An Effective Method for Constructing Knowledge Graph to Search Reusable ROS Nodes

$Yuxin\ Zhao^{1,2}, Xinjun\ Mao^{1,2*}, Sun\ Bo^{1,2}, Tanghaoran\ Zhang^{1,2}, Shuo\ Yang^{1,2}$
[1]*College of Computer,National University of Defense Technology, Changsha, China*
[2]*Key Laboratory of Software Engineering for Complex Systems, National University of Defense Technology, Changsha, China*
{yuxinzhao, xjmao, sunbo, zhangthr, yangshuo11} @nudt.edu.cn

***Abstract*—Developing robot software is difficult for most software engineers as it requires multi-discipline knowledge such as robotics, AI, and software engineering. Robot Operating Systems (ROS) provides a software development framework and lots of reusable ROS Nodes that encapsulate various robotics functions, which can simplify robot software development in terms of software reuse. However, searching and reusing required ROS Nodes from thousands of ROS Nodes is still challenging due to the scattered distribution of ROS Node information and the need for adequate search methods. In this paper, we present an effective method to construct a ROS Node knowledge graph in support of searching and reusing ROS Nodes. Our method uses multiple data sources, including open-source ROS software in Github and ROS wiki community. We extract two-tuple functional information and task-related noun phrases from the ROS Node description and ROS communication interactions from the ROS Node source code. The constructed ROS Node knowledge graph (RNKG) contains 14,065 entities and 15,767 relations. It provides rich semantic information to comprehensively and precisely describe ROS Nodes, their services, and related interaction topics and messages.**

***Index Terms*—Knowledge Graph, ROS Nodes, Robotics software development**

I. Introduction

Robot gradually plays an essential role in many fields, including autonomous vehicles, factories, healthcare, services, and commerce. A robot is a software-intensive system that needs several robotics software components to accomplish task requirements cooperatively. The advent of Robot Operating System (ROS) [12] simplifies robot software development. ROS is the most popular framework in robotics which is designed to be modular at a fine-grained scale, and its fundamental concepts *Node*, *Message*, *Topic*, and *Service*. ROS Node can offer a specific function for robotics tasks and a sequence of ROS Nodes to complete complicated robot mission [14] to construct robotics software. So, ROS Node is a proper component to reuse for developing robotics software.

However, without in-depth knowledge of ROS Nodes and their interactions, reusing ROS Nodes is difficult [1]. From a developer's point of view, developing robotics software usually requires a series of ROS Nodes from different ROS packages. It is especially difficult for developers with less experience to find ROS Nodes and make ROS Nodes cooperative. Thus, in ROS-based robotics software development, a large portion of development efforts has been investigated into searching ROS Node, especially when their descriptions information and interactions are difficult to obtain. Therefore, searching applicable ROS Nodes is necessary for more efficient software development.

In this paper, we constructed a ROS Node Knowledge Graph (RNKG), a domain knowledge graph that captures the basic computation graph concepts of ROS and the relationship between different ROS concepts. The KG serves as the foundation for recognizing developer demands, inferring developer needs, recommending a proper ROS Node, and offering ROS knowledge for accelerating robotics software development. Our main contributions are summarized as follows:

- present a systematic method to construct a ROS domain knowledge graph, extracting two-tuple functional information and task-related noun phrases from the ROS Node description, and ROS communication interactions from the ROS Node source code.
- propose RNKG, a domain knowledge graph that provides rich semantics information to comprehensively and precisely describe the ROS Nodes, their services, and related interaction topics and messages.
- evaluate the completeness and the correctness of RNKG to show that the ROS knowledge extraction method is effective.

The rest of the paper is organized as follows. In the next section, we review previous studies on this topic. We illustrate the methodology of constructing a ROS Node knowledge graph in Section III. We evaluate the completeness and correctness of constructed knowledge graph in Section IV. We discuss the threats to validity in Section V. Finally, we conclude our work and discuss future directions in Section VI.

II. Related Work

Due to the excellent performance of knowledge graphs in various application scenarios (question answering, recommendation, and information retrieval), the research on domain knowledge graphs [7] in various fields continues to deepen. However, due to the strong domain characteristics of the collected knowledge, researchers need to design different knowledge extraction methods and relation extraction methods to solve the knowledge graph construction tasks in different knowledge types like extracting knowledge from network

* Corresponding author.

encyclopedia and web content [15], from Wikipedia [16] and Chinese texts [8].

Due to the powerful knowledge reasoning ability and recommendation of knowledge graphs, knowledge graphs have also been deeply studied in the field of robotics and have made a lot of breakthroughs. Applying the knowledge graph to the knowledge reasoning state enhances the robot's ability to process environmental information [5]. Besides, it plays an indispensable role in promoting the development of robot software based on the power of recommendation. The ROS-related knowledge graph can be used in searching ROS Package [4] and ROS message [2] to simplify the process of developing robotics software.

In the research process, the knowledge reasoning ability and recommendation searchability of the knowledge graph will effectively help ROS developers find the ROS Node they need.

III. Methodology

This section introduces the ontologies of knowledge in our KG, and then we will describe our method to construct a robotics-specific knowledge graph RNKG. Fig.1 shows the whole method of our approach and introduces the method of constructing the ROS Node Knowledge Graph.

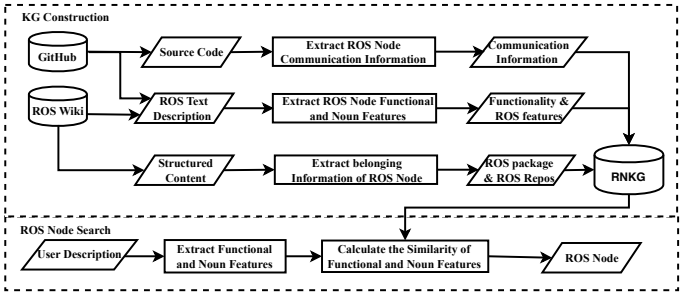

Fig. 1. The Overall method of our approach.

A. The ontology of RNKG

Before building the knowledge graph, we need to clarify the ontology and relationships contained in the ROS Node knowledge graph, so as to realize the guidance of entity extraction and relationship extraction. We show our core ontology in Fig.2. The ontology in the knowledge graph includes description information for ROS Node and common concepts of basic Computation Graph concepts of ROS.

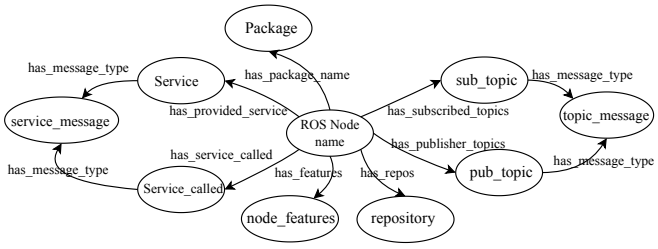

Fig. 2. The core ontology of RNKG.

B. Extract ROS Node Communication Information

We extract entity and relation related to ROS Node communication from the ROS Node source code in GitHub. To successfully extract the ROS Node information we need, we use pattern-matching based method [3] to realize the entity extraction of ROS Node.

C. Extract ROS Node Functional and Noun Features

We extract two-tuple functional information and task-related noun phrases from ROS Node description to distinguish different ROS Nodes.

Since ROS knowledge has strong domain characteristics, there are a large number of abbreviations (e.g., "ICP" means "Interactive Closest Point"). To fully understand the knowledge of ROS Node description, we build a local ROS dictionary and replace the ROS domain concepts in the text (specialized Nouns) with natural language description that users are more familiar.

For the ROS Node description, the type of each word tag is determined by the Part-Of-Speech Tagger in Fig.3. The part-of-speech tagger determines the syntactic category of each token and stores it in the POS feature, encoded in uppercase abbreviations. [9].

Fig. 3. An example of Part-Of-Speech tagger.

The next step in explaining ROS Node description is to eliminate the meaning of words. We need to combine single words into the most extensive phrase as much as possible to enhance the knowledge extraction of ROS Node description. We use two-tuple functional information and task-related Noun phrases to summarize ROS Node description.

For two-tuple functional information, we rely on the term Dependency Parsing (DP) [11] to examine the dependencies between the phrases of ROS Node description in Fig.4. We use $'obj' : (register, pointclouds)$ to describe the function of ROS Node.

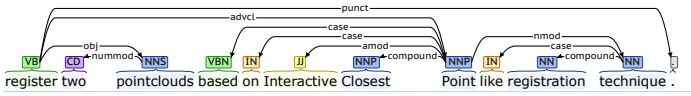

Fig. 4. An example of extracting two-tuple functional information.

For task-related Noun phrases, we use NP-chunking [17] to extract the ROS Node-related information in Fig.5, and we can get task-related Noun phrases "two pointclouds, interactive closest point, registration technique".

D. Constructed RNKG

So far, we have successfully constructed a knowledge graph containing ROS Node knowledge using the previous three extraction knowledge methods. The resulting RNKG consists of 14,065 entities and 15,767 relationships. Using Neo4j [10]

Fig. 5. Extracting noun phrases from ROS Node description.

to form the ROS Node domain knowledge graph, we store the structured triad data.

IV. Evaluation

In our constructed RNKG, the quality of ROS Node knowledge is essential, so we measure the completeness and correctness of ROS Node knowledge extraction to verify the usefulness of the ROS Node knowledge extraction method.

- **RQ1-** How is the completeness of the knowledge captured in the constructed RNKG?
- **RQ2-** How is the correctness of the knowledge captured in the constructed RNKG?

A. The completeness of RNKG.

The dataset required to build the knowledge graph was collected in GitHub. The completeness of RNKG refers to whether it contains all needed ROS Node information, including ROS Node Name, Topic, Service, and Message.

1) Protocol: we measure whether RNKG contains all ROS Node-related knowledge. Based on the relevant conclusions of other studies [6] [13], we constructed a verification dataset (randomly extracting 345 Node data, 345 Topic data, 345 Service data, and 345 message data) from ROS Wiki. The measure is whether the extracted validation data can be found in RNKG.

2) Results and Analysis: The results are shown in Table I. 90% of the ROS Node, Topic, and Service in the verification data set can be found in RNKG, and 98% of ROS Node in the verification dataset can be found in RNKG. Although the Message in the verification data set is less than 90%, its accuracy is still considerable. These data fully illustrate that the ROS Node knowledge contained in RNKG is complete.

> *More than 90% ROS Node knowledge extracted from ROS Wiki can be found in RNKG.*

B. The correctness of Node knowledge in RNKG

To evaluate the correctness of entity extraction method and relation extraction method, we should check whether we proposed method can successfully extract the required entities (sub_topics, pub_topics, service, service_called, service_message, package, and repository), and build correct relation between different entities.

1) Protocol: We obtained the knowledge of 355 ROS Nodes from ROS Wiki and RNKG, respectively. The knowledge provided in ROS Wiki is correct, and we need to measure the correctness of the knowledge in RNKG. We will calculate the correctness of entity extraction and relation extraction separately.

2) Results: The results of entity extraction are shown in Table.II. Generally speaking, the entity extraction of the RNKG knowledge graph is accurate and as complete as possible. The main reason for the high accuracy of entity extraction is that RNKG is in the ROS domain, and the dataset is composed of source code. Entity extraction from code data is simpler than analyzing language from natural language. The format of code data is fixed and strict, so it will naturally have a higher accuracy rate when combined with a template-based approach.

From Table.II shows that the F_1 of sub_topics and pub_topics is 87.60%, which is relatively low compared to other entities. One of the main reasons for this is that the code parameter settings of different ROS Node programs are different, and entity extraction is very difficult. It is difficult to distinguish what kind of parameter it is according to the string, so the accuracy of sub_topics and pub_topics is not very high.

For the relationship extraction between different entities, we determine it by analyzing the unique API in the ROS program because different entities must be implemented through specific API calls, so for the relation-

TABLE I
THE COMPLETENESS OF RNKG.

Category	Node name	Topic	Service	Message
From ROS Wiki	345	345	345	345
From RNKG	340	332	317	308
Contained Percentage	98.55%	96.23%	91.88%	89.27%

TABLE II
EXPERIMENTAL RESULTS OF ENTITY EXTRACTION.

Category	Precision	Recall	F_1
sub_topics	87.50%	87.72%	87.60%
pub_topics	87.93%	86.44%	87.17%
service	95.71%	97.14%	96.14%
service_called	89.71%	86.97%	88.31%
service_message	93.63%	88.55%	91.01%
topic_message	92.07%	94.37%	93.20%
package	93.45%	91.74%	92.58%
repository	94.31%	96.51%	95.38%

TABLE III
EXPERIMENTAL RESULTS OF RELATION EXTRACTION.

Category	Precision	Recall	F_1
(sub_topics, message_type)	69.55%	81.72%	75.14%
(pub_topics, message_type)	87.72%	86.44%	87.07%
(service, message_type)	83.80%	87.14%	85.43%
(service_called, message_type)	74.52%	86.97%	80.26%

ship in RNKG ("has_subscribed_topics, has_published_topics, has_service, has_service_called, has_package, has_repos, has_features") are accurate. But we need to specifically explain "has_message_type" in Table.III to explain some potential problems and impacts we found in verifying the correctness of the relationship. We can see that the F_1 of (sub_topics, message_type) and (service, message_type) are 87.07% and 85.43% respectively, which are relatively high. But the F_1 of (pub_topics, message_type) and (service_called, message_type) are 75.14% and 80.26% respectively, which are relatively low. The main reason is that when extracting subscribe topics and service called relationships, we need to analyze the Callback Function further. This process will cause errors in identification information and interfere with relationship extraction.

> *The ROS Node knowledge extracted by our method has more than 80% correctness and can be used to guide practice and development.*

V. Threats to validity

External validity. For the ROS-related extraction work of the dataset, we admit that our entity extraction method and relation extraction method from ROS Code has some flaws. There is still a large amount of knowledge in the code, which requires more detailed mining. Although there may be some erroneous data in the knowledge graph dataset and missing information in data extraction,we do not consider this a significant threat to external validity.

Internal validity. In the experimental verification process of RNKG, we randomly verified the completeness and correctness of ROS Node in RNKG. Additionally, we randomly selected ROS Node to evaluate the effectiveness of RNKG.

VI. Conclusion and future work

In this paper, we propose a method to construct a robotics knowledge graph RNKG with 14,065 entities and 15,767 relations for better serving ROS developers. We systematically introduce how it is automatically built from source code knowledge from GitHub and text description from ROS Wiki. We proposed an algorithm for searching ROS Node based on user description. Finally, we use three experiments to illustrate the information quality of RNKG and the possibility of implementing ROS Node search.

In the future, we plan to do in-depth follow-up research based on RNKG. And after in-depth mining of robot software project, the new data acquired by it is integrated into the constructed RNKG. We will continually enlarge RNKG to cover emerging ROS Node. On the other hand, we can better apply the reasoning ability of the knowledge map to the development tasks of robot software and use better semantic technology to support developers in searching for the ROS Node they need.

VII. Acknowledgement

This work was supported by the Key Laboratory of Software Engineering for Complex Systems and the National Science Foundation of China under granted number 62172426.

References

[1] Kai Adam, Katrin Hölldobler, Bernhard Rumpe, and Andreas Wortmann. Engineering robotics software architectures with exchangeable model transformations. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 172–179. IEEE, 2017.

[2] Sun Bo, Xinjun Mao, Shuo Yang, and Long Chen. Towards an efficient searching approach of ros message by knowledge graph. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 934–943. IEEE, 2022.

[3] Chia-Hui Chang and Shao-Chen Lui. Iepad: Information extraction based on pattern discovery. In *Proceedings of the 10th international conference on World Wide Web*, pages 681–688, 2001.

[4] Long Chen, Xinjun Mao, Yinyuan Zhang, Shuo Yang, and Shuo Wang. An efficient ros package searching approach powered by knowledge graph. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, pages 411–416, 2021.

[5] Angel Daruna, Mehul Gupta, Mohan Sridharan, and Sonia Chernova. Continual learning of knowledge graph embeddings. *IEEE Robotics and Automation Letters*, 6(2):1128–1135, 2021.

[6] Hongwei Li, Sirui Li, Jiamou Sun, Zhenchang Xing, Xin Peng, Mingwei Liu, and Xuejiao Zhao. Improving api caveats accessibility by mining api caveats knowledge graph. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 183–193. IEEE, 2018.

[7] Jinjiao Lin, Yanze Zhao, Weiyuan Huang, Chunfang Liu, and Haitao Pu. Domain knowledge graph-based research progress of knowledge representation. *Neural Computing and Applications*, 33:681–690, 2021.

[8] Shuang Liu, Hui Yang, Jiayi Li, and Simon Kolmanič. Preliminary study on the knowledge graph construction of chinese ancient history and culture. *Information*, 11(4):186, 2020.

[9] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[10] Justin J Miller. Graph database applications and concepts with neo4j. In *Proceedings of the southern association for information systems conference, Atlanta, GA, USA*, volume 2324, 2013.

[11] Maxim Mozgovoy and Roman Efimov. Wordbricks: a virtual language lab inspired by scratch environment and dependency grammars. *Human-centric Computing and Information Sciences*, 3:1–9, 2013.

[12] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[13] Xiaoxue Ren, Xinyuan Ye, Zhenchang Xing, Xin Xia, Xiwei Xu, Liming Zhu, and Jianling Sun. Api-misuse detection driven by fine-grained api-constraint knowledge graph. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 461–472, 2020.

[14] Redmond Ramin Shamshiri, Ibrahim A Hameed, Lenka Pitonakova, Cornelia Weltzien, Siva K Balasundram, Ian J Yule, Tony E Grift, and Girish Chowdhary. Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison. *International Journal of Agricultural and Biological Engineering*, 11(4):15–31, 2018.

[15] Haoze Yu, Haisheng Li, Dianhui Mao, and Qiang Cai. A relationship extraction method for domain knowledge graph construction. *World Wide Web*, 23:735–753, 2020.

[16] Haoze Yu, Haisheng Li, Dianhui Mao, and Qiang Cai. A domain knowledge graph construction method based on wikipedia. *Journal of Information Science*, 47(6):783–793, 2021.

[17] Xuejiao Zhao, Zhenchang Xing, Muhammad Ashad Kabir, Naoya Sawada, Jing Li, and Shang-Wei Lin. Hdskg: Harvesting domain specific knowledge graph from content of webpages. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 56–67, 2017.