A Knowledge Graph Construction Method for Software Project Based on CAJP

Yang Deng[1], Bangchao Wang[1,2], Zhongyuan Hua[1], Yong Xiao[1], Xingfu Li[1]*

[1] School of Computer Science and Artificial Intelligence, Wuhan Textile University, China
[2] Engineering Research Center of Hubei Province for Clothing Information, Wuhan Textile University, China
dd0028y@163.com, wangbc@whu.edu.cn, 1745224789@qq.com, 634218476@qq.com, lixingfu1999@163.com

Abstract

In recent years, there has been increasing interest in using knowledge graphs (KGs) to help stakeholders organize and better understand the connections between various artifacts during software development. However, extracting entities and relationships automatically and accurately in open-source projects is still a challenge. Therefore, an efficient method called Concise Annotated JavaParser (CAJP) has been proposed to support these extraction activities, which are vitally important for KG construction. The experimental result shows that CAJP improves the accuracy and type of entity extraction and ensures the accuracy of relationship exaction. Moreover, an intelligent question-and-answer (Q&A) system is designed to visualize and verify the quality of the KGs constructed from six open-source projects. Overall, the software project-oriented KG provides developers a valuable and intuitive way to access and understand project information.

Keywords: Knowledge graph, Open-source software project, Q&A system, CAJP

1 Introduction

Knowledge graphs (KGs) have gained significant attention recently as a powerful tool for representing and analyzing complex data, including open-source software projects. They provide a visual and intuitive way of expressing complex relationships between entities, such as software artifacts, developers, and users [1].

One of the main benefits of using KGs in open-source software is the ability to understand better the connections between different artifacts in projects, which include source codes, user manuals, bug reports, question-and-answer (Q&A) documents, and more [2]. By providing a holistic view of these artifacts, KGs can help developers better understand the functionality of a software project and make more informed decisions about how to reuse existing code [3].

Building a KG for open-source software projects is still challenging due to the large scale, complex structure, and rich meaning of software artifacts. One of the critical components of building a KG is the extraction of entities and relationships, which can be difficult when working with open-source software projects [4]. However, due to the special code, these existing methods for extracting entities and relationships from open-source software projects result in a significant amount of redundant information in the entity extraction, reducing the efficiency of developers.

To alleviate these problems, we propose a KG construction approach and Concise Annotated JavaParser (CAJP) method for extracting entities and relationships. To reduce the interference caused by redundancy, the CAJP divides annotation into Context and DocletTag, improving the accuracy and type of entity extraction and ensuring accurate KG construction. The entities and relationship extraction by CAJP are the basis of KG construction. An intelligent Q&A system for KG is designed to provide developers with a more intuitive way to access and understand the information.

The main contributions are as follows:

(1) The extraction method called CAJP is proposed for open-source code. Six software projects are used to validate the effectiveness of extracting entities and entity relationships.
(2) An intelligent Q&A system is designed, which provides developers with a more intuitive way to understand the information in the KG.
(3) The KG with 4335 entities and 8267 relationships is constructed. The rationality of software project KG construction is verified from entity extraction, entity relationship extraction, and Q&A system.

The remainder of the paper is organized as follows: Section 2 introduces the related work. Section 3 details our approach for software project KG construction. Section 4 experiments to validate the approach proposed. Section 5 is the result and analysis. Section 6 discusses the threats to the validity of the proposed method and experiments. Section 7 summarizes our study and provides future works.

2 Related Work

This section summarizes the entity extraction method and literature quality for software project KG.

2.1 Entity Extraction Method

KG construction in open-source software projects mainly focuses on organizing and managing knowledge from software development processes to improve software reuse and efficiency. There are currently some approaches to

*Corresponding Author: Bangchao Wang; E-mail: wangbc@whu.edu.cn
DOI: 10.53106/160792642023112406006

software project entity extraction. Shang et al. [4] constructed a user behavior KG from the development code and the user behavior. The Qdox was proposed to extract entity, a small-footprint, high-speed parser that extracts metadata from Java source. Thus, the class properties extracted by this method sometimes need to be corrected about the parent class, resulting in inaccurate extracted class properties. The other method is the JavaParser method proposed by Peng et al. [5], which uses Eclipse JDT's ASTParser to parse each Java file into an abstract syntax tree. However, annotation redundancy exists when extracting block annotations. We follow a similar underlying idea but propose the CAJP method, which reduces the interference caused by redundancy and optimizes the accuracy of entity extraction comparison.

2.2 Literature Quality Assessment

Several studies have proposed different software project KG construction methods. Lin et al. [6] used software KGs in the intelligent development environment to provide intelligent help for software development. Seven data formats are supported. However, this article does not have experiments to verify KG construction. Li et al. [2] devised with a method for building a software project KG for open-source projects, which aimed to organize and manage the structural knowledge of software projects. Zou et al. [7] focused on constructing software KG based on big data. They proposed a code-centric software knowledge model, a two-layer plugin framework for KG construction, and software Q&A. Osorio et al. [8] proposed a framework called DockerPedia aimed to improve the documentation and understanding of software images used in KG. According to Table 1, the level of quality validation of the above studies is at most Level 1.

Therefore, a more intuitive and high-quality verification level is needed to ensure the robustness and reliability of the constructed KGs. This paper uses six datasets for the CAJP and KG construction methods, which are fully validated by three experiments, and the quality of the literature is at Level 3.

Table 1. The detail of the literature quality assessment [9]

Level	Describe	Literature
0	No evidence.	[4]
1	Evidence obtained from demonstration or working out with toy examples.	[2, 6-8]
2	Evidence obtained from expert opinions or observations (e.g., survey or interview).	
3	Evidence obtained from academic studies (e.g., controlled lab experiments).	
4	Evidence obtained from industrial studies (e.g., causal case studies in an industrial setting).	

3 Our Approach

3.1 Overview

Figure 1 shows our proposed four-steps approach to constructing KG and designing a Q&A system for open-source software projects: ontology construction, information exaction, knowledge storage and fusion, and Q&A system design. The following subsections explain each step in detail.

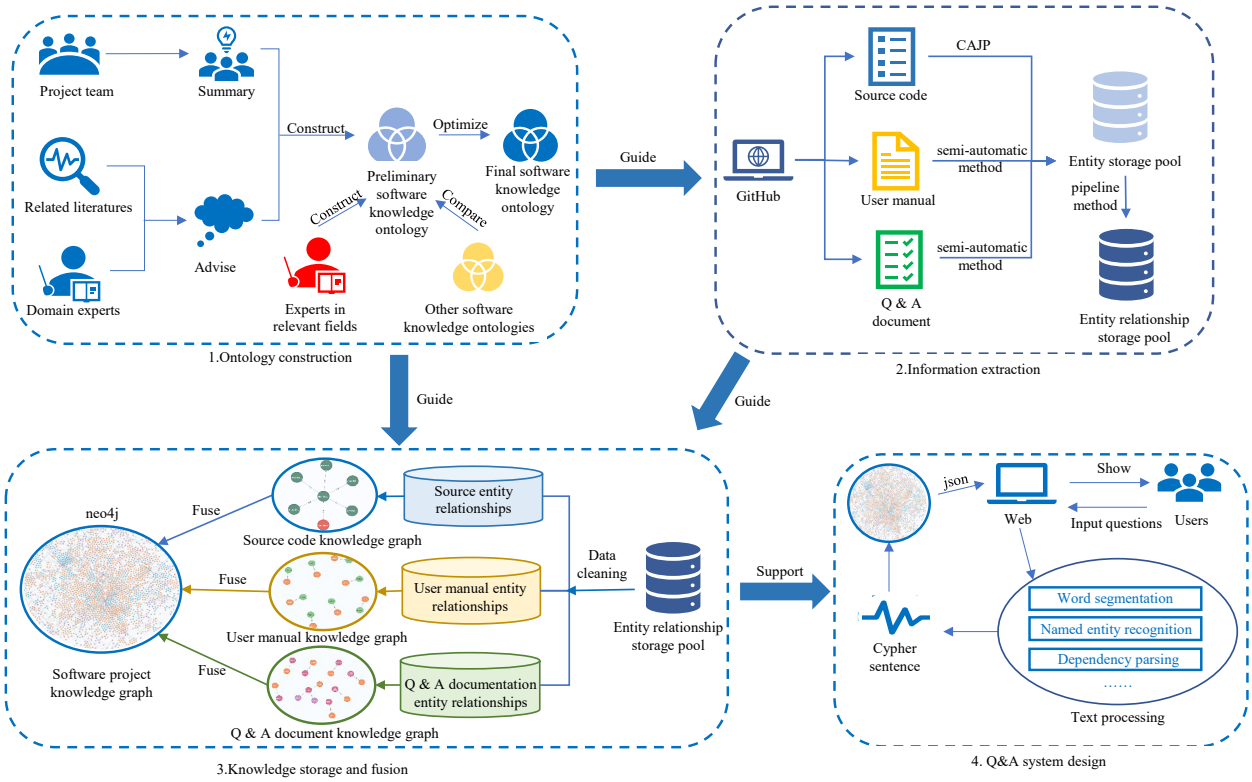

Figure 1. The KG construction framework for the open-source software project

3.2 KG Construction

3.2.1 Ontology Construction

Figure 2 shows the ontology detail, which guides the extraction of entities and entity relationships. The "blue" represents the three artifacts to construct a software project KG, which includes java source code, user manuals, and Q&A documents. The "yellow" represents the entities extracted for each artifact. In addition, the text on the arrows represents the entity relationships.

Step 1 (on the upper left of Figure 1) shows the Skeletal method [10], divided into three steps to build our ontology. To begin, the project team creates a preliminary ontology from domain projects on GitHub and pieces of literature. In the second step, we invite domain experts and compare them with other open-source software project ontologies to optimize our ontology. Lastly, we randomly obtain datasets in java open-source software projects. If all datasets have a corresponding ontology, the ontology build is complete. Otherwise, go back to the second step.

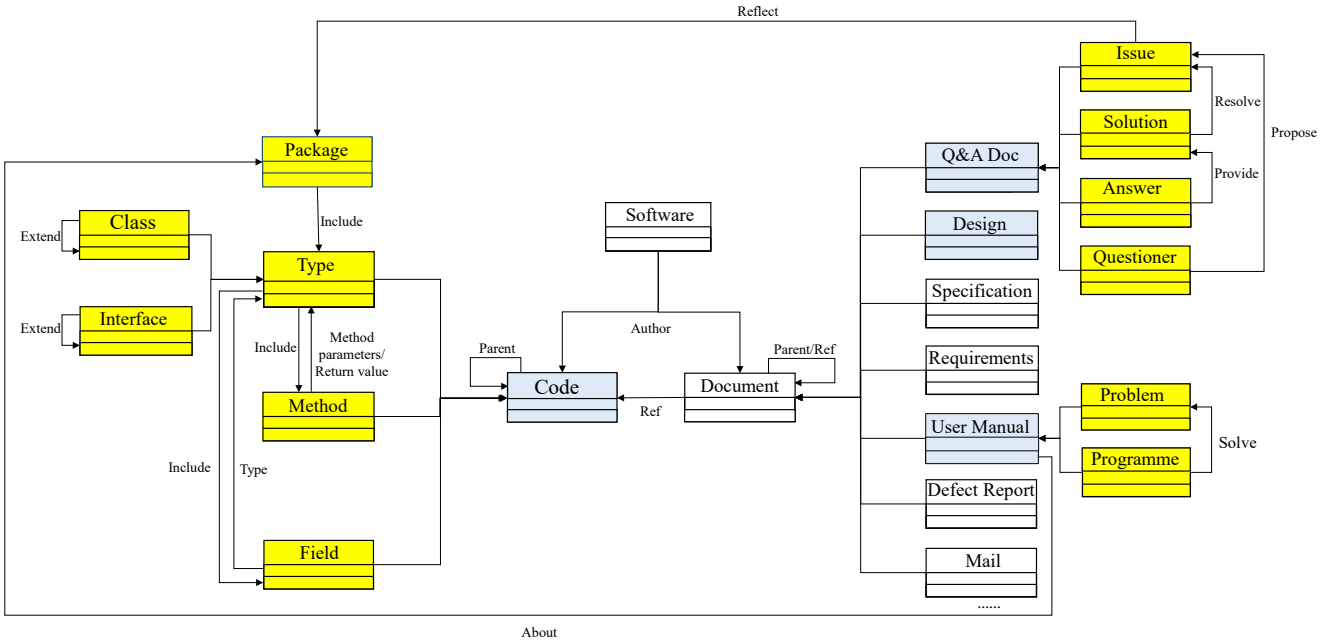

Figure 2. The ontology of open-source software project KG

3.2.2 Information Extraction

Step 2 (on the upper right of Figure 1) shows the information extraction process. These entities and entity properties are extracted from the three kinds of software projects. The retrieved entities are then utilized to create entity relationships. This section will be divided into entity extraction and entity relationship to introduce the extraction and the methods used.

1) Entity Extraction

Figure 3 shows the detail of entity extraction from three types of software projects, which includes entities and entity attributes extraction.

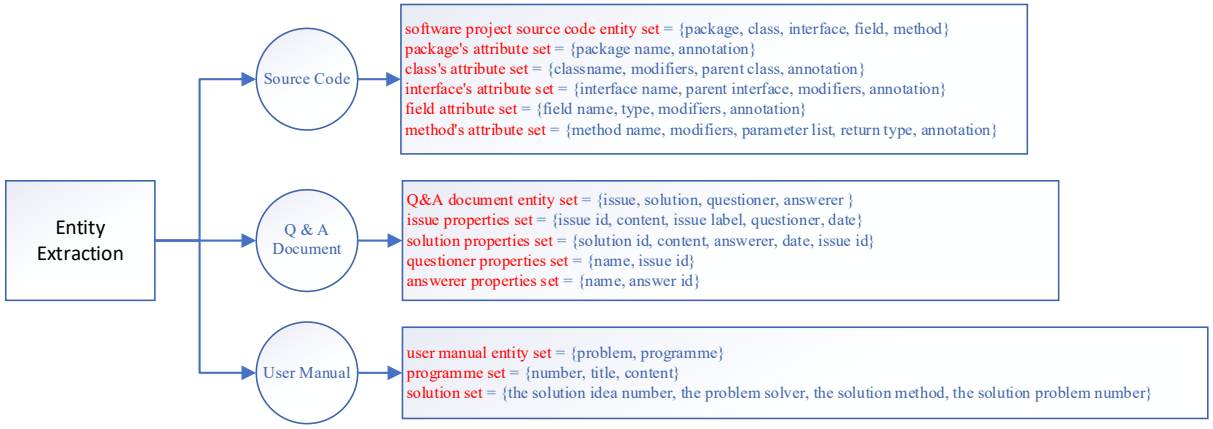

Figure 3. The information on entity extraction from software projects

Table 2. Annotation extraction algorithm

```
Input: Tags List T
Output: AnnotationDecomposer Object
1. Function TagsDecomposer(T)
2.    Save the text list of Tags and content C
3.    Annotation content storage list Contexts
4.    Document Part Storage List Doclets
5.    While T≠Ø:
6.       Get the Tag in the current tag list
7.       If Tag_size>2:
8.          Tag_TaxtContext = ExtractContext (Tag); // Extracting
Tag Text
9.          C.add (Tag_TaxtContext);
10.      End if
11.   End while
12.   While C≠Ø:
13.      Get the current Tag text content
14.      If Tag_TaxtContext.isDocletTag():
15.         // The current Tag text content is a document part
16.         docletTag = docletTagDecomposer (Tag_TaxtContext)//
Creating Part Objects
17.         DocletTags.add(docletTag)
18.      End if
19.      Else:
20.         Contexts. add (Tag_TaxtContext)
21.      End else
22.   End while
23. Return InitAnnotationDecomposer ()
```

The semi-automatic method [5] extracts entities from the user manual and Q&A document, which combines text similarity and expert feedback. CAJP extracts entities from the source code.

The CAJP method enhances the accuracy of entity extraction by parsing source code into an abstract syntax tree (AST) and optimizing block annotation and class property extraction. Figure 5 (RQ1) shows that CAJP is optimized for inaccurate parent class properties exaction. In addition, this method associates block annotations to the corresponding code element, not associate line annotations. As shown in Figure 6 (RQ1), the block annotations are divided into each line, one line as a Tag. Then, the Context, DocletTag of name, and DocletTag of value are extracted from each Tag. Table 2 shows the pseudo-code for annotation extraction.

Lines 2-4 initialize the relevant variables; lines 5-11 divide the block annotation into line annotations (Tags); lines 12-22 separate the tags into Context, DocletTag of value, and DocletTag of name; and line 23 returns the data.

2) Entity Relationships Extraction

As depicted in Figure 2, constructing entity relationships involves establishing connections within and between resources. A pipeline method is utilized when building relationships in source codes to achieve high accuracy in information extraction. Table 4 presents the pseudo-code for the pipeline method.

As shown in Table 3, line 2 initializes the Java parse tree node traverser, lines 3-6 traverse the document and build a java parse tree for each file, and lines 7-15 extract entity relationships by traversing each node and finally return a triplet set.

Table 3. The pipeline algorithm of extracting entity relationships

```
Input: Java Project Documentation Collection DS
Output: RDF triple Collection RS
1.    Function CreateRelation (DS)
2.       visitor = initVisitor (RS)//Initialize the Java parse tree node
      walker
3.       While DS≠Ø:
4.          Get the current java file D
5.          comp = createComplication(D)//Building a Java parse
      tree
6.             nodes = comp. accept(visitor) //visitor traverses the
      parse tree and returns a list of nodes
7.          While nodes≠Ø:
8.             Get current node N
9.             If N. isKey ():  //If N is a keyword such as a package,
      class, etc., record it
10.               Key = N
11.            End if
12.            If N. hasRelation ():
13.               re = createIncludeRelation (key, N)
14.               RS.add(re)
15.            End if
16.         End while
17.      End while
18.      Return RS
```

3.3 Knowledge Storage and Fusion

The neo4j [7] is a powerful tool for storing and managing data, as it utilizes a graph structure to represent information. It allows for a more intuitive representation of associated data, as the relationships between different entities can be easily visualized. Additionally, the traversal algorithm design of neo4j utilizes the natural extension properties of graph structures, making it more efficient and user-friendly than traditional relational databases that require complex connection operations [6].

Step 3 (on the bottom left of Figure 1) shows the knowledge fusion and storage process. Firstly, we use pandas [11] to remove the same entity. Knowledge from different sources but representing the same entity is merged under the guidance of our ontology. Secondly, <entity, relation, entity> triplets are stored in eno4j through py2neo [7]. Figure 4 shows an example of the three types of software KGs after fusion. The right part is to query and add a node through neo4j.

3.4 Q&A System Design

The lexicon-grammar-based semantic [12] parsing method has strong interpretability and a clear structure, which achieves good results in question-answering in limited areas. Step 4 (on the bottom right of Figure 1) shows that we use the Flask framework and charts technology to build a web page to implement the interaction between software project KGs and users (Figure 8). The open-source software project KG is used as the underlying data support of the system. Firstly, obtain user questions from the web page. Secondly, NLTK and LTP [5] implement text processing such as word segmentation, named entity recognition, and dependency syntax analysis to convert natural language into cypher sentences. Then, extract the answer by obtaining the data in neo4j with the cypher sentence. Finally, the answers in the form of json are passed to the web for display.

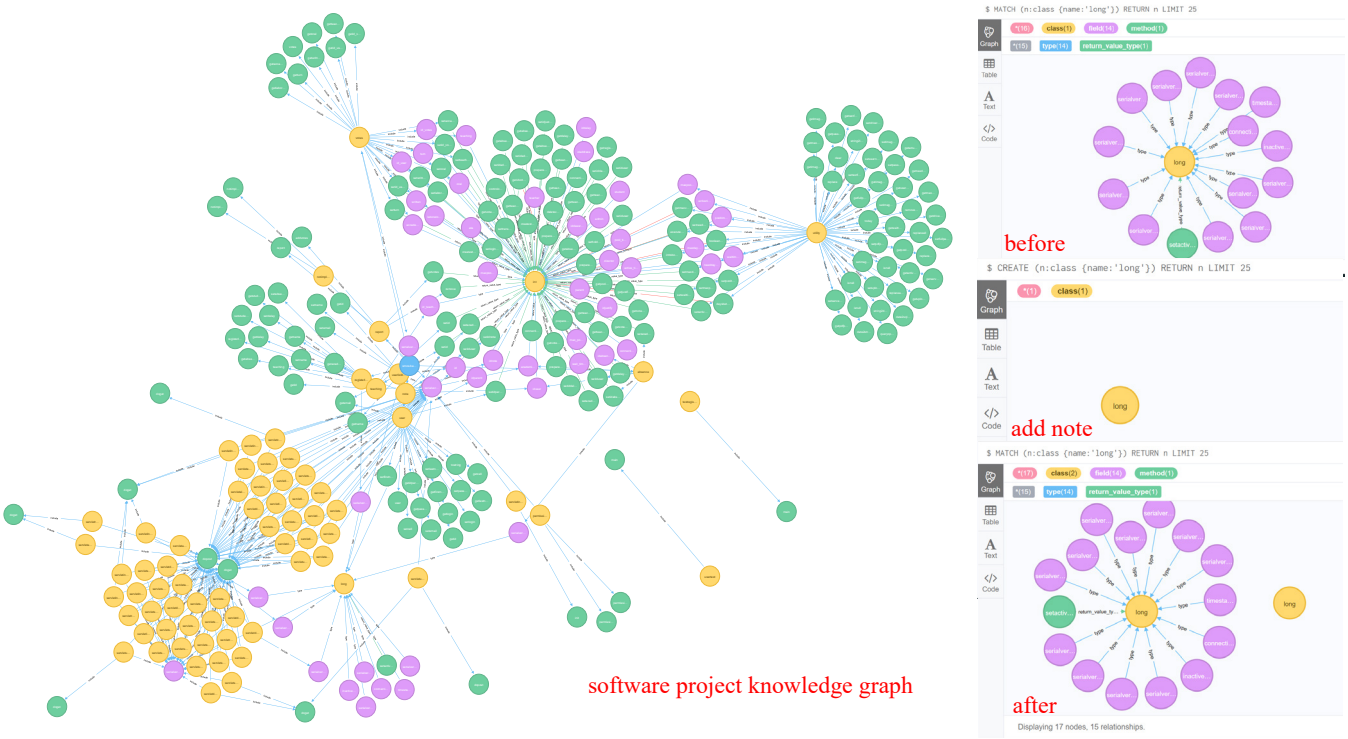

Figure 4. An example of the three types of software project knowledge graphs after fusion

4 Experimental

4.1 Research Question

To ensure the quality of the KG construction, this paper introduces the CAJP method for improved entity extraction. The effectiveness of this method is evaluated through RQ1. To assess the method's effectiveness in extracting entity relationships, RQ2 is proposed for experimental verification. Finally, the paper constructs a software project KG based on six datasets, designs a Q&A system to visualize the KG, and evaluates the system's quality through RQ3. Three research questions are defined as follows:

RQ1: How effective is the CAJP in improving annotation extraction?

RQ2: How does the CAJP method impact constructing entity relationships?

RQ3: What is the quality of the KG constructed using the Q&A system?

4.2 Quality Measure

In this paper, different quality evaluation measures are adopted for different RQs. The details are shown in Table 4.

Accuracy is a metric defined as the proportion of correct predictions the model or algorithm makes. All results are correct when the *Accuracy* is 100%. The equation for *Accuracy* is as follows:

$$Accuracy = \frac{|N_{rel}|}{|N_{total}|}. \tag{1}$$

Where N_{total} represents the number of predictions. N_{rel} represents the number of correct predictions.

Cohen's kappa (K) [13] expresses the level of agreement between two annotators on a classification problem. The formula for calculating Cohen's kappa is shown in (2).

$$Kappa = \frac{q_1 - q_2}{1 - q_2}. \tag{2}$$

Where q_1 represents the ratio of the sum of the diagonal elements to the sum of the whole elements. q_2 represents the proportion of the sum of the multiplicate elements.

Table 4. Research questions and corresponding metrics

Research question	Quality measure
RQ1: How effective is the CAJP in improving entity extraction?	None
RQ2: How does the CAJP method impact constructing entity relationships?	Cohen's kappa, Accuracy
RQ3: What is the quality of the KG constructed using the Q&A system?	Cohen's kappa, Accuracy

4.3 Datasets

4.3.1 Data Acquisition

This experiment employs six Java open-source software projects as datasets. The project details are shown in Table 5. The dataset has three types, which are from GitHub. Table 6 is the composition of the source code.

Table 5. The number of user manuals, Q&A documents, and source codes in the six datasets

Project name	User manuals	Q&A documents	Source codes	Relationships
Designpattern-master	0	0	789	1019
GeekQ-Tools-master	0	0	61	117
Memoryoptimization-master	27	0	116	187
Miaosha-master	27	47	1873	5124
Threadandjuc-master	6	12	3113	5049
ZookeeperDesign-master	19	0	164	279

Table 6. The composition of the source code

Project name	Package	Class	Interface	Domain	Method	Class enumerations	Domain enumerations
Designpattern-master	180	157	23	79	349	1	0
GeekQ-Tools-master	13	11	2	7	28	0	0
Memoryoptimization-master	19	20	0	28	49	0	0
Miaosha-master	268	228	29	774	1130	27	218
Threadandjuc-master	387	504	15	640	1479	1	0
ZookeeperDesign-master	15	15	1	45	88	0	0

4.3.2 Sampling Method

The experiments in this paper use stratified random sampling [14], dividing the data into five categories: packages, classes, interfaces, domains, and methods, and sampling them respectively. The formula for sample sampling is as follows:

$$n = \frac{P(1-P)}{\frac{e^2}{z^2} + \frac{P(1-P)}{N}}. \tag{3}$$

Where e represents average *Accuracy*, P represents sample variation, N represents the total number of samples, and z represents statistic. The e of the expected average correct rate in this experiment is between plus and minus 0.05, the survey result is within the 95% confidence range, the 95% confidence level requires the z statistic to be 1.96, and the estimated sample variability P is 0.5 [15].

5 Result and Analysis

5.1 RQ1: How Effective is the CAJP in Improving Entity Extraction?

To assess the performance of the CAJP method for entity extraction, this study compares it to existing methods and performs quantitative and qualitative analyses of the three ways.

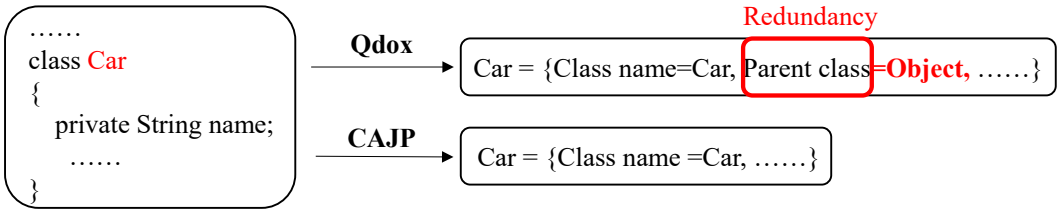

Figure 5. The class extraction process of CAJP and Qdox

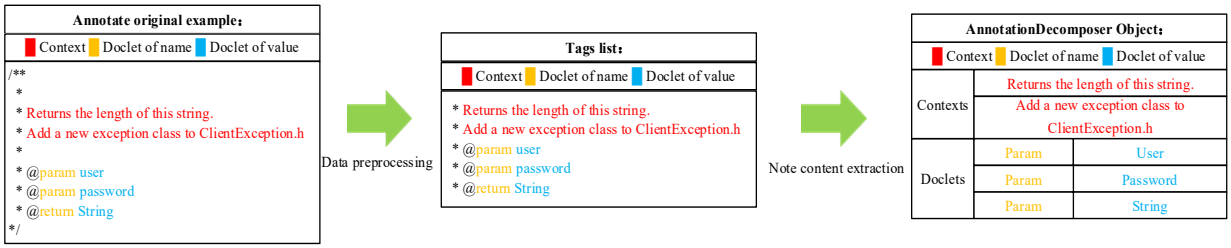

Figure 6. The optimization process for annotations in the CAJP method

Table 7. Function extraction comparison of JavaParser, Qdox, and CAJP

Function	JavaParser	Qdox	CAJP
Getting annotations on demand	×	√	√
Extracting annotation type modifier	√	×	√
Extracting class attributes correctly	√	×	√
Extracting the class constructor method	√	×	√

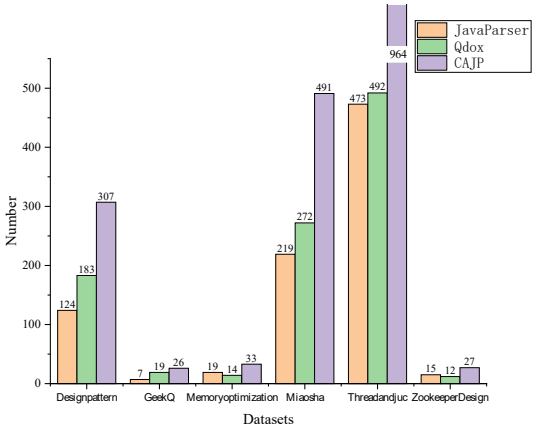

Figure 7. The number of optimizations in entity extraction

The process of class extraction is illustrated in Figure 5. The parent class is not present in the class properties, resulting in incorrect extraction by Qdox. Figure 6 shows how CAJP optimizes annotation. The annotations are divided into Context, Doclet of name, and Doclet of value, reducing interference caused by redundancy and improving the *Accuracy* of annotation extraction. Therefore, as shown in Table 7, the CAJP method improves the *Accuracy* of annotation and class attribute extraction and adds some functions compared to JavaParser.

As depicted in Figure 7, the number of entities optimized using the CAJP method is the largest and the most effective, followed by JavaParser and Qdox. This is because the CAJP method optimizes the entity extraction by reducing the annotation redundancy compared to JavaParser and reducing the annotation redundancy compared and incorrect class properties compared to Qdox. Thus, CAJP demonstrates superior optimization in entity extraction compared to the other methods.

Overall, the results of this experiment indicate that the CAJP method is an efficient and effective solution for entity extraction in open-source software projects.

5.2 RQ2: How Does the CAJP Method Impact Constructing Entity Relationships?

The RQ aims to evaluate the impact of entity relationship construction when the CAJP has improved the type of entity extraction. The *Accuracy* of the entity relationship is evaluated using JavaParser and Qdox as baselines. The two experiments are as follows:

(1) Firstly, the entity relationships and their numbers are constructed using CAJP, JavaParser, and Qdox. Secondly, a sample of datasets is chosen. Three researchers with expertise in Java and project experience are tasked with counting the number of entity relationships in each dataset, which serves as the "true set." Finally, the *Accuracy* is calculated for each sample dataset.

(2) To evaluate the reliability of the "true set" exacted by three researchers in (1), the K is used in this experiment as a statistical indicator of consistency.

As shown in Table 8, Qdox extracted a higher overall count of entity relationships than CAJP and JavaParser due to the difference in enumeration type division that affected the results of the three methods. Table 6 illustrates that the enumeration types include class enumeration and domain enumeration, with Qdox extracting class enumeration and domain enumeration as the domain. At the same time, CAJP and JavaParser divide enumeration types into different types. For the true set, this experiment considered class enumeration as a separate type, leading to a higher count of Qdox than the other two methods.

As shown in Table 9, the K of all datasets is 1.00. According to [13], the three researchers in (1) have the same judgment criteria for whether the entity relationship extraction is correct. Therefore, the experimental results in (1) are correct and valid, and the *Accuracy* of the sampled results are 1.00 in entity relationship extraction, which is better than Qdox and has the same performance as JavaParser for entity relationship extraction.

Overall, the results of this experiment demonstrate that after CAJP improves the entity extraction, the performance of the entity relationship extraction does not decrease, thus ensuring the quality of KG construction. This highlights the potential of CAJP as a valuable tool in developing KG and other similar applications.

Table 8. The number of extraction entity relationships by using CAJP, JavaParser, and Qdox

	JavaParser		Qdox		CAJP	
Project name	Overall count	Sample size	Overall count	Sample size	Overall count	Sample size
Designpattern-master	1019	279	1256	294	1019	279
GeekQ-Tools-master	117	117	111	111	117	117
Memoryoptimization-master	187	187	185	185	187	187
Miaosha-master	5124	357	5472	359	5124	357
Threadandjuc-master	5049	357	5971	361	5049	357
ZookeeperDesign-master	279	279	303	303	279	279

Table 9. Cohen's kappa and accuracy for entity extraction

	JavaParser		Qdox		CAJP	
Project name	*K*	*Accuracy*	*K*	*Accuracy*	*K*	*Accuracy*
Designpattern-master	1.00	1.00	1.00	0.92	1.00	1.00
GeekQ-Tools-master	1.00	1.00	1.00	0.93	1.00	1.00
Memoryoptimization-master	1.00	1.00	1.00	0.90	1.00	1.00
Miaosha-master	1.00	1.00	1.00	0.96	1.00	1.00
Threadandjuc-master	1.00	1.00	1.00	0.93	1.00	1.00
ZookeeperDesign-master	1.00	1.00	1.00	0.95	1.00	1.00

5.3 RQ3: What is the Quality of the KG Constructed using the Q&A System?

This RQ aims to evaluate the effectiveness of the software project KG constructed by the CAJP and the Q&A system designed to visualize it. To achieve this, the RQ utilizes six datasets (as shown in Table 5) to construct the software project KG and then invites two researchers with relevant experimental experience to evaluate its quality. The researchers are asked to propose and answer sixteen questions and then judge whether the answers provided by the Q&A system match their expectations.

Additionally, the experiment employs the *K* measurement index to assess the consistency of the two researcher's judgments with the answers provided by the Q&A system. The sixteen questions used in this evaluation are presented in Table 11. Figure 8 is a diagram of the designed Q&A system.

Table 10 shows that the opinions of the two authors regarding Q7 are not in agreement. The first author finds many method nodes called "generate" in the dataset, including a node with return value types of "C" and "ValidateCode." As a result, shown in Table 11, the first author considers that the Q&A system should give specific answers like "C" and "ValidateCode" rather than a simple "yes" or "no." However, the second author thinks there is a general method node in the code that returns "ValidateCode," and therefore, the answer of the Q&A system is appropriate.

Table 12 shows the *K* between [0.61-0.8]. According to [13], the consistency of the two authors in their answers to sixteen questions is substantial. With an *Accuracy* of about 90%, the answer of *Accuracy* is guaranteed. The KG construction method performs well in entity and relationship extraction.

The results demonstrate the effectiveness of CAJP in constructing high-quality software project KGs that can be effectively visualized through the designed Q&A system. The inconsistent judgments of the two authors on Q7 may indicate room for improvement in the ability of the Q&A system to provide specific answers. Overall, the results suggest that the Q&A system is a valuable tool for software developers.

Table 10. The sixteen questions proposed by two researchers

ID	Question
1	What questions do lany1721 ask?
2	What package does the CodeController class exist in?
3	Is there a main method in the Optimization Application class?
4	Which class does the domain validateCodeGenerators exist in?
5	What are the method parameters of the method getValidateCodeType?
6	Which is the parent class of class ImageCodeProcessor?
7	Which is the return value type of the method generateValidateCode?
8	How are website visits statistically achieved?
9	In which package is the method findValidateCodeProcessor1 defined?
10	Is the method addCourse defined in the interface CourseAggregate?
11	How do I start a project with an imported idea?
12	Don't constants in enumerations need to be final?
13	Is there a getWorkerId method in class IdWorker?
14	What is the type of domain called expireTime?
15	Does the class named MemoryOutOOMController have a domain named classList?
16	Is there a String type for the method parameter of the method User?

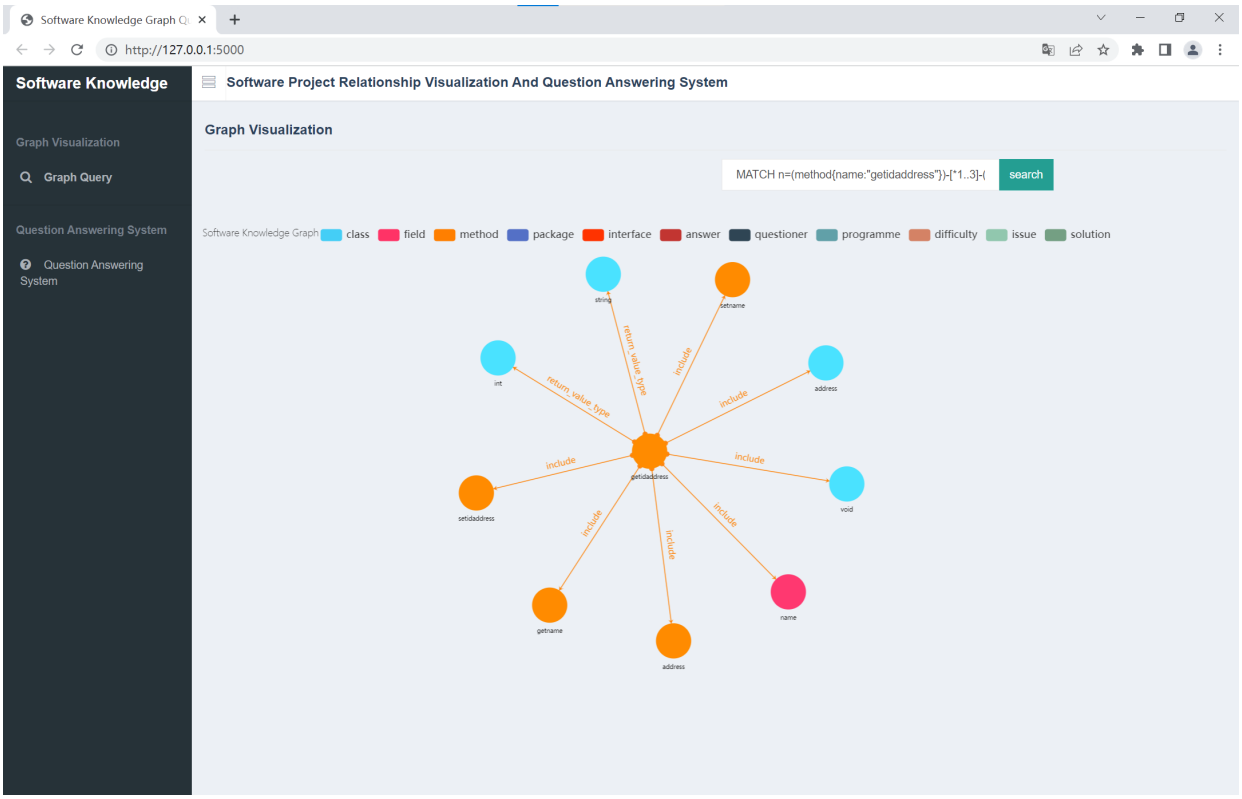

Figure 8. Open-source software project intelligent Q&A system

Table 11. The results of the two author's answers to the Q&A system

	Member I	Member II
Q1	√	√
Q2	√	√
Q3	√	√
Q4	√	√
Q5	√	√
Q6	√	√
Q7	×	√
Q8	√	√
Q9	√	√
Q10	√	√
Q11	√	√
Q12	√	√
Q13	√	√
Q14	×	×
Q15	√	√
Q16	√	√

(The answer of the Q&A system is consistent with the author's expectations √, otherwise ×)

Table 12. The Q&A system calculation

	K	*Accuracy*
Q&A System	0.6364	0.9063

6 Validity Threats

Some threats may limit the validity of these experiments, so this section focuses on potential threats and how we can control or alleviate them. The four validity threats are as follows:

Conclusion validity: A detailed research design is designed for this study. Section 4 shows that the research questions, the choice of quality measures, the dataset selection, and the sampling method are reasonably designed to ensure the conclusions are valid.

Construct validity: In this paper, the software project KG construction method shown is proposed in Figure 1. The CAJP can effectively support the construction method. In addition, accuracy metrics and Kappa are used to analyze the results, which can effectively quantify various situations.

Internal validity: The experimental results show that the CAJP influences the annotation redundancy and parent class attribute extraction in the KG construction. The impact factors need to be selected appropriately.

External validity: The experiment uses six datasets from GitHub. The datasets are publicly available, commonly used, and authentic, which reduces the threat of this study producing different results in different systems or projects. Therefore, the CAJP has a great opportunity and potential to be expanded to real software systems and projects.

7 Conclusion and Future Work

To help developers better understand the connections between different artifacts and mitigate the problem of redundancy in entity extraction, this paper proposes an efficient method for KG construction and an entity and relationship extraction method called CAJP. The method is

applied to six datasets to construct a KG with 4335 entities and 8267 relationships. The results show that CAJP improves the quality of entity extraction by reducing redundancy and ensuring the accuracy of entity relationship extraction. An intelligent Q&A system is also designed to visualize the KG, demonstrating the effectiveness of CAJP in constructing high-quality software project KGs.

The work needs further improvement. The next step is to expand the data types and improve the automation of the KG construction process.

Acknowledgment

This work is supported by the National Natural Science Foundation of China Project (No. 62102291), the Young Talents Programmer of Scientific Research Program of the Hubei Education Department (Project No. Q20211711), and the Opening Foundation of Engineering Research Center of Hubei Province for Clothing Information (No. 2022HBCI02, No. 2022HBCI05).

References

[1] K. W. Chen, C. Y. Huang, Automatic Categorization of Software with Document Clustering Methods and Voting Mechanism, *International Journal of Performability Engineering*, Vol. 18, No. 4, pp. 251-262, April, 2022.

[2] W. P. Li, J. B. Wang, Z. Q. Lin, J. F. Zhao, Y. Z. Zou, B. Xie, Software Knowledge Graph Building Method for Open Source Project, *Journal of Frontiers of Computer Science and Technology*, Vol. 11, No. 6, pp. 851-862, June, 2017.

[3] X. Ke, S. Li, Chinese Organization Name Recognition based on Co-training Algorithm, *2008 3rd International Conference on Intelligent System and Knowledge Engineering*, Xiamen, China, 2008, pp. 771-777.

[4] F. H. Shang, Q. Y. Ding, R. S. Du, M. J. Cao, H. Y. Chen, Construction and Application of the User Behavior Knowledge Graph in Software Platforms, *Journal of Web Engineering*, Vol. 20, No. 2, pp. 387-412, March, 2021.

[5] S. S. Xing, M. W. Liu, X. Peng, Automatic Code Semantic Tag Generation Approach based on Software Knowledge Graph, *Journal of Software*, Vol. 33, No. 11, pp. 4027-4045, November, 2022.

[6] Z. Q. Lin, B. Xie, Y. Z. Zou, J. F. Zhao, X. D. Li, J. Wei, H. L. Sun, G. Yin, Intelligent Development Environment and Software Knowledge Graph, *Journal of Computer Science and Technology*, Vol. 32, No. 2, pp. 242-249, March, 2017.

[7] Y. Z. Zou, M. Wang, B. Xie, Z. Q. Lin, Software Knowledge Graph Construction and Q&A Technology based on Big Data, *Big Data Research*, Vol. 7, No. 1, pp. 22-36, January, 2021.

[8] M. Osorio, C. Buil-Aranda, I. Santana-Perez, D. Garijo, DockerPedia: A Knowledge Graph of Software Images and Their Metadata, *International Journal of Software Engineering and Knowledge Engineering*, Vol. 32, No. 1, pp. 71-89, January, 2022.

[9] B. C. Wang, H. Wang, R. Q. Luo, S. Zhang, Q. Zhu, A Systematic Mapping Study of Information Retrieval Approaches Applied to Requirements Trace Recovery, *International Conference on Software Engineering and Knowledge Engineering*, Pittsburgh, Pennsylvania, USA, 2022, pp. 1-6.

[10] Y. J. Yang, B. Xu, J. W. Hu, M. H. Tong, P. Zhang, L. Zheng, Accurate and Efficient Method for Constructing Domain Knowledge Graph, *Journal of Software*, Vol. 29, No. 10, pp. 2931-2947, October, 2018.

[11] C. Müller, T. Pascher, A. Eriksson, P. Chabera, J. Uhlig, KiMoPack: A python Package for Kinetic Modeling of the Chemical Mechanism, *The Journal of Physical Chemistry A*, Vol.126, No. 25, pp. 4087-4099, June, 2022.

[12] A. Barreiro, C. Mota, J. Baptista, L. Chacoto, P. Carvalho, Linguistic resources for paraphrase generation in portuguese: a lexicon-grammar approach, *Language Resources and Evaluation*, Vol. 56, No. 1, pp. 1-35, March, 2022.

[13] M. Li, C. Zhang, T. F. Yu, Kappa values in testing the concordance: comments on a recent article about nasopharyngeal swabs for SARS-CoV-2 detection, *Microbiology Spectrum*, Vol. 10, No. 6, Article No. e0158222, December, 2022.

[14] J. N. K. Rao, W. A. Fuller, Sample Survey Theory and Methods: Past, Present, and Future Directions, *Survey Methodology*, Vol. 43, No. 2, pp. 145-160, December, 2017.

[15] J. A. A. Jothi, A. R. Sulthana, A Review on the Literature of Fashion Recommender System using Deep Learning, *International Journal of Performability Engineering*, Vol. 17, No. 8, pp. 695-702, August, 2021.

Biographies

Yang Deng was born in Hubei, China. Her major interests are in the areas of software engineering, requirements engineering, machine learning, and natural language processing.

Bangchao Wang is a Master Supervisor in the School of Computer Science and Artificial Intelligence at Wuhan Textile University, Wuhan, China. He received the PhD degree in Computer Science from Wuhan University. His research interests mainly include but are not limited to software engineering, requirements engineering, and knowledge engineering.

Zhongyuan Hua was born in Qianjiang City, Hubei Province. His main interests are in the areas of software engineering, requirements engineering, and knowledge engineering.

Yong Xiao was born in Xianning City, Hubei Province. His main interests are in the area of software requirements traceability.

Xingfu Li was born in Hubei, China. His major interests are in the areas of NLP and Software Engineering.