Frontiers in Genetics

TYPE Original Research
PUBLISHED 26 August 2022
DOI 10.3389/fgene.2022.941996

Bioinformatic workflow fragment discovery leveraging the social-aware knowledge graph

Jin Diao[1], Zhangbing Zhou[1,2]*, Xiao Xue[3], Deng Zhao[1] and Shengpeng Chen[4]

[1]School of Information Engineering, China University of Geosciences (Beijing), Beijing, China, [2]Computer Science Department, TELECOM SudParis, Evry, France, [3]School of Computer Software, College of Intelligence and Computing, Tianjin University, Tianjin, China, [4]Wuda Geoinformatics Co., Ltd., Wuhan, China

Check for updates

OPEN ACCESS

EDITED BY
Yucong Duan,
Hainan University, China

REVIEWED BY
Zhuofeng Zhao,
North China University of Technology, China
Buqing Cao,
Hunan University of Science and Technology, China

*CORRESPONDENCE
Zhangbing Zhou,
zbzhou@cugb.edu.cn

SPECIALTY SECTION
This article was submitted to Computational Genomics, a section of the journal Frontiers in Genetics

RECEIVED 12 May 2022
ACCEPTED 29 June 2022
PUBLISHED 26 August 2022

CITATION
Diao J, Zhou Z, Xue X, Zhao D and Chen S (2022), Bioinformatic workflow fragment discovery leveraging the social-aware knowledge graph. *Front. Genet.* 13:941996.
doi: 10.3389/fgene.2022.941996

Constructing a novel bioinformatic workflow by reusing and repurposing fragments crossing workflows is regarded as an error-avoiding and effort-saving strategy. Traditional techniques have been proposed to discover scientific workflow fragments leveraging their profiles and historical usages of their activities (or services). However, social relations of workflows, including relations between services and their developers have not been explored extensively. In fact, current techniques describe invoking relations between services, mostly, and they can hardly reveal implicit relations between services. To address this challenge, we propose a social-aware scientific workflow knowledge graph (S^2KG) to capture common types of entities and various types of relations by analyzing relevant information about bioinformatic workflows and their developers recorded in repositories. Using attributes of entities such as credit and creation time, the union impact of several positive and negative links in S^2KG is identified, to evaluate the feasibility of workflow fragment construction. To facilitate the discovery of single services, a service invoking network is extracted form S^2KG, and service communities are constructed accordingly. A bioinformatic workflow fragment discovery mechanism based on Yen's method is developed to discover appropriate fragments with respect to certain user's requirements. Extensive experiments are conducted, where bioinformatic workflows publicly accessible at the myExperiment repository are adopted. Evaluation results show that our technique performs better than the state-of-the-art techniques in terms of the precision, recall, and *F1*.

KEYWORDS
bioinformatic workflow, fragment discovery, social relations, knowledge graph, scientific workflow

1 Introduction

With the wide-adoption of web service technology, recurring data and computational resources are increasingly encapsulated as web services or mashup APIs and assembled as scientific workflows (Fischer et al., 2021; Coleman et al., 2022). Online repositories, such as *myExperiment*[1], are publicly accessible for publishing and sharing of scientific workflows constructed by scientists from various disciplines (Gkortzis et al., 2021). Bioinformatics, for example, has seen a spectacular rise in the availability of distributed services (Brandt et al., 2021) and allows rapid and accurate analysis using bioinformatic workflows. Examples of bioinformatic workflows from myExperiment are illustrated in Figure 1. With an increasing number of bioinformatic workflows available online, scientists can reuse and repurpose legacy workflows, rather than developing from scratch, to satisfy novel requirements which are examined to be completely or partially satisfiable by legacy workflows in repositories (Brandt et al., 2021; Rosa et al., 2021). As shown in Figure 1B, the workflow "BiomartAndEMBOSSDisease" retrieves all genes on human chromosome 22, which are associated with a disease, and aligns upstream regions with mouse and rat homologues. This workflow can be reused to reduce the cost when a scientist is willing to design a similar experiment. In fact, considering knowledge-intensiveness and error-proneness for constructing a novel bioinformatic workflow, reusing or repurposing current workflows has been evidenced as an error-avoiding and effort-saving strategy for conducting reproducible bioinformatics experiments (Ren and Wang, 2018; Almarimi et al., 2019). To facilitate the reuse and repurposing of bioinformatic workflows, techniques for discovering and recommending the most relevant fragments of current workflows are fundamental (Yao et al., 2021).

Current techniques have been developed to support the discovery of workflow fragments with similarity assessment. Traditionally, these works evaluate structural similarities between workflows (Bai et al., 2017; Zhang et al., 2018; Zhou et al., 2018), where partial-ordering relations specified upon services are concerned. Although the structure can well-represent the execution semantics of individual workflow fragments, semantic mismatches exist, due to domain differences of workflow developers. To mitigate this problem, annotation-based similarity computation techniques are proposed to complement the structural similarity assessment. Annotations are typically provided by developers to prescribe the category and essential functionalities of certain workflows (Ni et al., 2015; Zhong et al., 2016; Hao et al., 2017). Since workflows may not be accompanied with annotations in certain scenarios (Starlinger et al., 2014), annotation-based strategies with inaccurate similarity calculations may not work as expected. As a result, it may hardly recommend suitable fragments when performing certain scientific experiments.

Considering the fact that developers themselves, who prescribe the annotations, may provide insights about the execution relations between workflows, this study proposes to explore social relations between developers to facilitate recommending appropriate workflow fragments. Figure 1 shows a motivating example of two similar bioinformatic workflows, which are built by two developers who are actually friends. Therefore, incorporating the social relations of developers is promising to further improve the recommendation performance. Discovering fragments from bioinformatic workflows that are assembled by developers in social relations is a promising research challenge. While workflow repositories, such as myExperiment, have been constructed for decades, there still have insufficient socially relevant data on developers. As a result, current techniques focus on gathering and applying certain social information, such as developer reputation, to facilitate the discovery accuracy of appropriate workflows and services (Qiao et al., 2019; Khelloufi et al., 2021; Zhu et al., 2021). In fact, more relations between services (Herbold et al., 2021), and their positive or negative links on workflow fragments discovery and recommendation, have not been explored extensively. Therefore, considering social relevance between developers and services, for facilitating the reuse and repurposing of current workflow fragments, is a challenge to be explored further.

To address this challenge, this study proposes a novel workflow fragment discovery mechanism, by exploring social relations of developers and services that are formed in a knowledge graph. Major contributions presented in this article are summarized as follows:

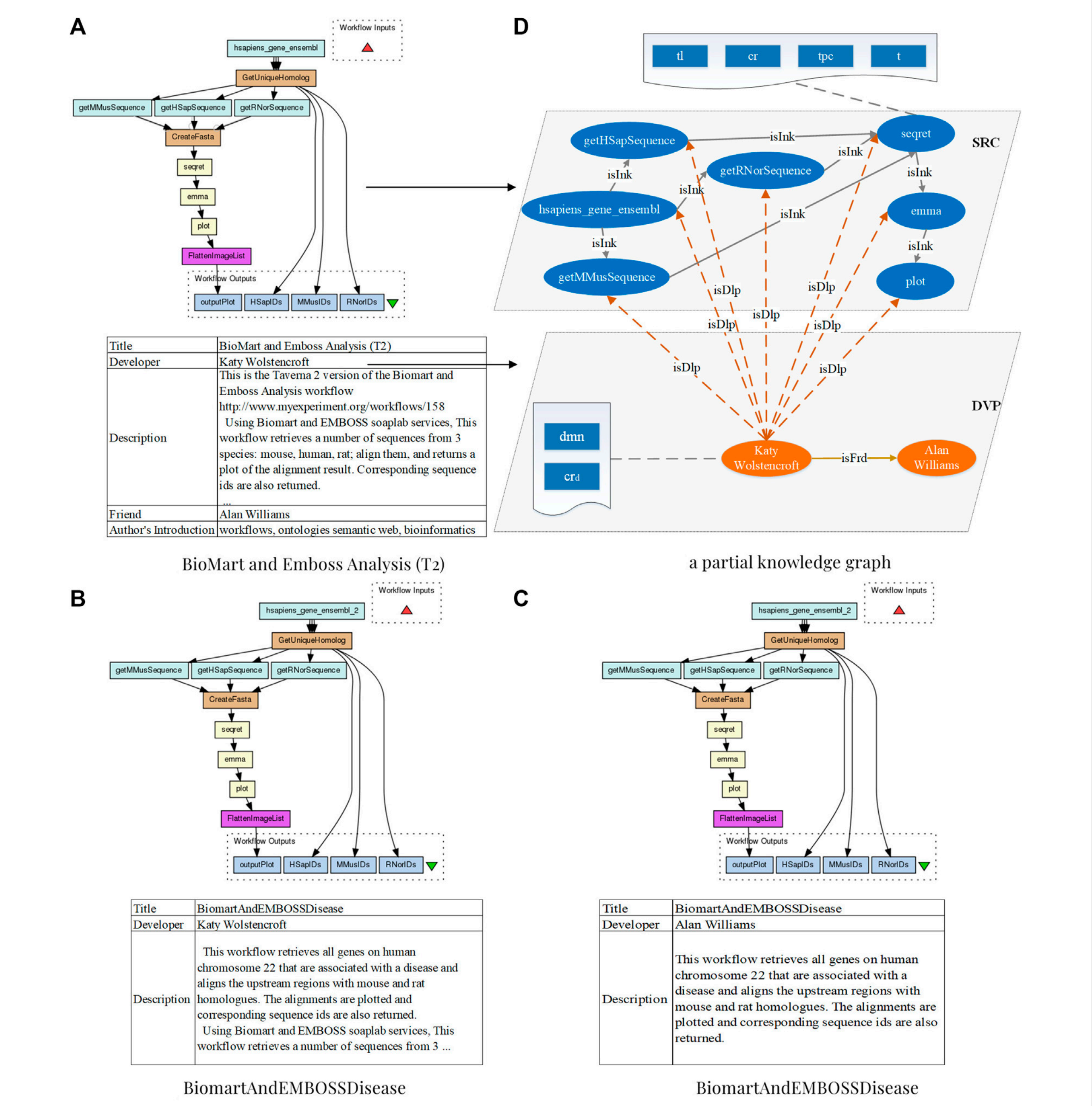
- We constructed a social-aware scientific workflow knowledge graph (S^2KG) from the myExperiment repository, where services and developers of bioinformatic workflows are encapsulated as entities, and relevant attributes of entities, such as topic, reputation, and domain, are obtained. In addition, multiple relations between entities, including (i) invocation relations between services, (ii) developer relations between services and their developers, and (iii) friend relations between developers, are captured.
- We proposed a novel bioinformatic workflow fragment discovery mechanism leveraging S^2KG. Specifically, positive or negative links between services are identified by analyzing their credits, co-invocation possibilities, and co-developer relations (Ni et al., 2015). A service invoking network (SINet) is formed based on invocation relations between services in S^2KG. Service communities are generated from SINet using the fast unfolding method (Blondel et al., 2008), to facilitate individual candidate services discovery from a functional perspective. Thereafter, services are pairwise connected through query operations upon S^2KG. The Yen's method (Yen,

1 https://www.myexperiment.org/workflows

1971) is adopted to construct and recommend appropriate workflow fragments to satisfy user's requirements.

Bioinformatic workflows in myExperiment are adopted as the data set in our experiments, where social relations between services and developers are discovered. Extensive experiments are conducted, and evaluation results show that our technique, which complements social relations, outperforms the state-of-the-art counterparts in terms of the precision, recall, and *F1*.

This study is organized as follows. Section 2 introduces relevant concepts of *S*^{2}*KG* and the attributes of entities. Section 3 presents the process of workflow recommendation based on *S*^{2}*KG*. Section 4 evaluates our method and makes a comparison with state-of-the-art techniques. Section 5 discusses related works. Section 6 concludes this study.



Title	BioMart and Emboss Analysis (T2)
Developer	Katy Wolstencroft
Description	This is the Taverna 2 version of the Biomart and Emboss Analysis workflow http://www.myexperiment.org/workflows/158 Using Biomart and EMBOSS soaplab services, This workflow retrieves a number of sequences from 3 species: mouse, human, rat; align them, and returns a plot of the alignment result. Corresponding sequence ids are also returned. ...
Friend	Alan Williams
Author's Introduction	workflows, ontologies semantic web, bioinformatics

Title	BiomartAndEMBOSSDisease
Developer	Katy Wolstencroft
Description	This workflow retrieves all genes on human chromosome 22 that are associated with a disease and aligns the upstream regions with mouse and rat homologues. The alignments are plotted and corresponding sequence ids are also returned. Using Biomart and EMBOSS soaplab services, This workflow retrieves a number of sequences from 3 ...

Title	BiomartAndEMBOSSDisease
Developer	Alan Williams
Description	This workflow retrieves all genes on human chromosome 22 that are associated with a disease and aligns the upstream regions with mouse and rat homologues. The alignments are plotted and corresponding sequence ids are also returned.

FIGURE 1
Three bioinformatic workflows from *Taverna* 2 of the myExperiment repository with the title "*BioMart and Emboss Analysis (T2)*", "*BiomartAndEMBOSSDisease*", and "*BiomartAndEMBOSSDisease*", respectively, and a partial knowledge graph of the *BioMart and Emboss Analysis (T2)* in *S*^{2}*KG*.

2 S^2KG construction

This section introduces relevant concepts and presents the construction procedure of S^2KG.

2.1 Concepts of S^2KG

myExperiment is an online research environment that supports social sharing of developers' workflows (Goble et al., 2010), which consists of several services. According to these characteristics, the knowledge graph constructed on this repository in this study includes two types of entities, that is, services and developers, as well as three types of relations between them. The workflow is used to reflect the invocation relation between services, so it is not used as a separate entity. The specific definitions are as follows.

A service in S^2KG is defined as follows:

Definition 1 *(Service). A service is a tuple src = (tl, tpc, cr, t), where:*

- *tl is the title of src;*
- *tpc is the topic vector that represents its functions;*
- *cr is its credit, which is calculated based on workflows containing this src;*
- *t represents the created time of src.*

A developer in S^2KG is defined as follows:

Definition 2 *(Developer). A developer is a tuple dvp = (dmn, cr_d), where:*

- *dmn is the topic vector representing his research domains;*
- *cr_d is the reputation calculated by his rating and the credit of his workflows.*

A social-aware scientific workflow knowledge graph (S^2KG) is defined as follows:

Definition 3 *(S^2KG). S^2KG is a tuple (V, LNK), where:*

- *$V = SRC \cup DVP$ is a set of entities for services, SRC, and a set of developers, DVP;*
- *LNK is a set of directed links which specify three kinds of relations: (i) services and services (isInk), (ii) services and developers (isDvp), and (iii) developers and developers (isFrd).*

A scientific workflow in S^2KG is defined as follows:

Definition 4 *(Scientific Workflow). A scientific workflow is a tuple wkf = (cr_w, SRC_w, LNK_w, dsc_w, dvp_w, TG_w), where:*

- *cr_w is the credit calculated upon its download times, viewing times, and rating;*
- *$SRC_w \subset SRC$ is a set of services in wkf;*
- *$LNK_w \subset LNK$ is a set of data links connecting services in SRC_w;*
- *dsc_w is the text description in the profile of wkf;*
- *$dvp_w \subset DVP$ is the developer of wkf;*
- *TG_w is a set of tags provided by dvp_w.*

Figure 1D shows a snippet of S^2KG, which includes several services represented by blue ovals, developers represented by orange ovals, and their relations are represented by arrows with different colors. Specifically, for scientific workflow *BioMart and Emboss Analysis (T2)* in myExperiment, which is a bioinformatic workflow, as shown in Figure 1A, its developer Katy Wolstencroft is represented by orange ovals. Its services are represented by blue ovals; for example, the service *hsapiens_gene_ensembl*. Blue rectangles in wavy rectangles describe the properties of entities, such as the *dmn* and cr_d of Katy Wolstencroft, and the *tl*, *tpc*, *cr*, and *t* of *hsapiens_gene_ensembl*. According to the workflow specification, relations between a developer and his services are extracted as *isDvp* and represented by the orange dotted line; for example, the relation between Katy Wolstencroft and his services *hsapiens_gene_ensembl*. Based on data links in workflows, relations between services are extracted as *isInk* and represented by the gray lines; for example, the relation between the service *hsapiens_gene_ensembl* and the service *getRNorSequence*. Specially, *GetUniqueHomolog* and *CreateFasta* are *beanshells* for cohesion, so they are not regarded as services. Finally, the relation between developers and their friends is extracted as *isFrd* and represented by the yellow arrow in this figure. For example, Katy Wolstencroft, the author of workflows shown in Figures 1A,B, and Alan Williams, the author of the workflow shown in Figure 1C, are friends, and their relation is represented by a yellow arrow and labeled as *isFrd*.

2.2 Topic of services

This section constructs topic vectors of services for representing their functions and domains. For a service, the title and text description in its profile prescribe its original functionality. However, since services are constantly being combined for new application scenarios, their profiles can hardly reflect their new application scenarios and functions. As is often the case, various workflow information sharing platforms provide rich descriptions to describe their domains and functions (Gu et al., 2021). Workflows can be regarded as a set of interdependent services that implement complex functions. Based on this observation, we argue that workflows can be considered as the domain of relevant services to provide their integrated functional description. For a more comprehensive representation of service topics, these functions and domains are used to generate topic vectors for the corresponding services. In total, three sample scientific workflows are shown in Figure 1, and they contain similar services but have different descriptions to represent novel domain of services.

```
Require:
  - $SRC$ : a set of services
  - $WKF$ : a set of workflows
Ensure:
  - $DOC$ : a set of documents
1: for $\forall\ src_i \in SRC$ do
2:   $doc_i \leftarrow src_i.tl \cup src_i.dsc$;
3:   for $\forall\ wkf_j \in WKF$ and $src_i \in wfk_j.SRC_w$ do
4:     for $\forall\ snt_i \in wfk_j.dsc_w$ and $\forall\ wd_{si} \in src_i.tl$ do
5:       if $snt_i.contains(wd_{si})$ then
6:         $doc_i \leftarrow snt_i \cup doc_i$
7:       end if
8:     end for
9:     for $\forall\ wdt_i \in wfk_j.TG_w$ and $\forall\ wd_{si} \in src_i.tl$ do
10:      if $wdt_i.contains(wd_{si})$ then
11:        $doc_i \leftarrow wdt_i \cup doc_i$
12:      end if
13:    end for
14:  end for
15:  $DOC \leftarrow DOC \cup \{doc_i\}$
16: end for
```

Algorithm 1. Service corpus construction

Algorithm 1 presents the construction procedure of service corpus contained in workflows. To prescribe the functionality of each service src_i, its title $src_i.tl$ and text description $src_i.dsc$ are assembled as a document doc_i (line 2). To present the novel domain of src_i, the related description in $wfk_j.dsc_w$ and tags in $wfk_j.TG_w$ of each workflow wkf_i containing src_i are added to doc_i (lines 3–14), where contains () is a comparison function, snt_i is the ith sentence of $wfk_i.dsc_w$, wd_{si} is the ith word of $wrc_i.tl$, $src_i.tl$ is the title of src_i, and wdt_i is the ith tag in $wfk_j.TG_w$. All documents construct the corpus for generating topics for services (line 15). Note that doc_i contains several paragraphs, mostly, and could hardly be regarded as a short text, which usually contains less than five words or no more than 140 characters (Li et al., 2016). Therefore, considering the size of DOC, the Latent Dirichlet Allocation (LDA) model (Blei et al., 2003) is adopted to generate topics for service corpus. Generally, LDA is a bag-of-words model and widely used in general-scale long text classification, where stop words are removed during the model preprocessing phase.

The time complexity of Algorithm 1 is O ($|SRC|*|WKF|*|SRC_w|*|SNT|$), where $|SRC|$ is the number of services in the repository, $|WKF|$ is the number of workflows in the repository, $|SRC_w|$ is the number of services in the jth workflow, and $|SNT|$ is the number of sentences in the description of wkf_j. Note that line 9 should iterate fewer times than line 4, and thus, the time complexity of Algorithm 1 is determined by lines 1, 3, and 4.

```
Require:
  - $DOC$ : a set of documents generated by Algorithm 1
  - $\alpha$ : a doc-topic Dirichlet prior parameter
  - $\beta$ : a topic-word Dirichlet prior parameter
  - $tp_n$ : the number of topics
  - $itt$ : the times of iterations
  - $vb$ : the size of vocabularies
Ensure:
  - $\phi$ : parameters for topic-word distribution
  - $\theta$ : parameters for doc-topic distribution
1: $nkt$, $nktS$, $nmk$, $nmkS$, $z \leftarrow$ iniModel($DOC$)
2: for $itt \in [1, itt]$ do
3:   for $tp \in [1, tp_n]$ and $tr \in [0, vb]$ do
4:     $\phi_{kt} \leftarrow (nkt_{kt} + \beta) \div (nktS_k + vb \times \beta)$
5:   end for
6:   for $m \in [1, DOC.length]$ and $tp \in [1, tp_n]$ do
7:     $\theta_{mk} \leftarrow (nmk_{mk} + \alpha) \div (nmkS_m + tp_n \times \alpha)$
8:   end for
9:   for $l \in [1, DOC.length]$ and $p \in [1, doc_l.length]$ do
10:    $z \leftarrow$ smplTpcZ($l$, $p$)
11:  end for
12: end for
```

Algorithm 2. Service topic model construction

Leveraging DOC generated by Algorithm 1, Algorithm 2 introduces the service topic vector construction procedure. Specifically, the model is initialized and parameters are generated leveraging the set of documents DOC, where nkt is the count of a term for a certain topic, nmk is the count of a topic for a certain document, $nktS$ is the sum for the kth row in nkt, $nmkS$ is the sum for the mth row in nmk, and z is the generated topic label array (line 1). During each iteration itt, we continuously updated the parameters for topic–word distribution ϕ (lines 3–5), as well as the parameters for doc–topic distribution θ (lines 6–8), where tp_n is the number of topic; vb is the vocabulary; $DOC.length$ is the size of DOC; $doc_l.length$ is the size of document doc_l; and tp, tr, and m are local variables. The Gibbs sampling *smpleTpcZ* () is adopted to update topic label array afterward, where l and p are local variables (lines 9–11). Please refer to (Blei et al, 2003) for the specific sampling process. The time complexity of Algorithm 2 is O ($itt*DOC.length*doc_l.length$). Note that lines 3 and 6 should iterate fewer times than line 9, and thus the time complexity of Algorithm 2 is determined by lines 2 and 9.

2.3 Reputation of services

This section constructs the credit of services through the collective perception of workflows containing these services. A service is applied in multiple workflows with some reputation information representing their popularity. As components of a workflow, the credit of every service contributes to an accurate partial-execution of this workflow, which indicates that users prefer to obtain a service with certain quality. To evaluate the quality of services, the method described in Yao et al. (2014) is used to calculate the credit (cr) of services leveraging the workflows information as follows.

Generally, the credit cr_w of a workflow $wkf_i.cr_w$ reflects the degree of adoption by developers, and it is calculated by three factors including viewing times ($wkf_i.n_v$), download times ($wkf_i.n_d$), and rating ($wkf_i.n_{rt}$) by the following formula.

$$wkf_i.cr_w = f_{crd}\left(wkf_i.n_v, wkf_i.n_d, wkf_i.n_{rt}\right) \tag{1}$$

Where f_{crd} is a monotonic increasing function to ensure that the quality of a workflow is directly proportional to its popularity.

For each service in a workflow, its credit can be calculated by adopting a fair-share method, as presented in Nepal et al. (2009). Specifically, due to different importance, the credit of services in a workflow $wkf_i.V$ should be assigned according to its importance as follows.

$$wkf_i.V = v_1, v_2, \ldots, v_n \tag{2}$$

Where $v \in [0, 1]$ and $\sum v = 1$. Based on Eq. 2, the credit of each service $src_j.cr$ is computed using the formula below.

$$\forall src_j \in wkf_i.SRC_w \qquad src_j.cr = v_j wkf_i.cr_w \tag{3}$$

Since a service src_j may be adopted in several workflows, the average credit is regarded as the credit $src_j.cr$.

$$src_j.cr = \frac{\sum_{i=1}^{WN} wkf_i.src_j.cr}{WN} \tag{4}$$

Where WN is the number of workflows containing src_j, and $wkf_i.src_j.cr$ is the credit of src_j calculated by wkf_i.

2.4 Domain and reputation of developers

This section constructs the topic vector of developers for presenting their research domains which influence their services' and workflows' functionality and domains. Through examining the information about developers in myExperiment, a developer generally has four features describing his research domains, including his introduction, interests, tags, and field (or industry). These features are adopted to generate topic vectors of corresponding developers leveraging Algorithm 3.

```
Require:
  - $DVP$ : a set of developers
Ensure:
  - $\phi$ : parameters for topic-word distribution
  - $\theta$ : parameters for doc-topic distribution
1: for $\forall\, dvp_i \in DVP$ do
2:     $doc_i \leftarrow dvp_i.itd \cup dvp_i.itr \cup dvp_i.fld \cup dvp_i.TG_d$
3:     $DOC \leftarrow DOC \cup \{doc_i\}$
4: end for
5: $\phi, \theta \leftarrow DOC$.Algorithm 2
```

Algorithm 3. Developer topic model construction

Algorithm 3 shows the construction of topic vectors for developers. To prescribe the domain of each developer dvp_i (line 1), the introduction $dvp_i.itd$, interests $dvp_i.itr$, field $dvp_i.fld$, and tags $dvp_i.TG_d$ are assembled into a document doc_i (line 2), and these documents construct the corpus for generating topics of developers (line 3). Specifically, $dvp_i.itd$ and $dvp_i.itr$ are texts with several functional paragraphs, and $dvp_i.fld$ and $dvp_i.TG_d$ are some concise words. As mentioned in Section 2.2, doc_i of each developer is not a short text. Thus, the *LDA* model is adopted to generate topic vectors for developers (line 5). Note that the number of iterations in line 1 should be less than that in line 5, so the time complexity of Algorithm 3 is determined by Algorithm 2, where *DOC* is the corpus of developers involved in this algorithm.

The reputation is calculated to reflect the trust degree of a developer. We use the method proposed in Yao et al. (2014) to calculate this value using the developer's rating and his services' credit. Specifically, each developer in myExperiment has an average rating to reflect his contribution. Hence, the rating is considered as an important feature for calculating the reputation. In addition, the credit of his previously developed services is another feature that indicates his reputation. Therefore, the reputation cr_d of a developer is calculated leveraging the follow formula (Yao et al., 2014).

$$cr_d = f_{rp}\left(rt_{d_i}, \{p_{d_i}\}\right) \tag{5}$$

Where the function f_{rp} is a monotonic increasing function, which ensures that the reputation of a developer is high when his credit is high and the quality of his services is high as well. rt_{d_i} is the rating of a developer calculated by the platform. $\{p_{d_i}\}$ is the credit set of his services.

3 Bioinformatic workflow fragment discovery

This section presents the identification of positive and negative links between services to support bioinformatic workflow fragment discovery, involving the selection of candidate atomic services leveraging community detection, and the discovery of their fragments in S^2KG.

3.1 Union impact based on positive and negative links

There exists positive or negative links between pairs of services. Positive links specify correlations, collaborations, and complementary relations between services, whereas on the contrary for negative links. Based on S^2KG, four types of positive links are identified to guide service cooperation (Ni et al., 2015). A service src_i may compose with another src_j, when

1) src_j has a good credit,
2) src_j has a highly similar topic with src_i,
3) the developer of src_j is same as that of src_i, or
4) the developer of src_j is a friend with similar topics to the developer of src_i.

Specifically, the higher the credit of a service is, the higher the possibility that this service is selected to compose a novel workflow. Thus, the positive link Cr_{ij} between src_i and src_j is calculated to reflect the impact of their credit src_i. cr and $src_j.cr$ as follows.

$$Cr_{ij} = src_i.cr \times src_j.cr \tag{6}$$

Where $src_i.cr$ and $src_j cr$ are the credit of src_i and src_j calculated by Eq. 4.

- The second positive link Sim_{ij} is identified to calculate the similarity of src_i and src_j by the following formula Eq. 7 leveraging the services' topic vectors constructed in Section 2.2.

$$Sim_{ij} = \frac{\sum_{k=1}^{tp_n}\left(src_i.tpc_k \times src_j.tpc_k\right)}{\sqrt{\sum_{k=1}^{tp_n}\left(src_i.tpc_k\right)^2} \times \sqrt{\sum_{k=1}^{tp_n}\left(src_j.tpc_k\right)^2}} \tag{7}$$

Where $src_i.tpc_k$ and $src_j.tpc_k$ are the values of the kth feature in $src_i.tpc$ and $src_j.tpc.tp_n$ is the total number of topics. The higher the results are, the more similar the two topic vectors are. A threshold trd_t is prescribed to examine whether two services are similar. Intuitively, when $Sim_{ij} \geq trd_t$, the topic of two services are similar, and not otherwise.

- The third positive link Sd_{ij} is identified by Eq. 8 to examine whether the developers of src_i and src_j are same. Considering the stickiness of a developer's domain, his services should be similar in terms of his topics. These services may be easier to adapt from the perspective of structure, and their composition may match the functional requirements more appropriately.

$$Sd_{ij} = \begin{cases} 1 & \text{if } \exists \left(dvp_i,\ isDvp,\ src_j\right) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Where dvp_i is the developer of src_i, and dvp_i, $isDvp$, and src_j means that the developer of src_j is also dvp_i. As shown in Figure 1A and Figure 1B, these two workflows are constructed by the same developer Katy Wolstencroft. Their structures are similar, but they are adopted in different domains and have different titles and introductions.

- The fourth positive link Sf_{ij} is identified, when two developers are friends, their domains and interests may be similar. Thus, the topic of services they developed should be similar.

$$Sf_{ij} = \begin{cases} 1 & \text{if } \exists \left(dvp_i,\ isFrd,\ dvp_j\right) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Where (dvp_i, $isFrd$, and dvp_j) means that the developer dvp_i of src_i is a friend of the developer dvp_j of src_j. As shown in Figure 1, the developers of Figure 1B and Figure 1C are friends. As we can see, they have constructed the similar workflows with the same title and different functional description.

Negative links indicate functionality uncorrelations, conflicts, or even competitions between services. Based on S^2KG, a union negative link TC_{ij} is identified leveraging Cr_{ij} and Tm_{ij}. Specifically, Tm_{ij} is a negative link specifying that services may cooperate with very low feasibility if they have not cooperated in the same workflow since their creation. Tm_{ij} is calculated as follows.

$$Tm_{ij} = now - \max\left(src_i.t, src_j.t\right) \tag{10}$$

Where now is the current time. The uncooperative duration of two services is determined by the latest service. The larger the value of Tm_{ij} is, the less likely that these two services are cooperated to construct a novel workflow.

Based on Tm_{ij}, TC_{ij} can be formed as follows to present that two services are unlikely to cooperate.

$$TC_{ij} = Tm_{ij} \times Cr_{ij} \tag{11}$$

Generally, the larger the value of TC_{ij} is, the lower the feasibility that src_i and src_j can be cooperated.

As mentioned before, given two services, we adopted the union impact U_{ij} through integrating positive and negative links to determine whether they can be cooperated, as follows.

$$U_{ij} = \alpha \times Sim_{ij} + \beta \times Cr_{ij} - \gamma \times TC_{ij}, \quad if\ Sd_{ij} = 1\ or\ Sf_{ij} = 1 \tag{12}$$

Where α, β, and γ are the importance of each influencing factor, and $\alpha + \beta + \gamma = 1$.

3.2 Service discovery leveraging community detection

Due to the different levels of users' expertise, a requirement in this study is composed of several sub-requirement descriptions in an effort to express the requirement more clearly. Generally, it can be formalized in terms of $Q = \{q_1, q_2, \ldots, q_m\}$. For each sub-requirement, an appropriate service is discovered accordingly. To facilitate single service discovery from the functional perspective, services and $isInk$ relations are extracted from S^2KG and construct a Service Invoking Network ($SINet$). For example, the service *hsapiens_gene_ensembl* and the service *getRNorSequence* in the workflow *BioMart and Emboss Analysis (T2)* are divided into the same purple community because of similar application scenarios. The fast unfolding method (Blondel et al., 2008), which is heuristic based on modularity optimization, is adopted to divide $SINet$ into several functional communities. This method adjusts the division of communities by continuously optimizing the modularity, where the modularity of a partition is a measure of the density of links within the community and the density of links between communities (Newman, 2006) as defined by Eq. 13.

$$CM = \frac{1}{2m} \times \sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2m}\right] \times \delta\left(c_i, c_j\right) \tag{13}$$

$$k_i = \sum_j A_{ij} \tag{14}$$

$$m = \frac{1}{2}\sum_{i,j} A_{ij} \tag{15}$$

$$\delta(c_i, c_j) = \begin{cases} 1 & c_i = c_j \\ 0 & others \end{cases} \tag{16}$$

Where A_{ij} represents the wight of the link between src_i and src_j, and the wight is Sim_{ij} as calculated by Eq. 7. k_i is the sum of wights of links which connect to src_i. m is the sum of link wights in SINet. c_i is the community to which src_i is assigned. $\delta(c_i, c_j)$ represents the fact that whether c_i and c_j are same. By dividing SINet into communities, the entire network is a set *CM* of communities, and each community c_i in *CM* is a tuple $c_i = \{c_t, CS\}$, where c_t is the topic vector of the representative service of c_i and *CS* is a set of services in c_i.

```
Require:
    -$SINet$ : a service invoking network
    -$Q$ : a set of a user's requirements
Ensure:
    -$S$ : a set of candidate services
 1: $CM \leftarrow$ Louvain($SINet$)
 2: $k \leftarrow 0$
 3: for $\forall\ q_i \in Q$ and $\forall\ c_j \in CM$ do
 4:     if $Sim(q_i,\ c_j.c_t) \geq k$ then
 5:         $CT_i \leftarrow CT_i \cup \{c_j\}$
 6:     end if
 7: end for
 8: for $\forall\ c_i \in CT$ and $\forall\ src_j \in c_i.CS$ do
 9:     if Sim($q_i,\ src_j.tpc$) $\geq trd_{sc}$ then
10:         $S \leftarrow S \cup \{src_j\}$
11:     end if
12: end for
```

Algorithm 4. Candidate service discovery

Based on *CM*, the most relevant communities and candidate services are discovered. Algorithm 4 represents candidate communities and services discovery procedure. First, SINet is divided into several communities leveraging the fast unfolding method (Blondel et al., 2008) according to service topics (line 1). A comparison variable *k* is set to 0 (line 2). For each sub-requirement q_i and each community c_j, the functional similarity between them is calculated by the comparison function *Sim*() and compared with *k*, where q_i is vectorized by embedding. The community c_j with the most similar functionality to q_i is inserted into a set of candidate communities *CT* (lines 3–7). For each candidate community in *CT*, the similarity of each $src_j.tpc$ and q_i is calculated and compared with the pre-specified threshold trd_{sc}. If the similarity is larger than trd_{sc}, src_j is inserted into the set *S* as candidate services (lines 8–12). The time complexity of Algorithm 4 is O ($|CT|^*|CS|$), where $|CT|$ is the number of *CT* and $|CS|$ is the number of services in the community c_i. Note that line 3 should iterate fewer times than line 8, and thus, the time complexity of Algorithm 4 is determined by line 8.

TABLE 1 Data set in *Taverna 2*.

Statistics	Value
# of service	2,870
# of workflow	1,058
# of developer	175
# of *isInk*	2,516
# of *isDvp*	2,870
# of *isFrd*	271

3.3 Bioinformatic workflow fragment discovery

Based on candidate services discovered by Algorithm 4, this section proposes to discover appropriate workflow fragments, where relations prescribed by S^2KG are obtained to connect candidate services for respective service stubs in the requirement. The Yen's method (Yen, 1971), which is a heuristic method widely used in graph traversal, is adopted to discover and compose relevant workflow fragments from various workflows.

```
Require:
    -$S^2KG$ : a services social knowledge graph
    -$S$ : a candidate services set
Ensure:
    -$MPS$ : a set of fragments
 1: for $\forall\ src_i,\ src_j \in S$, where $i \neq j$ do
 2:     $PH_0 \leftarrow$ Dijkstra($U_{ij},\ src_i,\ src_j$)
 3:     for $k \in [1,\ k_{\#}]$ do
 4:         for $i \in [0$, size($PH_i$) - 1] do
 5:             $src_{sp} \leftarrow PH_{k-1}.src_i$
 6:             $Ph_{rt} \leftarrow PH_{k-1}.srcs$(0, i)
 7:             for $\forall\ p \in PH$ do
 8:                 if $Ph_{rt} = p.srcs$(0, i) then
 9:                     $S^2KG.LNK \leftarrow S^2KG.LNK$ - $\{(p.edge(\text{i, i+1}))\}$
10:                 end if
11:             end for
12:             $Ph_{sp} \leftarrow$ Dijkstra($U_{spj},\ src_{sp},\ src_j$)
13:             $BPH \leftarrow BPH \cup \{(Ph_{rt} + Ph_{sp})\}$
14:             $S^2KG \leftarrow S^2KG$.rtEdg()
15:         end for
16:         if $BPH$ = NULL then
17:             break
18:         end if
19:         $PH_k \leftarrow BPH$.MAX()
20:     end for
21:     $MPS \leftarrow MPS \cup PH$
22: end for
```

Algorithm 5. CFDY: Crossing-workflow fragment discovery using Yen's method

The Algorithm 5 (denoted *CFDY*) shows the procedure of discovering appropriate bioinformatic workflow fragments. First, the *Dijkstra* () is adopted to find the optimal combinatorial fragment PH_0 from the service src_i to the service src_j leveraging the union impact U_{ij} (lines 1,2). Based on PH_0, the *k*th combinatorial fragment is found (lines 3–19). Above all, every deviated service is traversed (lines 4–15). Specifically, src_{sp} is retrieved from the (k-1)th best combinatorial fragment and Ph_{rt} records the sequence of services from src_i to src_{sp} (lines 4–6). The links that belong to part of the previous best combinatorial fragment of the same Ph_{rt} are removed from the S^2KG (lines 7–11). The combinatorial fragment from src_{sp} to src_j is found by the *Dijkstra* () and recorded to Ph_{sp} (line 12). Entire combinatorial fragment is made up of Ph_{rt} and Ph_{sp} and added to the set *BPH* (line 13). The links that were removed before are added back to S^2KG (line 14). If there are no other combinatorial fragments, the method ends (lines 16–18). The optimal combinatorial fragment in *BPH* is the *k*th combinatorial fragment PH_k (line 19). All paths in *PH* are added to the set *MPS* (line 21). The time complexity of Algorithm 5 is O ($|S|*k_{\#}*size(PH)*|PH|$), where $|S|$ is the number of candidate services, *size(PH)* is the value of *size* (PH_i) minus 1, and $|PH|$ is the number of path.

4 Implementation and evaluation

This section presents our experiments and evaluation results. Experiments are performed on a desktop computer with an Intel i7 6,700 processor at 3.40 and 3.41 GHz, 8.00 GB of RAM and a 64-bit Windows 10 operating system. The prototype is implemented by *Python* and Java.

4.1 Data set and preprocessing

This study adopts bioinformatic workflows in *myExperiment* for our experiments, where workflows in the *Taverna 2* category by May 2019 are crawled. For each service, its title, description, created time and developer are collected. For each workflow, its title, description, tags, publishing date, download times, viewing times, rating, developer, services and data links are collected, where the data links reflect the control flows between services (i.e., invocation relations). For each developer, his name, introduction, interests, field, rating and friends are collected. Note that services and workflows without a title or description are deleted. As a summary, the numbers of available services, workflows, developers and their relations are shown in Table 1.

The data cleaning procedure is conducted, where stop words are removed, and the stemming of words is extracted. Thereafter, entities and relationships are extracted, and their attributes are obtained by the techniques presented in Section 2. We adopt the graph database Neo4j (Robinson et al., 2015) to store these cleaned data.

To evaluate the efficiency of our technique, we have generated 40 crossing-workflow fragments leveraging legacy workflows as testing fragments based on S^2KG. According to the statistic reported in our previous work (Zhou et al., 2020), roughly 86% of workflows contains no more than 11 services. Therefore, 5 out of 40 testing fragments are set to contain over 11 services.

4.2 Measurement metrics

Three metrics are adopted to evaluate the accuracy and effectiveness of our technique as follows:

- *P*: The precision (denoted *P*) indicates the percentage of the number of correctly recommended services over the total number of recommended services.

$$P = \frac{|CS_{pt} \cap CS_{rc}|}{|CS_{rc}|} \tag{17}$$

- *R*: The recall (denoted *R*) refers to the percentage of the number of correctly recommended services over the total number of desired services.

$$R = \frac{|CS_{pt} \cap CS_{rc}|}{|CS_{pt}|} \tag{18}$$

- *F1*: The *F1* score is used for an overall evaluation based on *P* and *R*.

$$F1 = \frac{2 \times P \times R}{P + R} \tag{19}$$

Where CS_{pt} is the expected service set, and CS_{rc} is the set of recommended services.

4.3 Baseline techniques

In this section, the following four state-of-the-art techniques are chosen as baselines to evaluate the effectiveness of our technique:

- *CSBR* (Gu et al., 2021) is a semantics-based model to compose and recommend a set of complementary services for workflow construction. By applying this approach, we first construct a semantic service bundle repository using experimental data. Then a bundle of complementary services is recommended to fulfill the sophisticated requirements. Finally, a more suitable result is found using a greedy approximation method considering the time complexity.
- *ClstRec* (Conforti et al., 2016) is a modularized clustering algorithm to generate service clusters. We first identify target clusters for each service stub, find their services or fragments therein, and sort them into candidate services or fragments. Then a series of fragments are constructed

TABLE 2 *Prp* settings with various tp_n.

tp_{ns}	10	20	30	40	42	43	44	45	46
Prp_s	196.860	149.406	135.332	132.726	132.176	131.505	131.757	132.750	132.752
tp_{nd}	2	4	6	8	9	10	11	12	13
Prp_d	654.289	509.672	468.705	468.485	465.883	454.085	463.771	479.020	479.157

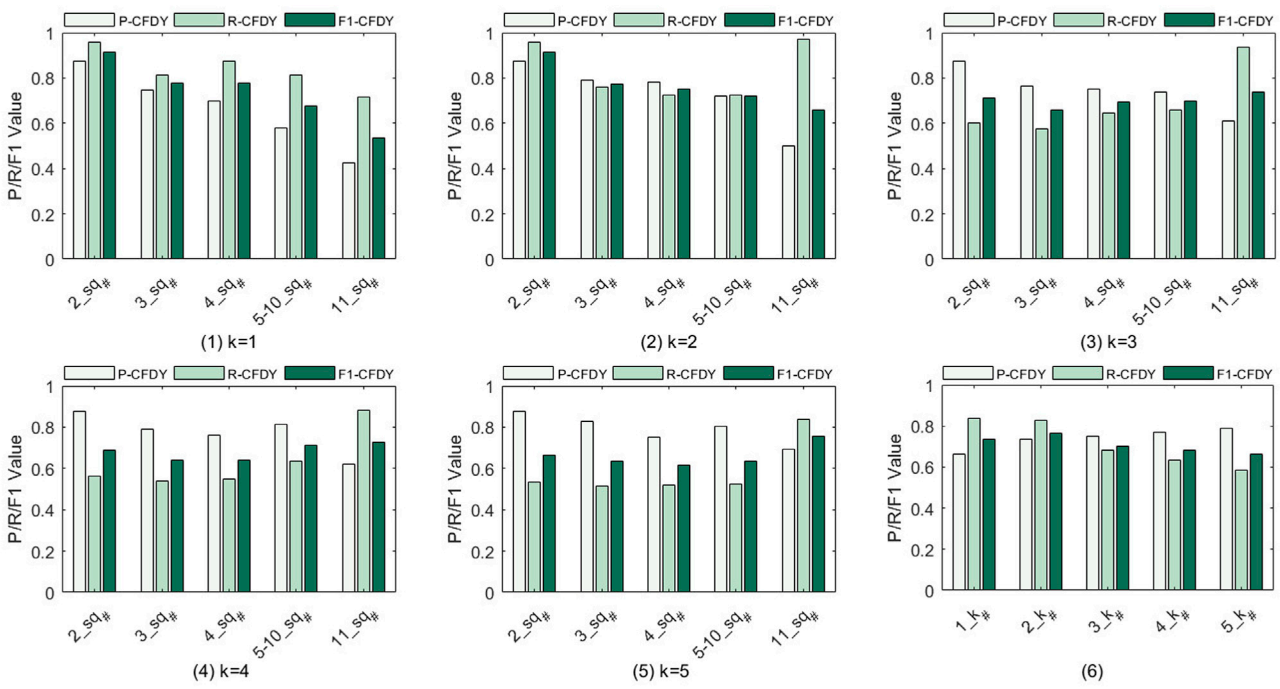


FIGURE 2
Precision, recall, and F1 value for the fragment discovery when $k_{\#}$ is set to 1, 2, 3, 4, and 5, respectively.

across workflows based on their relations. Based on their similarity, these fragments are identified, ranked, and recommended accordingly.

- *CDSR* (Xia et al., 2015) is a category-aware clustering and distributed service recommending method to automatically create fragments. First, we cluster the experimental data into various categories based on the similarity of functionality and popularity of their services. Then we map requirements to relevant categories to find candidate services. Finally, these candidate services in the most relevant categories construct cross-workflow fragments to fulfill the requirements.
- *Short Path* (denoted *SP*) method is a classical heuristic algorithm. First, we start to navigate from a service and select the neighbors with the highest relevance, which have a connection-aware relation with it, according to a given probability distribution. Then a similar operation is performed starting from that service to find a service fragment.

4.4 Evaluation results

In this section, we first optimize the algorithm *CFDY* by adjusting the following parameters tp_n and $k_{\#}$ and then use the parameters $sq_{\#}$ and trd_U to discuss the evaluation results of *CFDY* and baselines.

- tp_n: *The topic number.* The semantic description is susceptible to the topic number. Different number of topics should lead to different partitions of services and developers and recommend various results. Therefore, determining an appropriate tp_n is fundamental and crucial.

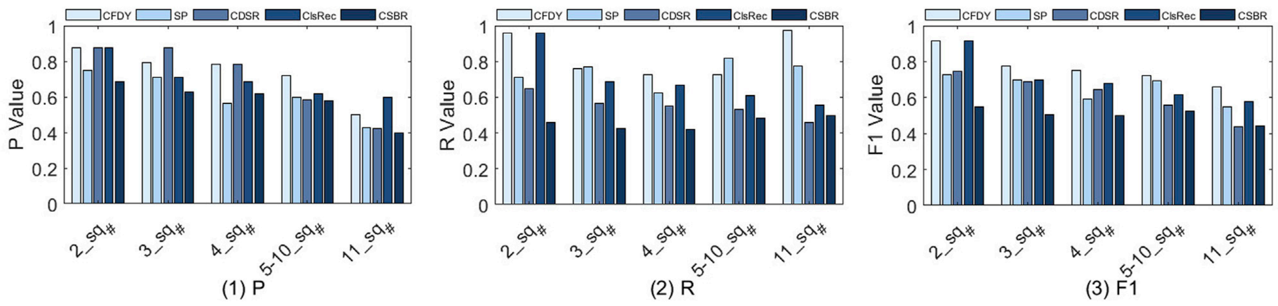


FIGURE 3
P, *R*, and *F1* value for the fragment discovery, when the sub-requirements number is set to 2, 3, 4, 5–10 and 11, respectively.

- $k_\#$: *The number of paths.* When it changes, it should affect the number of path searches of *CFDY*. Different $k_\#$ should affect the number of services in results, thereby affecting the efficiency of *CFDY*.
- $sq_\#$: *The number of sub-requirements.* With the increase of $sq_\#$, the number of service and the complexity of fragments should increase, thereby affecting the efficiency of the fragment discovery.
- trd_U: *The connection-aware threshold of two services.* It should influence the efficiency of our method by changing the scale of candidate services set.

4.4.1 Impact of tp_n

To select the optimal tp_n, a widespread perplexity is used to calculate the quality of the LDA model, as shown below, which describes the degree of uncertainty of the model about documents and their topic. Therefore, the lower the perplexity is, the better predictive effect.

$$Prp = \exp\left\{-\frac{\sum_{d=1}^{M}\log p(w_d)}{\sum_{d=1}^{M} N_d}\right\} \tag{20}$$

Where M is the number of *DOC*, N_d represents the number of words in a *doc*, and $p(w_d)$ is the probability of that the word w_d is contained in the *doc*.

As shown in Table 2, the perplexities of services' and developers' LDA models are calculated separately. For the LDA model of services, with the increasing of topic number (denoted as tp_{ns}), its perplexity (denoted as Prp_s) decreases. When Prp_s is 131.505, tp_{ns} is selected to the optimal value as 43. For the LDA model of developers, its perplexities (denoted as Prp_d) are calculated when its topic number (denoted as tp_{nd}) ranges from 2 to 13. When tp_{nd} is 10, Prp_d is the smallest value as 454.085. Therefore, 43 and 10 are determined as the tp_n of two LDA models.

4.4.2 Impact of $k_\#$

The influence of different $k_\#$ on *P*, *R*, and *F1* is shown in Figure 2 when $k_\#$ is set to 1, 2, 3, 4 and 5, respectively. tp_{ns} is set to 43, tp_{nd} is set to 10, and trd_U is set to 0.8. Considering that different $sq_\#$ will affect the complexity of fragments discovery, the efficiency of various $k_\#$ is evaluated with test examples containing 2–11 sub-requirements. Considering that the number of test examples containing 5–10 sub-requirements is small, these examples are grouped into one category.

- Figure 2(1) and Figure 2(2) show that, as $sq_\#$ gradually increases, *P*, *R* and *F1* as a whole gradually decrease. In particular, in Figure 2(2), when $sq_\#$ is 11, there is a sudden change in *R* that has a higher value. The increase of $sq_\#$ leads to the increase of the number of services contained in CS_{pt}. Therefore, the fragment structure becomes more complex. Generally, when $sq_\#$ is larger, the single path search can hardly fulfill the complexity requirement well. Compared with Figure 2(1), our technique increases the number of paths, and the number of services contained in CS_{rc}. Thus, *R* is increased to some extent.
- As shown in Figure 2(3)–2 (5), when $k_\#$ is larger than 2, their *P*, *R* and *F1* have similar tends. With the gradual increase in $sq_\#$, *P* and *F1* gradually decrease, but *R* has increased to a certain extent. In the same way, the decrease in *P* and *F1* is due to the increase in $sq_\#$ and the complexity of their structure. With the increase of $k_\#$, the number of services in CS_{rc} increases, which will lead to an increase in *R*. However, as $k_\#$ gets larger, too much exploration will lead to a decrease in *R* when $sq_\#$ is small. For example, compared with *R* in Figure 2(1), *R* in Figure 2(5) is significantly lower when $k_\#$ equal to 2.
- Figure 2(6) shows the average values of *P*, *R*, and *F1*, at each $k_\#$. Larger $k_\#$ means that more fragments are constructed. Due to the increase of fragments, more services are selected. Hence, this result may lead to an increase of *P*. However, since the number of expected services is fixed, blindly increasing the number of recommended services by adding too many paths may not maintain the increase in *R*. *F1* has a maximum value at 2. Therefore, 2 is finally selected as the value of $k_\#$ in subsequent experiments.

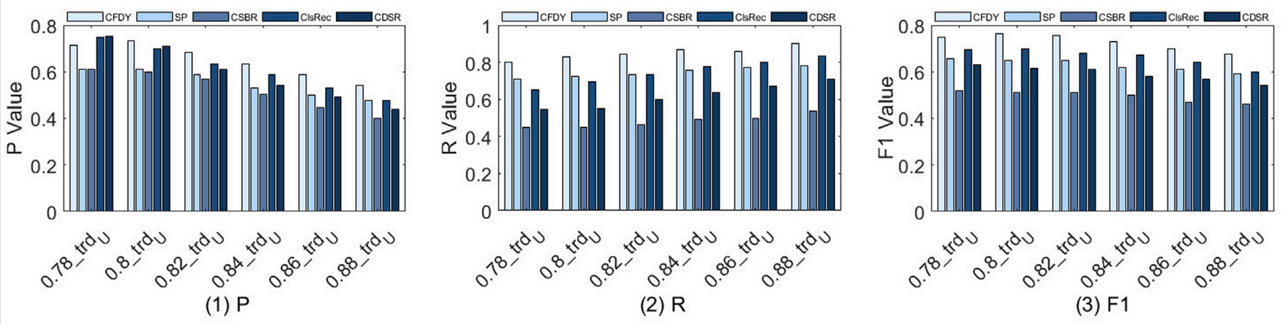


FIGURE 4
P, *R*, and *F1* value for the fragment discovery when the connection-aware threshold is set to 0.78, 0.80, 0.82, 0.84, 0.86, and 0.88, respectively.

4.4.3 Comparison on $sq_{\#}$

We compare five methods and estimate the influence of different $sq_{\#}$ for fragment discovery. The results of *P*, *R*, and *F1* are shown in Figure 3 when $sq_{\#}$ is set to 2, 3, 4, 5–10, and 11, respectively. tp_{ns} is set to 43, tp_{nd} is set to 10, $k_{\#}$ is set to 2, and trd_U is set to 0.8.

Figure 3(1) shows that *P* decreases with the increase of $sq_{\#}$, and the overall performance of *CFDY* is better than other methods.

- For CFDY, as $sq_{\#}$ increases, its p decreases. When $sq_{\#}$ is small, its structure is relatively simple, and the expected service can be selected more accurately by using the relationship between services. With the increase of $sq_{\#}$, the structure of the fragment becomes more complex and contains more branches. Therefore, the selection of candidate services brings certain difficulties, and thus the accuracy rate is reduced.
- Compared with *CFDY*, *SP* only explores a single path and lacks the exploration of branches. Its recommended fragment does not contain some of the expected services present in the branch. As a result, the number of expected services in its recommendation set is less than *CFDY*, which results in its *P* being lower than that of *CFDY*.
- In fact, *CSBR* pursues more semantic similarity matching, and the consideration of structural similarity is not a priority, which leads to the fact that most of the services it recommends are not the expected ones. Therefore, its overall performance is the lowest compared to others.
- *CDSR* uses category awareness to cluster services and considers the impact of service coexistence time on its relationship when considering historical combination information. By considering the functional similarity and relations, when $sq_{\#}$ is less than 5, its *P* is relatively high. But when $sq_{\#}$ is too large and the fragment structure is too complex, its consideration of semantics and structure can hardly fulfill the requirement.
- Similarly, *ClstRec* uses the description of services to cluster them, and selects candidate services from suitable clusters for each sub-requirement. However, this method does not pay too much attention to structural information and can hardly guarantee the rationality of service composition. This causes *P* to be lower when $sq_{\#}$ is large and the fragment structure is more complex.

Figure 3(2) shows *R* of five methods. Overall, the *R* of *CFDY* is higher than that of other methods.

- For *CFDY*, as $sq_{\#}$ increases, its *R* first decreases and then increases. On the whole, its *R* is the highest compared to other methods. When $sq_{\#}$ is small, its structure is relatively simple, and too many exploration branches will add some unexpected services to the recommended fragment. Therefore, its *R* decreases. As $sq_{\#}$ becomes larger, the structure of the fragment becomes more complex and contains more branches. Therefore, further exploration of branches will increase *R* to some extent.
- Similarly, since *CSBR* lacks consideration of structural similarity, most of the services contained in its recommended fragment are unexpected, so its *R* is the lowest.
- Since *SP* has not further explored branches, its *R* is overall lower than that of *CFDY*. When $sq_{\#}$ is at 5–10, its *R* is higher than that of *CFDY*. Because it only explores a single path, in the case of more branches, the number of services in the fragment it recommends is much smaller than *CFDY*, which causes its *R* to be higher.
- For *ClsRec*, when $sq_{\#}$ is smaller, it has a higher *R*. Because it focuses on the similarity of functions, when $sq_{\#}$ is small and the fragment structure is simple, it can relatively accurately find expected services. However, when the fragment becomes complicated, this method can hardly effectively find all expected functions, due to the lack of comparison of structural similarity.

- *CDSR* first finds candidate services according to functional category, then uses historical usages and the coexistence time of services to construct fragments. This method considers the structure of fragments to a certain extent, but can hardly guarantee the necessity of the recommended services. Therefore, the recommended fragments contain a large number of unexpected services, which leads to a lower *R*.

As shown in Figure 3(3), the *F1* of five method decreases as $sq_{\#}$ increases.

- Compared with other methods, *CFDY* has the highest *F1*. Because it considers the structural similarity of fragments while considering semantic similarity. As the $sq_{\#}$ increases, the requirement of a user becomes more and more sophisticated, which leads to more complex selection of candidate services, and more complex fragment discovery and recommendation. Its *F1* is the highest when $sq_{\#}$ is 2, indicating that its recommendation effect is the best when the recommended fragment contains two services and their relationships. This value does not reach 1, because the function descriptions of some services are too similar, resulting in too high similarity of their topic vectors, so that they can hardly be accurately distinguished when selecting candidate services. This is an inevitable problem of *LDA* model.
- For *SP*, due to its low applicability when the fragment structure is complex, its *F1* is lower than that of *CFDY*. The other three methods divide services into different categories, clusters or packages according to their functions, and use semantic similarity to select candidate services. They lack the comparison of structural similarity. In contrast, both *CDSR* and *ClsRec* use historical relations between services to calculate the similarity in fragment structure, while *CSBR* only considers the feasibility of combinations in terms of functional similarity, which leads to the lowest *F1*.

4.4.4 Comparison on trd_U

We estimate the influence of different trd_U for bioinformatic workflow fragment discovery and the results of *P*, *R* and *F1* are shown in Figure 4 when trd_U is set to 0.78, 0.80, 0.82, 0.84, 0.86 and 0.88, respectively. tp_{ns} is set to 43, tp_{nd} is set to 10 and $k_{\#}$ is set to 2. Since the test set contains various samples with different $sq_{\#}$, the final result is the average of all test results.

The results in Figure 4(1) show that *P* of *CFDY* is the highest overall compared to other methods.

- Similarly, when semantic similarity is considered, *CFDY* has more exploration branches compared to *SP*, so the recommended fragment contains more expected services. Specifically, when trd_U is higher, the number of candidate services that can be selected decreases, and the number of expected services that are missing in the recommended fragment increases. This results in *P* getting smaller and smaller as trd_U increases. Especially for *CFDY*, *P* at 0.8 is greater than that at 0.78, which is caused by the uneven distribution of service in S^2KG and the large difference in in-degree and out-degree of them.
- Since *CSBR* doesn't consider the structural similarity much, its *P* is the lowest among all methods. It only relies on the functional similarity between services to discover a crossing-workflow fragment. When trd_U is higher, the number of candidate services for selection decreases, which affects its recommendation effect. Compared with *CSBR*, although *SP* considers the similarity of the fragment structure to a certain extent, it does not further explore branches and its accuracy is only higher than that of *CSBR*. Compared with the above two methods, *ClsRec* and *CDSR* have higher *P*. Generally, they adopt the clustering and classification to compare the functional similarity of fragments, and also apply historical usages to evaluate the structural similarity of fragments. Therefore, they are more effective than the methods that only consider semantic similarity. However, lacking the exploration of social relations, they are not as effective as *CFDY*.

Figure 4(2) shows the comparison of *R*. Similarly, *R* of *CFDY* is the highest, while *R* of *CSBR* is much lower than the other four methods. The difference is that, compared with *P*, as trd_U increases, the *R* of five methods increases.

- As the threshold increases, the candidate services become more similar and these services are more likely to be expected services. Some unexpected services are filtered out and the expected services are more likely to be included in the recommended fragment. For *CFDY*, the variation of trd_U affects the selection of its candidate services. However, compared with other methods, the consideration of social information on the discovery of crossing-workflow fragments can ensure the functional similarity and structural rationality of fragments to a certain extent and a better effect can be obtained.
- For *CSBR*, since it pursues more semantic similarity without considering the fragment structure, and does not consider the structural matching between services, its recommended fragment contains more unexpected services than other methods, which leads to its *R* is the lowest. *CDSR* uses historical usages information to ensure the rationality cross-workflow fragment structure. As a result, the recommended fragment contains a relatively high number of expected services, which results in a higher *R* than that of *CSBR*.
- Similarly, because *SP* and *ClsRec* add the similarity evaluation of the recommended cross-workflow

fragment structure, their *R* are higher than that of *CSBR*. The difference in the recommended effects of *SP*, *ClsRec* and *CDSR* is caused by their different calculation methods of functional similarity. In addition, the fragment complexity recommended by *SP* is lower than other methods and its fragment contains a relatively small number of services, which is part of the reason for its high *R*.

Finally, the comparison results of *F1* of the five methods are shown in Figure 4(3). This figure shows that the *F1* of each method decreases according to the changes of *P* and *R*. In general, *CFDY* has the highest *F1* and *CSBR* has the lowest one.

- For *CFDY*, as the trd_U increases, there are fewer connections that can meet the requirements, which leads to some feasible solutions to be ignored, thereby reducing *P*. At the same time, the reduction of the candidate set can increase the possibility of selecting the expected services, so that *R* increases. However, in combination, the increase in *R* is less than the decrease in *P*, so *F1* decreases. Since the *P* of *CFDY* at 0.8 is greater than that at 0.78, the *F1* of *CFDY* at 0.80 is higher than that at 0.78.
- Due to the lack of comparison of structural similarity in *CSBR*, its *F1* is the lowest. It shows that structural similarity is an important factor in cross-workflow fragment discovery and recommendation. Blindly pursuing functional similarity while ignoring structural similarity cannot achieve better recommendation results. Compared with *CSBR*, the other three methods leverage some structural information, thereby obtaining better *F1*. But compared with *CFDY*, they lack the exploration of the social relations between services, so *F1* is lower.

A higher *F1* of *CFDY* indicates that reasonable social information can improve the effectiveness of cross-workflow fragment discovery and recommendation to some extent. In fact, the representation of the functional domain of a service can be enhanced by using social information. In addition, author information can be used to reveal the hidden relationships between services. Therefore, it has a positive impact on fragment discovery and recommendation.

5 Related works

5.1 Social-aware workflow fragment discovery

Workflow fragment recommendation is an important research problem in the field of service computing (Coleman et al., 2022). It can shorten development cycles and reduce the cost by recommending suitable services and workflow fragments for users (Almarimi et al., 2019) from an open, large-scale library of Web services (Modi and Garg, 2019). In the past, profiles of services and workflows were used as the only guide for users to discover workflow fragments. However, with the development of social network service (SNS), traditional service repositories have become increasingly social, and contain a wealth of social information reflecting the social connections of developers and services (Bastami et al., 2019). This social information can also have an impact on workflow fragment recommendation, whereas existing approaches did not take full advantage of this complex social information currently.

Authors (Gu et al., 2021) propose a service package recommendation model (*CSBR*) based on a semantic service package repository by mining existing workflows. Using the degree of service co-occurrence, the correlation between service and workflow is mined. Specifically, reusable service packages composed of multiple collaborative services are annotated with composite semantics instead of their original semantics. Based on the semantic service pack repository, *CSBR* can recommend service packs that cover the functional requirements of workflow fragments as completely as possible. However, this approach discusses only some social properties and lacks further exploration of social relations, making it difficult to reveal the implicit relations between services.

Xia et al. (2015) used the categories of services to construct workflow fragments. They propose a category-aware distributed service recommendation (CDSR) model based on a distributed machine learning framework. Experiments on real data sets prove that the proposed method not only achieves a significant improvement in accuracy, but also enhances the diversity of recommendation results. However, this method ignores the relations between services and can hardly guarantee the structural similarity of the recommended workflow fragments.

Yao et al. (2014) proposed a ReputationNet to facilitate the workflow fragment discovery. Based on the ReputationNet, the reputations of services and its developers are calculated and represented. According to this, the services and workflows that have better qualities can be recommended to users to satisfy their sophisticated and complicated business requirements. This method utilizes the social attribute reputation, which can improve the efficiency of fragment recommendation to a certain extent. However, many other social information, such as social relations which can promote users to mine latent knowledge, have not been considered.

Zhu et al. (2021) proposed a new model SRaSLR, which is a type of social-aware service label recommendation model. There are invocation and dependency relations between services, and these relations make services naturally constitute a service social network. The authors combine the textual information in service profiles and the social network relations between services. Based on the feature fusion of two perspectives, a model based on deep learning is constructed. Authors conduct a lot of experiments on real-world

Programmable Web data set, and the experimental results show that the use of social relations can improve the performance of recommendation.

Khelloufi et al. (2021) argued that combining users' social characteristics can improve the efficiency of services recommendation and help us provide context-aware services. Therefore, they exploit the social relationships defined in SIoT to build service recommendations among devices, and thus, to enhance service discovery and composition. They propose a SIoT-based service recommendation framework in which devices inherit social relationships from their owners to provide socially aware service recommendations. A boundary-based community detection algorithm is proposed to form a community of socially connected devices.

Kalaï et al. (2018) adopted the social information about the users and the profiles about services to build a social-aware graph for services recommendation. The widespread use of social media provides a large amount of social information for service repositories. Using social information, many user relationships can be extracted for capturing implicit relationships between services. For example, two users, who are friends with each other, may be interested in similar service features. Based on the interests of a user and his friends, personal service recommendations can be provided. However, workflow fragments that can accomplish complex requirements may be preferable to users than recommending a single service that can accomplish simple and specific tasks for them.

Liang et al. (2016) proposed a new framework to effectively discover appropriate services by combining social media information. Specifically, they propose different methods to measure the four social factors collected from Twitter that semantic similarity, popularity, activity and decay factors. Qiao et al. (2019) proposed a recommendation algorithm based on knowledge graph representation learning, which embeds the entities and relations of knowledge graph into a low-dimensional vector space. These methods consider some social attributes in service recommendation, reflecting the importance of social information in recommendation work. However, they mainly recommend a single service to users, and can hardly be used to discover workflow fragments to fulfill the complex requirements prescribed by certain users.

Based on the various types of data in service repositories, underlying logical relations among them can be found to facilitate workflow fragment discovery and recommendation (Wang et al., 2019). Authors propose a fine-grained knowledge graph (DUSKG) to represent the information about users, services and service value feature (VF) and their relations. Based on the DUSKG, the VFs that a service has, the VFs which a user is interested in, and the relations between users and services can be expressed intuitively. Leveraging the DUSKG, five methods are adopted to recommend reasonable single services. However, this method also ignores the importance of workflows which can accomplish complex tasks.

5.2 Semantics-based workflow fragment discovery

Techniques have been developed to recommend workflow fragments from a functional perspective (Hao et al., 2019). Conforti et al. (2016) proposed a technique for automatically discovering hierarchical workflow fragments containing interrupted and non-interrupted boundary services markers. This technique uses approximate functions and contains dependency discovery techniques to extract the process-subprocess hierarchy. Profiles and service invocation relations are used for workflow fragment discovery. However, this method has not yet considered the social information that has an impact on the workflow fragment recommendation, and the information in the repository is not considered comprehensively.

Since the profiles of services are static and the development process is iterative (Huang et al., 2012), Modi and Garg (2019) proposed a method to update the profile of a single service leveraging the description of related workflows. Supplementary information can update the application scenario of a service and optimize its profile. Using this approach, the accuracy of the functional description of a service can be improved and the available services can be recommended to the user. However due to the limitations of functionality, a single service may not accomplish sophisticated and complicated requirements. Wang et al. (2017) proposed a method to extract fine-grained service value features and distributions for personalization service recommendation. By analyzing comments, the most interest aspects of a user can be learned. According to them, the similarity of these features and the descriptions of services are calculated and services with high similarity will be recommended to the user. However, the application scenarios of a single service are limited, since a single service can hardly satisfy the user's requirement as well as implement the user's complex functions completely. This approach lacks to explore the impact of social information and social association on workflow fragment recommendation.

Zhong et al. (2016) and (Hao et al. (2017) extracted the valued information from workflow description to narrow gaps between developers and users. The application scenarios are adopted to supply the description of services to emphasize their functionalities. The *LDA* model is adopted to represent the semantic functions of services. Based on reconstructed descriptions of services, the similarity between services and queries can be improved. However, the similarity is not the only metric that should be considered. Other metrics (Sun et al., 2019), for example, the quality of services (Li et al., 2019), should also be considered in the workflow fragment discovery procedure, so as to guarantee the reliability of the workflow fragments.

Zhou et al. (2018) proposed a method for workflow fragment recommendation which both consider the semantic information of workflows and the hierarchical relations of services. The clustering approach is adopted to cluster the hierarchical structure according to the semantic information, so that the services and workflows with similar functions are in the same group as much as possible. However, this method only considers the invoking relationship between services and does not consider the impact of social connections on workflow fragment recommendation. In fact, these social relations emerge in large numbers in the repositories and also affect the composition of services to a certain extent.

Many services can provide similar functionality, and it is difficult for users to find the service they want (Ren and Wang, 2018). In the workflow fragment recommendation, whether two services can cooperate is an important problem (Lissandrini et al., 2018). The factors that affect service composition usually include two types, positive and negative links. Ni et al. (2015) leveraged tags and both positive and negative links to find service patterns. In addition to positive links of services which facilitate workflows fragment construction, several negative links between services are found, which are strangling service composition. The links between two credible services that have never been cooperated and the links between two services that have been created for a long time but never cooperated together are negative links. Although the consideration of negative links can guide whether two services can be combined, the consideration of positive links is relatively simple. This method explores the influence of social attributes and historical usage on workflow fragment recommendation, but ignores the role of social connections in recommendation work, and does not explore the impact of social relations on workflow segment discovery and recommendation.

5.3 Syntax-based workflow fragment discovery

The syntax-based method focuses on the structure of workflows and the problem of service composition is regarded as a service matching problem. The matching of interface parameters is adopted as the most important metric to promote the composition. Niu et al. (2016) modeled the workflow fragment discovery problem as an uncertain web service composition planning problem. A total of two new uncertain planning algorithms using heuristic search are proposed, called UCLAO* and BHUC, which use the similarity of service interface parameters to solve the U-WSC planning problem of reduced state space, thereby improving the efficiency of finding service portfolio solutions. Empirical experiments are carried out based on running examples in E-commerce applications and large-scale simulation data sets. However, it does not take the level of expertise of different users into account. In fact, there may exist users who do not know the details of the interface, and may not be able to provide input or output parameters. Moreover, the lack of considering service semantics and social associations may not ensure the correctness of the workflow from a functional point of view.

Due to the fast increase of web services over the Internet, Lin et al. (2012) proposed a backward planning method to discover reasonable workflow fragments in a large-scale web service repository based on the lowest cost. The authors exploit the similarity of input and output parameters to construct service groups for facilitating service search. Also, a backward strategy is used to reduce the search space, in order to improve the computational efficiency during workflow construction. However, this approach also neglects the important functional semantics of services and lacks the exploration of the impact about social association among services on their combination.

Liu et al. (2014) proposed a workflow-based framework for workflow fragments discovery. It not only uses the matching degree of the interface parameters to facilitate service composition but also employs a data-centric composition principle that the parameters matching are based on the tag-based semantics. Also, the semantics of service are determined by the folksonomy. The authors first used the related tags to stand for parameters and then constructed workflows based on them. This approach considers the semantic information of the service as well and can better reflect the functionality of the services. Therefore, it can facilitate the combination of services and the recommendation of workflow fragments from a functional perspective. In fact, besides labels, there is other rich semantic information in the repository that can facilitate the construction of workflow fragments. However, these semantic information are not used. Meanwhile, social repositories contain a rich variety of social information and social correlations among items, and these social correlations are not considered in this approach.

6 Conclusion

Considering the knowledge-intensiveness, effort-consuming, and error-proneness when constructing a novel bioinformatic workflow from scratch, discovering and reusing the best practices in legacy workflows is promising when it comes to accomplishing similar tasks. Traditional methods are proposed to discover appropriate workflow fragments depending on their profiles or partial social information in service repositories. However, social relations between developers have not been explored extensively. To capture these relations, this study constructs a knowledge graph S^2KG that includes two types of entities and three types of relations. Based on S^2KG, we propose a bioinformatic workflow fragment discovery mechanism, where we identify positive and negative links for service composition through analyzing their co-invocation possibilities and co-developer relations. A SINet is formed by *isInk* relations in S^2KG to facilitate single service discovery. Finally, the *Yen*'s method is adopted to construct bioinformatic workflow fragments with respect to user's requirements. Experimental results demonstrate that our method performs better than the state-of-the-art techniques with higher accuracy and efficiency.

Data availability statement

Publicly available data sets were analyzed in this study. This data can be found at: https://www.myexperiment.org/workflows.

Author contributions

JD: conceptualization, methodology, software, data curation, and writing—original draft preparation. ZZ: supervision, formal analysis, visualization, and writing—review and editing. XX: resources, data curation, investigation, and writing—review and editing. DZ: methodology, supervision, validation, and writing—review and editing. SC: resources, supervision, validation, and writing—review and editing.

Funding

This work was supported partially by the National Key R&D Program of China (2019YFB2101803) and partially by the National Natural Science Foundation of China (42050103).

Conflict of interest

SC was employed by Wuda Geoinformatics Co., Ltd.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest

The handling editor (initials) declared a past coauthorship with the authors (ZZ, XX).

Publisher's note

References

Almarimi, N., Ouni, A., Bouktif, S., Mkaouer, M. W., Kula, R. G., Saied, M. A., et al. (2019). Web service api recommendation for automated mashup creation using multi-objective evolutionary search. *Appl. Soft Comput.* 85, 105830. doi:10.1016/j.asoc.2019.105830

Bai, B., Fan, Y., Tan, W., and Zhang, J. (20172017). IEEE, 124–131.Sr-lda: Mining effective representations for generating service ecosystem knowledge maps*IEEE Int. Conf. Serv. Comput.*

Bastami, E., Mahabadi, A., and Taghizadeh, E. (2019). A gravitation-based link prediction approach in social networks. *Swarm Evol. Comput.* 44, 176–186. doi:10.1016/j.swevo.2018.03.001

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *J. Mach. Learn. Res.* 3, 993–1022.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *J. Stat. Mech.*, P10008. doi:10.1088/1742-5468/2008/10/p10008

Brandt, C., Krautwurst, S., Spott, R., Lohde, M., Jundzill, M., Marquet, M., et al. (2021). porecov-an easy to use, fast, and robust workflow for sars-cov-2 genome reconstruction via nanopore sequencing. *Front. Genet.* 12, 711437. doi:10.3389/fgene.2021.711437

Coleman, T., Casanova, H., Pottier, L., Kaushik, M., Deelman, E., da Silva, R. F., et al. (2022). Wfcommons: A framework for enabling scientific workflow research and development. *Future Gener. Comput. Syst.* 128, 16–27. doi:10.1016/j.future.2021.09.043

Conforti, R., Dumas, M., García-Bañuelos, L., and La Rosa, M. (2016). Bpmn miner: Automated discovery of bpmn process models with hierarchical structure. *Inf. Syst.* 56, 284–303. doi:10.1016/j.is.2015.07.004

Fischer, M., Hofmann, A., Imgrund, F., Janiesch, C., and Winkelmann, A. (2021). On the composition of the long tail of business processes: Implications from a process mining study. *Inf. Syst.* 97, 101689. doi:10.1016/j.is.2020.101689

Gkortzis, A., Feitosa, D., and Spinellis, D. (2021). Software reuse cuts both ways: An empirical analysis of its relationship with security vulnerabilities. *J. Syst. Softw.* 172, 110653. doi:10.1016/j.jss.2020.110653

Goble, C. A., Bhagat, J., Aleksejevs, S., Cruickshank, D., Michaelides, D., Newman, D., et al. (2010). myexperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic Acids Res.* 38, W677–W682. doi:10.1093/nar/gkq429

Gu, Q., Cao, J., and Liu, Y. (2021). Csbr: A compositional semantics-based service bundle recommendation approach for mashup development. *IEEE Trans. Serv. Comput.*, 1. doi:10.1109/TSC.2021.3085491

Hao, Y., Fan, Y., Tan, W., and Zhang, J. (2017). "Service recommendation based on targeted reconstruction of service descriptions," in *2017 IEEE international conference on web services* (IEEE), 285–292.

Hao, Y., Fan, Y., and Zhang, J. (2019). Service recommendation based on description reconstruction in cloud manufacturing. *Int. J. Comput. Integr. Manuf.* 32, 294–306. doi:10.1080/0951192x.2019.1571242

Herbold, S., Amirfallah, A., Trautsch, F., and Grabowski, J. (2021). A systematic mapping study of developer social network research. *J. Syst. Softw.* 171, 110802. doi:10.1016/j.jss.2020.110802

Huang, K., Fan, Y., and Tan, W. (2012). "An empirical study of programmable web: A network analysis on a service-mashup system," in *2012 IEEE 19th international conference on web services* (IEEE), 552–559.

Kalaï, A., Zayani, C. A., Amous, I., Abdelghani, W., and Sèdes, F. (2018). Social collaborative service recommendation approach based on user's trust and domain-specific expertise. *Future Gener. Comput. Syst.* 80, 355–367. doi:10.1016/j.future.2017.05.036

Khelloufi, A., Ning, H., Dhelim, S., Qiu, T., Ma, J., Huang, R., et al. (2021). A social-relationships-based service recommendation system for siot devices. *IEEE Internet Things J.* 8, 1859–1870. doi:10.1109/jiot.2020.3016659

Li, C., Wang, H., Zhang, Z., Sun, A., and Ma, Z. (2016). Noncoding RNAs in human cancer: One step forward in diagnosis and treatment. *Brief. Funct. Genomics* 15, 165–166. doi:10.1093/bfgp/elw004

Li, S., Huang, J., Cheng, B., Cui, L., and Shi, Y. (2019). Fass: A fairness-aware approach for concurrent service selection with constraints. In IEEE International Conference on Web Services. (IEEE), 255–259.

Liang, T., Chen, L., Wu, J., Xu, G., and Wu, Z. (2016). Sms: A framework for service discovery by incorporating social media information. *IEEE Trans. Serv. Comput.* 12, 384–397. doi:10.1109/tsc.2016.2631521

Lin, S.-Y., Lin, G.-T., Chao, K.-M., and Lo, C.-C. (20122012). A cost-effective planning graph approach for large-scale web service composition. *Math. Problems Eng.*, 21. doi:10.1155/2012/783476

Lissandrini, M., Mottin, D., Palpanas, T., and Velegrakis, Y. (2018809). "Multi-example search in rich information graphs," in *2018 IEEE 34th international conference on data engineering* (IEEE)–820.

Liu, X., Huang, G., Zhao, Q., Mei, H., and Blake, M. B. (2014). imashup: a mashup-based framework for service composition. *Sci. China Inf. Sci.* 57, 1–20. doi:10.1007/s11432-013-4782-0

Modi, K. J., and Garg, S. (2019). A qos-based approach for cloud-service matchmaking, selection and composition using the semantic web. *J. Syst. Inf. Technol.* 21, 63–89. doi:10.1108/jsit-01-2017-0006

Nepal, S., Malik, Z., and Bouguettaya, A. (20092009). Reputation propagation in composite services. *IEEE Int. Conf. Web Serv.*, 295–302.

Newman, M. E. (2006). Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U. S. A.* 103, 8577–8582. doi:10.1073/pnas.0601602103

Ni, Y., Fan, Y., Tan, W., Huang, K., and Bi, J. (2015). Ncsr: Negative-connection-aware service recommendation for large sparse service network. *IEEE Trans. Autom. Sci. Eng.* 13, 579–590. doi:10.1109/tase.2015.2466691

Niu, S., Zou, G., Gan, Y., Zhou, Z., and Zhang, B. (2016). "Uclao* and bhuc: Two novel planning algorithms for uncertain web service composition," in *2016 IEEE international conference on services computing* (IEEE), 531–538.

Qiao, X., Cao, Z., and Zhang, X. (2019). Web service recommendation technology based on knowledge graph representation learning. In Journal of Physics: Conference Series, 1213. Bristol: IOP Publishing, 042015.

Ren, L., and Wang, W. (2018). An svm-based collaborative filtering approach for top-n web services recommendation. *Future Gener. Comput. Syst.* 78, 531–543. doi:10.1016/j.future.2017.07.027

Robinson, I., Webber, J., and Eifrem, E. (2015). *Graph databases: New opportunities for connected data*. Sebastopol: O'Reilly Media, Inc.

Rosa, M. J., Ralha, C. G., Holanda, M., and Araujo, A. P. (2021). Computational resource and cost prediction service for scientific workflows in federated clouds. *Future Gener. Comput. Syst.* 125, 844–858. doi:10.1016/j.future.2021.07.030

Starlinger, J., Brancotte, B., Cohen-Boulakia, S., and Leser, U. (2014). Similarity search for scientific workflows. *Proc. VLDB Endow.* 7, 1143–1154. doi:10.14778/2732977.2732988

Sun, M., Zhou, Z., Wang, J., Du, C., and Gaaloul, W. (2019). Energy-efficient iot service composition for concurrent timed applications. *Future Gener. Comput. Syst.* 100, 1017–1030. doi:10.1016/j.future.2019.05.070

Wang, H., Chi, X., Wang, Z., Xu, X., and Chen, S. (2017). "Extracting fine-grained service value features and distributions for accurate service recommendation," in *2017 IEEE international conference on web services* (IEEE), 277–284.

Wang, H., Wang, Z., Hu, S., Xu, X., Chen, S., Tu, Z., et al. (2019). Duskg: A fine-grained knowledge graph for effective personalized service recommendation. *Future Gener. Comput. Syst.* 100, 600–617. doi:10.1016/j.future.2019.05.045

Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J., Wu, C., et al. (2015). Category-aware api clustering and distributed recommendation for automatic mashup creation. *IEEE Trans. Serv. Comput.* 8, 674–687. doi:10.1109/tsc.2014.2379251

Yao, J., Tan, W., Nepal, S., Chen, S., Zhang, J., De Roure, D., et al. (2014). Reputationnet: Reputation-based service recommendation for e-science. *IEEE Trans. Serv. Comput.* 8, 439–452. doi:10.1109/tsc.2014.2364029

Yao, L., Wang, X., Sheng, Q. Z., Benatallah, B., and Huang, C. (2021). Mashup recommendation by regularizing matrix factorization with api co-invocations. *IEEE Trans. Serv. Comput.* 14, 502–515. doi:10.1109/tsc.2018.2803171

Yen, J. Y. (1971). Finding the k shortest loopless paths in a network. *Manag. Sci.* 17, 712–716. doi:10.1287/mnsc.17.11.712

Zhang, J., Pourreza, M., Lee, S., Nemani, R., and Lee, T. J. (2018). "Unit of work supporting generative scientific workflow recommendation," in *International conference on service-oriented computing* (Springer), 446–462.

Zhong, Y., Fan, Y., Tan, W., and Zhang, J. (2016). Web service recommendation with reconstructed profile from mashup descriptions. *IEEE Trans. Autom. Sci. Eng.* 15, 468–478. doi:10.1109/tase.2016.2624310

Zhou, Z., Cheng, Z., Zhang, L.-J., Gaaloul, W., and Ning, K. (2018). Scientific workflow clustering and recommendation leveraging layer hierarchical analysis. *IEEE Trans. Serv. Comput.* 11, 169–183. doi:10.1109/tsc.2016.2542805

Zhou, Z., Wen, J., Wang, Y., Xue, X., Hung, P. C., Nguyen, L. D., et al. (2020). Topic-based crossing-workflow fragment discovery. *Future Gener. Comput. Syst.* 112, 1141–1155. doi:10.1016/j.future.2020.05.029

Zhu, Y., Liu, M., Tu, Z., Su, T., and Wang, Z. (2021).Sraslr: A novel social relation aware service label recommendation model. In *2021 IEEE international conference on web services*. IEEE, 87–96.