Improving Issue-PR Link Prediction via Knowledge-aware Heterogeneous Graph

Journal:	*Transactions on Software Engineering*
Manuscript ID	TSE-2023-11-0555
Manuscript Type:	Regular
Keywords:	GitHub, Issue Systems, Pull-based Model, Heterogeneous Graph, Metapath Aggregation

Improving Issue-PR Link Prediction via Knowledge-aware Heterogeneous Graph

Shuotong Bai[†‡], Huaxiao Liu[†‡*], Enyan Dai[§] and Lei Liu[†‡]

Abstract—Links between issues and pull requests (PRs) assist GitHub developers in tackling technical challenges, gaining development inspiration, and improving repository maintenance. In realistic repositories, these links are still insufficiently established. Aiming at this situation, existing works focus on issues and PRs themselves and employ text similarity with additional information like issue size to predict issue-PR links, yet their effectiveness is unsatisfactory. The limitation is that issues and PRs are not isolated on GitHub. Rather, they are related to multiple GitHub sources, including repositories and submitters, which, through their diverse relationships, can supply potential and crucial knowledge about technical domains, developmental insights, and cross-repository technical details. To this end, we propose **A**uto **IP** **L**inker (AIPL), which introduces the heterogeneous graph to model multiple GitHub sources with their relationships. Further, it leverages metapath-based technology to reveal and incorporate the potential knowledge for a more comprehensive understanding of issues and PRs. Firstly, we identify 4 types of GitHub sources related to issues and PRs (repositories, users, issues, PRs) as well as their relationships, and model them into task-specified heterogeneous graphs. Next, we analyze information transmitted among issues or PRs to reveal which knowledge is crucial for them. Based on our analysis, we formulate a series of metapaths and employ the metapath-based technology to incorporate various knowledge and learn issue and PR embeddings. Finally, we can infer whether an issue and a PR can be linked based on their embeddings. We evaluate the performance of AIPL on two real-world data sets collected from GitHub. The results demonstrate that it outperforms other baselines and achieves average improvements of 24.97%, 23.99%, 30.31%, and 27.98% in terms of our metrics.

Index Terms—GitHub, Issue Systems, Pull-based Model, Heterogeneous Graph, Metapath Aggregation

1 Introduction

High-quality open-source software relies on effective distributed collaborative development [1], [2], [3]. For enhancing collaboration and ensuring repository sustainability, open-source software platforms, such as GitHub, support volunteers to submit issues and pull requests (PRs) to repositories [4]. More specifically, issues are used for reporting unexpected software behaviors, such as bugs and feature requests [5], [6], and repository contributors can submit PRs containing code changes to solve the challenges of issue reports [7], [8]. Further, sophisticated developers establish links between relevant PRs and issues to provide solutions, gain inspiration, prompt tracking, and expedite software development. By tracing these issue-PR links, code reviewers within the repository can determine if the PR can satisfy the issue's requirements and merge it into the repository during the code review process [9], [10]. Also, newcomers can explore linked PRs to find potential solutions when encountering similar issues [11], [12]. Hence, establishing a link between issues and pull requests is not only necessary but also immensely beneficial for repository maintenance and evolution [13].

Despite the necessity of linking issues and PRs, adequate links are often unavailable for application scenarios due to various reasons. For issue participants, whether newcomers or veterans, finding relevant solutions from the vast number of pull requests in a multitude of repositories, especially for large-scale repositories, is a labor-intensive and tedious task [12], [14]. In addition, PR submitters may also disregard these links. As mentioned in the research [15], over half (54%) of PRs are not linked to an issue when submitted, despite the fact that contribution guidelines of their repositories recommend linking. Moreover, PR submitters are unaware of whether their proposed technical solutions may address other issues, especially those from external repositories [16]. The above factors contribute to the insufficient exploration and establishment of links between issues and PRs on GitHub.

To address the problem of lacking links between issues and PRs, several initial efforts based on open-source software have been dedicated to recommending links. The majority of existing works rely on the textual similarity between issues and PRs for link prediction [17], [18], [19]. Taking the representative method iLinker [17] as an instance, it utilizes a combination of techniques of TF-IDF, word2vec, and doc2vec to generate representations for issues and PRs. It then computes similarity scores between these representations to recommend issue-PR links with higher similarity scores. However, in some realistic scenarios, different developers express the same technical questions in different ways. As a result, issues and PRs with similar content may not necessarily be related, while those with dissimilar content could indeed be linked. For instance, when addressing the challenge of enhancing terminal validation functionality, one issue reporter describes it as `manually triggering validation`, while the linked-PR submitter may describe it as `improving the terminal able directive`.

Hence, solely relying on textual similarity to recommend issue-PR links may be insufficient and less effective. Some recent works [15], [20] incorporate additional information with textual similarity to further benefit the issue-PR link prediction. For example, A-M [15] concatenates textual similarity with features including participant overlap and the size of the issue report.

* *Huaxiao Liu is the corresponding author. {liuhuaxiao@jlu.edu.cn }*
† *College of Computer Science and Technology, Jilin University, Changchun, Jilin, China*
‡ *Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, Jilin, China*
§ *College of Information Sciences and Technology, Pennsylvania State University, the United States*
{baist20}@mails.jlu.edu.cn,{liulei,liuhuaxiao}@jlu.edu.cn,{emd5759}@psu.edu

However, redundant useful information from other sources is generally not incorporated in these works. For example, issues under a repository can share valuable information about specific features, and PRs from the same submitter might hold knowledge of specific development experience. Meanwhile, similar issues can complement each other with additional details, and PRs submitted by collaborators can offer insights from various user perspectives. These interactions among different types of knowledge can enrich the semantics and enhance the depth of understanding of issues and PRs on GitHub. Therefore, it is necessary to explore how to organize such useful and complex knowledge for the task of predicting issue-PR links on GitHub.

Considering various types of sources and relationships related to issues and PRs, a potential solution is utilizing heterogeneous graphs to organize their information. This heterogeneous graph structure can represent graphs coming with multi-types of nodes and edges, shown in Figure 3, and capture complex structures and nuanced semantics for heterogeneous data. Recent works have achieved great results in modeling heterogeneous graphs with graph neural networks for various tasks such as node classification [21], [22] and clustering [23], [24]. However, the investigation of leveraging graph neural networks on heterogeneous graphs for issue-PR link prediction is rather limited.

Therefore, we introduce heterogeneous graph-based methods to tackle our task of predicting issue-PR links and propose a novel approach called AIPL (Auto-Issue PR-Linker). First, we organize various GitHub sources into a task-specified heterogeneous graph based on their context and preliminary relationships among these sources. This task-specified heterogeneous graph can represent all the aforementioned source information while also capturing their correlations. Next, we leverage metapath-based techniques to obtain representations of issues and PRs from the heterogeneous graph. The metapath is a sequence of node and edge types within the specific schema, which can be regarded as paths of specific information transmission on heterogeneous graphs [21]. Compared with traditional graph learning methods [25], [26], where a target node's embedding is aggregated with information from all neighboring nodes, metapath-based techniques can define the scope of crucial neighbors and extract information from them to avoid the information redundancy. The heterogeneous graph learning based on metapaths can provide holistic views and a more comprehensive understanding of issues and PRs. To this end, we observe and analyze realistic issues and PRs to formulate a series of metapaths. By different metapaths, we conduct two aggregation processes with attention mechanisms to learn and fuse information propagated from metapaths and derive representations of target issues and target PRs. Finally, a link classifier can predict the link between them. When given a pair of an issue and a PR, AIPL can suggest whether there can be a link.

To facilitate the training of AIPL, we construct two large-scale graph-based datasets around two large repositories facebook/react and vuejs/vue. For different datasets, we explore 97,556 nodes (37,136 issues and 34,540 PRs) with 196,834 edges (8,457 issue-PR links) and 49,200 nodes (23,166 issues and 10,826 PRs) with 95,160 edges (3,655 issue-PR links), respectively. According to the experimental results, the average performance of AIPL outperforms the other baseline, consisting of the iLinker [17], the A-M [15], random walk algorithm [27], R-GCN [28] and HAN [22], achieving average improvements of 24.97%, 23.99%, 30.31%, and 27.98% in terms of accuracy, precision, recall, and F1-score. Furthermore, we conduct ablation studies for AIPL and

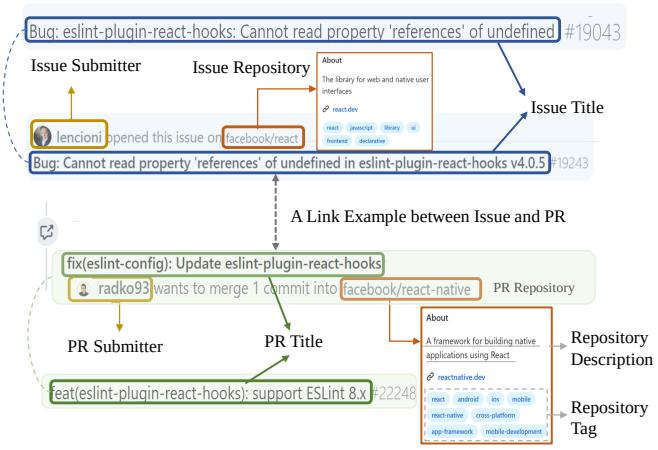


Fig. 1: A real-world example of Issue-PR link on GitHub.

show each piece of graph information and designed metapath play a crucial role in performance. These findings indicate the effectiveness and usefulness of our proposed AIPL. To sum up, we make the following contributions:

- To our best knowledge, this is the first work to introduce heterogeneous graphs to anticipate GitHub issue-PR links. We hope this work can assist developers and the community to improve traceability, fast search solutions for technological challenges, and promote closer collaboration.
- According to realistic examples, we formulate 12 types of metapaths, enabling our approach can effectively learn embeddings of GitHub issues and PRs on a complex graph structure.
- We release our annotated datasets and publicly accessible codes[1], and hope they can help other researchers' future research.

The rest of the paper is organized as follows. Section 2 presents a preliminary analysis of AIPL. In Section 3, we introduce the overall architecture of the AIPL. Section 4 evaluates its performance on real-world data sets. Section 5 summarizes related work and Section 6 presents the threats to the validity of the study. Finally, we conclude the paper in Section 7.

2 Preliminary Analysis

We first use real-world examples to introduce related sources and preliminary relationships about issue-PR link prediction. Then, we formulate our task by a problem definition. Further, we conduct an in-depth analysis of realistic GitHub data to summarize metapath.

2.1 GitHub Sources and Relationships Denotation for Issue-PR Link Prediction

GitHub sources denotation. GitHub is a social-coding platform, where users can submit text-based issues or PRs to a specific repository. Figure 1 shows examples of issue #19243 of repository facebook/react submitted by user *lencioni* and PR #29299 of repository facebook/react-native submitted by user *radko93*. According to this example, we denote 4 types of GitHub sources related to issues and PRs, which are *repositories*, *users*, and the *issues* and *PRs* themselves. As illustrated in Figure 1, information of users merely contains pure login names, and repositories are constituted by descriptions and tags. As for issues and PRs, both of them contain information on titles. The above information

1. https://github.com/baishuotong/AIPL

is utilized for generating initial node embeddings (explained in Section 3.1).

GitHub preliminary relationships denotation. After denoting GitHub sources, we explore preliminary relationships among these sources. According to the submitting history, we identify 5 directed relationships, which are *issues and repositories*, *issues and users*, *PRs and repositories*, *PRs and users*, as well as *users and repositories* based on their interactions. Specifically, corresponding connections will be established if a user submits an issue or a pull request to a repository, or if the user contributes to or forks the repository.

Then, we observe potential relationships among these GitHub sources. First, as suggested in [17], [29], linked issues or linked PRs tend to share similar titles. Taking the examples of Figure 1, two issues both reporting the bug in reading property 'references' are linked together. Meanwhile, PR #29299 is also connected to PR #22248 with close titles. Hence, we infer that relationships between issues (*issues and issues*) and relationships between PRs (*PRs and PRs*) can be established when they share similar titles. In addition, we can also find repositories in the same subarea that share similar descriptions and tags. Take repositories facebook/react and facebook/react-native, for example, these two repositories are related to constructing web and native user interfaces and they have similar descriptions with tags. Then, we add the connections between repositories with similar descriptions and tags (*repositories and repositories*). It is worth mentioning that, in our graph, there is no direct edge linking the issues and PRs, since it is the link to be predicted. The above details will be explained in Section 3.1.

2.2 Problem Formulation

According to the aforementioned sources and relationships, We employ $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \mathbf{X})$ to denote the task-specified heterogeneous graph, where $\mathcal{V}$ and $\mathcal{E}$ denote sets of nodes and edges. Also, $\mathcal{A}$ are the node types *{Issue, PR, Repository, User}* abstracted as $\{I, PR, R, U\}$, while $\mathcal{R}$ represents relationship types *{I-R, I-U, PR-R, PR-U, U-R, I-I, PR-PR, R-R}*, defined by submitting histories on GitHub and text similarity. $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, ...\mathbf{X}_N\}$ is the set of node features extracted from textual information or random representations. With the given definitions and notations, the problem of predicting issue-PR links with heterogeneous graphs can be formulated as follows:

Problem 1. *Given a $\mathcal{G}$ where part of issue-PR node pairs (v_L^I, v_L^{PR}) are provided with labels of linkage, our objective is to learn a function for predicting the existence of a link between issue node v_n^I and pull request node v_n^{PR} in the test set (v_T^I, v_T^{PR}), i.e,*

$$f(v_n^I, v_n^{PR}) \rightarrow y \tag{1}$$

where f is the function to learn and $y \in \{0, 1\}$ is the label of the issue-PR link. $y = 1$ indicates the existence of a link between v_n^I and v_n^{PR}, and $y = 0$, otherwise.

2.3 Formulating Metapath

After formalizing our problem of predicting issue-PR links, our main task becomes how to fit the function f which should be able to learn higher-quality embeddings of issues and PRs (V_I and V_{PR}). To accomplish this purpose, we leverage a metapath-based approach to integrate their textual content and the knowledge extracted from crucial graph-based neighbors. As such, we analyze real data from GitHub and identify 12 types of issue-/PR-oriented metapaths. We briefly introduce the analyzed data. These samples, comprising 300 issues and 300 PRs, are randomly selected from 40 repositories (consistent with our constructed dataset) under two mainstream domains on GitHub, web building and user interfaces. The analysis process is explained as follows.

As for issues, we observe which source information can complement them and summarize the corresponding path information. The first example comes from similar issues, and we define the path of information transmission as `I-I`. Intuitively, similar issues tend to link to a PR, suggesting a need to exchange knowledge. Also, as the first row of Table 1, issues with similar titles can complement the more detailed information, hence, we integrate the information from similar issues into target issue embeddings.

Next, we explore information transmission of issues from the same repositories. In general, issues from a repository share a similar repository background. Although repository descriptions with tags indicate basic information about repositories, some details of repository functions can be reflected in the issue titles. For example, both issues #11442 and issue #8810 discuss vue server renderer, and issue #8810 provides information about the vue renderer that can be applied on the tool web pack 4 (seeing in the second row of Table 1). This detail can help security issue #11442 to find more adapted tool scenarios. Hence, we defined the second type of transmission path as `I-R-I`. Additionally, similar repositories share similar functions, and issues from similar repositories can also complement more detail of functions. Take another instance in the third row of Table 1, issue #1647 and issue #12458 come from similar repositories, facebook/react and facebook/react-native, whose titles are about react prop type. Issue #12458 reflects that the variable of react prop type can influence the effect of layout animation. Hence, we establish the path `I-R-R-I` to connect the issues of similar repositories to supplement more details of repository functions and tool scenarios.

Then, we take the insight into issues from the same submitters. A user *rpominov* has submitted an issue #137 to the repository cujojs/most to discuss the speed of data transmission. After a period of time, he found a bug in data transmission in the repository facebook/react. Then, he is inspired by the prior issue discussion and provides a potential reason accounting for the bug of facebook/react (the example of the fourth row of Table 1). The prior issue provides inspiration and background information for the next, while the later questions complement the next task-specific step for the previous issue. Therefore, we set up the `I-U-I` path to provide task-specific background information as well as contextual information for the corresponding issue. Also, these information associations embody submitters with similar development experiences. For example, shown in the fifth row of Table 1, both user *miloskrca* and user *skegel13* report bugs in updating items on mobile chrome in different repositories. After our analysis, we found they have contributed to the same repository vuejs/vetur, which indicates that users with similar development experiences may encounter similar questions across different repositories. Hence, we define the `I-U-R-U-I` path, to incorporate the issue with information from different users' perspectives.

The final type of issue-oriented path is `I-R-U-R-I`, where issues from different repositories are connected through users who have contributed to both repositories. For example, despite issues #78 and #2549 originating from different repositories with non-collaborating submitters (shown in the sixth row of Table 1), they

TABLE 1: Some examples of exploring high-order issue-oriented relationships.

Relationship	#Issue	Issue Repository	Issue Submitter	Issue Title
I-I	#362	cujojs/most	marvinhagemeister	TypeError: Cannot read property 'run' of undefined.
	#18412	facebook/react-native	PrabhakarVoonik	uncaught TypeError: Cannot read property 'major' of undefined.
I-R-I	#11442	vuejs/vue	alfredriesen	snyk report vue server renderer 2.6.11 serialize javascript 2.1.2 security issue.
	#8810	vuejs/vue	AlexDubok	vue server renderer web pack 4 deprecation warn for plugins.
I-R-R-I	#1647	facebook/react	gasi	react prop type renderable null v undefined.
	#12458	facebook/react-native	janicduplessis	react prop type warn with layout animation.
I-U-I	#137	cujojs/most	rpominov	How comes it so much faster than LoDash?
	#4008	facebook/react	rpominov	React.cloneElement doesn't preserve context hile deprecated React.addons.cloneWithProps does.
I-U-R-U-I	#75	paliari/v-autocomplete	miloskrca	update-items not working on mobile Chrome.
	#9777	vuejs/vue	skegel13	v-model on text input on chrome android not updating.
I-R-U-R-I	#78	Rocket1184/electron -netease-cloud-music	Gabirel	[Bug] Memory leak when switch page & all sort of things.
	#2549	vuejs/vue-router	ruiyong-lee	memory leak by dynamically sets keep-alive include to router-view.

meet the same bug of memory leak when operating in different repositories. This observation indicates that disparate development domains may encounter similar issues, necessitating the consideration of cross-domain knowledge and information exchange. Nevertheless, identifying such cross-domain repositories presents a problem. Our in-depth analysis reveals that these repositories from different domains, such as Rocket1184/electron-netease-cloud-music and vuejs/vue-router, exhibit a shared contributor *rocka*. Leveraging the contributions of users who engage across various domains, we can bridge the gap and identify cross-domain repositories, facilitating the comprehensive integration of issue knowledge from cross-domain perspectives.

Following the establishment of various issue-oriented metapaths, we extend our observations to the PRs and found similar patterns. As a result, we derived 6 distinct types of PR-oriented metapaths, denoted as `PR-PR`, `PR-R-PR`, `PR-R-R-PR`, `PR-U-PR`, `PR-U-R-U-PR` and `PR-R-U-R-PR`. After formulating metapaths, f can aggregate the specific knowledge propagating on each metapath, avoiding the noise from overwhelming information (details in Section 3.3). In the next Section, we elaborate on the details of generating the embeddings of issues and PRs based on the heterogeneous graph and these formulated metapaths.

3 Approach

Based on the above illustrations, we design AIPL, a three-step graph-based approach for predicting links between issues and PRs, as shown in Figure 2. Initially, AIPL constructs a task-specified heterogeneous graph by generating initial embeddings for all source nodes including issues, PRs, repositories, and users, and modeling relationships between nodes as edges. To ensure the dimension of each node embedding is consistent, we employ a node transformation to map all node embedding into the same vector space. The second step of AIPL aims to generate embeddings of issues and PRs via aggregating information based on our heterogeneous graph and designed metapaths. This step contains two components: 1) Intra-metapath Aggregation, which encodes the metapath instances of each metapath to learn information for the target node via aggregating its metapath-based neighbors, its content, and the context in between. 2) Inter-metapath Aggregation, which combines the different knowledge revealed by all metapaths. Finally, AIPL contains an output layer, which employs a transformation with a nonlinear function to fuse the embeddings of an issue-PR pair and predict a link between them.

3.1 Heterogeneous Graph Construction

According to the analysis of Section 2.1, we have identified 4 types of GitHub sources and 8 types of relationships related to our task. Subsequently, we explain how to leverage these sources and relationships for constructing a task-specific heterogeneous graph. Figure 3 illustrates the above process.

3.1.1 Node Embedding

First, we determine the nodes of the heterogeneous graph and learn their initial representations. For each node of different types $\mathcal{A}$ *={Issue(I), PR(PR), Repository(R), User(U)}*, we represent the node embedding based on corresponding information.

Embeddings of Issue and PR. The embeddings for nodes V_I and V_{PR} are generated using the titles of issues and PRs. Note that, we exclude the bodies and comments from consideration due to redundancy information, such as an abundance of code and bug reports [29], [30], [31]. Inspired by the success of pre-trained language models [32], [33], we apply Glove [32] pre-trained on 6 billion words of Wikipedia and Gigword corpus, to encode each title as a feature vector. Then we consider these feature vectors as the node embeddings of V_I and V_{PR}, and we set these node embeddings as 50-dimensional.

Embeddings of Repositotry. To generate embeddings for the Repository nodes V_R, we follow the approach introduced in [34]. We utilize the repository descriptions along with repository tags to create the Repository node embedding. The goal here is to enhance the available information, especially in situations where repository descriptions may be lacking in detail. The procedure for encoding the embeddings of V_R is similar to that used for V_I and V_{PR}. Consequently, the dimensions of the V_R embeddings are set at 50.

Embeddings of Users. As for the representations of GitHub users (nodes V_U), following [35], we adopt a similar approach. Specifically, we assign a 10-dimensional random vector to each node of V_U. Each element within this vector is populated with a randomly selected integer, drawn from a range between 1 and 10.

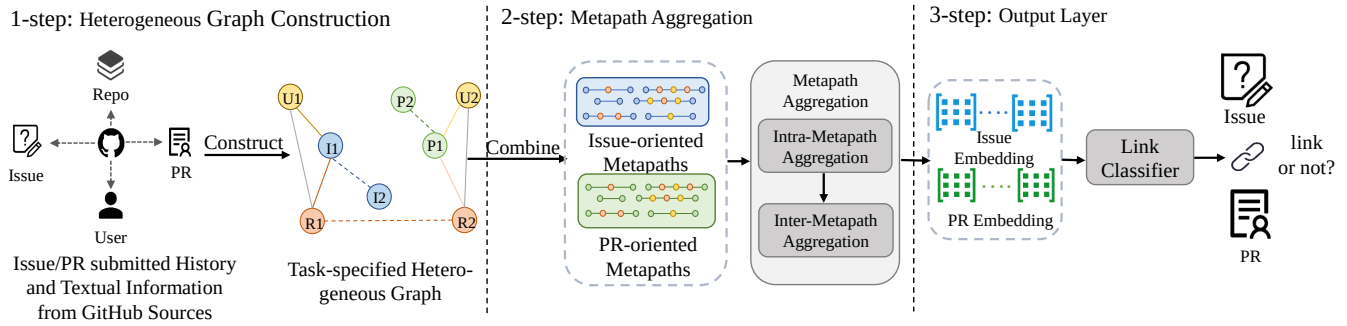


Fig. 2: Overviews of AIPL

While this may appear simplistic, it provides starting information for representing GitHub users within our heterogeneous graph.

3.1.2 Edge Construction

After determining the node embeddings, the next step aims to construct the edges for our task-specific heterogeneous graph. We define 8 types of relationships among nodes, which are *I-R*, *I-U*, *PR-R*, *PR-U*, *U-R*, *R-R*, *I-I*, and *PR-PR*. Note that, we consider our heterogeneous graph as unoriented, in that, the influence between nodes is symmetric, as consistent as [21], [23], [28]. For instance, we regard *U-I* and *I-U* as the same edge. Subsequently, we explain how to model the above edges in detail.

Our method starts by determining edges, which are directly extracted from the GitHub history data, including *I-R*, *I-U*, *PR-R*, *PR-U* and *U-R*. More specifically, when a user u submits an issue i or a PR pr to a repository r, the edges of $u-i$, $r-i$, $u-pr$, and $r-pr$ are established. Also, we construct a link $u-r$ if the user u has engaged in activities of committing, forking, or submitting issues or PRs to the repository r.

Next, as suggested in Section 2.1, relationships between similar repositories, issues, and PRs are necessary to construct. To find similar repositories, we compute the cosine similarity between node embeddings of V_R learned by the method of Section 3.1.1. Once the value of cosine-similarity between two node embeddings of V_R is higher than 0.9 (we set this threshold by experiments in Section 4.5.1), we regard these two repositories as similar repositories and connect them as *R-R*. For estimating similar issues and PRs, we conduct the same steps as above. We consider two issues and two PRs as similar to be linked if the cosine similarity of their node embeddings (also learned by the method of Section 3.1.1) exceeds 0.9 (this threshold is also defined experiments in Section 4.5.1). Further, we link similar issues or similar PRs as edges *I-I* or *PR-PR*. Having constructed all node embedding and edges, we generate a heterogeneous graph encompassing various sources and relationships on GitHub.

3.1.3 Node Content Transformation

For our heterogeneous graph, initial embeddings of different node types have various dimensions. For example, the embedding dimension of V_U is 10, while the embedding dimensions of other node types are defined as 50. This presents a challenge in processing node embeddings of varying dimensions in a unified framework [21]. To address this challenge, we employ a linear transformation by projecting node embeddings into the same latent factor space. For a node $v \in V_A$ of type $A \in \{I, PR, R, U\}$, we apply the following operation.

$$\mathbf{h}'_v = \mathbf{W}_A \cdot \mathbf{x}_v^A \quad (2)$$

where $\mathbf{x}_v^A$ is the original feature vector, and $\mathbf{h}'_v$ is the projected latent vector of node V_A. $\mathbf{W}_A$ is the parametric weight matrix for type A's nodes. After applying this operation, all nodes' projected embeddings share the same dimension defined as 10 referenced [35], which improves the aggregation process of the next stage.

3.2 Metapath Aggregation

After constructing heterogeneous graphs and obtaining the initial embeddings of each node, the next stage of AIPL is to learn the comprehensive embeddings of issues and PRs. To accomplish this aim, we utilize a metapath-based approach to integrating information from crucial neighbors for target nodes (i.e., V_I and V_{PR}), shown in Figure 4. To clarify the metapath-based approach, we provide the following formal definitions related to metapath before explaining our approach in detail.

Definition 1. *(Metapath). A metapath P is defined as a path in the form of $v_1 \xrightarrow{R_1} v_2 \xrightarrow{R_2} \dots \xrightarrow{R_l} v_{l+1}$ (abbreviated as $v_1, v_2 \dots v_{l+1}$). In our work, it describes a composite relation $R = R_1 \circ R_2 \circ \dots \circ R_l$ between nodes v_1 and v_{l+1}. $\circ$ denotes the composition operator on relations.*

Definition 2. *(Metapath Instance) Given a metapath P of a heterogeneous graph, a metapath instance p of P is defined as a node sequence in the graph following the schema defined by P.*

For example, there is a metapath `PR-R-U-R-PR`. One instance of this metapath is PR#113 → repository concord-consortium/codap → user Maaartinus → repository folio-org/stripes-core → PR#103. Now, we delve into the notations based on metapaths and metapath instances.

Definition 3. *(Metapath-based Neighbor) Given a metapath P of a heterogeneous graph, the metapath-based neighbors N_v^P of a node v are defined as the set of nodes that are linked to node v via instances of the metapath P. These neighbors include v itself if the metapath P is symmetric, and a neighbor in two different metapath instances is regarded as two different nodes in N_v^P.*

Continuing from the previous example, For node PR #103, node PR #113 is a metapath-based neighbor of it. The other nodes can be regarded as metapath-based intermediate nodes. Formally,

Definition 4. *(Metapath-based Intermediate Nodes) After determining the metapath-based neighbors N_v^P of a node v, we define the other nodes linked within the metapath P as intermediate nodes $\{m^{P(v,u)}\}$.*

Hence, the set of nodes {repository concord-consortium /codap, user Maaartinus, repository folio-org/stripes-core} is the metapath-based intermediate nodes of node PR #103.

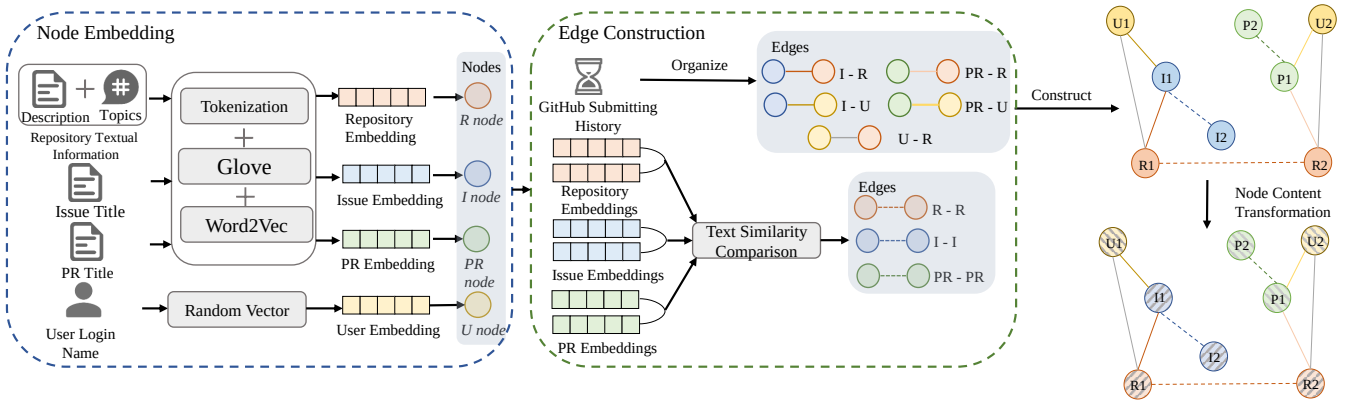


Fig. 3: The process of constructing a heterogeneous graph.

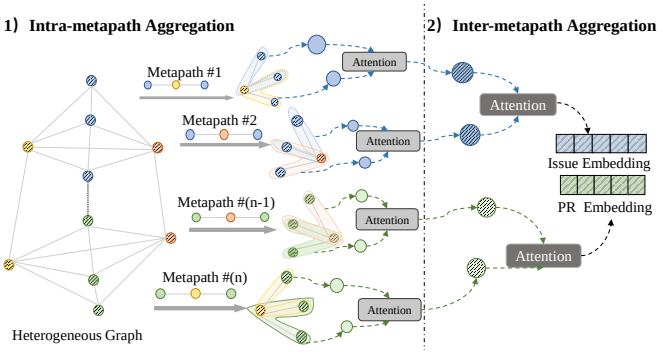


Fig. 4: The process of Metapath Aggregation.

3.2.1 *Intra-metapath Aggregation*

Intra-metapath aggregation in AIPL attempts to integrate the information under each metapath. According to the analysis 12 identified issue/PR-oriented metapaths in Section 2.3, each metapath defines the different scope of graph-based neighbors and represents information transmission in multiple perspectives. As such, AIPL begins by separately consolidating information from different metapaths and provides distinct representations for the target nodes within each specific metapath.

Specifically, given a metapath P, this aggregation process encodes all metapath instances of P to aggregate the information of target nodes, metapath-based neighbors, and the context in between. Before explaining the aggregation process, we denote some formulated terms according to the above definitions. As for each target node $v \in \{V_I, V_{PR}\}$, we define $P(v,u)$ as a metapath instance connecting the target node v and the metapath-based neighbor as $u \in N_v^P$. Further, we define the intermediate nodes of $P(v,u)$ as $\{m^{P(v,u)}\} = P(v,u) - \{u,v\}$.

First, we employ a metapath instance encoder to encode each metapath instance into a vector representation, as $\mathbf{h}_P(v,u)$. The metapath instance encoding can be formulated as:

$$\mathbf{h}_P(v,u) = f_\theta(P(v,u)) = f_\theta(\mathbf{h}'_v, \mathbf{h}'_u, \{\mathbf{h}'_t, \forall t \in \{m^{P(v,u)}\}\}) \tag{3}$$

where f_θ is a metapath instance encoder that summarizes the whole information of a metapath instance into a vector. Specifically, the information of a metapath instance includes all nodes with their relationships. The main idea comes from two aspects. Firstly, we remain the intermediate nodes of metapath instances rather than merely select target nodes with metapath-based neighbors as [22]. Since these intermediate nodes can supplement richer information and more nuanced portrayal of target nodes. For example, for a metapath instance of metapath `I-R-I`, the intermediate node v_n^R can provide repository information for the target node. In addition, we also retain relationship information and distinguish different relationships. It aims to maximize a more refined understanding of each metapath instance. To this end, we employ a metapath instance encoder based on relational rotation(RotatE) [36] to model the knowledge of a metapath instance, which can retain the intermediate node information and relationship information simultaneously. Afterward, we can obtain a vector representation of each metapath instance of different metapaths.

In detail, given $P_(v,u) = (v, t_1, .., t_v, u)$, let R_i be the relation between node t_{i-1} and node t_i, let $\mathbf{r}_i$ be the relation vector of R_i, the metapath instance encoder can be formulated as:

$$\begin{aligned} \mathbf{o}_0 &= \mathbf{h}'_{t0} = \mathbf{h}'_u, \\ \mathbf{o}_i &= \mathbf{h}'_{ti} + \mathbf{o}_{i-1} \odot r_i, \\ \mathbf{h}_{P(v,u)} &= \frac{\mathbf{o}_{num}}{num+1}. \end{aligned} \tag{4}$$

where $\mathbf{h}'_{ti}$ and $\mathbf{r}_i$ are vectors of nodes and relationships, $\odot$ is the element-wise product, and num is the number of the nodes in a metapath instance $P(v,u)$. Then, we can acquire the embedding $\mathbf{h}_{P(v,u)}$ of a metapath instance $P(v,u)$.

After encoding the whole metapath instances of P into vector representations, we apply a graph attention layer to weighted sum these metapath instances related to target node v. The motivation is that different metapath instances contribute to aggregate node embeddings in different degrees. Specifically, the attention matrix parameters for different metapaths are distinct. We model this by learning a normalized importance weight α_{vu}^P for each metapath instance and then weighted summing all instances. The formula is as follows:

$$\begin{aligned} e_{vu}^P &= \text{LeakyReLU}(\mathbf{a}_P^v \top \cdot [\mathbf{h}'_v \| \mathbf{h}_{P(v,u)}]), \\ \alpha_{vu}^P &= \frac{\exp(e_{vu}^P)}{\sum_{s \in N_v^P} \exp(e_{vs}^P)}, \\ \mathbf{h}_v^P &= \sigma\left(\sum_{u \in N_v^P} \alpha_{vu}^P \cdot \mathbf{h}_{P(v,u)}\right). \end{aligned} \tag{5}$$

Here $\mathbf{a}_P^v$ is the attention vector for the metapath instances of P related to node v, and $\|$ denotes the vector concatenation operator. e_{vu}^P indicates the importance of metapath instance $P(v,u)$ to the

node v, which is then normalized across all choices of $u \in N_v^P$ using the softmax function. When the normalized importance weight $\mathbf{a}_{vu}^P$ is obtained for all metapath instances $P(v,u), \forall u \in N_v^P$, where N_v^P is the metapath P-based neighbors of v. Through an activation function $\sigma(\cdot)$, we obtain the final output representation $\mathbf{h}_v^P$.

Meanwhile, the effectiveness of the graph attention mechanism is impacted by the number of attention heads. A suitable number of attention heads can assist in stabilizing the training process and reducing the heterogeneity introduced by our graph. Hence, we adopt K attention mechanisms, as:

$$\mathbf{h}_v^P = \|_{k=1}^K \sigma\left(\sum_{u \in N_v^P} [\alpha_{vu}^P]_k \cdot \mathbf{h}_{P(v,u)}\right). \tag{6}$$

K is the number of the heads of attention mechanisms (we set K as 8 based on our evaluation results), where $\|$ means a concentrate operator to combine outputs $[\alpha_{vu}^P]_k$. $[\alpha_{vu}^P]_k$ is the normalized importance of metapath instance $P(v,u)$ to node v at the k-th attention head.

Overall, given the set of metapaths $P_A = \{P_1^A, P_2^A, ...P_m^A\}$ which start or end with the node whose type is $A \in \{I, PR\}$, the intra-metapath aggregation generates P_m^A metapath-specific vector representations of the target node v of type A. For the metapath P_m^A, $\mathbf{h}_v^{P_m^A}$ can be interpreted as a summarization of the whole metapath instances of P_m^A about the node v, illustrating information of each metapath-based context of the target node v.

3.2.2 Inter-metapath Aggregation

Following intra-metapath aggregation, we perform inter-metapath aggregation to synthesize insights from all metapaths for issues and PRs. Considering that diverse metapaths encapsulate different information perspectives, it is reasonable to expect their contributions to the generation of target node embeddings will vary accordingly. Hence, we leverage attention mechanisms to assign weights for different metapaths, to build more comprehensive representations for issues and PRs.

Formally, considering a target node type $A \in \{I, PR\}$ and $V_A \in \{V_I, V_{PR}\}$, intra-metapath aggregation generates a set of latent vectors $\{\mathbf{h}_v^{P_1^A}, \mathbf{h}_v^{P_2^A}, ...\mathbf{h}_v^{P_m^A}\}$ for $v \in V_A$. Here, m is the number of metapaths oriented to the node type A. Next, given a certain node type A, we learn the importance ($\beta_{P_m^A}$) of each metapath $P_m^A \in P_A$ by global attention mechanisms.

To this end, we perform a series of operations. First, we employ a nonlinear transformation to transform metapath-specific vectors $\mathbf{h}_v^{P_m^A}$ for all nodes $v \in V_A$ in the metapath P_m^A. Then, for all nodes in V_A (i.e., V_I or V_{PR}), we average their transformed vectors to summarize the whole node information under the metapath P_m^A. The above processes are formulated as follows:

$$s_{P_m^A} = \frac{1}{|V_A|} \sum_{v \in V_A} \tanh\left(\mathbf{M}_A \cdot \mathbf{h}_v^{P_m^A} + \mathbf{b}_A\right) \tag{7}$$

where $\mathbf{M}_A$ is the weight matrix, $\mathbf{b}_A$ is the bias vector. $|V_A|$ is the number of nodes whose type is A.

After that, we use the attention mechanism and normalize it via a softmax function to improve the stability and interpretability, which as:

$$\begin{aligned} e_{P_m^A} &= \mathbf{q}^\top \cdot s_{P_m^A} \\ \beta_{P_m^A} &= \frac{\exp(e_{P_m^A})}{\sum_{P \in P_A} \exp(e_P)} \end{aligned} \tag{8}$$

where $\mathbf{q}$ is the parameterized attention vector for node type A and $\beta_{P_m^A}$ can be regarded as the contribution of the metapath P_m^A. More detailed, when the value of $\beta_{P_m^A}$ is higher, the more important metapath P_m^A is. Finally, we fuse all metapath-specific embeddings with the corresponding weight to obtain the final embedding of node v.

$$\mathbf{h}_v^{P_A} = \sum_{P \in P_A} \beta_P \cdot \mathbf{h}_v^P \tag{9}$$

After the inter-metapath aggregation, we obtain the node embeddings of the issue nodes V_I and PR nodes V_{PR}, which have aggregated the whole information extracted from different metapaths with the corresponding weights. In our prediction task, we set the dimension of the final vectors of nodes of V_I and V_{PR} as 64. We consider that metapath aggregation can alleviate the challenges arising from the heterogeneity of the heterogeneous graph and help effectively extract features based on different nodes and relationships, thereby enriching the information for the desired issue and PR nodes.

3.3 Prediction and Objective Function

Finally, AIPL designs an output layer to predict the relationships between issues and PRs. The output layer employs a linear transformation with a nonlinear function to calculate the probability of the link existence between an issue and a PR, which can be depicted as:

$$\hat{y} = \sigma(\mathbf{W}_O \cdot (\mathbf{h}_{v_n^I}^{P_I} \parallel \mathbf{h}_{v_n^{PR}}^{P_{PR}})) \tag{10}$$

where σ is an activation function of sigmoid and $\mathbf{W}_o$ is a learnable weight matrix. $\mathbf{h}_{v_n^I}^{P_I}$ is the generated embedding of target issue node v_n^I based on issue-oriented metapaths P_I, and $\mathbf{h}_{v_n^{PR}}^{P_{PR}}$ is the generated embedding of target PR node v_n^{PR} based on PR-oriented metapaths P_{PR}. When the probability is greater than 0.5, it indicates the presence of a link between an issue and a PR, otherwise, there is no link. Since our task is to predict the links between issues and PRs which can be regarded as a binary classification task, we adopt a binary-cross entropy loss:

$$L = -\frac{1}{|\mathcal{D}|} \sum_{(\{v_n^I, v_n^{PR}\}, y) \in \mathcal{D}} (y \log(\hat{y}) + (1-y)\log(1-\hat{y})) \tag{11}$$

where $\mathcal{D}$ denotes the training set that provides labels of linkage between issues and pull requests.

Finally, the overall forward propagation process is shown in Algorithm 1. Specifically, we input our task-specified heterogeneous graph G, metapaths $\mathcal{P}$ and the number of attention heads K. First, we initialize G and transform all node representations into the same dimension by Eq.(2). Next, for target nodes $v \in \{V_I, V_{PR}\}$, we aggregate all metapath instances of each metapath P related to v with multi-head attention mechanism by Eq.(5-6). Sequentially, we fuse all embeddings from different metapaths by Eq.(7-9) and generate the final embeddings of target node v. Finally, we calculate the probability of the link existence between an issue v_n^I and a PR v_n^{PR} by Eq.(10).

Algorithm 1: AIPL forward propagation

input : The heterogeneous graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \mathbf{X})$, $\mathcal{P}$, K
output: The probability of linking v_n^I and v_n^{PR}

Initialize G;
Transform $v, \forall v \in \mathcal{V}$ into the same dimension by Eq.(2);
for $v \in \{V_I, V_{PR}\}$ **do**
  for *metapath* $P \in \{P_I, P_{PR}\}$ **do**
    Aggregate all metapath instances of P related to v via multi-head attention by Eq.(4) and (6);
  end
  Aggregate embeddings from all metapaths related to v by Eq.(7-9);
end
Calculate probability of linking v_n^I and v_n^{PR} by Eq.(10);

TABLE 2: Statistics over our newly constructed datasets.

Dataset	facebook/react	vuejs/vue
# of *I* nodes	37,136	23,166
# of *PR* nodes	34,540	10,826
# of *R* nodes	20	20
# of *U* nodes	25,860	15,188
# of *I-R* edges	37,136	23,166
# of *I-U* edges	37,136	23,166
# of *PR-R* edges	34,540	10,826
# of *PR-U* edges	34,540	10,826
# of *U-R* edges	26,780	17,185
# of *I-I* edges	9,809	4,187
# of *PR-PR* edges	8,409	2,142
# of *R-R* edges	27	7
# of {*I-PR*}	8,457	3,655

4 EXPERIMENT

In this section, we conduct experiments over real-world datasets by answering the following research questions.

- **RQ1** Can our proposed AIPL effectively predict issue-PR links on GitHub?
- **RQ2** Can AIPL be effective when specific portions of the heterogeneous graph or metapaths are removed?
- **RQ3** How does the performance of AIPL vary when different thresholds or numbers of attention heads are applied?

4.1 Dataset Construction.

Before evaluating AIPL, we attempt to construct suitable datasets for our research. Although some studies have released datasets for similar tasks, they have primarily focused on predicting links between issues and PRs within the same repositories. In contrast, our task aims to link issues and PRs across various repositories in GitHub. Additionally, our approach involves multiple data sources, including information from issues, PRs, repositories, and users, which are absent in prior datasets. Therefore, we have chosen to construct new datasets, and the corresponding process is described as follows.

Collection. To justify AIPL, we curate two datasets. For each dataset, we choose a large-scale GitHub repository as its basis. These two repositories are facebook/react and vuejs/vue. The reasons for this selection are as follows: 1) These two repositories host the most stars on GitHub, apart from some recommended lists, tutorials, and books, 2) they contain plenty of issues and PRs for our research (facebook/react host 27,229 issues and PRs and vuejs/vue host 13,064 by August 2023.), and they might hold more issue-PR links, 3) active communities are still dedicated to maintaining and developing these repositories and our work may help their contributors and subscribers. To clarify, we illustrate the construction details of the facebook/react dataset as an example. The procedure for building the vuejs/vue dataset follows the same methodology.

First, we adopt GitHub Rest API to collect timeline events of the whole issues and PRs of facebook/react. Next, we analyze the timeline events of each issue or PR and find there is an event called 'cross-referenced' when an issue or PR hosts a link. Based on the information from this event, we can ascertain whether an issue or a PR has previously been linked with other issues or PRs, and we can retrieve the IDs of the interlinked issues or PRs, along with their respective repositories. After collecting these repositories, we regard them as candidate repositories related to facebook/react as [1] [17] did. However, constructing a dataset that encompasses all the issues and PRs from these candidate repositories is an immensely resource-intensive and laborious task. To this end, we retain the foundational repository facebook/react (vuejs/vue would be included when constructing the vuejs/vue dataset) and randomly select 19 repositories from the above candidates. Collectively, each dataset holds 20 repositories.

Subsequently, we collect the descriptions with tags of these repositories and the information on all the issues and PRs under each of these 20 repositories. Based on the information on issues and PRs, we retain their titles and the login names of their submitters. Finally, we collect the commit history and fork records of all submitters.

Filtering. To guarantee the quality of datasets, we adopt the filtering process. First, we filter out non-English issues and PRs. Subsequently, we remove the issues and PRs that are submitted by bots, since these issues and PRs might lack essential content and primarily serve as automated notifications [37]. Also, we delete duplicate issues and PRs to ensure the uniqueness of the items in this dataset.

Annotation. As our approach relies on supervised learning to predict links between issues and PRs, it requires dataset annotation to train the AIPL. Following the annotation proceed of [15], we retain the whole existing issue-PR links as positive samples. For negative samples, we randomly match an issue from the aforementioned positive samples with a PR. However, we notice negative samples may be potential links but overlooked by developers. To mitigate this threat, we introduce manual inspection to ensure the quality of our dataset. We invite a group of volunteer experts to check our datasets. 1/2 of the volunteer experts are pursuing Ph.D. degrees in software engineering, and 1/2 of the volunteers are graduates in software engineering (two males and two females, ages ranging from 24 to 33). Also, all of these volunteers hold software development experience on GitHub.

Next, we divide them into two groups, where each group contains one PH.D. and one other volunteer. Each group is required to inspect each link in both the positive and negative samples. If an issue and its linked PR are deemed irrelevant in the positive samples or if there are disagreements in the negative samples, the inspection teams mark them independently. After merging the inspection results from both groups, Cohen's kappa agreement is calculated to be 0.7332, indicating an acceptable level of consistency in the results. After that, we require all volunteer experts

to discuss together and reach an annotation consensus. Finally, all the authors double-check the annotation results to ensure a 100% agreement rate. This data annotation procedure takes a total of two weeks, with per participant spending two weeks on the annotation tasks. This results in 8,457 positive samples with 8,441 negative samples in facebook/react repository 3,655 positive samples and 3,669 negative samples in the vuejs/vue repository.

After the aforementioned procedure, we construct two datasets named facebook/react dataset and vuejs/vue dataset, which provide comprehensive information about repositories, users, issues, and PRs. The detailed statistics of the two datasets are summarized in Table 2.

Dataset Split. To construct the test set, we extract the most recent 10% of submitted issues and PRs based on their submission timestamps. More details of the test sets are as follows: 822 positive links, 822 negative links from the facebook/react dataset, 341 positive links, and 341 negative links from the vuejs/vue dataset. The remaining data and linkage labels are used to train our model (7,635 positive links and 7,619 negative links from the facebook/react dataset, 3,314 positive links, and 3,328 negative links from the vuejs/vue dataset). Before proceeding with the training process, we perform a 90/10 chronological split on the remaining dataset for the training set and validation set. Note that the training and validation set do not overlap with the test set.

4.2 Experimental Settings

In this part, we present baselines and implementation details.

4.2.1 Baselines.

We briefly introduce the baseline methods for comparison below.

- **iLinker** [17]: iLinker can recommend the related issues for PRs based on textual similarity. The computation of text similarity information relies on TF-IDF, and deep learning techniques, i.e. Word Embedding and Document Embedding.
- **A-M** [15]: A-M supports connecting issues and PRs on GitHub by technologies of feature selection and Mondrian forest. Also, it achieves state-of-the-art performance for the task of predicting issue-PR links on its datasets. Note that, feature construction of A-M can be extended to cross-repository issue-PR links, and it's feasible on our dataset.

Additionally, we compare AIPL with other graph-based technologies to assess the performance of our methodology.

- **Random Walk** [27]: Random Walk algorithm is a classical and traditional method for graph learning. We set the whole heterogeneous graph constructed in Section 3.1 as the input for this algorithm. Then, given a target node such as an issue, the algorithm would explore the graph by performing random walks to generate a sequence of PR nodes. After the grid search, we set the initial step size to 5 and set the probability of a random jump to 0.1. For comparison, we keep the first PR node of the generated sequences as the final result.
- **R-GCN** [28]: R-GCN (Relational Graph Convolutional Network) model leverages graph convolutional networks to learn node representations and capture the structural information of the graph. The same training dataset (in Section 4.1) involving positive samples and negative samples is fed to train the R-GCN model. To make a fair comparison, we employ the same data preprocessing method and hyper-parameter optimization for R-GCN via the grid search.
- **HAN** [22]: HAN is a metapath-based method to learn embeddings of heterogeneous graphs. It also leverages attention mechanisms to combine information of metapath-based neighbors into one vector representation for each node. Also, we employ the same data prepossessing method and hyper-parameter optimization for HAN via the grid search.

4.2.2 Implementation Details.

First, AIPL projects the descriptions of repositories along with their tags, as well as the titles of issues and PRs as 50-dimensional vectors. In cases where the content of certain repositories, issues, or PRs is empty, we initialize their 50-dimensional vectors with random values [30], [38], [39]. During the training procedure, to overcome the over-fitting problem, we apply dropout [40] on the input embeddings with a 0.5 drop rate, which means, 50% of representations will be randomly masked to reduce the parameters that need to be trained in each batch training. Also, we train AIPL with batch size = 8, and learning rate = 0.001 for 30 epochs. Moreover, we also adapted the strategy of early stopping [41]. If the performance on the validation dataset did not promote for 10 epochs, the training process would be stopped. All the experiments are implemented by PyTorch over a server equipped with NVIDIA GTX 1060 GPU (3G memory).

4.3 Effectiveness Analysis on Predict Results (RQ1)

Table 3 shows the evaluation results and examples of different approaches in the task of predicting issue-PR links. The key outcome of this evaluation is that, compared with these existing state-of-the-art methods, our proposed AIPL achieves the best scores (seeing the bold cells in Table 3). Specifically, AIPL achieves average gains of 27.98% F1-score and realizes the highest accuracy of 93.39% on the facebook/react dataset and 92.08% vuejs/vue dataset. As for metrics of Precision and Recall, AIPL achieves an average of 23.99% and 30.31% improvement over other baselines.

To better illustrate our results, Table 4 lists examples of issue-PR links predicted by different approaches, which have been shown on our repository site. Combining these examples, we analyze the reasons why AIPL outperforms baselines. First, iLinker relies solely on text similarity for link prediction. This approach is limited by semantic gaps and incomplete textual information. In many cases, developers employ expressions to describe the same challenges or provide incomplete information, resulting in a similarity-based method insufficient. For example 1 of Table 4, the issue title only contains the content `Vue2.7 setup injectH`. iLinker fails to predict the link between this issue with the PR `fix: pass vue instance as this pointer to setup`. The incomplete information makes it challenging for iLinker to learn features of issues and PRs effectively. A-M outperforms iLinker by incorporating text similarity and additional information, yet it primarily focuses on textual information and basic attributes, neglecting other vital knowledge relevant to issues and PRs. As example 2 in Table 4, PR titled `episode detail page fixes` addresses the issue `video attribute needed but not guaranteed by React`. These items with different content come from various repositories and involve different contributors, making it challenging for A-M to establish a link between them. Through deeper insights, we find that both their repositories are related to video apps, and their submitters share similar development experiences. A-M lacks the capacity to explore and harness this valuable knowledge and perform not well in similar link examples.

As for the traditional graph-based method, Random Walk, its algorithm is characterized by traversing the neighboring nodes of

TABLE 3: Performace Comparison with Baselines.

Datasets	Metrics	iLinker	A-M	Random Walk	R-GCN	HAN	AIPL
facebook/react	Accuracy (%)	51.52	64.73	56.42	72.00	91.76	**93.39**
	Precision (%)	51.89	64.71	61.80	70.03	86.39	**90.56**
	Recall (%)	41.58	66.30	33.82	76.61	95.27	**96.61**
	F1-score (%)	46.16	65.49	43.71	73.17	90.61	**93.49**
vuejs/vue	Accuracy (%)	50.44	70.38	54.55	74.93	90.91	**92.08**
	Precision (%)	50.44	70.09	57.48	71.55	90.45	**92.38**
	Recall (%)	48.39	72.14	36.95	79.18	90.91	**93.26**
	F1-score (%)	49.39	71.10	44.98	75.17	90.61	**92.81**

TABLE 4: Examples of Issue-PR prediction by different approaches.

ID	Issue	PR	iLinker	A-M	Random Walk	R-GCN	HAN	AIPL
1	Vue2.7 setup injectH	fix: pass vue instance as this pointer to setup	✗	✓	✓	✓	✓	✓
2	video attribute needed but not guaranteed by React	episode detail page fixes	✗	✗	✗	✓	✓	✓
3	Replacing prototype while setting objects	do not proxify objects that are values of non-patchable array properties	✗	✗	✗	✗	✓	✓
4	Remove deprecations for 1.0.0	build: Add storybook a11y not guaranteed by React	✗	✗	✗	✗	✗	✓

the target node first. This makes it easier to search for nearby neighboring nodes, especially when the issue and PR are from the same repository or same submitter. However, this algorithm may struggle to predict links between distant nodes due to the risk of getting stuck in local optima and missing the global optimal solution. R-GCN regards each relationship (i.e. edge) as a homogeneous graph, fuses different relationships separately, and then superimposes them to update the node features. While R-GCN can account for different relationships, it does not fully consider the potential interactions between complex relationships. Taking the $i_1 - r_1 - i_2$ metapath instance as an example, R-GCN can only learn the information between i_1 and r_1, but it neglects the information transmission from i_1 to i_2. Consequently, it results in a significant loss of information in the target node's embedding. The above limitations constrain the performance of Random Walk and R-GCN in predicting the links between seemingly less closely related issues and PRs (example 3, 4 in Table 4). Compared with the above baselines, AIPL introduces a heterogeneous graph to model issues and PRs with related sources on GitHub, which can represent the correlations and transmit information among them. Next, we identify which crucial knowledge should be aggregated into embeddings of issues and PR and formulate the series of metapaths. These metapaths propagate the information of crucial neighbors to issues and PRs and such knowledge-aware technology can generate more comprehensive embeddings. Additionally, the attention mechanism also plays an important role in performance. It helps us weigh the importance of different metapaths and decrease the influence of irrelevant information.

Finally, HAN, a model leveraging heterogeneous graphs and metapath-based techniques, emerges as the top-performing baseline. However, the performance gain obtained by AIPL over HAN is around 1-4% and struggles to process the complicated situation (example 4 in Table 4) that issues and PRs do not hold any direct relationships including similar repositories or collaborators. This result underscores the pivotal role played by the metapath instance encoder of AIPL, in the process of intra-metapath aggregation (as detailed in Section 3.2.1). This encoder is responsible for aggregating information from all nodes including target nodes, metapath-based neighbors, and intermediate nodes. With richer information, AIPL can learn more accurate embeddings for target nodes, surpassing the performance of HAN, which merely focuses on metapath-based neighbors.

Further, we manually check some randomly sampled erroneous issue-PR pairs and identify two main reasons why the results predicted by AIPL do not match the ground truth. First, AIPL struggles to handle fine-grained distinctions. For example, AIPL predicts the link between the issue `wrong parameter in custom directive 1.0.27` and the PR `fix directive sort compare`, yet they are annotated as negative samples in the ground truth. Despite their similar content, the issue pertains to version 1.0.27 of the repository vuejs/vue, while the PR serves version 2. AIPL cannot detect such nuanced factors as version numbers, leading to confusion in distinguishing issues and PRs that aim at different versions but share relevant content. Besides, subtle differences such as variations in variable names and method names can also lead to a decrease in the accuracy of AIPL. To accommodate this complex situation, it is necessary to employ text detection methods that can capture and preserve these crucial details.

In addition, partial newly established links have not been entirely learned by AIPL. Our ground truth set selects recent links as the ground truth instead of randomly sampling as the traditional deep learning methods [15]. Consequently, there will inevitably be new or similar problems that have not appeared in the historical data. This unlearned knowledge poses challenges for AIPL to predict the newly established links in the ground truth set. To alleviate this limitation, we aim to continuously collect information and enrich our historical data.

4.4 Ablation Study(RQ2)

In AIPL, the first step is constructing the task-specific heterogeneous graph. It is achieved by mapping the textual information of issues, PRs, and repositories as well as users to initial nodes and leveraging the relationships among the above sources to establish edges. Next, we learn the high-level embeddings of issues and PRs

TABLE 5: Evaluating the Rationality of Constructing Heterogeneous Graphs.

Datasets	Metrics	AIPL on Issue-PR graph	AIPL on User-Issue-PR graph	AIPL on Repo -Issue-PR graph	AIPL without textual information	Complete AIPL
facebook/react	Accuracy (%)	62.97	72.97	77.52	50.00	**93.39**
	Precision (%)	61.89	79.33	70.75	50.00	**90.56**
	Recall (%)	63.64	63.88	81.45	**100.00**	96.61
	F1-score (%)	62.75	70.77	75.73	66.67	**93.49**
vuejs/vue	Accuracy (%)	67.30	72.73	75.51	50.00	**92.08**
	Precision (%)	63.38	73.18	79.67	50.00	**92.38**
	Recall (%)	58.36	70.97	72.23	**100.00**	93.26
	F1-score (%)	60.77	72.05	76.04	66.67	**92.81**

TABLE 6: Evaluating the Rationality of Designed Metapaths.

Datasets	Metrics	Variant ①	Variant ②	Variant ③	Variant ④	*Complete AIPL*
facebook/react	Accuracy (%)	89.88	79.82	90.42	82.55	**93.39**
	Precision (%)	89.97	76.25	90.46	80.25	**90.56**
	Recall (%)	95.39	84.00	90.79	84.48	**96.61**
	F1-score (%)	92.60	79.94	90.62	82.31	**93.49**
vuejs/vue	Accuracy (%)	90.91	76.25	90.03	87.68	**92.08**
	Precision (%)	90.22	73.97	91.18	85.89	**92.38**
	Recall (%)	92.67	79.77	90.62	90.03	**93.26**
	F1-score (%)	91.42	77.76	90.90	87.91	**92.81**

based on this heterogeneous graph and metapaths extracted by our observation (see Sections 2.1 and 2.3). Accordingly, our ablation study is divided into two parts. The first part is to verify the quality and effectiveness of our constructed heterogeneous graph and the second is to evaluate the rationality of our designed metapaths.

- Focusing on the first aim, we design the following variants. 1) *AIPL on Issue-PR graph.* We construct the initial heterogeneous graph which only consists of issue nodes, PR nodes as well as edges of *I-I* and *PR-PR*. 2) *AIPL on Repo-Issue-PR graph.* We construct the initial heterogeneous graph which removes the user nodes as well as associated edges and metapaths. 3) *AIPL on User-Issue-PR graph.* We construct the initial heterogeneous graph which removes the repository nodes, associated edges, and metapaths. 4) *AIPL without semantic information.* We remove all textual information of issue nodes, PR nodes, and repository nodes, as well as remove the edge information based on textual information involving *R-R*, *I-I*, and *PR-PR* (illustrated in Section 3.1). The metapaths such as `I-I`, `PR-PR`, `I-R-R-I`, and `PR-R-R-PR` are also deleted in this variant. Then, we compare AIPL with four variants in the ground truth.
- Focusing on the second aim, we conduct the following variants to evaluate the performance of AIPL with different metapath sets. Variant ① *Removing metapaths* `I-I` *and* `PR-PR`. Variant ② *Removing metapaths* `I-R-I`, `PR-R-PR`, `I-R-R-I` *and* `PR-R-R-PR`. Variant ③ *Removing metapaths* `I-U-I`, `PR-U-PR`, `I-U-R-U-I` *and* `PR-U-R-U-PR`. Variant ④ *Removing metapaths* `I-R-U-R-I` *and* `PR-R-U-R-PR`.

4.4.1 Analysis of the Constructed Heterogeneous Graph

The results in Table 5 illustrate that AIPL performs the best compared with other baselines with different heterogeneous graphs. These results reveal the significance of each piece of information we utilized, including repository information, user information, and textual information, in achieving effective performance. According to our analysis of results, we summarize the following findings.

- Textual information plays the most crucial role in predicting the links between issues and PRs. *AIPL without textual information* is almost unable to work. Without textual information, AIPL almost loses its ability to learn knowledge, as its loss function fails to converge. It leads to the fact that *AIPL without textual information* classifies all samples as linked and gets results of 50% accuracy and 100% recall. The specific reasons are as follows. In the process of constructing a heterogeneous graph, textual information is essential to represent the node content of issues, PRs, and repositories to facilitate subsequent learning. Additionally, edges based on text similarity are indispensable, including edges *R-R*, *I-I*, and *PR-PR*. These edges can shorten the distance between matching issue nodes and PR nodes, and when learning the embedding of nodes, similar nodes can inject valuable information for them.
- Repositories and users are also useful for our task. When removing the information of repositories (*AIPL on Repo-Issue-PR graph.*) or the information of users (*AIPL on User-Issue-PR graph*), all metrics decrease (19.57% on average). Especially, when we only retain the relevant information on issues and PRs, *AIPL on Issue-PR graph* experiences a more pronounced decrease (30.44% on average). Firstly, issues and PRs are not only related to their content but also hold associations with their repository and the submitters involved. Therefore, when learning embeddings of issues and PRs, it is crucial to consider comprehensive information. Secondly, the connections between repositories and users also assist AIPL in learning more accurate and comprehensive embeddings for issues and PRs. These connections provide additional information, propagate the information from the neighborhood nodes, and enable metapath-based deep learning to acquire knowledge.

4.4.2 Results of Evaluating the Designed Metapaths

The performances of AIPL with different metapath sets are exhibited in Table 6. We can see that AIPL with all metapaths is consistently superior to all other variants, indicating the whole metapaths are necessary. Specifically, we observe that metapaths related to

repositories (I-R-I, PR-R-PR, I-R-R-I and PR-R-R-PR) exhibit the most significant influence (all metrics decrease 14.60% in Variant ②). This is primarily attributed to the repository-related information and functional scenarios propagated through these metapaths, which prove highly valuable for comprehending issues and PRs. Subsequently, metapaths I-R-U-R-I and PR-R-U-R-PR emerge as vital contributors (all metrics decrease 7.93% in Variant ④). These metapaths, by acting as intermediaries between repositories across diverse domains, facilitate the exchange of information between issues and PRs. Consequently, they provide additional perspectives to enhance the embedding of issues and PRs. Moreover, metapaths rooted in text similarity and user-centric metapaths prove to be equally indispensable (all metrics decrease 1.44% and 2.44% in Variant ① and Variant ③). These metapaths furnish rich, detailed information and encapsulate user experiences. As a result, they augment the quality and comprehensiveness of the acquired embeddings. To sum up, our designed metapaths are reasonable for AIPL.

Our results emphasize the significance of diverse knowledge in enhancing issue and PR comprehension and link prediction. It is our aspiration that this knowledge-aware approach can be extended to other sources on GitHub, such as specific analyses of repositories and comprehensive modelings of developers, and further prompt knowledge sharing across various platforms. Additionally, we expect our research can help the open-source community, enabling them to optimize issue and PR management and foster knowledge reuse and collaborative development.

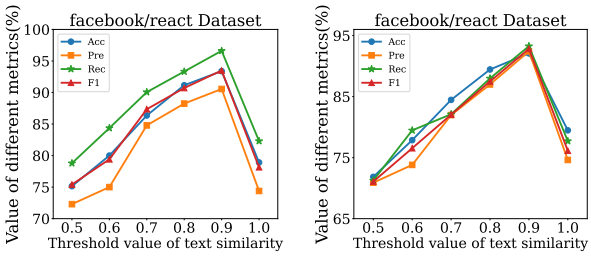
4.5 Parameter Selection(RQ3).

During the implementation of AIPL, we manually configure two essential parameters. The first parameter concerns the text similarity threshold, crucial for establishing edges within heterogeneous graphs like *R-R*, *I-I*, and *PR-PR*. The second parameter governs the number of attention heads employed in the attention mechanism. Consequently, we conducted the following two experiments to assess the efficacy of these parameter settings.

- We increase threshold values of text similarity from 0.5 to 1 with an interval of 0.1. In the process of constructing heterogeneous graphs, when the similarity between repository representations, issue representations, or PR representations is larger than the threshold, it will establish an edge between them. Then, we compare the performance of AIPL with different thresholds of text similarity.
- The set process of the number of attention heads(K) is discussed. We examine AIPL with different K from set $K \in \{2, 4, 6, 8, 10, 12, 14, 16\}$ since this parameter is often set as an even number in deep learning generally [21].

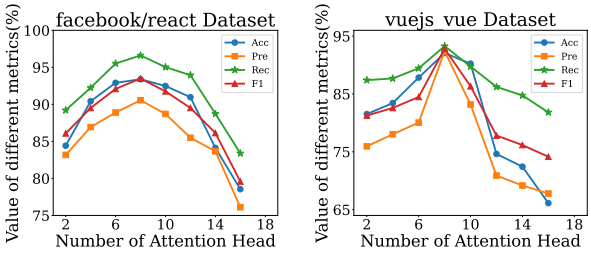
4.5.1 *Results of Parameter Selection(RQ3)*

Figure 5 shows the performance of our model with different thresholds of text similarity. The x-axis represents the threshold values of text similarity between representations of repositories, issues, or PRs, and the y-axis shows different metrics. According to the shown results, we can find all metrics increase when we improve the threshold from 0.5 to 0.9. It suggests that when we adopt a more lenient criterion for assessing similar nodes, it results in the formation of edges between nodes that are not adequately similar. Consequently, during the training of embeddings for the target nodes, extraneous and noisy information is introduced via these erroneous edges, leading to a degradation in performance. Conversely, when we rigorously restrict connections only to highly similar nodes (by setting the threshold to 1.0), essential information that should have been conveyed through metapaths is obstructed, causing a substantial decline in AIPL's performance. Hence, we set the threshold of text similarity as 0.9.



(a) facebook_react Dataset (b) vuejs_vue Dataset

Fig. 5: Evaluation results of AIPL with the different threshold of text similarity.



(a) facebook_react Dataset (b) vuejs_vue Dataset

Fig. 6: Evaluation results of AIPL with the different number of attention heads.

We report the effectiveness of AIPL with different numbers of attention heads in Figure 6. Where, the abscissas axis is the number of attention heads, and the ordinate axis is the corresponding value of different metrics. Also, links in different colors represent different metrics. As we can see, in the beginning, the performances of our model in both repositories continue to improve, but when the number of attention heads exceeds 8, the performance of the model shows a significant downward trend. Such a problem is consistent with "overfitting" and "degradation" mentioned in [42]. It explains if the performance of the attention-based model has reached the maximum, adding additional attention heads to this network will backfire. Because the model with more attention heads becomes more prone to over-attending or attending to irrelevant information. This can lead to a decrease in the model's ability to focus on important features or relationships, resulting in reduced performance. Based on the corresponding results, we set the number of attention heads as 8.

5 Related Work

We present the related work from two aspects: Links on GitHub and Heterogeneous Graph Embedding.

5.1 Links on GitHub

Link records of GitHub aid repository contributors in tracing the process of repositories and reusing code from more experienced developers. Hence, many previous studies have proven that linking operations is common in development practices and is helpful for the process of global collaboration on GitHub.

For links between issues and PRs or among issues and PRs, Le et al. [12] find that linking issue reports is a common practice when developers conduct issue-related tasks. The links between issue reports can facilitate the process of issue fixing and make developers more efficient in handling a series of issues. Zampetti

et al. [43] investigate how developers document pull requests in GitHub with external references, and they find that linking external resources can be useful for learning something new or solving specific problems. Furthermore, in a recent paper, Zhang et al. [17] presented a mixed methods study of the relationship between the practice of issue linking and issue resolution in the Rails ecosystem. They found that linking across projects will not retard issue resolution, and it is associated with more discussion. Based on the above findings, Zhang et al. propose an approach called iLinker for knowledge sharing, which combines information retrieval techniques and deep learning models to infer the links between issues and PRs, as well as among issues and PRs. PARACHI et al. [15] present a new tool, Aide-memoire, for improving traceability and repository maintenance, which employs feature selection approaches and Mondrian Forest to suggest such links when a developer submits a pull request or an issue.

However, studies predicting issue-PR links still are limited in numbers. Moreover, the majority of existing work relies on text similarity. In contrast, we propose a graph-based deep learning approach for predicting issue-PR links across repositories that can enrich information on issues and PRs.

5.2 Heterogeneous Graph Embedding

Due to the data heterogeneity in real-world settings, such as the GitHub platform, scholars are dedicated to heterogeneous graphs, achieve heterogeneous graph embeddings via learning node representations, and further support the downstream tasks, such as link prediction, and node classification.

One predominant approach conducts heterogeneous graph learning through the utilization of metapaths. For example, Dong et al. [44] formalize metapath-based random walks to search neighborhoods for nodes and employ a heterogeneous ship-gram model to represent node embeddings. Similar to [44], Shi et al. [45] employs a type constraint strategy to filter the node sequence and applies the DeepWalk model to learn the node embeddings of the target type. Shang et al. [46] propose a novel embedding-based framework. It models vertices as low-dimensional vectors to explore network structure-embedded similarity. Fu et al. [47] propose a model called HIN2Vec, which carries out multiple prediction training tasks jointly based on a target set of relationships to learn latent vectors of nodes and metapaths. HAN [22] proposed by Wang et al. incorporates both node and metapath-based neighbors and employs an attention mechanism to learn the importance of different metapaths. Also, MAGNN [21] leverages intra-metapath aggregation and inter-metapath aggregation to combine information from multiple metapaths to generate the embeddings of target nodes. Xing et al. [48] design a Multi-view Graph Neural Network. The proposed method develops a novel and inductive subspace-based strategy for link inference by aggregating multi-typed node and edge feature views and capturing the enhanced representations.

Inspired by the great success of these methods on heterogeneous graphs, we introduce heterogeneous graph embedding to our task issue-PR link predictions. It is worth emphasizing that we design task-specific metapaths and heterogeneous graphs to represent enhanced embeddings of issues and PRs.

6 Threats to Validity

We have identified the following threats to the validity of our work.

Internal Threat. The first threat is that AIPL makes it hard to represent the practical situation of all repositories in GitHub. In this paper, we construct heterogeneous graphs based on two repositories, namely facebook/react and vuejs/vue, along with their relevant repositories to train models. However, the relevant repositories included in our study encompass diverse tools, cover different time periods, and are developed in various programming languages. The issues and pull requests from these repositories are split into training, validation, and test sets for our experiments. We consider it can migrate this threat to some extent. Still, we will try to employ datasets on a larger scale to pre-train a more generalized model in the future.

External Threat. The second threat comes from human-involved annotation. To avoid the situation of mistaken links and missed links, we invite specialized researchers to annotate our dataset and the manual annotation introduces personal bias. To reduce subjectivity, we strictly follow the standard labeling process and let the two external teams label these documents independently. Further, we require both teams to discuss together to help resolve disagreements. Afterward, we double-check all labels to ensure that there is no error due to inconsistency.

7 Conclusion

In this paper, we proposed a novel approach, named AIPL, based on the heterogeneous graph with metapath-based techniques for predicting links between issues and PRs on GitHub. We evaluate our approach around two large-scale datasets constructed manually. Experimental results show that it can effectively predict the issue-PR links and outperform the other baselines. In addition, we conduct the ablation analysis to investigate the influence of different information and designed metapaths on AIPL. The results show that the whole heterogeneous graph and metapaths of AIPL play an important role in the effectiveness.

In the feature, we further improve the performance of AIPL by incorporating more information, such as code files of PRs, and users' profiles. Also, we intend to add edge weights in the heterogeneous graph to reflect the varying strengths of relationships between different sources. Moreover, we plan to take our approach one step further by employing a larger-scale dataset to predict links from small and new repositories.

Acknowledgments

The work is funded by Jilin Provincial Natural Science Foundation 20230101070JC, the Fundamental Research Funds for the Central Universities, JLU, (2022-JCXK-16), the National Natural Science Foundation of China (NSFC) No. 62102160, and Science and Technology Research Project of Education Department of JiLin Province of China (JJKH20211104KJ).

References

[1] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, "Curating github for engineered software projects," *Empirical Software Engineering*, vol. 22, pp. 3219–3253, 2017.

[2] M. Ruffin and C. Ebert, "Using open source software in product development: a primer," *IEEE Software*, vol. 21, pp. 82–86, 2004.

[3] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye, "Evolution patterns of open-source software systems and communities," in *International Workshop on Principles of Software Evolution*, 2002.

[4] Y. Peng, C. Fu, Y. Yu, C. Huang, Z. Zhao, and J. Dong, "Multiplex heterogeneous graph convolutional network," *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.

[5] O. Baysal, R. Holmes, and M. W. Godfrey, "No issue left behind: reducing information overload in issue tracking," *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014.

[6] R. Kikas, M. Dumas, and D. Pfahl, "Using dynamic and contextual features to predict issue lifetime in github projects," *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pp. 291–302, 2016.

[7] B. Vasilescu, Y. Yu, H. Wang, P. T. Devanbu, and V. Filkov, "Quality and productivity outcomes relating to continuous integration in github," *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015.

[8] M. Zhang, H. Liu, C. Chen, Y. Liu, and S. Bai, "Consistent or not? an investigation of using pull request template in github," *Inf. Softw. Technol.*, vol. 144, p. 106797, 2021.

[9] W. Maalej and H.-J. Happel, "Can development work describe itself?" *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pp. 191–200, 2010.

[10] D. Ståhl, K. Hallén, and J. Bosch, "Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework," *Empirical Software Engineering*, vol. 22, pp. 967–995, 2017.

[11] H. Rocha, M. T. Valente, H. T. Marques-Neto, and G. C. Murphy, "An empirical study on recommendations of similar bugs," *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, pp. 46–56, 2016.

[12] T.-D. B. Le, M. L. Vásquez, D. Lo, and D. Poshyvanyk, "Rclinker: Automated linking of issue reports and commits leveraging rich contextual information," *2015 IEEE 23rd International Conference on Program Comprehension*, pp. 36–47, 2015.

[13] J. Lin, Y. Liu, Q. Zeng, M. Jiang, and J. Cleland-Huang, "Traceability transformed: Generating more accurate links with pre-trained bert models," *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 324–335, 2021.

[14] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. T. Devanbu, "Fair and balanced?: bias in bug-fix datasets," in *ESEC/FSE '09*, 2009.

[15] P.-P. Pârtachi, D. R. White, and E. T. Barr, "Aide-mémoire: Improving a project's collective memory via pull request-issue links," *ACM Transactions on Software Engineering and Methodology*, vol. 32, pp. 1 – 36, 2023.

[16] A. Bachmann, C. Bird, F. Rahman, P. T. Devanbu, and A. Bernstein, "The missing links: bugs and bug-fix commits," in *Fast Software Encryption Workshop*, 2010.

[17] Y. Zhang, Y. Wu, T. Wang, and H. Wang, "ilinker: a novel approach for issue knowledge acquisition in github projects," *World Wide Web*, vol. 23, pp. 1589–1619, 2020.

[18] P. Ma, D. Xu, X. Zhang, and J. Xuan, "Changes are similar: Measuring similarity of pull requests that change the same code in github," *Communications in Computer and Information Science*, 2017.

[19] Y. Yu, H. Wang, G. Yin, and T. Wang, "Reviewer recommendation for pull-requests in github: What can we learn from code review and bug assignment?" *Inf. Softw. Technol.*, vol. 74, pp. 204–218, 2016.

[20] W. Xiao, H. He, W. Xu, X. Tan, J. Dong, and M. Zhou, "Recommending good first issues in github oss projects," *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, pp. 1830–1842, 2022.

[21] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding," *Proceedings of The Web Conference 2020*, 2020.

[22] X. Wang, H. Ji, C. Shi, B. Wang, P. Cui, P. S. Yu, and Y. Ye, "Heterogeneous graph attention network," *The World Wide Web Conference*, 2019.

[23] C. Zhang, D. Song, C. Huang, A. Swami, and N. Chawla, "Heterogeneous graph neural network," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[24] W. Guan, F. Jiao, X. Song, H. Wen, C.-H. Yeh, and X. Chang, "Personalized fashion compatibility modeling via metapath-guided heterogeneous graph learning," *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022.

[25] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, pp. 61–80, 2009.

[26] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *ArXiv*, vol. abs/1609.02907, 2016.

[27] F. Wang and D. P. Landau, "Efficient, multiple-range random walk algorithm to calculate the density of states." *Physical review letters*, vol. 86 10, pp. 2050–3, 2000.

[28] M. Schlichtkrull, T. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Extended Semantic Web Conference*, 2017.

[29] S. Bai, L. Liu, H. Liu, M. Zhang, C. Meng, and P. Zhang, "Find potential partners: A github user recommendation method based on event data," *Inf. Softw. Technol.*, vol. 150, p. 106961, 2022.

[30] H. Fazayeli, S. M. Syed-Mohamad, and N. S. M. Akhir, "Towards auto-labelling issue reports for pull-based software development using text mining approach," *Procedia Computer Science*, 2019.

[31] R. Kallis, A. D. Sorbo, G. Canfora, and S. Panichella, "Ticket tagger: Machine learning driven issue classification," *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 406–409, 2019.

[32] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[33] L. Shi, M. Xing, M. Li, Y. Wang, S. Li, and Q. Wang, "Detection of hidden feature requests from massive chat messages via deep siamese network," *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pp. 641–653, 2020.

[34] H. Shao, D. Sun, J. Wu, Z. Zhang, A. Zhang, S. Yao, S. Liu, T. Wang, C. Zhang, and T. F. Abdelzaher, "paper2repo: Github repository recommendation for academic papers," *Proceedings of The Web Conference 2020*, 2020.

[35] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 4–24, 2019.

[36] Z. Sun, Z. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," *ArXiv*, vol. abs/1902.10197, 2018.

[37] S. Bai, L. Liu, C. Meng, and H. Liu, "Automating discussion structure re-organization for github issues," *Expert Syst. Appl.*, vol. 225, p. 120024, 2023.

[38] L. Bao, X. Xia, D. Lo, and G. C. Murphy, "A large scale study of long-time contributor prediction for github projects," *IEEE Transactions on Software Engineering*, vol. 47, pp. 1277–1298, 2021.

[39] S. Baltes and S. Diehl, "Usage and attribution of stack overflow code snippets in github projects," *Empirical Software Engineering*, vol. 24, pp. 1259–1295, 2018.

[40] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[41] L. Prechelt, "Early stopping-but when?" in *Neural Networks*, 1996.

[42] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," *ArXiv*, vol. abs/1905.09418, 2019.

[43] F. Zampetti, L. Ponzanelli, G. Bavota, A. Mocci, M. D. Penta, and M. Lanza, "How developers document pull requests with external references," *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pp. 23–33, 2017.

[44] Y. Dong, N. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:3919301

[45] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, pp. 357–370, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:2409634

[46] J. Shang, M. Qu, J. Liu, L. M. Kaplan, J. Han, and J. Peng, "Meta-path guided embedding for similarity search in large-scale heterogeneous information networks," *ArXiv*, vol. abs/1610.09769, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:3964926

[47] T. yang Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:3958144

[48] Y. Xing, Z. Li, P. Hui, J. Huang, X. Chen, L. Zhang, and G. Yu, "Link inference via heterogeneous multi-view graph neural networks," in *International Conference on Database Systems for Advanced Applications*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221839358