (S) of the corresponding dataset. This uniform sampling method guarantees diversity in the queries, thereby ensuring a broad range of experts and teams are retrieved for our analysis.

4.3.1 Expert Search Experiments

ExES is designed to work properly for any expert search black-box, satisfying the conditions defined in Section 3.1. However, to select the black-box expert search model, we implemented an expert search system, inspired by the ideas from several state-of-the-art solutions [31, 14, 32]. The chosen architecture leverages graph convolutional networks [40] to learn representations of the experts. The model was trained in a contrastive learning process; for each query from the dataset (e.g., paper titles in DBLP, or project keywords in GitHub), the participants of that query are considered more relevant than randomly selected individuals.

To evaluate the explanations for expert search models, we first ranked experts for each query using the pre-trained model, having $k = 10$. Then, from the total retrieved individuals, we sampled 100 experts within the top-k and 100 non-experts ranked between $k+1$ and $2k$. We applied all saliency and counterfactual explanation methods (see Sections 3.4 and 3.5) for each sampled individual to explain their ranking. We ran this series of experiments with beam size $b = 30$, maximum explanation size $\gamma = 5$, number of required explanations $e = 5$, and number of candidate features $t = 10$. In addition, the neighborhood distance threshold d was set to 1 for Skill Saliency, skill counterfactual, and collaboration addition counterfactual explanations, ensuring an individual's neighborhood contains themselves and their immediate collaborators. For Collaboration Saliency and collaboration removal counterfactual explanations, we extended the neighborhood depth to 2 to incorporate 2-hop collaborations. We also set τ in calculating Collaboration SHAP values equal to 0.1. In Section 4.3.3, we will discuss how variations in these parameters affect the performance and outcomes of ExES.

We compared ExES's explanations against those created by a baseline that conducted an exhaustive search over the explanation search space. It is important to note that in saliency query explanations, there is no corresponding pruning method, hence no exhaustive baseline for comparison. To maintain efficiency during experimentation, we imposed a timeout of 1000 seconds for generating each explanation.

Furthermore, in terms of skill addition counterfactuals (recall Section 3.5.1), the exhaustive search algorithm involves probing the black-box system by adding every skill into each node of the collaboration network. This creates a substantial search space, making it infeasible to run the exhaustive baseline within practical time constraints. To effectively assess this scenario, we conducted two separate baseline tests. The first, which we denote as the *Exhaustive neighborhood* baseline, utilizes the entire network for potential node perturbations while adopting the pruned skill set from ExES as the candidate skills for addition. The metrics for this baseline are represented as N in our tables. Conversely, the second baseline, referred to as the *Exhaustive skills* baseline, retains the universe of skills (S) as potential additional skills but limits modifications to the neighborhood of the target expert. The results corresponding to this baseline are denoted as S in our tables.

Table 4.2: Latency and size comparison for generating explanations for Expert Search

Category	**Target**	**Explanation Method**	**Dataset**	**Latency (s)**		**Explanation Size**	
				ExES	Exhaustive	ExES	Exhaustive
Saliency	—	Skill SHAP	DBLP GitHub	3.17 1.11	147.77 147.22	22.4 21.6	97.30 43.82
		Query SHAP	DBLP GitHub	0.13 0.19	—	2.76 3.52	—
		Collaboration SHAP	DBLP GitHub	11.29 7.23	98.02 76.57	13.17 4.24	174.96 144.78
Counterfactual	Experts	Skill Removal	DBLP GitHub	57.53 5.74	917.06 14.93	2.23 1.68	1.53 1.36
		Query Augmentation	DBLP GitHub	0.36 0.35	0.21 0.13	1.17 1.21	1.00 1.00
		Collaboration Removal	DBLP GitHub	17.18 12.25	671.95 154.25	2.09 2.31	1.73 2.23
	Non-experts	Skill Addition	DBLP	79.92	N: 213.27 S: 173.83	1.97	N: 1.41 S: 1.22
			GitHub	5.16	N: 15.24 S: 13.19	2.21	N: 1.63 S: 1.18
		Query Augmentation	DBLP GitHub	0.71 0.51	0.93 0.64	1.35 1.06	1.00 1.00
		Collaboration Addition	DBLP GitHub	6.17 1.51	159.79 11.82	1.33 1.54	1.12 1.03

Results. Table 4.2 presents the average latency and explanation sizes for ExES's algorithms alongside those of the exhaustive baseline, across the DBLP and GitHub datasets. Note that for saliency explanations, the explanation size shows the number of features having non-zero feature attribution values. Since the exhaustive baseline takes every feature into account, the exhaustive explanation size is not necessarily less than the saliency explanation sizes of ExES.

In our efficiency analysis, ExES demonstrated an average latency reduction of 52% for explaining expert search systems compared to the exhaustive search method. We observed that pruning the search space reduces the running time for finding saliency explanations, and counterfactual skill and collaboration explanations. In counterfactual query explanations, we observe that our pruning strategy reduces the latency for turning non-experts into

Table 4.3: Precision@1 and Precision@5 of generated saliency explanations by ExES for Expert Search

Category	Explanation Method	Dataset	Precision@1	Precision@5
Saliency	Skill SHAP	DBLP	0.85	0.70
		GitHub	0.81	0.64
	Collaboration SHAP	DBLP	1.00	0.98
		GitHub	1.00	0.99

Table 4.4: Precision and Precision* of generated explanations by ExES for Expert Search

Category	Target	Explanation Method	Dataset	# Explanations		Precision	Precision*
				ExES	Exhaustive		
Counterfactual	Experts	Skill Removal	DBLP	383	265	0.87	0.98
			GitHub	465	385	0.98	0.99
		Query Augmentation	DBLP	470	470	0.85	0.97
			GitHub	435	475	0.81	0.96
		Collaboration Removal	DBLP	301	230	0.93	1.00
			GitHub	242	226	0.83	0.98
	Non-experts	Skill Addition	DBLP	387	N: 270	N: 0.87	N: 0.97
					S: 420	S: 0.43	S: 0.85
			GitHub	375	N: 295	N: 0.91	N: 0.96
					S: 440	S: 0.32	S: 0.73
		Query Augmentation	DBLP	456	460	0.59	0.91
			GitHub	440	475	0.64	0.94
		Collaboration Addition	DBLP	441	460	0.73	0.95
			GitHub	400	470	0.59	0.89

experts. However, the reverse direction exhibits an overhead without latency reduction. the pruning method seems to be an overhead, without reducing the latency. This likely stems from the fact that to turn experts into non-experts, adding an arbitrary keyword to the query is sufficient. However, selecting the additional keywords using the keyword embedding model is useful for sustaining the plausibility and validity of the counterfactual queries. This approach ensures the chosen keywords closely relate to the original query, thus maintaining the relevance and meaningfulness of the counterfactual queries.

Table 4.3 demonstrates the Precision@1 and Precision@5 for Skill-based and Collaboration-based saliency explanations for Expert Search. The results confirm that, on average, ExES achieves a Precision@1 of 0.91 and a Precision@5 of 0.82. This means that 91% of the features identified as most important, and 82% of the top-5 important features, are correctly recognized as important by ExES. Table 4.4 shows the number of total counterfactual explanations generated by ExES (with pruning) and by the baseline (without pruning, by exhaustive search) within the time limit, along with the precision and precision* of explanations created by ExES. Given a total of 100 target experts and $e = 5$, the ideal output would involve 500 explanations for each method. However, ExES may miss some explanations due to pruning of the search space, while the exhaustive search might not complete within the time limit. This analysis revealed ExES maintains an average precision of 74% over all counterfactual explanation types. In addition, ExES maintains an average precision* of 94%, which proves that it could generate *nearly-optimal* explanations in 94% of the cases.

Upon examining the precision values, it becomes evident that for Skill Addition counterfactuals, the precision of ExES, in comparison to *Exhaustive skillset* baselines, drops below 0.5. This observation leads us to conclude that selecting appropriate candidate keywords for the search space is crucial in attaining minimal explanations. One potential action to enhance the probability of taking those appropriate keywords in the search space is increasing the parameter t. Nevertheless, this approach introduces a trade-off between latency and precision. A larger t value would lead to ExES behaving more closely to the *Exhaustive skillset* baseline, but at the cost of increased computation time. Still, the precision results indicate that 70% ExES's skill addition explanations are nearly-optimal.

4.3.2 Team Formation Experiments

As outlined in 3, ExES is capable of explaining the decisions for any team formation system that meets the criteria specified in Section 3.1. However, for our evaluation, we selected the method from [32] as our black-box for team member retrieval. This method receives an expert as the main member, and subsequently constructs the team around the main member. The search procedure selects the team members in a breadth-first manner starting from the main member, with the aim to cover all skills in the query. The team formation system utilizes the expert ranker, as described in 4.3.1, as a tie-breaker between candidate members at the same distance from the main member, prioritizing individuals with higher ranks.

To evaluate explanations for team member inclusion and exclusion, we employed the random queries generated in the previous section. For each query, we randomly selected an expert from the top-k results and then used our team formation method to build a team around that expert. Within each formed team, we randomly sampled one team member to explain their inclusion and one non-member from the main member's neighborhood to explain their exclusion.

The exhaustive baselines and parameters used for explanation generation are as defined in Section 4.3.1.

Table 4.5: Latency and size comparison for generating explanations for Team Formation

Category	Target	Explanation Method	Dataset	Latency (s)		Explanation Size	
				ExES	Exhaustive	ExES	Exhaustive
Saliency	—	Skill SHAP	DBLP	8.63	236.53	30.65	93.40
			GitHub	2.88	229.80	23.98	40.69
		Query SHAP	DBLP	0.24	—	2.65	—
			GitHub	0.31		3.52	
		Collaboration SHAP	DBLP	62.41	645.62	18.38	209.38
			GitHub	41.02	79.04	8.62	168.73
Counterfactual	Members	Skill Removal	DBLP	132.18	958.05	1.77	1.31
			GitHub	10.12	31.82	1.54	1.26
		Query Augmentation	DBLP	0.82	0.46	1.20	1.05
			GitHub	0.80	0.35	1.30	1.00
		Collaboration Removal	DBLP	35.91	932.63	1.95	1.60
			GitHub	22.74	328.47	2.08	1.64
	Non-members	Skill Addition	DBLP	181.46	N: 450.35 S: 371.57	2.19	N: 1.59 S: 1.31
			GitHub	13.78	N: 28.40 S: 24.72	2.88	N: 1.95 S: 1.58
		Query Augmentation	DBLP	0.84	1.28	2.14	1.83
			GitHub	0.59	0.75	2.03	1.67
		Collaboration Addition	DBLP	13.32	366.51	1.51	1.26
			GitHub	4.26	25.33	1.79	1.15

Results. Table 4.5 compares the average latency and explanation sizes for generating explanations for team formation systems, using ExES and the exhaustive baseline. This comparison supports our observations from Section 4.3.1. We noted that ExES achieves an average reduction in latency of 50% for generating team formation explanations, underlining the efficiency of ExES.

Table 4.6 presents the Precision@1 and Precision@5 scores for Skill-based and Collaboration-based saliency explanations of Team Formation, generated by ExES. Aligned with the metrics from 4.3.1, our findings show that ExES is able to generate saliency explanations for Team Formation systems, with a Precision@1 of 0.83 and a Precision@5 of 0.72 on average. Furthermore, Table 4.7 displays the number of total explanations generated by ExES and by the baseline within the time limit, along with the precision and precision* of explanations created by ExES. Similar to our approach in evaluating expert search explanations, *Exhaustive neighborhood* and *Exhaustive skillset* baselines are employed for

Table 4.6: Precision@1 and Precision@5 of generated saliency explanations by ExES for Team Formation

Category	**Explanation Method**	**Dataset**	**Precision@1**	**Precision@5**
Saliency	Skill SHAP	DBLP	0.77	0.57
		GitHub	0.58	0.51
	Collaboration SHAP	DBLP	1.00	0.89
		GitHub	1.00	0.91

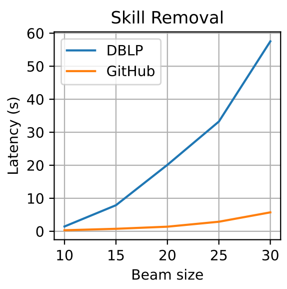
Table 4.7: Precision and Precision* of generated explanations by ExES for Team Formation

Category	**Target**	**Explanation Method**	**Dataset**	**# Explanations**		**Precision**	**Precision***
				ExES	Exhaustive		
Counterfactual	Members	Skill Removal	DBLP	306	210	0.82	0.92
			GitHub	405	365	0.92	0.93
		Query Augmentation	DBLP	260	260	0.86	0.98
			GitHub	199	250	0.82	0.97
		Collaboration Removal	DBLP	296	225	0.92	1.00
			GitHub	223	215	0.82	0.97
	Non-members	Skill Addition	DBLP	350	N: 250 S: 405	N: 0.81 S: 0.41	N: 0.96 S: 0.82
			GitHub	348	N: 265 S: 410	N: 0.90 S: 0.30	N: 0.97 S: 0.70
		Query Augmentation	DBLP	425	445	0.50	0.90
			GitHub	410	455	0.60	1.00
		Collaboration Addition	DBLP	410	445	0.69	0.91
			GitHub	360	450	0.57	0.83

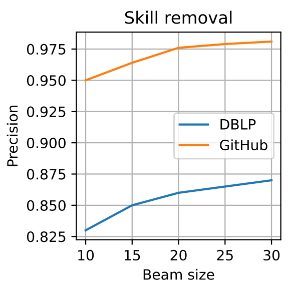
evaluating Skill Addition counterfactuals. The results indicate that ExES secures an average precision of 71% and an average precision* of 91% over all counterfactual explanation types, signifying its effectiveness in obtaining minimal and nearly-optimal explanations, respectively.

4.3.3 Parameter Sensitivity Analysis

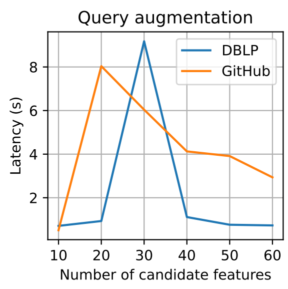
We furthermore conduct a sensitivity analysis on the parameters used in the search processes in ExES. These parameters include the beam size b, the number of candidate features selected in beam search t, the neighborhood radius d, and the threshold in calculating collaboration SHAP values τ. We evaluate their impacts on latency, precision, the number of found explanations, and the explanation size of the explanation methods. To analyze each



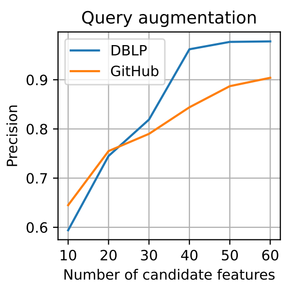
(a) Effect of b on Latency



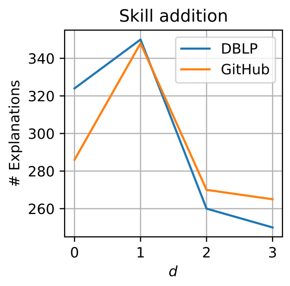
(b) Effect of b on Precision



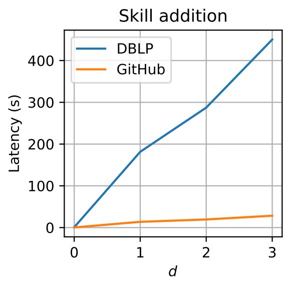
(c) Effect of t on Latency



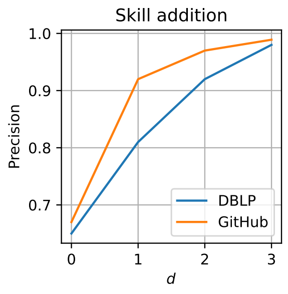
(d) Effect of t on Precision



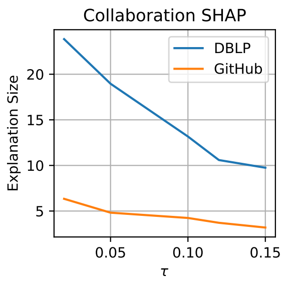
(e) Effect of d on # Expl.



(f) Effect of d on Latency



(g) Effect of d on Precision

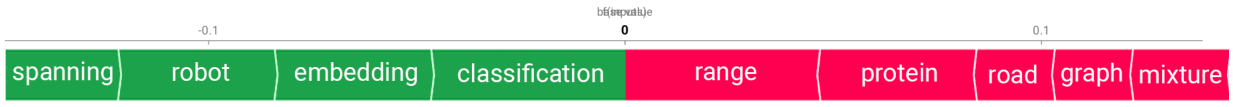


(h) Effect of τ on Expl. Size

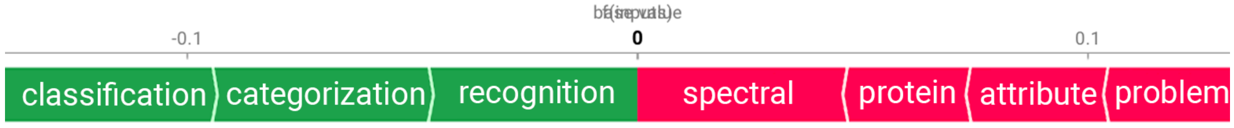
Figure 4.1: Parameter sensitivity analysis on explanation metrics

parameter, we set the other parameters to the values mentioned in Section 4.3.1.

Results. Figure 4.1 shows eight plots of the parameter sensitivity analysis. Figures 4.1a and 4.1b measure the effect of beam size b on the latency and precision of skill removal explanations in expert search. We can observe that the runtime and precision increase by increasing the mentioned parameters. Figures 4.1c and 4.1d measure the effect of the number of candidate tokens t on the latency and precision of query augmentation explanations for non-experts in expert search. Figure 4.1c displays an increasing trend in latency for lower values of t. However, for larger values of t, the set of candidate additional keywords would contain more keywords, enhancing the likelihood of including effective keywords for counterfactuals. This allows the search to terminate in the early steps, which reduces the latency. Figure 4.1d verifies that increasing t enhances the explanation precisions, where the increase rate declines for large values of t. Plus, Figure 4.1e demonstrates the impact of neighborhood size d on the number of generated valid Skill Addition explanations in expert search and the optimal value of d. In this figure, smaller values of d result in ExES failing to find valid explanations due to restricted search space, while larger values of d cause ExES to exceed the time limit. Moreover, Figures 4.1f and 4.1g show the tradeoff between latency and precision (compared with the *Exhaustive neighborhood* baseline) for



(a) With pruning



(b) Without pruning

Figure 4.2: Skill SHAP values for Yann LeCun's relevance status

different values of d. Finally, Figure 4.1h measures the effect of the threshold τ on explanation size in Collaboration SHAP values in expert search. It demonstrates that increasing τ leads to achieving smaller explanation sizes, as fewer experts are considered "impactful" and subject to further expansion.

4.4 Case Study

In this section, we qualitatively analyze the functionality of ExES in explaining expert search and team formation outcomes. We use the black-box models from 4.3.

In our first scenario, we take the query "deep neural training", and perform the expert search on the DBLP collaboration network, having $k = 10$. After running the expert search black-box, we observe that "Yann LeCun" is ranked 11th. While this indicates a strong position among nearly 17000 researchers, we focus on explaining why LeCun was not ranked in top-10, since he is a prominent figure in the field of machine learning research. In this scenario, we analyze saliency explanation methods.

First, we find Skill SHAP values, to see which skills in LeCun's neighborhood may be preventing him from getting a place in the top-10. Figure 4.2 shows the SHAP values of skills that either positively or negatively affect LeCun's relevance status, both with (generated by ExES) and without (generated by the exhaustive baseline) pruning methods. According to 4.2a, ExES demonstrates that skills related to Machine Learning and deep neural networks, such as "classification" and "embedding" positively influence LeCun's relevance. Conversely, skills such as "range", "protein", "road" and "graph", which are more associated with other machine learning applications, are irrelevant to the query and negatively impact LeCun's relevance.

Comparing the results of ExES with the exhaustive output (Figure 4.2b), we observe that the exhaustive baseline includes skills such as "recognition" and "categorization" as the most important positive skills. While these skills do not appear among the top features in ExES's results, they share semantic similarities with the top feature highlighted by ExES ("classification"). On the other hand, the exhaustive baseline finds "spectral" to have the most negative impact on LeCun's relevance. This skill, related to image and sound recognition topics, aligns with the keyword "range", identified by ExES as having the most negative effect.

Additionally, we analyze the SHAP values of collaborations in LeCun's neighborhood, to identify specific collaborators who positively or negatively influence his relevance to the query. Figure 4.3 displays the collaboration SHAP values in LeCun's neighborhood. According to this figure, collaborations with "Yoshua Bengio", "Pascal Vincent", and "Rob Fergus", who are all established ML researchers, appear to have a beneficial impact. However, collaborating with "Laurent Najman", "Xavier Bresson", and "Patrick Haffner" distracts LeCun from the query. This may be due to the specialization of these experts in applied ML, with a specific focus on computer vision, graph neural networks, and network systems, respectively.

In our second scenario, we take the same query and k from the first scenario. However, in this scenario, we aim to explain the relevance of "Yoshua Bengio", who is ranked 19th, using counterfactual explanations. We suggest that our counterfactual skill and edge additions could support career advancements (recall Figures 3.4 and 3.5).

First, we analyze counterfactual skill explanations to determine how Bengio might achieve a top-10 ranking. Figure 4.4 presents a list of such counterfactual skill explanations, revealing a minimal explanation size of 9. We observe that ExES proposes skills additions for both Bengio and his network, which demonstrates the importance of neighborhood structure in ranking experts. In these explanations, ExES identifies skills such as "recognition", "supervised", and "machine" to be added to Bengio and his collaborators. However, the first 7 skill additions remain consistent among our found perturbations, while the last two additions distinguish one explanation from another.

Next, we analyze counterfactual query explanations that position Bengio within the top 10 ranks. Figure 4.5 compares the query perturbation generated by ExES (Figure 4.5a) with those generated by the exhaustive baseline and without pruning techniques (Figure 4.5b). In this case, ExES generates an explanation (appending keywords "discriminative" and "recurrent" to the initial query) of size 2, while the minimal counterfactuals identified by the exhaustive baseline have a size of 1. Although the counterfactual found by ExES is

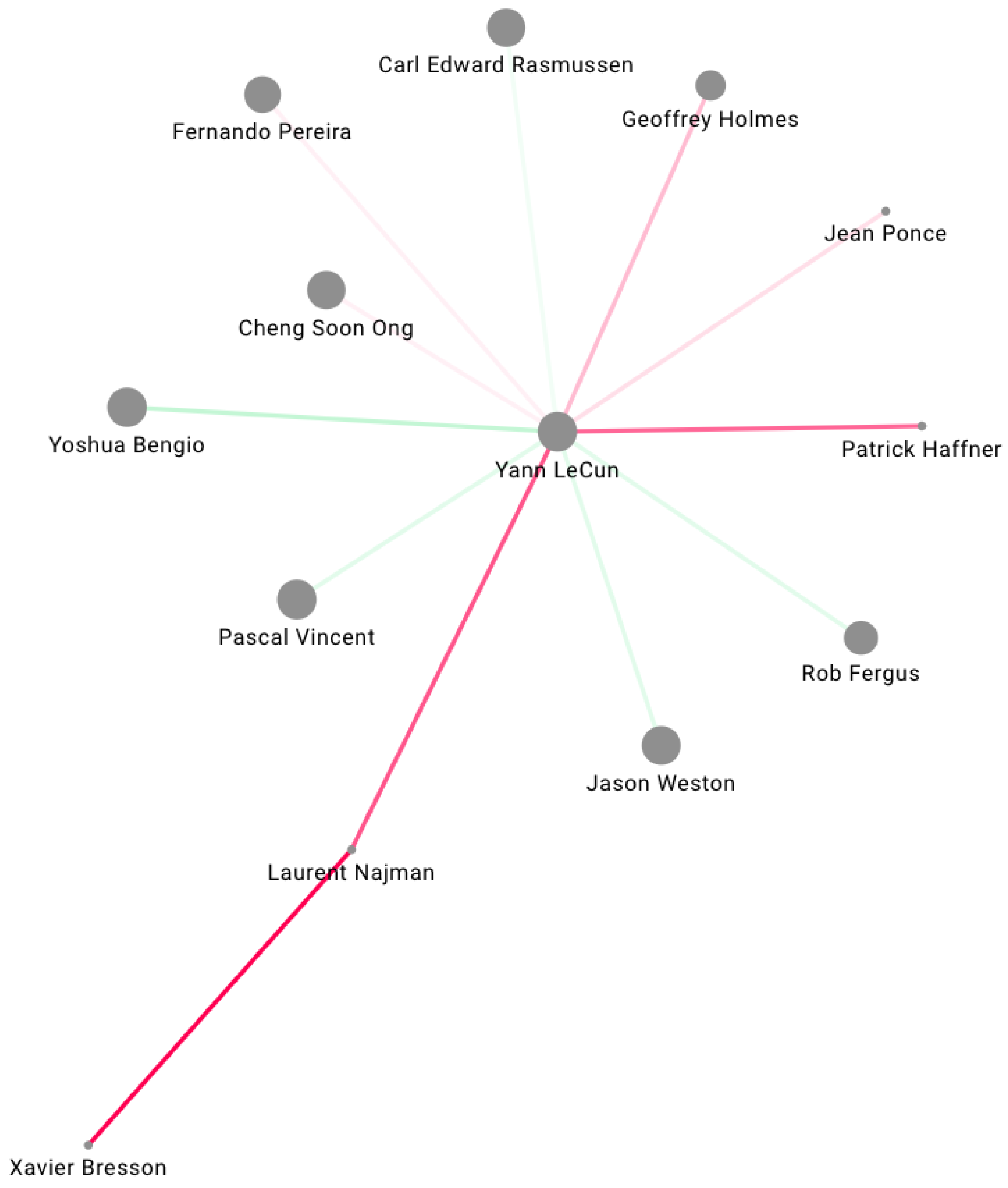
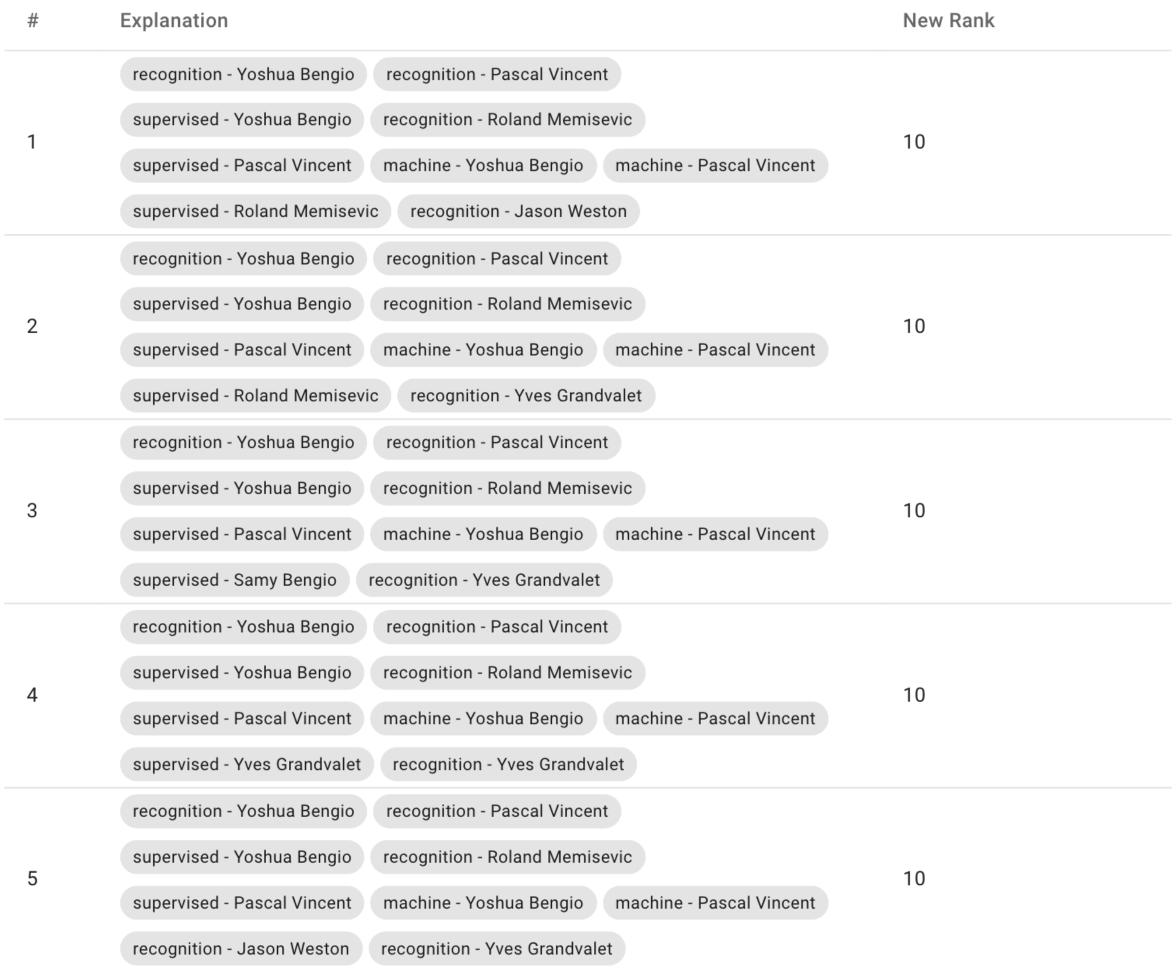


Figure 4.3: Collaboration SHAP values for Yann LeCun's relevance status



#	Explanation	New Rank
1	recognition - Yoshua Bengio; recognition - Pascal Vincent; supervised - Yoshua Bengio; recognition - Roland Memisevic; supervised - Pascal Vincent; machine - Yoshua Bengio; machine - Pascal Vincent; supervised - Roland Memisevic; recognition - Jason Weston	10
2	recognition - Yoshua Bengio; recognition - Pascal Vincent; supervised - Yoshua Bengio; recognition - Roland Memisevic; supervised - Pascal Vincent; machine - Yoshua Bengio; machine - Pascal Vincent; supervised - Roland Memisevic; recognition - Yves Grandvalet	10
3	recognition - Yoshua Bengio; recognition - Pascal Vincent; supervised - Yoshua Bengio; recognition - Roland Memisevic; supervised - Pascal Vincent; machine - Yoshua Bengio; machine - Pascal Vincent; supervised - Samy Bengio; recognition - Yves Grandvalet	10
4	recognition - Yoshua Bengio; recognition - Pascal Vincent; supervised - Yoshua Bengio; recognition - Roland Memisevic; supervised - Pascal Vincent; machine - Yoshua Bengio; machine - Pascal Vincent; supervised - Yves Grandvalet; recognition - Yves Grandvalet	10
5	recognition - Yoshua Bengio; recognition - Pascal Vincent; supervised - Yoshua Bengio; recognition - Roland Memisevic; supervised - Pascal Vincent; machine - Yoshua Bengio; machine - Pascal Vincent; recognition - Jason Weston; recognition - Yves Grandvalet	10

Figure 4.4: Counterfactual skill explanations to make Yoshua Bengio get into top-10

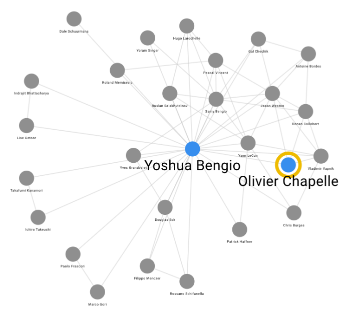
#	Explanation	New Rank
1	discriminative recurrent	9

(a) With pruning

#	Explanation	New Rank
1	evidence	6
2	support	7
3	engine	8
4	link	10
5	strong	10

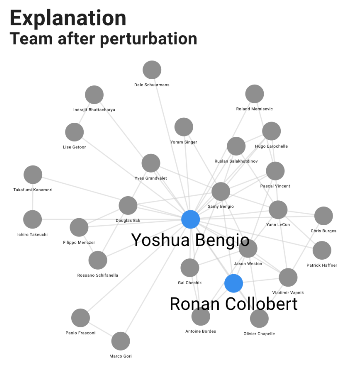
(b) Without pruning

Figure 4.5: Counterfactual query explanations to make Yoshua Bengio get into top-10



(a) Original team

#	Explanation
1	Olivier Chapelle - Thomas Deselaers
2	Olivier Chapelle - Olivier Bousquet



(b) Counterfactual explanations and modified team

Figure 4.6: Counterfactual collaboration explanations to explain Chapelle's selection for the team

not minimal, it qualifies as *nearly-optimal*, exceeding the optimal size by one unit. Furthermore, we notice that the counterfactual query generated by ExES maintains relevance to the initial query (as "discriminative" and "recurrent" are terms associated with ML model categories). In contrast, appending keywords like "evidence", "support", etc. would not present a logical or meaningful addition to the query.

In the last scenario (Figure 4.6), we investigate a team formation case with the query "deep neural training supervised", where the team formation model (recall Section 4.3) forms a team around Yoshua Bengio. In this case, the output team includes Bengio and "Olivier Chapelle" (Figure 4.6a). Chapelle has written two books on semi-supervised learning and has substantial expertise in training with labeled data. We utilize counterfactual collaboration explanations to interpret Chapelle's selection for the team. Figure 4.6b displays a list of edge removals, each of which would result in Chapelle being replaced by another member (Ronan Collobert) within the team. Focusing on the first edge removal, we see that Deselaers has significant experience in data annotation and labeling medical images. Eliminating the collaborative link between Deselaers and Chapelle disrupts the synergistic effect between two experts in supervised training. Therefore, Chapelle is substituted by Collobert to represent the keyword "supervised".

Chapter 5

Conclusion and Future Directions

5.1 Summary

In this study, we introduced ExES, the first tool specifically designed to generate explanations for expert search and team formation systems. By framing these complex tasks as binary classification problems, we applied saliency and counterfactual explanation methods, common in the XAI domain, to these tasks. In advance, to manage the vast search space for explanations, we established pruning techniques, which contributed to a reduction in the running time of the algorithms, without compromising precision. We demonstrated the efficiency and effectiveness of our explanation methods through experiments on two real-world datasets and benchmarking against an exhaustive explanation baseline.

5.2 Future Work

For future work, we suggest the following extensions:

- **Multi-objective counterfactual optimization:** The optimization function in counterfactual explanations could be expanded to include a broader range of parameters beyond explanation size. Inspired by the related work [19, 2], the optimization function could incorporate factors such as fidelity and robustness, in tandem with explanation size. For instance, by ensuring high fidelity in explanations, we ensure that explanations generated for one individual can be generalized for other individuals as well. On the other side, one could demand finding perturbations that do not change the ranking/team significantly, and therefore try to maximize the robustness.

- **Rule-based Explanations:** As outlined in [66, 20], rule-based explainers are a family of interpretable surrogate models which represent black-box model decisions as *if-then* rules. These explainers identify input feature patterns that consistently lead to specific outcomes. There is potential for extending ExES to generate rule-based explanations; i.e. identifying skills or connections that are essential for any individual to be selected by the model, or underscoring key skills or collaborations that cause any arbitrary individual to be included in the results with high confidence.

- **Distributed Implementation:** To ensure scalability, implementing a distributed version of ExES would be beneficial, specifically for reducing explanation latency in large-scale collaboration graphs.

- **User Study:** In this thesis, we quantitatively compared the metrics between explanations generated by ExES and the exhaustive baseline and validated the outputs through a qualitative case study. However, conducting a comprehensive user study would be beneficial to evaluate the precision and usefulness of ExES's explanations in real-world scenarios. In such a study, real end-users with diverse profiles and backgrounds would test ExES under various configurations, with a key focus on monitoring their satisfaction with the explanations provided. Specifically, we would measure users' preferences for receiving shorter versus longer explanations based on their intentions for using ExES, their specific queries, and the individuals targeted by these queries.

- **Identifying Data Quality Issues:** Due to the rapid evolution of collaboration networks caused by possible topic drifts and variability of experts' skills and collaboration ties [18], these networks are prone to data quality issues, such as missing edges and node labels, or unsuccessful collaborations [32]. We hypothesize that ExES could help in highlighting these data quality problems; for example, by identifying missing skills or collaborations involving experts who were overlooked by the black-box expert search/team formation systems due to these deficiencies. We propose an evaluation of ExES' potential in addressing these data quality issues for future studies.

- **Extending Application Domains:** While this work concentrates on the context of expert search and team formation, the framework formulation and pruning methods employed in ExES could be potentially extended beyond these areas. Future research could explore and assess the effectiveness of ExES's explanations in other graph search domains, such as keyword search in relational databases and protein-protein interaction networks.

References

[1] Pytorch Geometric, Implementation of GAE. https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/models/autoencoder.html#GAE. [Online; accessed 2024-03-08].

[2] Amin Abolghasemi, Suzan Verberne, Leif Azzopardi, and Maarten de Rijke. On the explainability of exposing query identification. In *6th FAccTRec Workshop on Responsible Recommendation at RecSys 2023*, 2023.

[3] Arian Askari, Suzan Verberne, and Gabriella Pasi. Expert finding in legal community question answering. In *ECIR*, pages 22–30, 2022.

[4] Krisztian Balog, Leif Azzopardi, and Maarten De Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR*, pages 43–50, 2006.

[5] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. A language modeling framework for expert finding. *Information Processing & Management*, 45(1):1–19, 2009.

[6] Mark Berger, Jakub Zavrel, and Paul Groth. Effective distributed representations for academic expert search. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 56–71, 2020.

[7] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *Data Mining and Knowledge Discovery*, 37(5):1719–1778, 2023.

[8] Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Matteo Silvestri, and Giuliano Vesci. Choosing the right crowd: expert finding in social networks. In *EDBT*, pages 637–648, 2013.

[9] Robin Brochier, Antoine Gourru, Adrien Guille, and Julien Velcin. New datasets and a benchmark of document network embedding methods for scientific expert finding. *arXiv preprint arXiv:2004.03621*, 2020.

[10] Spencer Bryson, Heidar Davoudi, Lukasz Golab, Mehdi Kargar, Yuliya Lytvyn, Piotr Mierzejewski, Jaroslaw Szlichta, and Morteza Zihayat. Robust keyword search in large attributed graphs. *Information Retrieval Journal*, pages 502–524, 2020.

[11] Tanya Chowdhury, Razieh Rahimi, and James Allan. Rank-lime: local model-agnostic feature attribution for learning to rank. In *ICTIR*, pages 33–37, 2023.

[12] Paolo Cifariello, Paolo Ferragina, and Marco Ponza. Wiser: A semantic approach for expert finding in academia based on entity linking. *Information Systems*, 82:1–16, 2019.

[13] Arash Dargahi Nobari, Mahmood Neshati, and Sajad Sotudeh Gharebagh. Quality-aware skill translation models for expert finding on stackoverflow. *Information Systems*, 87:101413, 2020.

[14] Arman Dashti, Saeed Samet, and Hossein Fani. Effective neural team formation via negative samples. In *CIKM*, pages 3908–3912, 2022.

[15] Hongbo Deng, Jiawei Han, Michael R Lyu, and Irwin King. Modeling and exploiting heterogeneous bibliographic networks for expertise ranking. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, pages 71–80, 2012.

[16] Stuart E Dreyfus and Robert A Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

[17] Zohreh Fallahnejad and Hamid Beigy. Attention-based skill translation models for expert finding. *Expert Systems with Applications*, 193:116433, 2022.

[18] Hossein Fani, Reza Barzegar, Arman Dashti, and Mahdis Saeedi. A streaming approach to neural team formation training. In *ECIR*, pages 325–340, 2024.

[19] Thorben Funke, Megha Khosla, Mandeep Rathee, and Avishek Anand. Zorro: Valid, sparse, and stable explanations in graph neural networks. *TKDE*, 2022.

[20] Zixuan Geng, Maximilian Schleich, and Dan Suciu. Computing rule-based explanations by leveraging counterfactuals. *PVLDB*, 16(3):420–432, 2022.

[21] Athina Georgara, Juan A. Rodríguez-Aguilar, and Carles Sierra. Allocating teams to tasks: an anytime heuristic competence-based approach. In *European Conference on Multi-Agent Systems*, pages 152–170, 2022.

[22] Athina Georgara, Juan A Rodriguez Aguilar, and Carles Sierra. Building contrastive explanations for multi-agent team formation. In *AAMAS*, pages 516–524, 2022.

[23] Negin Ghasemi, Ramin Fatourechi, and Saeedeh Momtazi. User embedding for expert finding in community question answering. *TKDD*, 15(4):1–16, 2021.

[24] Azin Ghazimatin, Oana Balalau, Rishiraj Saha Roy, and Gerhard Weikum. Prince: Provider-side interpretability with counterfactual explanations in recommender systems. In *WSDM*, pages 196–204, 2020.

[25] Diego Gómez-Zará, Leslie A DeChurch, and Noshir S Contractor. A taxonomy of team-assembly systems: Understanding how people use technologies to form teams. *ACM HCI*, 4(CSCW2):1–36, 2020.

[26] Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pages 1–55, 2022.

[27] Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems*, 34(6):14–23, 2019.

[28] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5), 2018.

[29] Viet Ha-Thuc, Ganesh Venkataraman, Mario Rodriguez, Shakti Sinha, Senthil Sundaram, and Lin Guo. Personalized expertise search at linkedin. In *2015 IEEE International Conference on Big Data*, pages 1238–1247, 2015.

[30] Radin Hamidi Rad, Ebrahim Bagheri, Mehdi Kargar, Divesh Srivastava, and Jaroslaw Szlichta. Retrieving skill-based teams from collaboration networks. In *SIGIR*, pages 2015–2019, 2021.

[31] Radin Hamidi Rad, Hossein Fani, Ebrahim Bagheri, Mehdi Kargar, Divesh Srivastava, and Jaroslaw Szlichta. A variational neural architecture for skill-based team formation. *ACM TOIS*, 42(1):1–28, 2023.

[32] Yu Hao, Xin Cao, Yufan Sheng, Yixiang Fang, and Wei Wang. Ks-gnn: Keywords search over incomplete graphs via graphs neural network. *Advances in Neural Information Processing Systems*, 34:1700–1712, 2021.

[33] Omayma Husain, Naomie Salim, Rose Alinda Alias, Samah Abdelsalam, and Alzubair Hassan. Expert finding systems: A systematic review. *Applied Sciences*, 9(20):4250, 2019.

[34] Sergio Jimenez, Fabio N Silva, George Dueñas, and Alexander Gelbukh. Proficiencyrank: Automatically ranking expertise in online collaborative social networks. *Information Sciences*, 588:231–247, 2022.

[35] Yong-Bin Kang, Hung Du, Abdur Rahim Mohammad Forkan, Prem Prakash Jayaraman, Amir Aryani, and Timos Sellis. Expfinder: A hybrid model for expert finding from text-based expertise data. *Expert Systems with Applications*, 211:118691, 2023.

[36] Mehdi Kargar and Aijun An. Discovering top-k teams of experts with/without a leader in social networks. In *CIKM*, pages 985–994, 2011.

[37] Maryam Karimzadehgan, Ryen W White, and Matthew Richardson. Enhancing expert finding using organizational hierarchies. In *ECIR*, pages 177–188, 2009.

[38] Sagar Kaw, Ziad Kobti, and Kalyani Selvarajah. Transfer learning with graph attention networks for team recommendation. In *IJCNN*, pages 1–8. IEEE, 2023.

[39] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[40] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[41] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[42] Bernhard H Korte and Jens Vygen. *Combinatorial optimization*, volume 1. Springer, 2011.

[43] Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In *SIGKDD*, pages 467–476, 2009.

[44] Michael Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *International symposium on string processing and information retrieval*, pages 1–10. Springer, 2002.

[45] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. *NeurIPS*, 36, 2024.

[46] Rennan C. Lima and Rodrygo L. T. Santos. On extractive summarization for profile-centric neural expert search in academia. In *SIGIR*, page 2331–2335, 2022.

[47] Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *AISTATS*, pages 4499–4511, 2022.

[48] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NeurIPS*, page 4768–4777, 2017.

[49] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *NeurIPS*, 33:19620–19631, 2020.

[50] Lijun Lyu and Avishek Anand. Listwise explanations for ranking models using multiple explainers. In *ECIR*, pages 653–668. Springer, 2023.

[51] Craig Macdonald and Iadh Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In *CIKM*, pages 387—-396, 2006.

[52] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM computing surveys (CSUR)*, 49(4):1–33, 2016.

[53] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[54] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[55] Saeedeh Momtazi and Felix Naumann. Topic modeling for expert finding using latent dirichlet allocation. *Data Mining and Knowledge Discovery*, 3(5):346–353, 2013.

[56] Mahmood Neshati, Hamid Beigy, and Djoerd Hiemstra. Expert group formation using facility location analysis. *Information processing & management*, 50(2):361–383, 2014.

[57] Mahmood Neshati, Zohreh Fallahnejad, and Hamid Beigy. On dynamicity of expert finding in community question answering. *Information Processing & Management*, 53(5):1026–1042, 2017.

[58] N. Nikzad-Khasmakhi, M.A. Balafar, M. Reza Feizi-Derakhshi, and Cina Motamed. Berters: Multimodal representation learning for expert recommendation system with transformers and graph embeddings. *Chaos, Solitons & Fractals*, 151:111260, 2021.

[59] Narjes Nikzad-Khasmakhi, Mohammadali Balafar, M. Reza Feizi-Derakhshi, and Cina Motamed. Exem: Expert embedding using dominating set theory with deep learning approaches. *Expert Systems with Applications*, 177:114913, 2021.

[60] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710, 2014.

[61] Mario Alfonso Prado-Romero, Bardh Prenkaj, Giovanni Stilo, and Fosca Giannotti. A survey on graph counterfactual explanations: definitions, methods, evaluation. *arXiv preprint arXiv:2210.12089*, 2022.

[62] Radin Hamidi Rad, Shirin Seyedsalehi, Mehdi Kargar, Morteza Zihayat, and Ebrahim Bagheri. A neural approach to forming coherent teams in collaboration networks. In *EDBT*, pages 440–444, 2022.

[63] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *SIGKDD*, pages 1135–1144, 2016.

[64] Joel Rorseth, Parke Godfrey, Lukasz Golab, Mehdi Kargar, Divesh Srivastava, and Jaroslaw Szlichta. Credence: Counterfactual explanations for document ranking. In *ICDE*, pages 3631–3634, 2023.

[65] Peyman Rostami and Mahmood Neshati. T-shaped grouping: Expert finding models to agile software teams retrieval. *Expert Systems with Applications*, 118:231–245, 2019.

[66] Cynthia Rudin and Yaron Shaposhnik. Globally-consistent rule-based summary-explanations for machine learning models: application to credit-risk evaluation. *JMLR*, 24(16):1–44, 2023.

[67] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach.* Pearson, 2021.

[68] Anna Sapienza, Palash Goyal, and Emilio Ferrara. Deep neural networks for optimal team composition. *Frontiers in big Data*, 2:14, 2019.

[69] Pavel Serdyukov, Henning Rode, and Djoerd Hiemstra. Modeling multi-step relevance propagation for expert finding. In *CIKM*, pages 1133–1142, 2008.

[70] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[71] Jaspreet Singh and Avishek Anand. Exs: Explainable search using local model agnostic interpretability. In *WSDM*, pages 770–773, 2019.

[72] Jaspreet Singh and Avishek Anand. Model agnostic interpretability of rankers via intent modelling. In *ACM FAT**, pages 618–628, 2020.

[73] Juntao Tan, Shuyuan Xu, Yingqiang Ge, Yunqi Li, Xu Chen, and Yongfeng Zhang. Counterfactual explainable recommendation. In *CIKM*, pages 1784–1793, 2021.

[74] Cem Tekin, Onur Atan, and Mihaela Van Der Schaar. Discover the expert: Context-adaptive expert selection for medical diagnosis. *IEEE Transactions on Emerging Topics in Computing*, 3(2):220–234, 2015.

[75] Khanh Hiep Tran, Azin Ghazimatin, and Rishiraj Saha Roy. Counterfactual explanations for neural recommenders. In *SIGIR*, pages 1627–1631, 2021.

[76] Christophe Van Gysel, Maarten de Rijke, and Marcel Worring. Unsupervised, efficient and semantic expertise retrieval. In *WWW*, pages 1069–1079, 2016.

[77] Olga Vechtomova. Query expansion for information retrieval. In *Encyclopedia of Database Systems*, pages 2254–2257. 2009.

[78] Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan E Hines, John P Dickerson, and Chirag Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

[79] Shirui Wang, Wenan Zhou, and Chao Jiang. A survey of word embeddings based on deep learning. *Computing*, 102(3):717–740, 2020.

[80] Xiaoliang Xu, Jun Liu, Yuxiang Wang, and Xiangyu Ke. Academic expert finding via $(k, \mathcal{P})$-core based embedding over heterogeneous graphs. In *ICDE*, pages 338–351, 2022.

[81] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *NeurIPS*, 32, 2019.

[82] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5782–5799, 2022.

[83] Morteza Zihayat, Aijun An, Lukasz Golab, Mehdi Kargar, and Jaroslaw Szlichta. Authority-based team discovery in social networks. In *EDBT*, pages 498–501, 2017.