# Academic Expert Finding via $(k, \mathcal{P})$-Core based Embedding over Heterogeneous Graphs

***Abstract*—Finding relevant experts in specified areas is often crucial for a wide range of applications in both academia and industry. Given a user input query and a large amount of academic knowledge (e.g., academic papers), expert finding aims to find and rank the experts who are most relevant to the given query, from the academic knowledge. Existing studies mainly focus on the embedding-based solutions that (1) consider academic papers' textual semantic similarities to a given query through document representation models and (2) extract the top-$n$ experts with the greatest similarities. Beyond the implicit textual semantics of papers, however, the papers' explicit relationships (e.g., co-authorship, citation, and same-topic relationship) in a heterogeneous academic graph (e.g., DBLP) are critical for document representation, insofar as they help improve the expert finding quality. Despite their importance, the explicit relationships of papers generally have been ignored in the literature. In this paper, we study the academic expert finding on heterogeneous graphs by considering the explicit relationships besides the implicit textual semantics of papers in one representation model. Specifically, we first define the $(k, \mathcal{P})$-core to denote a cohesive community of papers that are closely connected via a meta-path $\mathcal{P}$ ($\mathcal{P}$ indicates the different relationships of papers). We then propose an offline $(k, \mathcal{P})$-core based document embedding model to capture papers' various explicit relationships for representation. Moreover, by using papers' embeddings, we present an online threshold algorithm (TA)-based method to efficiently return top-$n$ experts via a carefully designed proximity graph-based index (PG-Index). We extend our approach to support multiple relationships simultaneously for representation. Extensive experiments over real-world datasets demonstrate the effectiveness and efficiency of our approach.**

## I. Introduction

Academic expert finding has recently attracted considerable attention in the information retrieval (IR) community, as the fast growing of the scientific literature and the urgent increasing demands of expert searching in real applications [1]–[7]. Given a user input query that describes the user's desired expertise for target experts and a large amount of academic knowledge, expert finding aims to find and rank experts who are most relevant to the given query, by taking academic knowledge as evidence [8]. Expert finding has been widely used in *reviewer assignment* [9]–[11], *consulting* [2], [12], *supervisors recommendation* [13], and *technology transfer between academia and industry* [1], [14], etc. In this paper, we target the domain of retrieving academic experts based on papers they authored, which is known as the *document-centric* expert finding [15], [16] (academic papers are defined as the academic knowledge). The two most important query forms of expert finding are topic-based and text-based query, as follows.

**Topic-based query.** A user usually forms a query by providing a set of desired topics (or research areas) of interest and expects to obtain a set of experts whose expertise can be covered or represented by the input topics [12], [17]–[19]. For topic-based queries, the *statistical language model* is commonly used to find relevant experts [2], [20]. To be more precise, a conditional probability model with topics as the condition is proposed to predict the probability of an expert related to these topics, and then we can return the top-$n$ experts according to such probability. To further improve the effectiveness, some works focus on integrating the topics' semantic representations (i.e., embeddings) into the statistical language model [18], [21]. The biggest problem of topic-based query is that the topics' textual content is too limited in terms of language to express a user's latent query intention [3]. Usually, a richer description of a specific topic is much better to express exactly which expert a user is looking for [19]. For example, if a recruiter from a company is looking for a researcher to work on a specific project, it is more likely that she will use the detailed description of the project instead of a few related topics of this project to find the right person. This is the motivation behind *text-based query* research.

**Text-based query.** As an alternative, text-based query takes a descriptive text as the input and aims to find the top-$n$ experts whose research work is semantically similar to the given text [19]. A popular solution to text-based query is to: (1) leverage a representation model (e.g., `SciBERT` [22], `SBERT` [23]) to embed the semantics of each paper and query text into $d$-dimensional vectors, (2) compute the similarity of each paper to the query text, and (3) extract the top-$n$ experts from the most similar papers to the query text [4], [19], [21], [23]. Besides the implicit textual semantics of papers, the papers' explicit relationships are also important to document representation [24]. For example, if two papers have the same co-authors or cite each other or focus on the same topics, then they are likely to have the semantically similar textual content [18]. Therefore, graph embedding-based solutions [3], [25] consider the relationships of papers in a homogeneous paper-paper graph to enhance the effectiveness of expert finding. Specifically, a deep walk is conducted on the homogeneous graph to preserve the features of relationships in the learned vectors. Unfortunately, it does not distinguish the types and strengths of the relationships in papers, but simply treats them equally. Thus, it does not stimulate the potential value of considering relationships for expert finding.

**Problems of existing solutions.** Nowadays, text-based query has become the main-stream query form of expert finding. Although existing solutions w.r.t. text-based query are easy

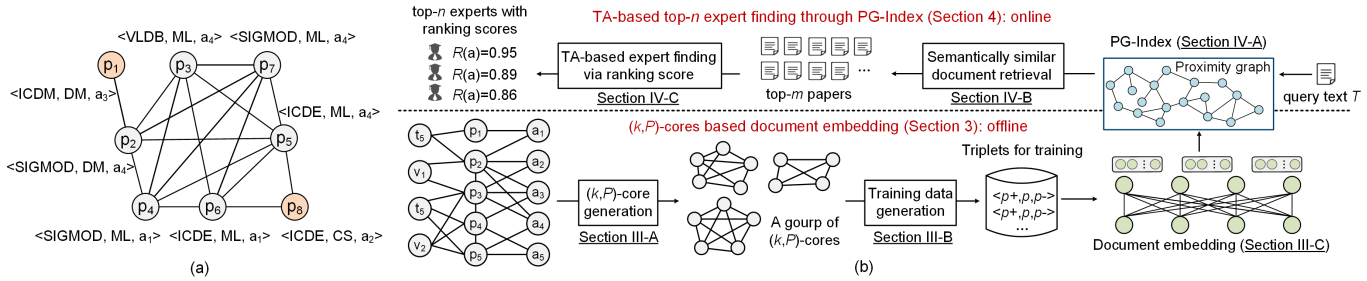



Fig. 1: Academic expert finding. (a) A homogeneous graph, where each paper node has a triple to describe the venue, topic, and author. If two papers have an edge, that means they have at least one element of the triple that is the same. (b) Our solution to academic expert finding over a heterogeneous graph.

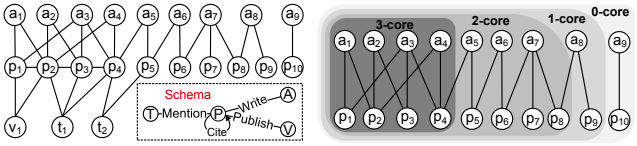
to implement, they are problematic for the following reasons. (1) The original representation-based solutions [4], [19], [21], [23] only consider papers' implicit semantics but ignore the important explicit structural relationships among papers, which leaves much room for effectiveness improvement. (2) Although the graph embedding-based solution [3], [25] considers the structural relationships, it still suffers, because the paper-paper relation in a homogeneous graph will introduce noises for embedding, which hampers the effectiveness. For instance, as Figure 1(a) shows, two papers $p_5$ and $p_8$ both appear in ICDE; hence, they have a relation in the homogeneous paper-paper graph, but they are not semantically similar because they focus on completely different research topics (*machine learning* for $p_5$ and *community search* for $p_8$). In other words, it is necessary and important to distinguish different types of relationships among papers and carefully select the valuable ones for document embedding. (3) A "*free-rider effect*" will occur, because a paper with only one relation (e.g., $p_1$) is considered equally in the embedding model as other papers having more relations. In fact, the cohesiveness of $p_1$ to other papers (e.g., $\{p_2, p_3 \cdots p_7\}$) is not that high, indicating it is of less value for embedding and should be treated differently. (4) Worse still, in real-world scenarios, papers are usually involved in a *heterogeneous graph* (see Figure 1(b) (left bottom)) having different types of nodes (e.g., paper $p_1$, author $a_1$, topic $t_1$, and venue $v_1$, etc.) and edges (e.g., $a_1$-Write-$p_1$, $p_2$-Cite-$p_3$, and $p_1$-Mention-$t_5$, etc.), such as the DBLP academic network. So, existing solutions on homogeneous graphs cannot be deployed directly on heterogeneous graphs. Though we can convert a heterogeneous graph to a homogeneous graph by [26] then apply the aforementioned solutions to find the top-$n$ experts, it adds expensive overhead in generating the homogeneous graph and the above problems (2)-(3) still exists.

As discussed above, none of the existing works can well support the academic expert finding by carefully considering both the implicit textual semantics and the explicit structural relationships of papers, over more general heterogeneous graphs. This motivates us to present our solution as follows.

**Our solution and contributions.** To handle the above issues, we first propose an offline $(k, \mathcal{P})$-core based document embedding model to represent the academic papers and query input text in the same vector space. To the best of our knowledge, we are the first to integrate papers' textual semantics and structural relations into the same model for heterogeneous graph embedding, by using the $(k, \mathcal{P})$-core. A specific meta-path $\mathcal{P}$ in a graph indicates a certain relation between two nodes, e.g., $P$-$A$-$P$ ($P$ and $A$ stands for paper and author) is a meta-path showing the co-authorship of papers (defined in **§II**). A $(k, \mathcal{P})$-core is a cohesive subgraph of which each node is connected to at least other $k$ nodes via the given meta-path $\mathcal{P}$ (defined in **§II**). By using the $(k, \mathcal{P})$-core, we can consider not only different relationships for papers through different meta-paths, but also consider the relations' cohesiveness through the size of $k$ to eliminate the "free-rider effect", which improves the embedding quality. Then, we present an online threshold algorithm (TA)-based method to efficiently return top-$n$ experts via a carefully designed proximity graph-based index (PG-Index). Figure 1(b) shows the whole pipeline and our contributions can be concluded as follows.

**(1) $(k, \mathcal{P})$-core based embedding.** Given a heterogeneous graph $G$, our $(k, \mathcal{P})$-core based embedding consists of three major steps. First, we present an optimized algorithm to search a group of $(k, \mathcal{P})$-cores from $G$, as the cohesive communities of papers (**§III-A**). Each paper in the same $(k, \mathcal{P})$-core community is regarded as closely structural-related via $\mathcal{P}$. For example, if we set $\mathcal{P} = P$-$A$-$P$, then the obtained $(k, \mathcal{P})$-core contains papers that share a large number of co-authors. It's natural that these papers focus on the similar research areas and are likely to be similar in their textual contents. Second, we design a sampling method to obtain positive and negative papers w.r.t. a seed paper as training data (**§III-B**), based on above $(k, \mathcal{P})$-core communities. Third, we apply a pre-trained textual representation model as an encoder to combine the textual semantics and structural relations of papers from the training data into vectors, we then define a triplet loss function to fine-tune the pre-trained representation model to output the final embeddings $\boldsymbol{E}$ of all papers (**§III-C**). As a result, only the papers with similar textual contents that are cohesively connected in a $(k, \mathcal{P})$-core are showing the higher similarity.

**(2) TA-based top-$n$ expert finding through PG-Index.** Given a user input query text $T$ and the learned embeddings $\boldsymbol{E}$, our top-$n$ expert finding consists of three major steps. First, we build a proximity graph-based index (PG-Index) in **§IV-A** according to $\boldsymbol{E}$ (build offline but use it online). Notice that, each node in the PG-Index represents a paper and each paper only has edges to several other papers that are similar to itself (measured by $L2$ norm distance of papers' representations). Second, we apply the fine-tuned representation model to get the representation of $T$, denoted by $\vec{v}_T$, and then we search the best $m$ papers having the




(a) A heterogeneous academic graph (b) $(k, \mathcal{P})$-core for $k$ = {0,1,2,3} and $\mathcal{P}$ = P-A-P

Fig. 2: A heterogeneous academic graph and $(k, \mathcal{P})$-cores

representations $\vec{v}_p$ that are similar to $\vec{v}_T$ by using the PG-Index (**§IV-B**). Third, given $m$ retrieved papers, we extract all the authors who have appeared in $m$ papers as candidates. We then define a ranking score of each author according to her contributions to her authored paper. Finally, we present a threshold algorithm (TA)-based method to obtain the top-$n$ experts having the smallest ranking score efficiently, without scanning and ranking all candidate experts (**§IV-C**). In our implementation, we set $m > n$ to ensure that we can obtain sufficient top-$n$ experts from $m$ papers.

**(3) Extensive evaluation on real-world datasets.** We conduct extensive experiments to evaluate: *effectiveness and efficiency* (**§VI-B & §VI-C**), *parameter sensitivity* (**§VI-D**), and *overhead of PG-Index* (**§VI-E**). The experimental results demonstrate our solution's superiority over all competitors from the domain of text-based expert finding.

We also optimize our solution by adopting different valuable meta-paths simultaneously in the $(k, \mathcal{P})$-core based embedding model (**§V**). Related work is discussed in **§VII**, and we concluded this paper in **§VIII**.

## II. Preliminaries

*Definition 1:* **Heterogeneous graph** $G$ [27]. A heterogeneous graph is defined as a graph $G = (V, E, L)$, where $V$ is the node set, $E$ is the edge set, and $L$ is a label function. (1) Each node $v \in V$ has a node type $\phi(v)$ by a node type mapping function $\phi : V \to \mathcal{A}$ ($\mathcal{A}$ is a set of node types). (2) Each edge $e \in E$ has an edge type $\psi(e)$ by an edge type mapping function $\psi : E \to \mathcal{R}$ ($\mathcal{R}$ is a set of edge types). (3) The label function $L$ is used to specific information for each node as $L(v)$ in the form of text. (4) $G$ consists of multiple node types and edge types satisfying $|\mathcal{A}| > 1$ and $|\mathcal{R}| > 1$.

*Definition 2:* **Schema of** $G$. Given a heterogeneous graph $G = (V, E, L)$ with the node type mapping $\phi : V \to \mathcal{A}$ and the edge mapping $\psi : E \to \mathcal{R}$, the schema of $G$ is defined as a graph $T_G = (\mathcal{A}, \mathcal{R})$ with the node set $\mathcal{A}$ and edge set $\mathcal{R}$.

In this paper, we tackle the academic expert finding problem in an academic network, such as DBLP, which is a typical heterogeneous graph. An example is illustrated as follows.

*Example 1:* Figure 2(a) illustrates a heterogeneous graph $G$ with the DBLP schema (right bottom). In this example, the node set $V$ contains four node types ($\mathcal{A} = \{\texttt{A}, \texttt{P}, \texttt{V}, \texttt{T}\}$), including a set of authors with type `A`, papers with type `P`, venues with type `V`, and topics with type `T`, e.g., $\phi(a_1) = \texttt{A}$, $\phi(p_1) = \texttt{P}$, $\phi(v_1) = \texttt{V}$, and $\phi(t_1) = \texttt{T}$. All the relationships among authors, papers, venues, and topics are described by the edge types $\mathcal{R} = \{\texttt{Publish}, \texttt{Mention}, \texttt{Write}, \texttt{Cite}\}$, e.g., $\psi(a_1p_1) = \texttt{Write}$, $\psi(p_1v_1) = \texttt{Publish}$, $\psi(p_2t_1) = \texttt{Mention}$, and $\psi(p_1p_2) = \texttt{Cite}$. Moreover, each paper is labeled with its title and abstract as $L(p) = \text{title} + \text{abstract}$.

*Definition 3:* **Meta-path** $\mathcal{P}$ [27]. A meta path $\mathcal{P}$ is a path defined on the schema $T_G = (\mathcal{A}, \mathcal{R})$, and an $l$-hop meta-path is denoted in the form of $\mathcal{P} = A_1 \overset{R_1}{-} A_2 \overset{R_2}{-} \cdots \overset{R_l}{-} A_{l+1}$, where $A_i \in \mathcal{A}$ and $R_i \in \mathcal{R}$ $(1 \le i \le l)$.

Since the relation between different type of nodes are clear in this paper, we ignore $R_i$ in $\mathcal{P}$ for simplicity and only use the node types to denote a meta-path. For example, given two papers in a heterogeneous graph $G$, different relationships between them can be described by different meta-paths, such as $P$-$A$-$P$ (co-authorship), $P$-$T$-$P$ (same topics), and $P$-$P$ (citation). Moreover, we say a path $(v_1, v_2 \cdots v_{l+1})$ in $G$ is a path instance of a meta-path $\mathcal{P}$ in $T_G$, if $\forall i$, $\phi(v_i) = A_i$ and $\psi(e_i) = R_i$ for each node $v_i$ and edge $e_i = v_iv_{i+1}$.

*Definition 4:* $\mathcal{P}$**-neighbor**. Given two nodes $\{v_i, v_j\}$ of a heterogeneous graph $G$, and a meta-path $\mathcal{P}$, $v_j$ is a $\mathcal{P}$-neighbor of $v_i$, if $v_j$ is connected to $v_i$ via a path instance of $\mathcal{P}$.

*Example 2:* As Figure 2(a) shows, the path $(p_1, a_1, p_2)$ is a path instance of $\mathcal{P}$=$P$-$A$-$P$, indicating that $p_1$ and $p_2$ have the same author $a_1$. Moreover, $p_2$ is a $\mathcal{P}$-neighbor of $p_1$.

In this paper, we aim to use a specific meta-path $\mathcal{P}$ between two papers to capture the cohesive $k$-core [28], [29] communities of papers w.r.t. this given $\mathcal{P}$, and then leverage them to improve the embedding quality. In the rest of the paper, when we mention a meta-path $\mathcal{P}$, it stands for the one between two papers, such as $P$-$A$-$P$. We next define the $(k, \mathcal{P})$-core in a heterogeneous graph $G$ as follows.

*Definition 5:* $(k, \mathcal{P})$**-core**. Given a heterogeneous graph $G$, a meta-path $\mathcal{P}$, and a non-negative integer $k$, the $(k, \mathcal{P})$-core on $G$ is defined as $G^k_{\mathcal{P}}$ satisfying: (1) $G^k_{\mathcal{P}} \subseteq G_{\mathcal{P}}$ is a maximal subgraph of $G$. (2) For each paper node $p$ of $G^k_{\mathcal{P}}$, it has $deg(p) \ge k$, where $deg(p)$ is the number of $\mathcal{P}$-neighbors of $p$.

*Example 3:* We use $(k, \mathcal{P})$-core to describe a cohesive community of papers and the cohesiveness increases as $k$ increases. Figure 2(b) illustrates four $G^k_{\mathcal{P}}$ with $k = \{0, 1, 2, 3\}$ and $\mathcal{P} = P$-$A$-$P$ for the heterogeneous graph $G$ in Figure 2(a). Notice that, $G^0_{\mathcal{P}}$ is the loosest community of papers that contains all the papers (even the independent paper $p_{10}$). As $k$ increases, $G^k_{\mathcal{P}}$ is getting more and more cohesive. Finally, $G^3_{\mathcal{P}}$ is the most cohesive community of papers, where each paper in $G^3_{\mathcal{P}}$ shares co-authors with another three papers.

By setting different meta-paths, we can achieve different communities of papers. For instance, the $(k, \mathcal{P})$-core for $P$-$T$-$P$ indicates that all the involved papers share the same topics with at least $k$ other papers. Or, the involved papers cite or be cited by at least $k$ other papers for $P$-$P$. When $k$ is large enough, we can say that all the papers in the $(k, \mathcal{P})$-core show greater similarity w.r.t. the given $\mathcal{P}$.

### A. Problem Definition

Given the aforementioned definitions, we are interested in the following problem: finding a set of experts who are more

relevant to a user-input query text, based on the document-document similarity, by carefully considering the $(k, \mathcal{P})$-core community of papers in a heterogeneous academic graph.

*Problem 1:* Given a user-input query text $T$, a heterogeneous graph $G = (V, E, L)$, a non-negative integer $k$, and a meta-path $\mathcal{P}$, we aim to find top-$n$ experts $S_A \subseteq V$ that satisfies:

- $S_A$ contains the best $n$ experts (with node tyep A) who have the minimized ranking score w.r.t. $T$ (Eq. 1). $R(a_i)$ is the ranking score of the expert $a_i$ (formally defined in **§IV**).

$$S_A = \{\arg\min_{a_i} R(a_i)\}, \quad s.t. \quad |S_A| = n \tag{1}$$

- $\forall a_i \in S_A$, $\exists p_{ij} \in P(a_i)$ that belongs to the $(k, \mathcal{P})$-core $G^k_{\mathcal{P}}$, where $p_{ij}$ is the $j$-th paper of all the papers $P(a_i)$ of the author $a_i$. According to our definition of $R(a_i)$, the more papers of $a_i$ that belong to $G^k_{\mathcal{P}}$ and show the higher semantically similarity to $T$, the smaller $R(a_i)$ it is.

To handle this problem, we first design a $(k, \mathcal{P})$-core based embedding model to improve the representation quality of papers (**Sec III**) by considering both the implicit textual semantics and explicit structural relationships of papers, then we present a threshold algorithm (TA)-based algorithm to return the top-$n$ experts efficiently, through a proximity graph-based index (**Sec IV**). To be more precise, we expect to find the most similar papers to an input text $T$, then return the top-$n$ authors from these papers as the output experts.

## III. $(k, \mathcal{P})$-CORE BASED DOCUMENT EMBEDDING

Figure 3 illustrates the framework of our $(k, \mathcal{P})$-core based embedding model. First, we use a meta-path $\mathcal{P}$ to capture the cohesive $(k, \mathcal{P})$-core communities of papers (§III-A). Then, we design a sampling method to obtain the positive (green nodes that belong to the $(k, \mathcal{P})$-core) and negative papers (blue nodes that do not belong to the $(k, \mathcal{P})$-core) w.r.t. the given seed papers (red nodes that are selected from the $(k, \mathcal{P})$-core) as the training data (§III-B). Finally, we apply a pre-trained representation model as an encoder to combine the textual semantics and structural relations of papers into one vector, and define a triplet loss function to fine-tune the pre-trained model (§III-C). As a result, the positive papers are more likely to be similar to the seed paper than the negative papers.

### A. $(k, \mathcal{P})$-core Community Generation

We first take the meta-path $\mathcal{P} = P$-$A$-$P$ (co-authorship) as an example to discuss how to generate the $(k, \mathcal{P})$-core community of papers. In §V, we show how to further improve the embedding quality by combining more valuable meta-paths in our model. We next introduce our solution to quickly find a $(k, \mathcal{P})$-core community of papers through *community search*.

**Community search** [28]. Given a graph $H$ and an input node $u$, community search aims to find a maximal subgraph $H' \subseteq H$ containing $u$, in which all the nodes have a greater cohesiveness. Back to our case, given a heterogeneous graph $G$, a meta-path $\mathcal{P}$, and a seed paper $p_s$, we expect to find a $(k, \mathcal{P})$-core containing $p_s$ as the community. The papers' cohesiveness is measured by the size $k$ of $(k, \mathcal{P})$-core. In §III-B, we discuss how to select the seed paper.

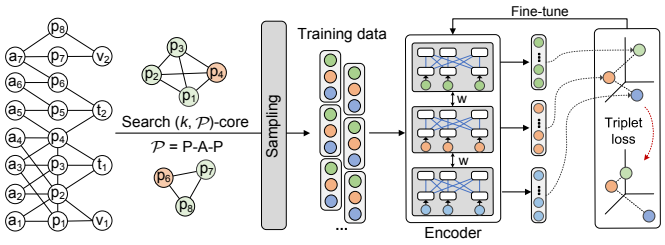


Fig. 3: Overview of $(k, \mathcal{P})$-core based document embedding model

**Straightforward solution.** A naive solution is to convert a heterogeneous graph $G$ into a homogeneous graph $G'$ according to the given meta-path $\mathcal{P} = P$-$A$-$P$, and then use a classic core decomposition algorithm [29] to find the $k$-core from $G'$ containing the seed paper $p_s$. However, this method is computationally expensive, because it enumerates all paper nodes to find all their $\mathcal{P}$-neighbors to form $G'$, even if a paper node is not connected to $p_s$.

**Extended solution.** As an alternative, an extended solution (FastBCore) is proposed in [30], which consists of two major steps: (1) a labeled search and (2) cleaning up nodes. In the first step, it uses the labeled search method to find every node connecting to the seed paper $p_s$ via path instances of $\mathcal{P}$. To be more precise, it starts a BFS from $p_s$ to find all its $\mathcal{P}$-neighbors and assign a label of each visited paper node to ensure that each paper is visited only once. Then, it repeats the above procedure from all the explored papers until no more $\mathcal{P}$-neighbors can be found. Finally, we obtain a set of candidate papers that could be involved in $(k, \mathcal{P})$-core, denoted by $S$. This BFS traversal avoids enumerating some unnecessary papers. For example, in Figure 2(a), if we set $p_s = p_4$, then $p_{10}$ will not be considered in BFS searching, because it is not a $\mathcal{P}$-neighbor of $p_4$. Next, in the second step, FastBCore finds up to $k$ $\mathcal{P}$-neighbors for each paper $p \in S$, and then iteratively removes those papers that do not satisfy the $k$-constraint from $S$ until all nodes in $S$ are satisfying the $k$-constraint, and finally returns $S$ as the output $(k, \mathcal{P})$-core.

However, this extended solution still has two limitations. (1) In the first step, a paper that does not satisfy the $k$-constraint still will be added in the candidate set $S$. Actually, these kinds of papers will never be included in the final $(k, \mathcal{P})$-core, so it is not worth continuing to search from them; instead, we should prune them early. (2) In some cases, the $k$-constraint is too strict, and may exclude some valuable papers from the $(k, \mathcal{P})$-core community. For example, in Figure 2(a), suppose that $p_s = p_4$, $p_s$ and $p_5$ are two papers with similar topics that are co-authored by the same author. Note that, $p_5$ has two $\mathcal{P}$-neighbors (we have $deg(p_5) = 2$). FastBCore will exclude $p_5$ from the final $(k, \mathcal{P})$-core community of papers for any $k >= 3$, but in this case, $p_5$ should be considered, as it would benefit to improve the embedding quality. Obviously, for the second limitation, we cannot solve it by simply setting a small $k$, because a small $k$ indicates that the $(k, \mathcal{P})$-core is not cohesive enough, such as the $(k, \mathcal{P})$-cores for $k = 1$ shown in Figure 2(b), thus including more worthless papers

**Algorithm 1:** $(k, \mathcal{P})$-core search

**Input:** A heterogeneous graph $G$, a seed paper $p_s$, a non-negative integer $k$, and a meta-path $\mathcal{P}$
**Output:** Community result $S$

```
1  Q ← {p_s}, S ← {p_s}, D ← ∅, Visit ← {p_s};
2  while Q ≠ ∅ do // candidate nodes selection
3      v = Q.poll();
4      for ∀u ∈ v's P-neighbors do
5          if !Visit.contains(u) then
6              Ψ[v].add(u);
7              Visit.add(u);
8      S.add(Ψ[v]);
9      if |Ψ[v]| >= k then
10         Q.addAll(Ψ[v]);
11     else D.add(v) ;
12 while D ≠ ∅ do // unpromising nodes prune
13     v ← D.poll();
14     S.remove(v);
15     for ∀u ∈ Ψ[v] && S.contains(u) do
16         Ψ[u].remove(v);
17         if Ψ[u].size() < k then
18             D.add(u);
19 N = getAllPNeighbor(p_s)// (k,P)-core extension;
20 return S.addAll(N)
```

(e.g., $p_6$, $p_7$, $p_8$, and $p_9$). We therefore present the following approach to resolve these issues.

**Our solution.** We optimize FastBCore from two aspects. (1) In the labeled search step, we expand the search space from a visited paper node, if this paper has at least $k$ $\mathcal{P}$-neighbors. Otherwise, we prune this paper and stop expanding the search space from it. We prove that this pruning operation is safe and has no effect on the output $(k, \mathcal{P})$-core by Theorem 1. (2) We not only return a strict $(k, \mathcal{P})$-core community containing the seed paper $p_s$, but also includes a small amount of $\mathcal{P}$-neighbors of $p_s$ having degree less than $k$. This can be viewed as a complement to the closest $(k, \mathcal{P})$-core community. In §VI, we show that the embedding quality is improved by these above two optimization strategies.

*Theorem 1:* Suppose $G_{\mathcal{P}}^k$ and $G_{\mathcal{P}}^k{}'$ are the $(k, \mathcal{P})$-cores returned by FastBCore and our solution. We have $G_{\mathcal{P}}^k = G_{\mathcal{P}}^k{}'$.

*Proof:* Suppose that $G_{\mathcal{P}}^k \neq G_{\mathcal{P}}^k{}'$, then there must $\exists p$ with $deg(p) \geq k$ that is pruned by our solution. To be more precise, we have $p \in G_{\mathcal{P}}^k$ and $p \notin G_{\mathcal{P}}^k{}'$. Since $p \in G_{\mathcal{P}}^k$, there must $\exists p^* \in G_{\mathcal{P}}^k$ that connects to $p$ via path instances of a given meta-path $\mathcal{P}$. Note that, this node $p^*$ also will be included in $G_{\mathcal{P}}^k{}'$ because it has $deg(p^*) \geq k$. Therefore, we still can include the paper $p$ by expanding the search space from $p^*$, which indicates that $p \in G_{\mathcal{P}}^k{}'$. This contradicts the assumption of $p \notin G_{\mathcal{P}}^k{}'$ and thus we have $G_{\mathcal{P}}^k = G_{\mathcal{P}}^k{}'$. □

We next introduce our solution based on the aforementioned two optimizations (as shown in Algorithm 1).

**Notations.** We use a queue $Q$ to record all the nodes that can be used for further search expansion. It is initialized as the seed paper $Q = \{p_s\}$. A hash set $S$ records all the nodes belonging to the $(k, \mathcal{P})$-core. An array $\Psi[v]$ is defined to record all the explored $\mathcal{P}$-neighbors of $v$, such as $\Psi[v] = \{u_1, \dots u_n\}$. For instance, if we set $\mathcal{P} = P\text{-}A\text{-}P$, then each pair of $\langle v, u_i \rangle$ represents two co-authored papers. We use a queue $D$ to record all the papers that should be removed, which do not satisfy the $k$-constraint. We also use a hash set Visit to record all visited nodes, avoiding duplicate access.

**Overview.** Our solution consists of three main steps: (1) *Candidate nodes selection*. For each node $v \in Q$, we find and record all $v$'s $\mathcal{P}$-neighbors in $\Psi[v]$ and $S$ (Lines 4-8). If $|\Psi[v]| >= k$, then we add all paper nodes in $\Psi[v]$ to the queue $Q$ for further search expansion. Otherwise, we add $v$ to the queue $D$ for pruning (Lines 9-11). (2) *Unpromising nodes prune*. In this step, we iteratively remove all paper nodes that do not meet the $k$-constraint from $S$ and this prune is safe according to Theorem 1. Since removing a paper node $v$ from $S$ will affect $v$'s $\mathcal{P}$-neighbors, we need to check if each $\mathcal{P}$-neighbor $u$ of $v$ still satisfies the $k$-constraint after removing $v$. If $|\Psi[u]| < k$, the node $u$ should also be removed from $S$ and then we add it to the queue $D$. We repeat this operation until all paper nodes in $S$ satisfy the $k$-constraint. In another word, all paper nodes from $S$ belong to the $(k, \mathcal{P})$-core $G_{\mathcal{P}}^k$ (Lines 13-18). (3) $(k, \mathcal{P})$-*core extension*. We expand $G_{\mathcal{P}}^k$ by adding all $\mathcal{P}$-neighbors of the seed paper $p_s$ that do not satisfy $k$-constraint to $S$, thereby relaxing the strict constraint of $(k, \mathcal{P})$-core to some extent (Lines 19-20).

*Example 4:* We show the $(k, \mathcal{P})$-core community search for $k = 3$, $\mathcal{P} = P\text{-}A\text{-}P$ on the heterogeneous graph provided in Figure 2(a), by setting the seed paper $p_s = p_4$. We start searching from $p_s$ ($Q = \{p_s\}$ at first) and collect all the $\mathcal{P}$-neighbors of $p_s$ in $\Psi[p_s]$ and $S$, e.g., $\Psi[p_s] = \{p_1, p_2, p_3, p_5\}$. Since $|\Psi[p_s]| >= 3$, we need to add all the paper nodes in $\Psi[p_s]$ to $Q$. We expand the search space by accessing each paper node in $Q$. For example, for paper $p_5 \in Q$, since $|\Psi[p_5]| < 3$, $p_5$ will be added to $D$, thereby pruning the search space from $p_5$, e.g., $\{p_6, p_7, p_8, p_9\}$. Now, we have $D = \{p_5\}$, $S = \{p_1, p_2, p_3, p_4, p_5\}$. Next, we should remove all the nodes in queue $D$. There is only one paper $p_5$ in $D$ so we remove $p_5$ from $S$. Since the removal of $p_5$ could decrease it's neighbor's degree, we should update it's neighbors' degree and keep removing the one that does not satisfy the $k$-constraint. We repeat these operations until $D$ is empty and we get $S = \{p_1, p_2, p_3, p_4\}$. Finally, we expand the above community appropriately by adding one $\mathcal{P}$-neighbor (but does not satisfy the $k$-constraint) of $p_s$. After that, we obtain the final result $S = \{p_1, p_2, p_3, p_4, p_5\}$. This result includes not only all the papers in $(k, \mathcal{P})$-core for $k = 3$ and $\mathcal{P} = P\text{-}A\text{-}P$, but also a valuable paper $p_5$ (have the same author and topic as the seed paper $p_4$, but does not satisfy the $k$-constraint).

### B. Sampling-based Training Data Generation

We aim to embed all the papers in a vector space in where two similar papers have a closer representation, and vice versa. Intuitively, we said that two papers are similar if (1) they both belong to the same cohesive $(k, \mathcal{P})$-core, for example, two papers having co-authorship ($\mathcal{P} = P\text{-}A\text{-}P$) show a certain degree of similarity and (2) two papers' textual semantics are similar. For the first condition, in this section, we propose

a sampling-based method to generate the training data based on the $(k, \mathcal{P})$-core communities. We define the training data as a set of triples of $\langle p^+, p_s, p^- \rangle$, where $p^+$ ($p^-$) is the positive (negative) sample w.r.t. a specific seed paper $p_s$ inside (outside) the $(k, \mathcal{P})$-core. For the second condition, we adopt a pre-trained model to capture the textual semantics (discussed in §III-C). We next introduce the main procedures of sampling based training data generation as follows.

**(1) Seed papers selection.** Given the meta-path $\mathcal{P} = P\text{-}A\text{-}P$, we randomly select a fraction $f$ of the paper nodes through a simple random sampling from the heterogeneous graph $G = (V, E, L)$ as our seed papers, that is $\{p_s^1, \cdots, p_s^r\}$ with $r = f \cdot |V(P)|$, where $V(P)$ denotes all paper nodes of $G$ (with node type `P`). For each seed paper $p_s$, we compute a $(k, \mathcal{P})$-core containing $p_s$ and leverage this community to generate a set of training data $\langle p^+, p_s, p^- \rangle$. By combing all the training data generated from each $(k, \mathcal{P})$-core community for every $p_s$, we can get the final training data. The reason that we need a set of seed papers instead of one is that we expect as much as possible papers could be covered by the returned communities, so that we can get more representative samples instead of getting a limited number of training data from only one $(k, \mathcal{P})$-core community. The parameter $f$ is used to control the number of seed papers, thus control the size of training data. The greater $f$ is, the more training data can be generated. In Sec VI, we show the effect of $f$ on expert finding.

**(2) Positive and negative samples collection.** We first show the definitions of positive and negative samples and then discuss how to collect them effectively.

*Definition 6:* **Positive sample**. Given a $(k, \mathcal{P})$-core $G_{\mathcal{P}}^k$ containing a seed paper $p_s$, we define each paper node in $G_{\mathcal{P}}^k$ (exclude $p_s$) is a positive sample w.r.t. $p_s$, denoted by $p^+$.

*Definition 7:* **Negative sample**. Given a $(k, \mathcal{P})$-core $G_{\mathcal{P}}^k$ containing a seed paper $p_s$, we define each paper node in $G_{\mathcal{P}} \backslash G_{\mathcal{P}}^k$ is a negative sample w.r.t. $p_s$, denoted by $p^-$.

Note that, the $(k, \mathcal{P})$-core $G_{\mathcal{P}}^k$ containing $p_s$ is usually a small, cohesive subgraph that contains a slight number of papers, so that we directly use all papers in $G_{\mathcal{P}}^k$ as positive samples w.r.t. $p_s$. On the contrary, the number of papers in $G \backslash G_{\mathcal{P}}^k$ is obviously larger than $G_{\mathcal{P}}^k$. We cannot directly take all papers in $G \backslash G_{\mathcal{P}}^k$ as negative samples w.r.t. $p_s$, because as the size of training data getting larger, the training time increases significantly. Suppose we have 100 papers in $G_{\mathcal{P}}^k$ and 100000 papers in $G \backslash G_{\mathcal{P}}^k$, then we have $100 \times 100000 = 10^7$ triples for only one community. Worse still, the imbalance in the number of positive and negative samples will cause a trained model with obvious bias [31]. The negative sample collection strategy has an important impact on training efficiency and effectiveness. Hence, we present two different strategies for negative samples collection and show their effect in §VI.

**Random Negative.** We select a negative sample from $G \backslash G_{\mathcal{P}}^k$ by simple random sampling. Specifically, for each $p^+ \in G_{\mathcal{P}}^k$, we randomly select $s$ negative samples, thus we have $s$ different triples $\langle p^+, p_s, p^- \rangle$ for a specific $p^+$. Although a random negative solution is easy to implement, it still has an obviously drawback: the paper that is close to the seed paper $p_s$ and the paper that is far away from $p_s$ have the same probability of being sampled from $G \backslash G_{\mathcal{P}}^k$ and contribute equally to the embedding model. In fact, the paper that is close to $p_s$ is more valuable for training than the far away papers [25], because if we can distinguish the close papers into papers unsimilar to $p_s$, then the far away papers probably be distinguished into papers unsimilar to $p_s$. So, we propose an alternative strategy to select those papers that do not belong to $G_{\mathcal{P}}^k$ but are close to $G_{\mathcal{P}}^k$ as negative samples.

**Near Negative.** As mentioned in Algorithm 1, we add those papers with less than $k$ $\mathcal{P}$-neighbors into a delete queue $D$ for pruning, once we found them at the first time. Actually, these papers are such of close papers to $G_{\mathcal{P}}^k$. Therefore, for each positive sample $p^+ \in G_{\mathcal{P}}^k$, we randomly select $s$ papers from $D$ as negative samples to form $s$ triples. In §VI, we empirically found that $s = 3$ is enough to achieve a balance between training efficiency and expert finding effectiveness. We also show that using the near negative is better than random negative for expert finding effectiveness.

**(3) Training data generation.** From these positive and negative samples, we get the training data as a set of triples $\langle p^+, p_s, p^- \rangle$. For each paper in a triple, we take the paper's textual information as the input of our embedding model, namely $\langle L(p^+), L(p_s), L(p^-) \rangle$, where $L(p) = \mathsf{title}{+}\mathsf{abstract}$.

### C. Document Embedding

Given a set of triples $\langle p^+, p_s, p^- \rangle$, our document embedding has three steps: (1) We use a pre-trained model to encode the textual information of each paper as its initial semantically representation. (2) We present a new *triplet loss* based on the positive and negative samples to fine-tune each paper's initial semantically representation. To be more precise, we use the structural relations of papers to fine-tune the semantic representation (3) We train the model via gradient descent.

**Document Encoder.** In this paper, we use the pre-trained model `SciBERT` [22] as the encoder, as it has been optimized for scientific texts and is more suitable for representing academic textual contents [25]. According to the input format of `SciBERT`, we first use `Wordpiece` [32] to split the textual information $L(p)$ of each paper $p$ into tokens. If the number of tokens exceeds the default maximum length limit (512) of `SciBERT`, we then truncate the tokens to meet the limit.

Figure 4 shows the architecture of our encoder. Given a $L(p) = \{w_1, w_2, \ldots, w_l\}$ that is tokenized by `Wordpiece`, each $w_i$ indicates a token of $L(p)$. Our document encoder aims to obtain an initial representation of $L(p)$, denoted by

$$\vec{v}_p = \Phi_P(\Phi_B(\{w_1, w_2, \ldots, w_l\}; \Theta_B)) \quad . \tag{2}$$

Notice that, $\Theta_B$ are the parameters of `SciBERT` and $\Phi_B : w_i \rightarrow \vec{w}_i$ is used to encode each token $w_i \in L(p)$ by using $\Theta_B$. Then, we apply a pooling method to extract the most valuable features from $\Phi_B(L(p); \Theta_B)$ and generate the representation of $L(p)$, that is, $\Phi_P : \{\vec{w}_i\} \rightarrow \vec{v}_p$. Two pooling strategies are used widely: (1) *Mean pooling* outputs the average representations of all tokens, that is $\Phi_P = \frac{1}{l} \sum_{i=1}^{l} \vec{w}_i$. (2)

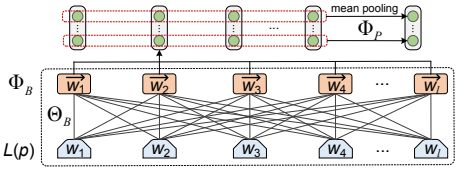


Fig. 4: Illustration for the document encoder

*Max pooling* takes the maximum value of the $j$-th dimension of all tokens' representations as the $j$-th dimension of $\vec{v}_p$, that is, $\Phi_P = \{max\{\vec{w}_{1j}, \vec{w}_{2j} \cdots \vec{w}_{lj}\}\}_{j=1}^{d}$, where $\vec{w}_{ij}$ is the $j$-th dimension of $\vec{w}_i$ (each $\vec{w}_i$ has $d$ dimensions). We adopt mean pooling as $\Phi_P$ in this paper due to its better performance than max pooling [23]. By applying this encoder, the implicit textual semantics of a paper is embedded to the representation $\vec{v}_p$. We next present a *triplet loss* based on the positive and negative samples to fine-tune $\vec{v}_p$.

**Triplet Loss.** In order to fine-tune the initial representations, we present a margin-based triplet loss to update the parameters $\Theta_B$. The basic idea of triplet loss is straightforward to make those structurally cohesive papers $(p_s, p^+)$ closer in the vector space and papers that are not cohesively connected $(p_s, p^-)$ as far away as possible. The triplet loss is provided in Eq. 3 and we aim to minimize this loss function:

$$\mathcal{L}(\Theta_B) = max\{d(\vec{v}_{p_s}, \vec{v}_{p^+}) - d(\vec{v}_{p_s}, \vec{v}_{p^-}) + c, 0\}, \quad (3)$$

where $\vec{v}_{p_s}, \vec{v}_{p^+}, \vec{v}_{p^-}$ are the representations of the seed node, positive sample, and negative sample, respectively. The margin hyperparameter $c$ (we empirically choose $c$ = 1) is used to ensure that $p^+$ is closer to $p_s$ than $p^-$ in terms of $L2$ norm distance $\delta$, that is, $\delta(\vec{v}_i, \vec{v}_j) = \|\vec{v}_i - \vec{v}_j\|_2$.

**Model Training** We use the `Adam optimizer` [33] for training following the suggested hyperparameter values in [34], i.e., $LR = 2e-5$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. Notice that, the hyperparameters $\beta_1$, $\beta_2$ control the exponential decay rates. We refer the interested readers to [33] for more details. We minimize the loss function (Eq. 3) thus to optimize the model parameters $\Theta_B$. For each paper, we can obtain its new representation by applying Eq. 2 with this optimized $\Theta_B$. In this paper, we denote the embeddings of all papers in the heterogeneous graph $G$ by $\boldsymbol{E} = \{\vec{v}_{p_1}, \cdots \vec{v}_{p_{|V(P)|}}\}$, where $|V(P)|$ is the number of paper nodes in $G$. In §IV, we will introduce how to use $\boldsymbol{E}$ for top-$n$ experts finding.

## IV. Top-$n$ Expert Finding via Embeddings

We now discuss how to find the top-$n$ experts who are semantically match with the given query text $T$, based on the document embeddings $\boldsymbol{E}$ obtained above. Figure 1(b) (top part) shows three major steps of our expert finding: (1) We build a proximity graph-based index (PG-Index) for all papers via $\boldsymbol{E}$ (§IV-A). This is an offline operation, which is processed once and can be used for online expert finding. (2) Given a query text $T$, we apply the fine-turned encoder (Figure 4) to get the representation of $T$, denoted by $\vec{v}_T$, and then we search the best $m$ papers having the representations $\vec{v}_p$ that are semantically similar to $\vec{v}_T$, by using the PG-Index (§IV-B). (3) Given $m$ retrieved papers, we extract all the authors who have appeared in $m$ papers as candidates and present a threshold algorithm (TA)-based method to get the top-$n$ experts having the smallest ranking scores efficiently (§IV-C).

### A. Proximity Graph-based Document Index

A straightforward way to retrieve the best $m$ papers that are semantically similar to the given query $T$ is to enumerate all papers, computer pair-wise semantic similarities between each paper and $T$, and return $m$ papers with the greatest similarities. Although it is easy to implement, this straightforward method is computationally expensive because of the time-consuming enumeration. Comparing to the tardy exact $m$ papers, it is more desirable if we can quickly obtain approximate $m$ papers with acceptable accuracy. This motivates us to build a proximity graph based index (PG-Index) for approximate paper retrieval. The fundamental component of PG-Index is actually a $k$ nearest neighbor ($k$NN) graph [35] and we present a refinement strategy to further improve it. We have the following steps to build PG-Index, as shown in Algorithm 2.

**(1) Navigating node selection.** The navigating node is the entry node of the PG-Index, where we can start the approximate paper search. We define the navigating node $q$ as the approximate centroid of all papers $V(P)$ ($V(P)$ is the set of paper nodes in a heterogeneous graph $G$). To be more precise, the representation of $q$, denoted by $\vec{v}_q$, is closest to the representation $\vec{v}_g$ of the centroid $g$ of all papers, where $\vec{v}_g = \frac{1}{|V(P)|}\sum_{\vec{v}_p \in \boldsymbol{E}} \vec{v}_p$ (Lines 1-2).

**(2) Initialize $k$NN graph.** We apply the paper embeddings $\boldsymbol{E}$ to build a $k$NN graph via NNDescent [36] (a commonly used method to construct $k$NN graphs). Specifically, we first find the $k$ nearest $\mathcal{P}$-neighbors of a paper $p$ ($N(p)$) by evaluating the $L2$ norm distance (Line 4), and then add an edge between $p$ and $\forall x \in N(p)$ in the original $k$NN graph (Line 6).

**(3) Refining neighbors.** According to [35], searching on the original $k$NN graph suffers from the low search efficiency. Given a query text $T$, the standard search procedure can be described as: starting from the navigating node $q$, we select the paper that is closest to $T$ from $q$'s $k$ nearest neighbors as the next-hop node and expand the search space greedily to approach $T$ step by step. This greedy search is problematic when the query $T$ is far away from the navigating node $q$, because we need more search expansions to approach $T$, thereby increasing the search overhead. A possible solution to this issue is to add some long-distance neighbours for some papers in the original $k$NN graph, so that we can quickly approach $T$ through these "highway" edges, significantly saving the computation cost. For instance, as Figure 5(a) shows, we need to expand search space 9 times (red edges) and visit 26 papers, on the original $k$NN graph. By contrast, for PG-Index (Figure 5(b)), we only require 5 expansions (blue edges) and visit 18 papers. To achieve this, we refine the neighbors for each paper in $k$NN graph via the following two operations.

**Long-distance neighbors extension.** For each paper $p$ of the original $k$NN graph, we access every paper $x \in N(p)$ and add an edge between $p$ and each paper $y \in N(x)$. By doing

**Algorithm 2:** PG-Index construction

**Input:** all document embeddings $\boldsymbol{E}$, all papers nodes $V(P)$
**Output:** proximity graph based document index PG-index

1 $\vec{v}_g = \frac{1}{|V(P)|}\sum_{\vec{v}_p \in \boldsymbol{E}} \vec{v}_p$ // navigating node selection
2 $\vec{v}_q$ = getNearestNode($\boldsymbol{E}$, $\vec{v}_g$);
3 **for** $p \in V(P)$ **do**
  // initialize $k$NN graph
4  $N(p)$ = getKNearestNodes($\boldsymbol{E}$, $\vec{v}_p$);
5  **for** $x \in N(p)$ **do**
6  PG-Index.addEdge($p$, $x$);
7  **for** $y \in N(x)$ **do** // neighbors expansion
8  PG-Index.addEdge($p$, $y$);
9 **for** $p \in$ PG-Index **do**
10  **for** $\{x, y\} \in N(p)$ **do** // neighbors removal
11  **if** $\delta(x, y) \leq \delta(y, p)$ **then**
12  PG-Index.removeEdge($p$, $y$);
13 **return** PG-index;

this, we build some "highway" edges that can quickly guide the search to the nodes two hops away from $p$ (Lines 7-8). Although this operation can improve the search efficiency, it still introduces overhead on the index construction and storage. To reduce such overhead to some extent, we need to remove some redundant long-distance neighbors in the next operation.

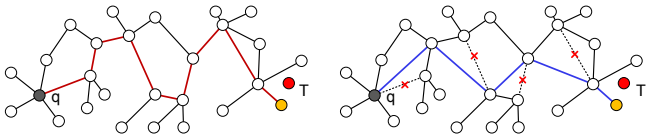
**Redundant neighbors removal.** Given the expanded $k$NN above, let us consider two neighbors $x$ and $y$ of $p$ ($\{x, y\} \in N(p)$), we say $y$ is a redundant paper compared to $x$ iff $\delta(y, x) \leq \delta(x, p)$ ($\delta$ is the $L2$ norm distance), and we can remove $y$ from $N(p)$ (Lines 9-12). The logic behind this removal is that if the distance between $y$ and $x$ is close enough, then we can approach $y$ quickly by accessing $x$ from $p$, instead of a long-distance edge from $p$ directly to $y$, thereby reducing the redundant long-distance edges. In other words, we expect that the long-distance edge can guide the search to the nodes far away from $p$, so as to well support the search scenario that query text $T$ is far away from the navigating node $q$.

### *B. Semantically Similar Document Retrieval*

Given a PG-Index and a query text $T$, we take the navigating node $q$ as the current visiting node $s$ to start the search. Specifically, we first add $s$ and all its neighbors $N(s)$ in PG-Index to the priority queue $Q$, where all the paper nodes in $Q$ are sorted in ascending order according to their distance to $T$. Then, we sequentially extract the paper node (from $Q$) that is closest to $T$ and have not been visited, as the new current visiting node $s$. We repeat the above steps until all the papers in $Q$ have been visited. Finally, we take the first $m$ papers in $Q$ as the top-$m$ papers, denoted by $\mathcal{D}$. Next, we will introduce how to use the retrieved top-$m$ papers to find the best $n$ experts who are semantically match with the given query text $T$.

### *C. Top-$n$ Expert Finding*

Given a query $T$ and the top-$m$ papers $\mathcal{D}$ that are semantically similar to $T$, we do the top-$n$ expert finding as follows. We first extract all the authors that have appeared in $\mathcal{D}$ as the candidate experts denoted by $\mathcal{C}$. Then, we define a ranking score to measure how an expert semantically matches with $T$. Finally, we present a threshold algorithm (TA)-based method to quickly return the top-$n$ experts based on the ranking score, without scanning and ranking all candidate experts in $\mathcal{C}$.



(a) Original $k$NN proximity graph (b) Our PG-Index after refinement
Fig. 5: An example of PG-Index

**Candidate experts.** Because we have obtained the best $m$ papers $\mathcal{D} = \{p_1, p_2, \cdots, p_m\}$ that are semantically similar to the given query text $T$, we directly extract all the authors for each paper $p \in \mathcal{D}$, denoted by $C_p$, and combine them to be the candidate experts to $T$, denoted by $\mathcal{C} = C_{p_1} \cup C_{p_2} \cup \cdots \cup C_{p_m}$.

**Expert ranking score.** We first define the expert score regarding a specific paper and then compute the final expert ranking score based on it. In the document-centric expert finding literatures, the retrieved paper is usually assumed as the "expertise evidence" of experts used to score the candidate experts [2], [15]. The reciprocal ranking [37] is a simple and effective method for expert finding, which takes the inverse value of a paper's rank as the expert score. This ranking gives a huge boost to candidates with at least one paper well ranked and tends to promote candidates with more papers than others. However, it ignores the fact that the contribution of each author in the same paper is different. As an alternative, we define the expert score w.r.t. a specific paper, denoted by $S(a, p)$, by considering an expert's contribution to this paper as:

$$S(a, p) = \frac{1}{I(p)} \cdot w(a, p) \quad , \tag{4}$$

where $I(p)$ indicates the rank of the paper $p$ in $\mathcal{D}$, and $w(a, p)$ indicates the contribution of the author $a$ to $p$. Instead of setting the same weight for all authors in $C_p$, we assume that the contribution increases as the author's rank increases [38], [39]. Therefore, in this paper, we assign the contribution weight by using a `Zipf` distribution [40] as follows:

$$w(a, p) = \frac{1}{I(a) \cdot \sum_{i=1}^{|C_p|}(1/i)} \quad , \tag{5}$$

where $I(a) = \{1, 2, \cdots, |C_p|\}$ is the rank of the candidate expert $a$ in the paper $p$, and $|C_p|$ represents the total number of authors in $p$. This method emphasizes that a highly ranked author has a significantly higher weight. Given an expert $a$, we define the expert ranking score $R(a)$ w.r.t. all the papers in $\mathcal{D}$ as the sum of its expert score w.r.t. every paper (Eq. 6).

$$R(a) = \sum_{p_i \in \mathcal{D}} S(a, p_i) \tag{6}$$

**TA-based top-$n$ experts finding.** We present a threshold algorithm (TA)-based method to find the top-$n$ experts efficiently without scanning and ranking all candidate experts. First, for each paper $p_j \in \mathcal{D}$, we record all candidate experts' scores w.r.t. $p_j$ in a ranked list $L_j$ in descending order (see Figure 6). Notice that, if an author $a$ does not appear in $p_j$, then

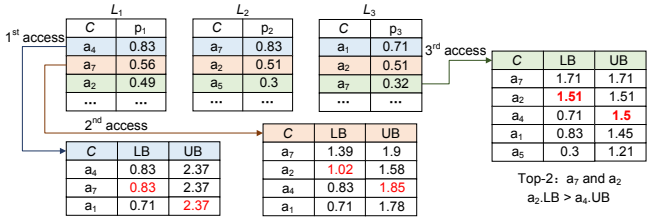



Fig. 6: TA-based top-$n$ expert finding

its expert score w.r.t. $p_j$ is set to be zero ($S(a, p_j) = 0$). Since we have $m$ papers in $\mathcal{D}$, we finally get $m$ ranked lists. Second, we access all lists from top to bottom and update the upper and lower bound of expert ranking score for each visited author $a_i$, denoted by $\overline{R}(a_i)$ and $\underline{R}(a_i)$, respectively. Finally, we terminate it early if $n$ experts are found, of which the smallest $\underline{R}$ is larger than other experts' greatest $\overline{R}$.

We now show how to update $\overline{R}(a_i)$ and $\underline{R}(a_i)$. According to Eq. 6, the expert ranking score $R(a_i)$ is computed by aggregating $S(a_i, p_j)$ over all papers in $\mathcal{D}$. Intuitively, if we know the bounds of $S(a_i, p_j)$, denoted by $\overline{S}(a_i, p_j)$ and $\underline{S}(a_i, p_j)$, of an expert from each $L_j$, then we can compute the bounds of the expert ranking score as follows.

$$\overline{R}(a_i) = \sum_{p_j \in \mathcal{D}} \overline{S}(a_i, p_j) \quad \text{and} \quad \underline{R}(a_i) = \sum_{p_j \in \mathcal{D}} \underline{S}(a_i, p_j) \quad (7)$$

**Upper bound of $R(a_i)$.** At each access to the ranked list $L_j$, if an author $a_i$ is not visited so far, then we have $\overline{S}(a_i, p_j) = S(a_x, p_j)$, where $S(a_x, p_j)$ is the expert score w.r.t. $p_j$ of the current accessed expert $a_x$ in $L_j$. This is because the expert is ranked in descending order in each $L_j$ according to its expert score w.r.t. $p_j$, so that all the unaccessed experts from $L_j$ must have a smaller expert score than $S(a_x, p_j)$. If we find $a_i$ from $L_j$, then we have $\overline{S}(a_i, p_j) = S(a_i, p_j)$. The upper bound $\overline{S}(a_i, p_j)$ is decreased as TA-based access executes. By applying Eq. 7, we can update the upper bound $\overline{R}(a_i)$ based on the value of $\overline{S}(a_i, p_j)$ from all each ranked list $L_j$.

**Lower bound of $R(a_i)$.** At each access to the ranked list $L_j$, if $a_i$ is not visited so far, then we have $\underline{S}(a_i, p_j) = 0$, because it is possible that $a_i$ is not an author of $p_j$. If we find $a_i$ from $L_j$, then we have $\underline{S}(a_i, p_j) = S(a_i, p_j)$. The lower bound $\underline{S}(a_i, p_j)$ is increased from 0 to $S(a_i, p_j)$ as the TA-based access executes. We can update the lower bound $\underline{R}(a_i)$ based on the value of $\underline{S}(a_i, p_j)$ from each ranked list $L_j$ (Eq. 7).

**Termination check**. We terminate the TA-based access if the top-$n$ experts are found: (1) we sort all visited experts in descending order of $\underline{R}(a_i)$, (2) we select the $n$-th largest $\underline{R}(a_i)$ as the lower bound of the top-$n$ experts' ranking score, denoted as $LB$, (3) we select the greatest $\overline{R}(a_i)$ among other experts as their upper bound on expert ranking score, denoted as $UB$, and (4) we terminate the TA-based access if $LB \geq UB$.

*Theorem 2:* The TA-based access can terminate safely and obtain the top-$n$ final experts, when $LB \geq UB$ holds.

*Proof:* Since the lower bound $\underline{S}(a_i, p_j)$ increases from 0 to the exact $S(a_i, p_j)$ as the TA-based access processes, the lower bound of the top-$n$ experts' ranking score $LB$ also increases (Eq. 7). Similar to $LB$, $UB$ decreases because the upper bound $\overline{S}(a_i, p_j)$ decreases to the exact $S(a_i, p_j)$ as the TA-based access processes. Hence, if $LB \geq UB$ holds at the $r$-th access, then it will hold for all $r'$-th access ($r' > r$). Therefore, the TA-based access can terminate safely and return the top-$n$ final experts when $LB \geq UB$ holds. □

*Example 5:* As Figure 6 shows, there ranked lists $L_1$,$L_2$ and $L_3$, each list $L_j$ records all candidate experts and their expert scores w.r.t. $p_j$. Suppose we want to find the top-2 experts, then the TA-based access is processed as follows. In the 1st access, we visited three experts $a_4$, $a_7$, and $a_1$ and we update their bounds of the expert ranking score as $\underline{R}(a_4) = 0.83$, $\overline{R}(a_4) = 2.37$, $\underline{R}(a_7) = 0.83$, $\overline{R}(a_7) = 2.37$, $\underline{R}(a_1) = 0.71$, and $\overline{R}(a_4) = 2.37$. Since $\overline{R}(a_1) > \min\{\underline{R}(a_4), \underline{R}(a_7)\}$, we need to keep accessing. After the 2nd access, we update the bounds of expert ranking score and we find that $\max\{\overline{R}(a_4), \overline{R}(a_1)\} > \min\{\underline{R}(a_7), \underline{R}(a_2)\}$, hence we continue accessing. The TA-based access terminates after the 3rd access, because the termination condition is reached, which means that it's impossible to find other experts having a greater ranking score than the smallest $\underline{R}$ of $\{a_2, a_7\}$. We finally return the top-2 experts without scanning and ranking all candidate experts.

## V. Optimization

In §III, we adopt the meta-path $\mathcal{P} = P$-$A$-$P$ (co-authorship) to find $(k, \mathcal{P})$-core communities of papers. The explicit structural relationships of papers are captured in these communities and are used to fine-tune the implicit textual semantics of papers represented by a pre-trained model. Although it is effective, it still has one limitation that should be handled.

**Analysis.** The biggest problem by only using the meta-path $P$-$A$-$P$ is that it ignores the fact that one author could be interested in many different research topics. For instances where one author wrote several papers having different research topics, these papers could be returned in the same $(k, \mathcal{P})$-core community with $\mathcal{P} = P$-$A$-$P$. If they are collected as positive samples, then the representation quality of our document embedding would be affected to some extent.

**Optimization.** In order to solve this problem, we enhance our $(k, \mathcal{P})$-core based document embedding model by considering more valuable meta-paths simultaneously. Different meta-paths represent the different relationships among papers, such as $P$-$T$-$P$ (same-topic relationship) and $P$-$P$ (citation). Both the two meta-paths can be viewed as a complement to the co-authorship $P$-$A$-$P$. Suppose that we have $l$ different meta-paths $\{\mathcal{P}_1, \cdots, \mathcal{P}_l\}$. We consider them in our model simultaneously as follows: (1) As mentioned in Section III-B, we select a set of seed papers randomly and find $l$ different $(k, \mathcal{P})$-core communities for each seed paper w.r.t. $l$ meta-paths, denoted by $\{G^k_{\mathcal{P}_1}, \cdots, G^k_{\mathcal{P}_l}\}$. (2) We next extract a common sub-community of $\{G^k_{\mathcal{P}_1}, \cdots, G^k_{\mathcal{P}_l}\}$ as the final community for each seed paper (Eq. 8), which can be viewed as having a strong structurally cohesiveness. To be more precise, for each paper in this common sub-community $G^k_{\mathcal{P}_{1\cdots l}}$, it has the same author, the same topic with at least $k$ other papers,

and cite (or be cited by) at least $k$ other papers. (3) After getting such communities, we can directly use them in our $(k, \mathcal{P})$-core embedding model (§III-C) and conduct the top-$n$ expert finding based on the PG-Index (§IV).

$$G^k_{\mathcal{P}_{1\cdots l}} = G^k_{\mathcal{P}_1} \cap G^k_{\mathcal{P}_2} \cap \cdots \cap G^k_{\mathcal{P}_l} \quad (8)$$

In §VI, we show the effect of meta-paths on the expert finding effectiveness and efficiency, by testing the three meta-paths ($\mathcal{P}_1$ = $P$-$A$-$P$, $\mathcal{P}_2$ = $P$-$T$-$P$, and $\mathcal{P}_3$ = $P$-$P$) individually and all the possible combination of them, such as $\mathcal{P}_1 + \mathcal{P}_2$, $\mathcal{P}_1 + \mathcal{P}_3$, $\mathcal{P}_2 + \mathcal{P}_3$, and $\mathcal{P}_1 + \mathcal{P}_2 + \mathcal{P}_3$.

## VI. Experimental Study

We present experiments to evaluate (1) effectiveness (§VI-B), (2) efficiency (§VI-C), (3) parameter sensitivity (§VI-D), and (4) overhead of PG-Index (§VI-E). Our code [41] were implemented in Python3.8, and all experiments were conducted on 24 cores of a 3.0GHZ server (running Ubuntu Linux) with 8 Tesla V100s GPUs (32G memory).

### A. Experimental Setup

**Datasets.** We used three real-work academic networks: Aminer [42], DBLP [43], and ACM [44]. Each provides the various relationships among authors, papers, and topics in a heterogeneous graph. All the original networks are provided in [41]. The statistics of our datasets are shown in Table I.

**Queries.** Similar to other text-based expert finding solutions [3], [19], [45], we formed 2680 queries (575, 999, and 1106 for Aminer, DBLP, and ACM, respectively) by randomly selecting papers from our datasets and directly use each paper's textual information $L(p)$ as the query text. Moreover, we adopted the same method as [3], [19], [45] to generate the ground truths: for each query, we selected all the authors having the same topics as the one of query as the ground truths. We refer interested readers to [19] for more information.

**Metrics.** We used two popular information retrieval measures [46], including the *precision at rank* $n$ (P@$n$) and *mean averaged precision* (MAP) as the effectiveness metrics. P@$n$ is the percentage of correct experts in the top-$n$ retrieved experts w.r.t. to the ground truths, which is estimated as P@$n$ = $\#$`correct experts`$/n$. Suppose we have $N$ correct experts for a query $T$, then the average precision AP w.r.t. $T$ is defined as AP = $\sum_{i=1}^{N}(\text{P@}i * I(a_i))/N$, where $I(a_i) = 1$ if $a_i$ is a correct expert, otherwise $I(a_i) = 0$, and MAP is the average AP for all queries. Moreover, we also adopted the *average document similarities* (ADS) of the top-$n$ experts' academic papers to the given query $T$ as our metric to measure how an expert $a_i$ is semantically related to $T$, that is ADS = $\sum_{i=1}^{n}\sum_{\forall p_{ij} \in P(a_i)} sim(p_{ij}, T)/|P(a_i)|/n$. We run each query for 5 times, and used the averaged value of the above metrics to evaluate the effectiveness. Besides, we used the *average response time* to evaluate the efficiency.

**Comparing methods.** We compared with two types of text-based expert finding methods. The first category includes all the methods adopting the representation model that only consider the textual semantics of papers, such as TFIDF [47], Avg.GloVe [48], and SBERT [23]. TFIDF is a bag-of-words model that only considers lexical similarity. Avg.GloVe is average word representations by factorizing the word-word co-occurrence matrix. SBERT is a contextual vector representation method based on the pre-trained model. For the second category, it includes several methods adopting the network embedding models on homogeneous graphs, such as TADW [49], GVNR-t [50], G2G [51] and IDNE [52]. Notice that, TADW and GVNR-t are based on matrix factorization. G2G are deep learning models. IDNE is based on the topic-word attention. Moreover, no matter what representation model or network embedding model is selected, they all need to access and rank all the candidate experts, thus returning the final top-$n$ experts. Different from these methods, we are the first method that consider both the explicit relations and implicit textual semantics of papers over heterogeneous graphs in one model, and we can return the top-$n$ experts without accessing and ranking all the candidates.

TABLE I. Statistics of datasets

| Datasets ↓ | # papers | # experts | # venues | # topics | # relations |
|---|---|---|---|---|---|
| Aminer | 1,125,082 | 996,110 | 15,874 | 7 | 4,945,282 |
| DBLP | 1,347,410 | 968,046 | 7,497 | 13 | 6,222,800 |
| ACM | 1,964,946 | 1,573,114 | 11,688 | 13 | 6,681,328 |

**Parameters.** We implemented our solution based on an open source NLP project `HuggingFace` [53] providing many pre-trained models. The default parameters are: $(k, \mathcal{P})$-core for $k$ = 4 and $\mathcal{P} \in \{P\text{-}A\text{-}P,\ P\text{-}T\text{-}P,\ P\text{-}P\}$ (we use $P$-$A$-$P$ and $P$-$T$-$P$ together as default), sampling ratio $f$ = 0.3 for training data generation, *near negative* strategy for negative samples collection (collect $s$ = 3 negative samples for per positive sample), top-$m$ papers retrieval for $m$ = 1000, and top-$n$ experts finding for $n$ = 20. Moreover, we trained our model by setting the margin of our loss function as $c$ = 1, `epochs` = 4, and the batch size of 64 for gradient accumulation.

### B. Effectiveness Evaluation

Table II shows the effectiveness results for three datasets. Notice that, the methods that only consider the textual semantics of papers in the representation model perform the worst. While the methods that adopting network embedding models on homogeneous graphs are better than the ones that do not consider the network relations. Finally, our solution outperforms all the methods compared (improved by at least 1.5X, 1.6X, and 1.4X for MAP, P@5, and ADS). This is because the explicit relationships of papers on heterogeneous graphs (e.g., co-authorship, citation, and same-topic relationship) are also important besides the implicit textual semantics of papers, and our solution is the only one that consider both aspects in one model, thus we achieve the best effectiveness results.

**Case study.** Some concrete case studies of expert finding on Aminer dataset are shown in Table III. We selected the best method (GVNR-t) besides ours as the competitor method to show the top-5 experts of two queries from *semantic web* and *NLP* research areas. Table III (left) shows the top-5 experts by taking the $L(p)$ of [54] as input query text and Table III (right) shows the one of [55] (bold items are correct experts). Ours has

TABLE II. Effectiveness of expert finding over three datasets

| Datasets → | Dataset 1: Aminer | | | | | Dataset 2: DBLP | | | | | Dataset 3: ACM | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Methods ↓ | MAP | P@5 | P@10 | P@20 | ADS | MAP | P@5 | P@10 | P@20 | ADS | MAP | P@5 | P@10 | P@20 | ADS |
| TADW | 0.445 | 0.544 | 0.386 | 0.264 | 0.486 | 0.288 | 0.528 | 0.460 | 0.365 | 0.616 | 0.266 | 0.488 | 0.365 | 0.267 | 0.614 |
| GVNR-t | 0.458 | 0.548 | 0.448 | 0.310 | 0.548 | 0.300 | 0.532 | 0.471 | 0.373 | 0.652 | 0.292 | 0.537 | 0.428 | 0.319 | 0.652 |
| G2G | 0.431 | 0.512 | 0.242 | 0.256 | 0.474 | 0.267 | 0.478 | 0.438 | 0.352 | 0.579 | 0.292 | 0.413 | 0.331 | 0.256 | 0.502 |
| IDNE | 0.443 | 0.537 | 0.404 | 0.263 | 0.500 | 0.282 | 0.508 | 0.441 | 0.360 | 0.601 | 0.219 | 0.405 | 0.329 | 0.255 | 0.487 |
| TFIDF | 0.375 | 0.431 | 0.329 | 0.234 | 0.345 | 0.210 | 0.448 | 0.334 | 0.232 | 0.372 | 0.218 | 0.464 | 0.358 | 0.266 | 0.415 |
| AvgGlove | 0.269 | 0.306 | 0.271 | 0.213 | 0.203 | 0.135 | 0.306 | 0.259 | 0.202 | 0.219 | 0.138 | 0.345 | 0.272 | 0.212 | 0.279 |
| SBERT | 0.405 | 0.505 | 0.312 | 0.243 | 0.345 | 0.213 | 0.461 | 0.368 | 0.276 | 0.383 | 0.194 | 0.425 | 0.351 | 0.275 | 0.394 |
| Ours (P-A-P ∩ P-T-P) | **0.690** | **0.883** | **0.654** | **0.381** | **0.812** | **0.381** | **0.721** | **0.609** | **0.465** | **0.843** | **0.371** | **0.647** | **0.569** | **0.445** | **0.866** |

TABLE III. Case study for expert finding (*Aminer*)

| semantic_web | | machine_learning | |
|---|---|---|---|
| GVNR-t | Ours | GVNR-t | Ours |
| **Ian Horrocks** | **Ian Horrocks** | **Ivan Bratko** | **Ivan Bratko** |
| **Biplav Srivastava** | **Biplav Srivastava** | **John Shawe-Taylor** | **Thomas G. Dietterich** |
| Daniel S. Weld | **Katia P. Sycara** | Jason Weston | **Peter A. Flach** |
| **Katia P. Sycara** | **Siegfried Handschuh** | Thomas Hofmann | **Jaime G. Carbonell** |
| Boris Motik | **James A. Hendler** | **Thomas G. Dietterich** | **Prasad Tadepalli** |

better precision than GVNR-t, because (1) the papers of these experts show greater semantic similarity to the query text, (2) these experts have co-authored a lot of papers, and (3) the papers of these experts have a lot of mutual citations. Since our solution considers both the textual similarity and relations of papers in one model, we have the better effectiveness results.

**Effect of different meta-paths.** We studied the effect of different meta-paths on the effectiveness. We conducted our solution by configuring one meta-path, two meta-paths, and three meta-paths, respectively. For the case of one meta-path, we have three different choices $P$-$A$-$P$ (A), $P$-$T$-$P$ (T), and $P$-$P$ (C). While for the case of two meta-paths, we have another three choices $P$-$A$-$P$ ∩ $P$-$T$-$P$ (AT), $P$-$P$ ∩ $P$-$T$-$P$ (CT), and $P$-$A$-$P$ ∩ $P$-$P$ (AC). For the case of three meta-paths, we consider $P$-$A$-$P$ ∩ $P$-$P$ ∩ $P$-$T$-$P$ (ACT). Table IV shows that instances considering the $(k, \mathcal{P})$-core in the embedding model performs better than the one without considering $(k, \mathcal{P})$-core (1st row: w/o $(k, \mathcal{P})$-core). This is because the similar papers can be clustered in the same $(k, \mathcal{P})$-core community and we can collect the representative positive samples from this community, thereby improving the embedding quality. Moreover, for the case of one meta-path, we found that $P$-$A$-$P$ is the best one and $P$-$P$ is the worst one. This is because it's normal that one paper cites other less-relevant papers in terms of the textual content. Compared to the citation relationship, the other two relationships can better reflect the similarity of papers. Moreover, we found that when we consider instances involving multiple meta-paths simultaneously in our model, the combination of $P$-$A$-$P$ ∩ $P$-$T$-$P$ is the best one. And another interesting finding emerged: the case of considering three meta-paths simultaneously performs worse than considering two meta-paths. This is because the more meta-paths are considered, the less $(k, \mathcal{P})$-core communities that satisfy the three meta-paths at the same time can be found, resulting in the insufficient training data which affect the embedding quality.

### C. Efficiency Evaluation

Figure 7 shows the efficiency results for three datasets. Since we improve the efficiency of expert finding through a *threshold algorithm (TA)-based method* by using the *PG-Index*, we have four versions of our solution: (1) Ours-1: w/ PG-Index & w/ TA (default version), (2) Ours-2: w/ PG-Index & w/o TA, (3) Ours-3: w/o PG-Index & w/ TA, and (4) Ours-4: w/o PG-Index & w/o TA. As shown in Table 7, Ours-1, Ours-2, and Ours-3 outperform all the other methods, and Ours-1 is the best one among these three versions. This proves that the PG-Index and TA method are useful to improve the efficiency. Moreover, from the results, we found that nearly 80% efficiency improvement is driven by the PG-Index while nearly 20% improvement is caused by TA-based method.

TABLE IV. Effect of meta-paths on effectiveness

| Datasets → | Dataset 1: Aminer | | | Dataset 2: DBLP | | | Dataset 3: ACM | | |
|---|---|---|---|---|---|---|---|---|---|
| Methods ↓ | MAP | P@5 | ADS | MAP | P@5 | ADS | MAP | P@5 | ADS |
| w/o $(k, \mathcal{P})$-core | 0.31 | 0.34 | 0.27 | 0.186 | 0.397 | 0.383 | 0.19 | 0.43 | 0.39 |
| P-A-P (A) | 0.63 | 0.81 | 0.80 | 0.37 | 0.71 | 0.81 | 0.35 | 0.62 | 0.83 |
| P-P (C) | 0.49 | 0.71 | 0.79 | 0.34 | 0.64 | 0.79 | 0.32 | 0.56 | 0.82 |
| P-T-P (T) | 0.61 | 0.78 | 0.80 | 0.37 | 0.67 | 0.81 | 0.33 | 0.59 | 0.79 |
| AT | **0.69** | **0.88** | **0.81** | **0.38** | **0.72** | **0.84** | **0.37** | **0.65** | **0.87** |
| AC | 0.63 | 0.77 | 0.79 | 0.36 | 0.71 | 0.81 | 0.34 | 0.63 | 0.82 |
| CT | 0.59 | 0.76 | 0.80 | 0.37 | 0.68 | 0.81 | 0.35 | 0.60 | 0.83 |
| ACT | 0.55 | 0.69 | 0.78 | 0.36 | 0.65 | 0.80 | 0.34 | 0.58 | 0.82 |

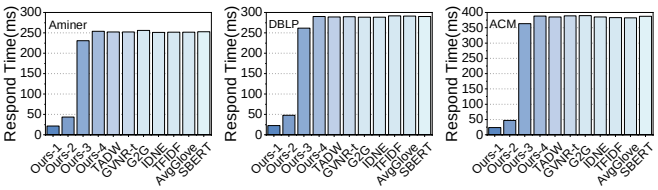



Fig. 7: Efficiency of expert finding over three datasets

### D. Parameter Sensitivity

Figure 8 reports the parameter sensitivity of our approach for (1) the sample ratio $f$, (2) the size $k$ of $(k, \mathcal{P})$-core, (3) the size $m$ of top-$m$ papers, (4) the size $n$ of top-$n$ experts, and (5) the strategies of negatives sampling.

**Effect of sample ratio $f$.** We varied $f$ from 10% to 50% to show its effect on expert finding effectiveness and training efficiency. Note that the larger $f$ is, the larger the amount of training data collected (§III-B). Since larger training data usually offers a high embedding quality, we can see that the MAP and P@$k$ increases as $f$ increases in Figure 8(a). Moreover, it's natural that training time increases as $f$ increases. Finally, we found that a balance is achieved when $f = 30\%$.

**Effect of the size $k$ of $(k, \mathcal{P})$-core.** We varied $k$ from 2 to 9 to see it's effect on expert finding effectiveness and training efficiency (Figure 8(b)). The larger $k$ is, the more cohesiveness of the $(k, \mathcal{P})$-core has, but with fewer papers are included in $(k, \mathcal{P})$-core. We cannot obtain sufficient training data from such a small community, affecting the embedding quality. On the contrary, if we set a smaller $k$, then we can obtain a relatively large community in where the relationships of papers are not close enough. Hence, we could collect more unrepresentative positive samples from such a loose community, but this also would harm the embedding quality. Moreover, it's natural that we require more time to search a $(k, \mathcal{P})$-core with a large $k$ than the one of small $k$. Finally, we found that a balance can be achieved when $k = 4$, $k = 5$, and $k = 5$ for Aminer, DBLP, and ACM datasets, respectively.

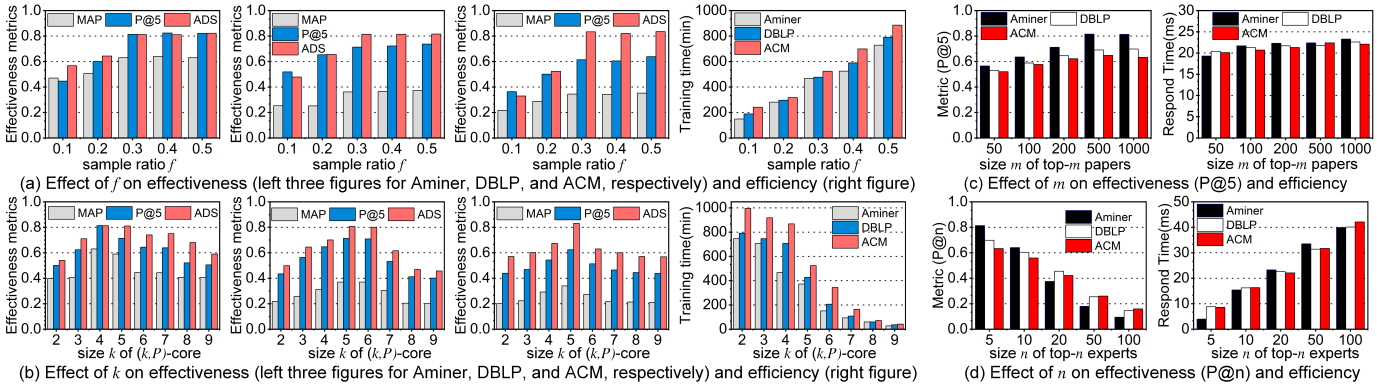



Fig. 8: Parameter sensitivity: effect of the size $k$ of $(k, \mathcal{P})$-core, sample ratio $f$, size $m$ of top-$m$ papers, and size $n$ of top-$n$ experts

**Effect of the size $m$ of top-$m$ papers.** We varied $m$ from 50 to 1000 to see its effect on the expert finding effectiveness and efficiency. Figure 8(c) shows that both results increase as $m$ increases, because we need more time to obtain more papers, and if more papers are taken as the "expertise evidence", then the returned experts are more accurate.

**Effect of the size $n$ of top-$n$ experts.** We varied $n$ from 5 to 100 to see its effect on the expert finding effectiveness and efficiency. In this test, we show P@$n$ for each $n$ as the effectiveness result. In Figure 8(d), the effectiveness result decreases as $n$ increases. This is because most relevant experts are observed at the beginning and less high-quality experts can be returned as $n$ increases. Besides, the efficiency result increases as $n$ increases, because we need more TA access to meet the termination condition and return the top-$n$ experts.

**Effect of negatives sampling strategies.** We implemented two sampling strategies: *random negative* and *near negative* (§III-B). As Table V shows, the near negative performs better than the random negative. This is because the near negative pays more attention to those more valuable papers (not belonging to the community but close to the seed paper). Moreover, for each positive sample we would collect $s$ negative samples, and we found that $s = 3$ is enough to achieve a balance between the effectiveness and training efficiency.

TABLE V. Effect of the negative sampling strategy (*Aminer*)

| Methods → | MAP | P@5 | ADS | Time (h) |
|---|---|---|---|---|
| Random (1:3) | 0.44 | 0.61 | 0.74 | 13.3 |
| Near (1:1) | 0.46 | 0.63 | 0.77 | 4.2 |
| Near (1:2) | 0.51 | 0.67 | 0.79 | 8.7 |
| Near (1:3) | 0.63 | 0.81 | 0.80 | 13.3 |
| Near (1:4) | 0.63 | 0.82 | 0.81 | 17.4 |

TABLE VI. Overhead of PG-Index (*Aminer*)

| Methods → | Mem (G) | Time (s) |
|---|---|---|
| $G$(1.1M, 4.9M) | 3.5 | 543.1 |
| $G_1$(0.8M, 3.8M) | 2.6 | 334.9 |
| $G_2$(0.4M, 2.1M) | 1.3 | 178.5 |
| $G_3$(0.2M, 0.9M) | 0.62 | 43.46 |
| $G_4$(0.1M, 0.6M) | 0.31 | 16.46 |

### E. Overhead of PG-Index

We extracted four sub-graphs $\{G_1 \cdots G_4\}$ with different size from the original dataset $G$ (e.g., $G_1$ contains 0.8M nodes and 3.8M edges), to study the overhead of PG-Index. Table VI shows that the construction time and memory usage of our PG-Index are modest, e.g., within 543.1 s and 3.5 GB.

## VII. Related Work

**Topic-based expert finding.** This kind of expert finding focuses on finding the experts whose expertise can be represented by the input topics. The typical approach to topic-based expert finding is *statistical language model* [2], [12], [17], [19], [20], [56], [57]. The basic idea behinds these algorithms is that the expertise of an expert can be estimated by aggregating textual evidence from relevant documents. To be more precise, a conditional probability model with topics as the condition is proposed to predict the probability of an expert related to these topics, and then return the top-$n$ experts according to such a probability. To further improve the effectiveness, some works focus on integrating the topics' semantic representations (i.e., embeddings) into the statistical language model [21].

**Text-based expert finding.** The problem of topic-based query is that the textual content of the topics is too limited in terms of language to express the user's latent query intention, thus motivating the text-based expert finding. A popular solution to text-based query is to find the most relevant experts by using the document representation model [1], [4], [18], [23]. Besides the implicit semantics of papers' textual contents, the explicit relations of papers are also important to document representation. Therefore, [3], [25] consider the structural relations among papers in a homogeneous paper-paper graph to enhance the effectiveness of expert finding. Specifically, a deep walk is conducted on the homogeneous graph to preserve the features of paper-paper relations in the learned vectors.

## VIII. Conclusion

We studied the academic expert finding on heterogeneous graphs by considering both the explicit relationships and the implicit textual semantics of papers in one model. We first proposed an offline $(k, \mathcal{P})$-core based embedding model based on a pre-trained model. Moreover, by using papers' embeddings, we next presented an online threshold algorithm (TA)-based method to efficiently return top-$n$ experts on the top of a carefully designed proximity graph-based index (PG-Index). All the unpromising papers could be pruned through PG-Index, thereby improving the efficiency. Besides, we can quickly return the top-$n$ experts without scanning and ranking all the expert candidates by using the TA-based method. Moreover, we extended our approach to support multiple relationships simultaneously in our representation model. Experimental results on real-world datasets confirmed the effectiveness and efficiency of our approach.

## References

[1] P. Cifariello, P. Ferragina, and M. Ponza, "Wiser: A semantic approach for expert finding in academia based on entity linking," *Information Systems*, vol. 82, pp. 1–16, 2019.

[2] S. Lin, W. Hong, D. Wang, and T. Li, "A survey on expert finding techniques," *Journal of Intelligent Information Systems*, vol. 49, no. 2, pp. 255–279, 2017.

[3] R. Brochier, A. Gourru, A. Guille, and J. Velcin, "New datasets and a benchmark of document network embedding methods for scientific expert finding," in *European Conference on Information Retrieval*, 2020, pp. 16–29.

[4] M. Berger, J. Zavrel, and P. Groth, "Effective distributed representations for academic expert search," in *EMNLP*, 2020, pp. 56–71.

[5] S. Yuan, Y. Zhang, J. Tang, W. Hall, and J. B. Cabotà, "Expert finding in community question answering: a review," *Artif. Intell. Rev.*, vol. 53, no. 2, pp. 843–874, 2020.

[6] W. Fan, X. Wang, and Y. Wu, "Expfinder: Finding experts by graph pattern matching," in *ICDE*, 2013, pp. 1316–1319.

[7] J. Sun, J. Xu, R. Zhou, K. Zheng, and C. Liu, "Discovering expert drivers from trajectories," in *ICDE*, 2018, pp. 1332–1335.

[8] J. Zhang, J. Tang, and J. Li, "Expert finding in a social network," in *DASFAA*. Springer, 2007, pp. 1066–1069.

[9] R. Roberts, "Understanding the validity of data: a knowledge-based network underlying research expertise in scientific disciplines," *Higher Education*, vol. 72, no. 5, pp. 651–668, 2016.

[10] A. T. P. Silva, "A research analytics framework for expert recommendation in research social networks," Ph.D. dissertation, City University of Hong Kong, 2014.

[11] S. Price and P. A. Flach, "Computational support for academic peer review: A perspective from artificial intelligence," *Communications of the ACM*, vol. 60, no. 3, pp. 70–79, 2017.

[12] H. Deng, I. King, and M. R. Lyu, "Formal models for expert finding on DBLP bibliography data," in *ICDM*, 2008, pp. 163–172.

[13] F. Alarfaj, U. Kruschwitz, D. Hunter, and C. Fox, "Finding the right supervisor: Expert-finding in a university domain," in *NAACL*, 2012, pp. 1–6.

[14] A. D. Nobari, S. S. Gharebagh, and M. Neshati, "Skill translation models in expert finding," in *SIGIR*, 2017, pp. 1057–1060.

[15] K. Balog, Y. Fang, M. De Rijke, P. Serdyukov, and L. Si, "Expertise retrieval," *Foundations and Trends in Information Retrieval*, vol. 6, no. 2–3, pp. 127–256, 2012.

[16] R. Gonçalves and C. F. Dorneles, "Automated Expertise Retrieval: A Taxonomy-based Survey and Open Issues," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–30, 2019.

[17] S. Momtazi and F. Naumann, "Topic modeling for expert finding using latent dirichlet allocation," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 5, pp. 346–353, 2013.

[18] H. Gui, Q. Zhu, L. Liu, A. Zhang, and J. Han, "Expert finding in heterogeneous bibliographic networks with locally-trained embeddings," *arXiv*, vol. abs/1803.03370, 2018.

[19] R. Brochier, A. Guille, B. Rothan, and J. Velcin, "Impact of the query set on the evaluation of expert finding systems," *CoRR*, vol. abs/1806.10813, 2018.

[20] K. Balog, Y. Fang, M. De Rijke, P. Serdyukov, and L. Si, "Expertise retrieval," *Foundations and Trends in Information Retrieval*, vol. 6, no. 2–3, pp. 127–256, 2012.

[21] C. Van Gysel, M. de Rijke, and M. Worring, "Unsupervised, efficient and semantic expertise retrieval," in *WWW*, 2016, pp. 1069–1079.

[22] I. Beltagy, K. Lo, and A. Cohan, "Scibert: A pretrained language model for scientific text," *CoRR*, vol. abs/1903.10676, 2019.

[23] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *EMNLP*, 2019, pp. 3980–3990.

[24] C. Shi, X. Kong, P. S. Yu, S. Xie, and B. Wu, "Relevance search in heterogeneous networks," in *EDBT*, 2012, pp. 180–191.

[25] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. Weld, "SPECTER: Document-level Representation Learning using Citation-informed Transformers," in *ACL*, 2020, pp. 2270–2282.

[26] Y. Fang, Y. Yang, W. Zhang, X. Lin, and X. Cao, "Effective and efficient community search over large heterogeneous information networks," *PVLDB*, vol. 13, no. 6, pp. 854–867, 2020.

[27] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *PVLDB*, vol. 4, no. 11, pp. 992–1003, 2011.

[28] Y. Fang, X. Huang, L. Qin, Y. Zhang, W. Zhang, R. Cheng, and X. Lin, "A survey of community search over big graphs," *VLDBJ*, vol. 29, no. 1, pp. 353–392, 2020.

[29] V. Batagelj and M. Zaversnik, "An o (m) algorithm for cores decomposition of networks," *arXiv*, vol. cs.DS/0310049, 2003.

[30] Y. Yang, Y. Fang, X. Lin, and W. Zhang, "Effective and Efficient Truss Computation over Large Heterogeneous Information Networks," in *ICDE*, 2020, pp. 901–912.

[31] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *CoRR*, vol. abs/1703.07737, 2017.

[32] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[35] M. Wang, X. Xu, Q. Yue, and Y. Wang, "A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search," *CoRR*, vol. abs/2101.12631, 2021.

[36] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *WWW*, 2011, pp. 577–586.

[37] C. Macdonald and I. Ounis, "Voting for candidates: Adapting data fusion techniques for an expert search task," in *CIKM*, 2006, pp. 387–396.

[38] E. A. Corrêa Jr, F. N. Silva, L. d. F. Costa, and D. R. Amancio, "Patterns of authors contribution in scientific manuscripts," *Journal of Informetrics*, vol. 11, no. 2, pp. 498–510, 2017.

[39] L. Weigang, "First and others credit-assignment schema for evaluating the academic contribution of coauthors," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 2, pp. 180–194, 2017.

[40] R. L. Axtell, "Zipf distribution of us firm sizes," *Science*, vol. 293, no. 5536, pp. 1818–1820, 2001.

[41] J. Liu, "Our code and datasets," https://github.com/leleyi/Kcore_Expert_Finding, 2021.

[42] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: Extraction and mining of academic social networks," in *SIGKDD*, 2008, pp. 990–998.

[43] M. Ley, "The dblp computer science bibliography: Evolution, research issues, perspectives," in *International symposium on string processing and information retrieval*. Springer, 2002, pp. 1–10.

[44] C. L. Hennessey, "Acm digital library," *The Charleston Advisor*, vol. 13, no. 4, pp. 34–38, 2012.

[45] S. D. Gollapalli, P. Mitra, and C. L. Giles, "Ranking experts using author-document-topic graphs," in *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, 2013, pp. 87–96.

[46] S. Büttcher, C. L. Clarke, and G. V. Cormack, *Information Retrieval: Implementing and Evaluating Search Engines*. Mit Press, 2016.

[47] W. B. Croft, D. Metzler, and T. Strohman, *Search Engines: Information Retrieval in Practice*, vol. 520.

[48] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014, pp. 1532–1543.

[49] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information." in *IJCAI*, 2015, pp. 2111–2117.

[50] R. Brochier, A. Guille, and J. Velcin, "Global vectors for node representations," in *WWW*, 2019, pp. 2587–2593.

[51] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," *CoRR*, vol. abs/1707.03815, 2017.

[52] R. Brochier, A. Guille, and J. Velcin, "Inductive document network embedding with topic-word attention," in *European Conference on Information Retrieval*, 2020, pp. 326–340.

[53] "HuggingFace," https://github.com/huggingface/transformers, 2021.

[54] L. Li and I. Horrocks, "A software framework for matchmaking based on semantic web technology," *International Journal of Electronic Commerce*, vol. 8, no. 4, pp. 39–60, 2004.

[55] D. Wettschereck, "A study of distance-based machine learning algorithms," 1994.

[56] K. Balog, L. Azzopardi, and M. De Rijke, "Formal models for expert finding in enterprise corpora," in *SIGIR*, 2006, pp. 43–50.

[57] H. Fang and C. Zhai, "Probabilistic models for expert finding," in *European conference on information retrieval*. Springer, 2007, pp. 418–430.