Knowledge Graphs: Semantics, Machine Learning, and Languages
M. Acosta et al. (Eds.)

doi:10.3233/SSW230017

Semantifying the PlanQK Platform and Ecosystem for Quantum Applications

Darya MARTYNIUK [a], Naouel KARAM [a], Michael FALKENTHAL [b], Yufan DONG [a] and Adrian PASCHKE [a,c]
[a] *Data Analytics Center DANA, Fraunhofer FOKUS, Berlin, Germany*
[b] *Anaqor AG, Berlin, Germany*
[c] *Institute for Computer Science, Freie Universität Berlin, Germany*

Abstract. Quantum computing is currently experiencing rapid progress. Due to the complexity and continuous growth of knowledge in this field, it is essential to store information in a way that allows an easy access, analysis and navigation over reliable resources. Knowledge graphs (KGs) with machine-readable semantics offer a structural information representation and can enhance the capabilities of knowledge processing and information retrieval. In this paper, we extend the platform and ecosystem for quantum applications (PlanQK) for a KG. Specifically, we describe how the quantum computing knowledge, which is submitted on the platform by researchers and industry actors, is incorporated into the graph. Moreover, we outline the semantic search over the PlanQK KG.

Keywords. Knowledge Graph, Ontology, Semantic Web, Semantic Search, Faceted Search, Ontology-based Search, Quantum Computing

1. Introduction

Quantum computing has experienced notable growth in recent years leading to a corresponding increase of knowledge in this field. Access to a wide spectrum of scientific and business information, including publications, programming code, software, and hardware documentations is without doubt highly beneficial for further research and development of new technologies. However, it also means that finding relevant knowledge in quantum computing related sources has become an increasingly tedious and time-consuming task even when employing search engines. Knowledge graphs (KGs) have grasped significant potential for enhancing capabilities of search systems, facilitating more efficient information retrieval (IR) [1]. Representing data as a graph allows to define formal semantics and improves flexibility for integrating data from heterogeneous sources [1], [2].

In this paper, we introduce an approach for organizing and curating the information available on PlanQK[1] - a collaborative platform and an ecosystem centered on quantum-enhanced applications - in form of a knowledge graph. The mission of PlanQK is to make the research and industrial knowledge about quan-

[1]https://platform.planqk.de/

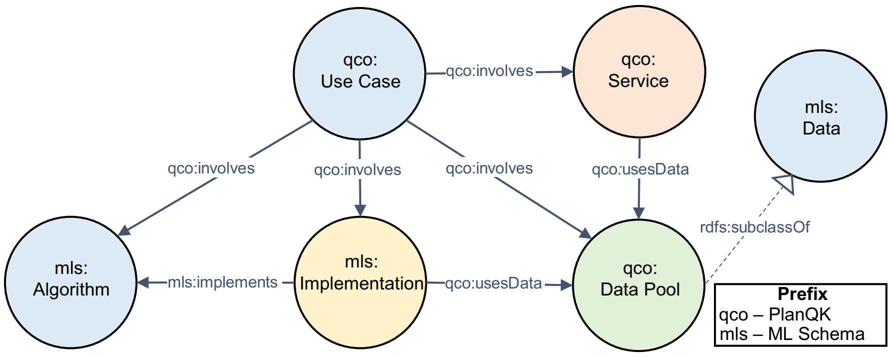


Figure 1. Core knowledge artifacts of the PlanQK ontology.

tum software and solutions easily accessible as well as to offer the capability to deploy, host, execute and monetize quantum services. We present a pipeline for a continuous semantification of the data submitted by user. Furthermore, we outline native semantic search and faceted semantic search, which leverage indexed parts of the PlanQK KG to improve the retrieval of relevant information.

This paper is structured as follows. In Section 2, we briefly introduce the PlanQK KG along with the underlying ontology and provide an overview of the semantification process of the platform data, i.e., its integration into the graph. In Section 3, we describe the realization of KG-based semantic search and semantic faceted search. We give an outlook on future work and a conclusion in Section 4.

2. PlanQK Knowledge Graph

PlanQK KG provides a structured machine-interpretable representation of the platform knowledge by organizing information into named nodes and directed edges that represent specific relations between these nodes[2]. Following the Resource Description Framework (RDF)[3] standard, each node and property is identified by a Uniform Resource Identifier (URI). To query the graph, we utilize SPARQL [3] query language, which is widely used in Semantic Web applications.

The *PlanQK Ontology* [4] serves as a formal schema of the PlanQK KG by defining selected computing concepts and relations between them in the context of the PlanQK platform. The ontology is publicly accessible to the research community including documentation and usage examples[4]. During the creation of the PlanQK ontology, we reviewed existing vocabularies that already encompass definitions of relevant concepts and reused certain ontologies, e.g., ML Schema [5], Subject Resource Application ontology (SRAO) [6], and Software Package Data

[2]Note that the term "knowledge graph" encompasses various definitions in the literature. For an overview and a comprehensive introduction of KGs we refer to [2]. In this paper, we denote a *knowledge graph* as an ontology, which defines domain knowledge along with application data integrated into the ontology, i.e., incorporated with machine-readable semantics.

[3]https://www.w3.org/TR/rdf12-concepts/

[4]https://github.com/PlanQK/semantic-services

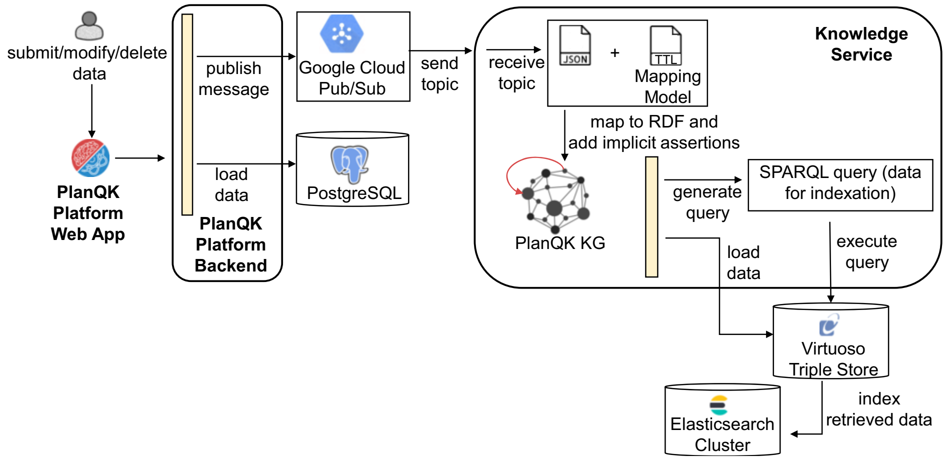


Figure 2. Overview of the platform data integration into the knowledge graph.

Exchange (SPDX) License List [7]. Figure 1 shows the core concepts of the PlanQK ontology, i.e., algorithms, implementations, use cases, data pools and services. Still the primary focus of the semantic annotations in PlanQK lies in describing quantum-related information, it nevertheless encompasses also knowledge related to the classical computing. The reason for this is twofold: Firstly, quantum concepts are often related to classical, e.g., quantum algorithms can be inspired by the idea of their classical counterparts, and, secondly, due to several limitations of the quantum devices in the Noisy Intermediate-Scale Quantum (NISQ) [8] era, quantum-enhanced applications are currently mostly developed in a hybrid manner, i.e., contains both quantum and classical parts [9], [10].

The semantification process of the PlanQK platform is depicted on Figure 2. The frontend of the platform allows users to create, modify and delete knowledge. For example, an user can create a new entry that describes an algorithm with a specific name and computation design, i.e., quantum, classical, or hybrid. The user can further modify the entry by filling various *textual attributes*, e.g., acronym, intent, and solution, and select *annotations* from the controlled vocabulary, e.g., problem class and application area. Once the data is submitted, the backend of the platform manages the information insertion into the relational database[5] and triggers a request to a cloud messaging service, which publishes the message regarding the data changes to the particular channel. To receive notifications from the channel, the knowledge service requires an active subscription. If this is the case, the service obtains a message in the JSON format. The conversion of new data from JSON into RDF is facilitated by Karma [11] integration tool. Karma provides a semi-automatic approach to define a mapping model between

[5]In the context of the PlanQK platform, the decision to use both a relational database and a knowledge graph for data storage is driven by considerations of security and component decoupling. In addition to the explicit content of the platform, the relational database contains the information about user rights. The decoupling of data storage is beneficial to facilitate the future expansion of the knowledge service for an ecosystem of similar platforms.

structured sources and KGs. Given a mapping model, an entity or a collection of similar entities stored in JSON can be dynamically transformed into RDF. However, this mapping is limited to the explicit information conveyed in the received data, i.e., the required inferences must be incorporated into the RDF code prior to its usage for the KG extension. We utilize pre-defined axioms and rules to derive implicit knowledge. An example of such inference is the assignment of a specific type to an algorithm instance, e.g., if the user submits an algorithm and specifies that it solves a classification problem, a relation *rdf:type* to the concept "Classification Algorithm", which is a subclass of "Machine Learning Algorithm", is attached. Finally, the generated RDF is used to update the instance level of the KG. Additionally, the data is indexed and inserted into a search cluster, which is queried during the content search.

We use PostgreSQL[6] as relational database and Google Cloud Pub/Sub[7] as messaging infrastructure. The knowledge service is an extension of the Terminology Service[8] (TS) [12], which was originally created for accessing, developing and reasoning of vocabularies withing the biological and environmental domains. We adjusted and extended the TS implementation to tackle PlanQK requirements, e.g., handling of instance data. The component is developed using Java and Spring Boot framework[9]. The KG is stored in Virtuoso Triple Store[10] and can be queried directly both over a SPARQL endpoint and the REST API of the knowledge service. To store the search index, we use the Elasticsearch (ES) engine[11].

3. Semantic Search

The user of the PlanQK platform has three options to retrieve the knowledge: (i) simple page navigation, (ii) semantic search, and (iii) semantic faceted search. In this section, we describe the last two alternatives.

Traditional keyword-based search approaches that solely relies on queries for literal matching of keywords can not meet the demands of knowledge retrieval since the meaning of the search term and the ambiguous nature of the natural language are not taken into account [13], [14]. The *semantic search* uses the context and the semantics of search terms for improved IR [15]. To generate relevant search results on the PlanQK platform, we index labels and descriptions of instances along with their synonyms, acronyms and broader terms extracted both from the local information, i.e., data submitted by user, and the global information, i.e., data stored in the KG including inferenced knowledge and annotations from the entire PlanQK ontology. Additionally, we employ common techniques for improving the IR such as spelling correction and stop words removal.

Initially, the user submits the search request using the platform frontend. The search term is forwarded to the API of the knowledge service, which is responsible

[6]https://www.postgresql.org/
[7]https://cloud.google.com/pubsub/docs/overview
[8]https://terminologies.gfbio.org/
[9]https://spring.io/projects/spring-boot
[10]https://virtuoso.openlinksw.com/
[11]https://www.elastic.co/de/elasticsearch/

for the retrieval of search results. The service automatically constructs a query that is then performed on an ES cluster. Thereby, the ranking of results is a part of the ES operations. Finally, the results are sent to the backend of the platform, where they undergo filtering depending on user access rights, and are displayed as snippets on the frontend. Note that the information stored in the ES cluster is filled in during the data transformation mentioned in the previous section. This is beneficial because the transformation and the search index in this scenario are generated for individual instances instead of the entire platform data.

The *faceted search* is an intuitive method of IR, where users can explore and refine search results by applying filters, or *facets*, along various dimensions [16], [17]. In our application, we distinguish between two facet types based on their functionality in querying and classifying data: (i) semantic type facets and (ii) property facets. *Semantic type facets* filter instances based on its semantic type. An example is the “problem type” facet for algorithms. The values of the facet are organized as a hierarchical tree with leafs representing the concept “Problem Type” and its subclasses, e.g., 'Machine Learning Problem' and 'Optimization Problem', as well as their subsequent subclasses. If the user selects a filter value, e.g., “optimization”, all data nodes directly connected by the property *rdf:type* with the concept 'Optimization Algorithm' or indirectly related through *rdf:type* with one of its subclasses, are retrieved. The successful retrieving of search results for these facets requires prior unique definitions of the concepts involved, e.g., the concept 'Optimization Algorithm' should be defined as 'Optimization Algorithm $\equiv$ Algorithm $\sqcap \exists$ solves.Optimization Problem'. The advantage of these facets is the ability to enhance IR with hierarchical dependencies. However, due to requirements for automatic inferences and deep graph querying, they can be time-consuming. To avoid deep queries by the extraction of broader terms, we restrict the query to the maximal depth *d*. To reduce the inference time, we deduce implicit knowledge about the type of instances, e.g., whether a specific algorithm is an optimization algorithm, during the data integration into the KG. Yet, this approach does not guarantee discovering all potential inferences related to new or modified instances. Hence, after n data insertions into the KG, we extract the graph and employ the Hermit [18] reasoner to derive additional assertions throughout the entire graph. *Property facets* rely on the range of specific relations. These facets are useful when the values are structured as a flat tree. In contract to semantic type facets, there is no need to define new concepts and corresponding definitions. An example is the “software tool” facet with values, e.g., “Qiskit”, or “Pennylane”. The results are retrieved by filtering out the instances that are connected by the relation *qco:depensOn* with the selected facet value.

Faceted search on the PlanQK platform starts with opening the advanced search interface for a specific core entity. This action triggers the generation of the facets and their possible values. Figure 3 depicts this process. To enable the construction of the SPARQL query for retrieving available facet values, we read the search configuration for the selected entity. This configuration specifies following attributes for each facet: (i) facet name, (ii) URI(s) of top node(s), and (iii) facet type (semantic type vs. property facet). As the structure of the facets depends on the underlying model, we query here the triple store directly. After the tree of facets is displayed, the user starts choosing desired filters. As shown in

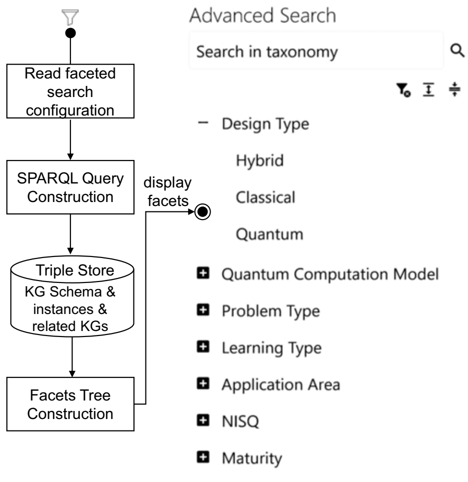
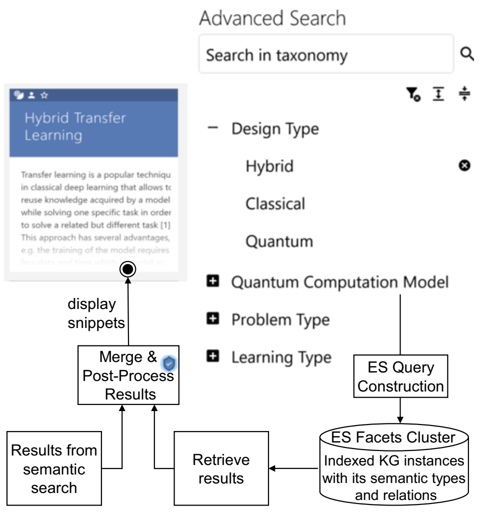


Figure 3. Generation of the facets tree.



Transfer learning is a popular techniqu
in classical deep learning that allows to
reuse knowledge acquired by a model
while solving one specific task in order
to solve a related but different task [1]
This approach has several advantages,
e.g. the training of the model requires

Figure 4. Retrieval of results in the faceted search.

Figure 4, we query the corresponding ES cluster to retrieve results, which holds only indexed search related information. In the post-processing step, the results from the faceted search (i) are merged with the result set from the semantic search (if it was executed first), and (ii) filtered to match user access rights for the content. Final results are displayed as snippets in the frontend.

4. Conclusion and Outlook

Curated KGs, which undergo a careful engineering and data integration, act as useful resources for various applications such as IR, decision support and recommendation systems. In this work, we presented the pipeline for an on-the-fly integration of the PlanQK platform data into a unified KG. The semantification process can be further improved by extracting relevant concepts and relations from textual attributes through machine learning enhanced techniques, e.g., named entity recognition. To ease IR over the platform, we implemented the native semantic search as well as the faceted semantic search. Further work is required to evaluate the search functionality and refine the search capabilities. We plan to expose a part of PlanQK KG as Linked Open Data and establish connections with related external sources such as DBPedia [19], Wikidata [20] and graphs in the research domain, e.g., Open Research Knowledge Graph (ORKG) [21]. As the knowledge in the field of quantum computing is currently changing rapidly, the possibility of the controlled graph curation by the community should be established.

Acknowledgments

This work was partially funded by the BMWK project *PlanQK (01MK20005F /01MK20005A).*

References

[1] Peng C, Xia F, Naseriparsa M, OsborneF. Knowledge Graphs: Opportunities and Challenges. Artificial Intelligence Review; 2023 Apr; p. 1 - 32.

[2] Hogan A, Blomqvist E, Cochez M, d'Amato C, de Melo G, Gutiérrez C, Kirrane S, Gayo JE, Navigli R, Neumaier S, Ngomo AN, Polleres A, Rashid SM, Rula A, Schmelzeisen L, Sequeda J, Staab S, Zimmermann A. Knowledge Graphs. ACM Computing Surveys (CSUR). 2021 July; 54(4): 1 - 37.

[3] Harris S, Seaborne A, Prud'hommeaux E. SPARQL 1.1 Query Language, W3C Recommendation. W3C Recommendation. World Wide Web Consortium [Internet]. 2013 Mar [cited 2023 May 30th]. Available from: https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[4] Martyniuk D, Falkenthal M, Karam N, Paschke A, Wild K. An analysis of ontological entities to represent knowledge on quantum computing algorithms and implementations. In: Paschke A, Rehm G, Al Qundus J, Neudecker C, Pintscher L, editors. Proceedings of the Conference on Digital Curation Technologies; 2021 Feb 8-12; Berlin, Germany.

[5] Publio GC, Esteves D, Lawrynowicz A, Panov P, Soldatova LN, Soru T, Vanschoren J, Zafar H. ML-Schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies. 2018; ArXiv, abs/1807.05351.

[6] FAIRsharing.org. SRAO, Subject Resource Application Ontology [Internet]. 2018 [cited 2023 May 30th]. Available from: https://fairsharing.org/FAIRsharing.b1xD9f.

[7] Software package data exchange SPDX License List [Internet]. 2023 [cited 2023 May 30th]. Available from: https://spdx.org/licenses/ (Accessed: 29 May 2023).

[8] Preskill J. Quantum computing in the NISQ era and beyond. Quantum. 2018 Aug; 2: 79.

[9] Endo S, Cai Z, Benjamin SC, Yuan X. Hybrid Quantum-Classical Algorithms and Quantum Error Mitigation. Journal of the Physical Society of Japan. 2021 Feb; 90 (3).

[10] Weder B, Barzen J, Beisel M., Leymann F. Provenance-Preserving Analysis and Rewrite of Quantum Workflows for Hybrid Quantum Algorithms. SN Computer Science, SCI. 2023 Feb; 4 (233).

[11] Knoblock CA, Szekely P, Ambite JL, Goel A, Gupta S, Lerman K, Muslea M, Taheriyan M, Mallick P. Semi-automatically mapping structured sources into the semantic web. In: Simperl E, Cimiano P, Polleres A, Corcho O, Presutti V, editors. Proceedings of the 9th international conference on The Semantic Web: research and applications. ESWC 2012; Heidelberg; 7295: 375–390.

[12] Karam N, Müller-Birn C, Gleisberg M, Fichtmüller D, Tolksdorf R, Güntsch A. A terminology service supporting semantic annotation, integration, discovery and analysis of interdisciplinary research data. Datenbank-Spektrum. 2016 Mar; 16: 195 – 205.

[13] Zhang X, Hou X, Chen X, Zhuang T. Ontology-based semantic retrieval for engineering domain knowledge. Neurocomputing. 2013 Sep; 116: 382-391.

[14] Sharma D, Pamula R, Chauhan DS. Semantic approaches for query expansion. Evolutionary Intelligence. 2021 Mar; 14: 1101–1116.

[15] Guha R, McCool R, Miller E. Semantic search. In: Hencsey G, White B, editors. Proceedings of the 12th international conference on World Wide Web. 2003 May; p. 700–709.

[16] Grau BC, Kharlamov E, Marciuska S, Zheleznyakov D, Arenas M: SemFacet. Faceted Search over Ontology Enhanced Knowledge Graphs, International Workshop on the Semantic Web. In: Kawamura T, Paulheim H, editors. Proceedings of the ISWC 2016 Posters and Demonstrations Track. 2016 Oct; Kobe, Japan.

[17] Heidari G, Ramadan A, Stocker M, Auer S. Demonstration of Faceted Search on Scholarly Knowledge Graphs. In: Leskovec J, Grobelnik M, Najork M, Tang J, Zia L, editors. WWWW '21: Companion Proceedings of the Web Conference. 2021 Apr; p. 685–686 .

[18] Glimm B, Horrocks I, Motik B, Stoilos G, Wang Z. HermiT: An OWL 2 Reasoner. Autom Reasoning. 2014 May; 53: 245–269.

[19] Auer S, Bizer C, Kobilarov G, Lehmann J, Cyganiak R, Ives Z. Dbpedia: a nucleus for a web of open data. In: The semantic web. 2007; p. 722–735.

[20] Vrandečić D, Krötzsch M. Wikidata: a free collaborative knowledgebase. Communications of the ACM. 2014 Sep; 57(10): 78–85.

[21] Jaradeh MY, Oelen A, Prinz M, Stocker M, Auer S. Open Research Knowledge Graph: A System Walkthrough. In: Doucet A, Isaac A, Golub K, Aalberg T, Jatowt A, editors. Digital Libraries for Open Knowledge. Lecture Notes in Computer Science, Springer; 2019; 11799.