Q^{2}Forge: Minting Competency Questions and SPARQL Queries for Question-Answering Over Knowledge Graphs

Yousouf Taghzouti, Franck Michel, Tao Jiang, Louis-Félix Nothias, Fabien Gandon

Q^{2}Forge: Minting Competency Questions and SPARQL Queries for Question-Answering Over Knowledge Graphs

Yousouf Taghzouti
yousouf.taghzouti@univ-cotedazur.fr
Univ. Côte d'Azur, Inria, ICN, I3S
Nice, France

Franck Michel
franck.michel@inria.fr
Univ. Côte d'Azur, CNRS, Inria, I3S
Nice, France

Tao Jiang
tao.jiang@cnrs.fr
Univ. Côte d'Azur, CNRS, ICN
Nice, France

Louis-Félix Nothias
louis-felix.nothias@cnrs.fr
Univ. Côte d'Azur, CNRS, ICN
Nice, France

Fabien Gandon
fabien.gandon@inria.fr
Inria, Univ. Côte d'Azur, CNRS, I3S
Nice, France

Abstract

The SPARQL query language is the standard method to access knowledge graphs (KGs). However, formulating SPARQL queries is a significant challenge for non-expert users, and remains time-consuming for the experienced ones. Best practices recommend to document KGs with competency questions and example queries to contextualise the knowledge they contain and illustrate their potential applications. In practice, however, this is either not the case or the examples are provided in limited numbers. Large Language Models (LLMs) are being used in conversational agents and are proving to be an attractive solution with a wide range of applications, from simple question-answering about common knowledge to generating code in a targeted programming language. However, training and testing these models to produce high quality SPARQL queries from natural language questions requires substantial datasets of question-query pairs. In this paper, we present Q^{2}Forge that addresses the challenge of generating new competency questions for a KG and corresponding SPARQL queries. It iteratively validates those queries with human feedback and LLM as a judge. Q^{2}Forge is open source, generic, extensible and modular, meaning that the different modules of the application (CQ generation, query generation and query refinement) can be used separately, as an integrated pipeline, or replaced by alternative services. The result is a complete pipeline from competency question formulation to query evaluation, supporting the creation of reference question-query sets for any target KG.

CCS Concepts

• **Information systems** → *Query languages*; • **Computing methodologies** → *Knowledge representation and reasoning*; **Natural language generation**; **Information extraction**.

K-CAP '25, Dayton, Ohio, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

Keywords

Competency Question, SPARQL, LLM

ACM Reference Format:
Yousouf Taghzouti, Franck Michel, Tao Jiang, Louis-Félix Nothias, and Fabien Gandon. 2025. Q^{2}Forge: Minting Competency Questions and SPARQL Queries for Question-Answering Over Knowledge Graphs. In *Proceedings of The Thirteenth International Conference on Knowledge Capture (K-CAP '25).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Semantic technologies, and in particular knowledge graphs (KGs), have been utilised in a variety of applications over time, including search engines, data integration, enterprise settings and machine learning. Numerous methods were proposed to assist their life-cycle and exploitation [12] leading to their adoption and the rapid growth of the Linked Open Data (LOD) cloud.[1] However, the exploitation of these KGs has been hindered by the steep learning curve associated with the stack of standards, in particular query languages such as SPARQL [19].

Over the last few years, common information retrieval methods have been profoundly renewed by the emergence of pre-trained Large Language Models (LLMs). The abilities of LLMs to understand and generate natural language (NL) and code alike have opened new research and development fields notably in the domain of data access and interaction. In particular, these abilities endow LLMs with the ability to translate a question expressed in NL into its counterpart in a structured query language, SPARQL in the case of RDF KGs. This allows domain experts to "speak to structured data" thus facilitating data access. To design and evaluate such text-to-SPARQL translation systems effectively, we need reference datasets providing curated question-query pairs that are either tailored to a specific KG or at least relevant for the domain it concerns.

Some question-query datasets (that we hereafter refer to as **Q^{2}sets**) have been produced in the context of benchmarks and challenges such as QALD [18], DBNQA [11], and LC-QuAD [7], but they are mostly based on subsets of DBpedia and/or Wikidata. When it comes to other domain-specific, possibly private KGs, or highly specialized KGs like in life sciences, creating a Q^{2}set involves skills that are rarely mastered by one and the same person. More likely, this requires the collaboration of domain experts who can

[1] Statistics on the Linked Open Data cloud: https://lod-cloud.net/#about

think of possibly complex competency questions (CQ) that scientists may want to ask, and Semantic Web experts who shall leverage the used ontologies and KG schema to come up with counterpart SPARQL queries.

Besides, a good practice in terms of documentation and metadata is to publish KGs with examples of queries they support. Yet, in practice this is rarely the case. Similarly, whereas CQs have been identified as a valuable documentation and starting point for understanding the capabilities of a KG, many KGs are accompanied with very few CQs, if any at all.

To support the creation of Q^2sets aimed at training, testing, benchmarking, and documenting our systems and knowledge graphs, we identified the need to provide tools that help researchers–as well as scientific and technical information professionals–to understand existing KGs and generate or refine corresponding Q^2sets, whether they are Semantic Web newcomers or experienced practitioners. Various methods and tools exist to help to create CQs and equivalent queries [1, 4, 5, 8, 15, 16, 24]. However, to the best of our knowledge, these tools are either domain-specific, extensively manual, or address only specific steps, but do not provide an end-to-end, integrated pipeline.

In this paper, we address the needs described above by presenting the methods, tools and services implemented in **Q^2Forge, a web application guiding the user through the steps of a generic, extensible, end-to-end pipeline to generate a reference Q^2set, i.e. a dataset of (NL question, SPARQL query) pairs tailored to a specific KG.** Through an interactive and iterative process, the user interface assists the user in three main areas: (1) producing CQs based on information about a KG and the domain it pertains to; (2) proposing **SPARQL query counterparts of the CQs**, given the KG and its schemata; (3) **testing** the proposed SPARQL queries, **judging** the relevance of question-query pairs, and **recommending refinements**. Rather than constraining users to a fixed end-to-end pipeline, Q^2Forge emphasizes flexibility by allowing users to use one task independently of the others.

Q^2Forge relies on an extensive, user-controlled configuration where, in particular, multiple language models can be selectively used at different steps of the pipeline. Through a documented Web API, Q^2Forge leverages a set of pre-defined services, e.g. to explore the KG or invoke a language model for a certain task, implemented using robust, community-proven libraries and frameworks such as LangChain.[2] Yet, a community may easily extend Q^2Forge with new services and steps, or re-implement some of the provided services for instance to use their own text-to-SPARQL tool instead of the one provided.

This paper is structured as follows: Section 2 provides an overview and comparison with relevant existing works. Section 3 describes our methodology, the pipeline architecture and the various components. Then, the source code and its sustainability plan are described in Section 6. Section 4 discusses real practical use cases where the resource could be used, while its potential impact and reusability are discussed in Section 5. Finally, the limitations and perspectives of the resource are outlined in Section 7.

[2]LangChain homepage: https://www.langchain.com/

2 Related Work

Linked Data Query Assistants. Approaches for assisting users in querying KGs can be broadly separated in two main non-disjoint categories: the ones relying on dedicated Graphical User Interfaces (GUI) (e.g. [2, 9, 10]) and the ones relying on Natural Language Interfaces (NLI) (e.g. [13, 17]). GUIs can provide high expressivity but remain difficult to use by non-technical experts, unless they trade off part of the expressivity in favor of reusing a popular interaction paradigm (e.g. faceted search). NLIs range from keyword-based retrieval to controlled natural language and full natural language dialogical interactions. Language models, large (LLM) and small (SLM), have significantly improved the methods for natural language processing in general, and in particular for question-answering over linked data. Language models are used in several ways: translate a question to a structured query (directly or indirectly [13]) or by directly answering the question when, for instance, the knowledge source was included in the training corpus of the language model. These two trends can be combined with augmentation techniques such as Retrieval Augmented Generation (RAG) [14] that performs information retrieval tasks of different natures (document, database, KG) to enrich the context used to invoke the language model and improve the quality of the answers. While some GUIs help lower the barrier for non-expert users, NLI approaches, and particularly those using LLMs, are even more user-friendly and extensible but have shown mixed results in generating accurate SPARQL queries from natural language, and they require reference Q^2sets to be trained, augmented and evaluated. This is precisely the purpose of Q^2Forge.

Linked Data Question-Query sets. Multiple tools and services are currently being released to address the question of text-to-SPARQL translation. BigCQ [20, 21] aims to create CQs and SPARQL equivalents based on the axioms of a specific OWL ontology. It is meant to help ontology engineering and evaluation, but cannot apply to common KGs that typically rely simultaneously on multiple ontologies and vocabularies. Amazon Bedrock[3] is a commercial, text-to-SPARQL translation service proposed by Amazon, that leverages LLMs and requires a collection of few-shot question-query pairs. It has applications particularly in bioinformatics. Unfortunately it is not released under an open source license, thus making comparison difficult. AllegroGraph's Natural Language Query (NLQ)[4] vector Database stores pairs of NL questions and corresponding SPARQL queries. This repository helps to train and refine models for accurate query generation, and its integration with SHACL shapes ensures the structural validity of the generated queries. However, this solution suffers from a lack of explainability. Users receive SPARQL query counterparts without understanding why a particular result was returned or the full process used to infer that outcome. In the opposite direction, AutoQGS [22] is a framework that generates NL questions from SPARQL queries, facilitating the creation of question-query datasets without extensive manual annotation. However, this solution requires existing SPARQL queries to generate training data, which limits its applicability.

[3]https://tinyurl.com/z4wrxvb3
[4]https://franz.com/agraph/support/documentation/nl-query.html

Challenges such as QALD [18], DBNQA [11], and LC-QuAD [7] provide Q²sets to train models that generate queries from NL questions, but they focus primarily on DBpedia and Wikidatan, although some editions of QALD, e.g. QALD 4[5], have included biomedical Q²sets. Similarly, the LC-QuAD 2.0 dataset[6] contains a Q²set of more than 20,000 pairs across DBpedia and Wikidata, including subdomains such as geography and science. While these resources serve general purpose question answering systems, they do not comprehensively capture domain specific or highly specialised knowledge. Some domain specific datasets exist, such as SIB bioinformatics SPARQL queries [3], a collection of hand-crafted Q²sets for various SIB-related KGs. By contrast, Q²Forge aims to fill this gap by providing an open-source solution to generate Q²sets for any domain and KG, including for private KGs.

3 From a Knowledge Graph to Q²Forge Pipeline

Q²Forge helps users to carry out three main tasks: generate competency questions in NL for a target KG, generate SPARQL query translations of the competency questions, and test and refine the SPARQL queries. To do so, Q²Forge orchestrates the use of various services to manage multiple per-KG configurations, extract and pre-process the schema of a KG, invoke various language models depending on the task to achieve at each step of the pipeline, etc. The services are invoked through a documented Web API implemented by a back-end server. We provide a prototype implementation of the back-end server called Gen²KGBot. A community may reuse Gen²KGBot as-is, or they may customize or extend its services to meet their specific needs. The links to the source repositories are given in Section 6.

The rest of this section further describes the steps of the Q²Forge pipeline, that are depicted in Figure 1: (1) create the configuration for a KG and (2) extract and pre-process its schema; (3) generate CQs and (4) optionally export them for reuse with another application or for documenting purpose; (5) translate a CQ into SPARQL; (6) execute the query and propose an interpretation of the results; (7) judge the relevance of the question-query pair and allow the user to iteratively refine the query; (8) export the Q²set for reuse with other systems. Note that Q²Forge remains very flexible: a user may follow the whole pipeline, but may also run each task independently by simply importing/pasting input data and exporting/copying the outputs.

3.1 KG Configuration and Pre-processing

3.1.1 Create a KG configuration. The pipeline starts with creating a KG configuration (depicted in Figure 2) where the user provides minimal information about the target KG: a name, a short name used later as an identifier, a textual description, a SPARQL endpoint URL, and the namespaces and prefixes to be used in the SPARQL queries and Turtle descriptions. Optionally, the user may fill in the URL of a SPARQL endpoint hosting the ontologies in case they are not on the same endpoint as the KG itself.

Once created, the configuration is stored on the back-end server. Additional parameters that can be edited manually to configure the available language models (seq-to-seq and embedding), where they

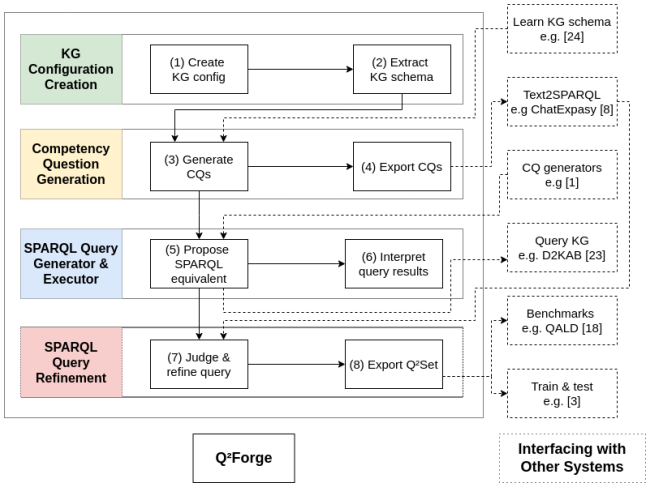

Figure 1: Q²Forge pipeline: resources and services.

are hosted (e.g. local vs. cloud resources, vector database etc.), and how they are assigned to each step of the pipeline. For instance, one may choose to use a large model with reasoning capabilities for generating a SPARQL query, but use a smaller model to interpret SPARQL results. Other parameters configure the strategy adopted to serialize ontology classes in the prompts submitted to seq-to-seq models, such as the number of ontology classes to describe and the linearization format used to describe them. Multiple formats are supported (currently Turtle, tuples or a NL format, see examples in Listing 1), since different language models may behave differently depending on the selected format.

3.1.2 KG pre-processing. There is usually a gap between how an ontology defines classes and how instances of these classes are concretely represented in the KG. Typically, instances may use properties and resources from additional vocabularies that are not explicitly mentioned in the ontology. Therefore, some downstream tasks like translating text to SPARQL and judging the relevance of a question-query pair require a description of the ontology classes as well as a description of how instances of these classes are concretely represented.

To do so, we first extract from the KG various types of information that will be helpful to carry out the downstream tasks.In our implementation, Gen²KGBot, this step creates a textual description of the classes from the labels and descriptions available in the ontologies, and computes text embeddings thereof. In Figure 2, these functions are available from the tabs 2 and 3. Second, Gen²KGBot samples class instances and analyzes the properties and value types they use (examples are provided in Listing 1). Lastly, the user may provide existing examples of NL question and associated SPARQL query. The pre-processing includes computing embeddings of these question-query pairs.

3.2 Competency Question Generation

This step invokes a language model to generate CQs based on various types of information about the KG: name and description, endpoint URL, list of the used ontologies. This information is either

[5]https://github.com/ag-sc/QALD/tree/master/4/data
[6]https://github.com/AskNowQA/LC-QuAD2.0/

```
@prefix obo: <http://purl.obolibrary.org/obo/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
obo:RO_0000056 rdfs:label "participates_in" .
[]a obo:CHEBI_53289 ;
   obo:RO_0000056 [a pubchem:MeasureGroup ];
...

('obo:CHEBI_53289', 'http://purl.obolibrary.org/obo/RO_0000056', '
      participates_in', 'http://rdf.ncbi.nlm.nih.gov/pubchem/vocabulary#
      MeasureGroup')
...

Instances of class 'obo:CHEBI_53289' have property 'obo:RO_0000056' (
      participates_in) with value type 'pubchem:MeasureGroup'.
...
```

Multiple Listing 1: Formats to describe properties and value types used by instances of a class: Turtle (top), tuple (middle), natural language (bottom)

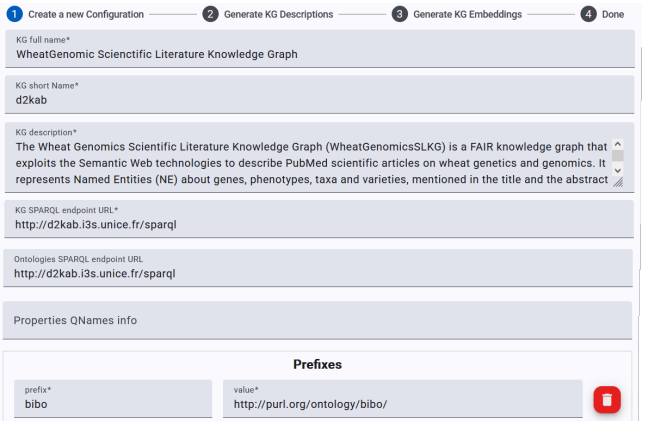

Figure 2: The KG configuration and pre-processing interface.

taken from the KG configuration (created in the previous step) or manually entered in a form. The user may also provide any other relevant information, e.g. the abstract of an article describing the KG.

Figure 3 depicts the CQ generation interface. The user can select the language model to be used, and the number of CQs to be generated. The model is instructed to return each question with an evaluation of its complexity (Basic, Intermediate or Advanced) and a set of tags. The *Enforce Structured Output* toggle can be used to compel the model to return the CQs as a JSON-formatted document.

Upon completion of the process, the user may download the output as a JSON document, and save it in a browser's cookie for reuse in the next step.

3.3 SPARQL Query Generation/Execution

In this step, the user has the ability to generate a SPARQL query counterpart of a NL question, execute it against a KG, and get an interpretation of the results. The question may originate from the preceding task, or the user may paste a question either hand-crafted or generated by another system.

Q^{2}Forge relies on various strategies provided by Gen^{2}KGBot to accomplish this task, which we refer to as "scenarios". In the

Figure 3: The competency question generation interface.

following, we focus on Scenario 5, depicted in Figure 4, that we will further describe below. When running a scenario, the steps of that scenario are progressively rendered on the interface, and for the ones that make an LLM call, the response is dynamically streamed to ensure a good user experience. Figure 5 is a snapshot of the interface of the SPARQL Query Generator/Executor. The steps are as follows:

(1) **Initial question:** the workflow is initiated by the user posing a NL question.
(2) **Question validation:** the question is evaluated by an LLM to assess the relevance of the question wrt. the description of the KG. The expected answer is boolean. If it is deemed invalid, the workflow stops.
(3) **Question pre-processing:** common techniques are used to extract named entities (NEs) from the question. SpaCy is used by default as it is widely adopted and considered production-class software. However, our implementation can support other NE extraction tools.
(4) **Select similar classes:** similarity search between the question and the ontology class descriptions computed in the KG pre-processing step is used to select relevant classes.
(5) **Get context information about the classes:** retrieve a description of the properties and value types used with instances of the selected classes.
(6) **Generate query:** generate a prompt from a template[7] using the KG configuration and the inputs from the previous steps, and submit it to the configured LLM.
(7) **Verify query and retry:** check if a SPARQL query was generated and if it is syntactically correct. If not, generate a retry prompt that includes the last generated answer and the reason for the retry, e.g. syntax errors, and submit this retry prompt to the configured LLM.
(8) **Execute the SPARQL query:** if a valid SPARQL query was generated, submit it to the KG endpoint and get the results.
(9) Use the configured LLM to **interpret the SPARQL results**.

Scenario 5, described above, is useful as a starting point when no prior question-query pair exists. However, once some pairs have been validated or if some pairs were hand-crafted, they can be added to the context and serve as examples. Scenario 6 can then be applied instead, as it provides the model with relevant example

[7]https://github.com/Wimmics/gen2kgbot/blob/master/app/scenarios/scenario_5/prompt.py#L3

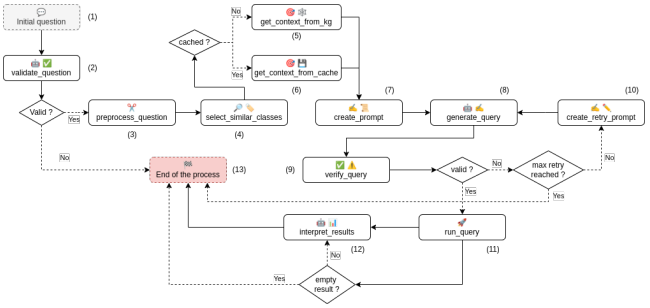

Figure 4: Gen²KGBot SPARQL Query Generator/Executor: workflow of Scenario 5

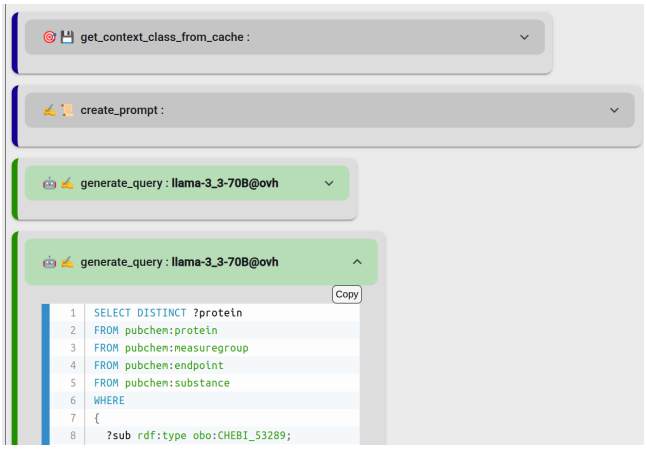

Figure 5: The SPARQL Query Generator/Executor interface.

SPARQL queries that can help in generating more accurate queries with fewer refinement iterations.

3.4 SPARQL Query Refinement

In this step, the user can incrementally refine a SPARQL query so that it reflects precisely the question. Figure 6 is a snapshot of the interface, and the process is as follows:

(1) First, the query is displayed in a SPARQL editor that highlights potential syntactic errors, and that can be used to submit the query to the endpoint.
(2) To help the user understand the query, Q²Forge can extract the qualified (prefixed) names (QNs) and fully qualified names (FQNs) from the query and get their labels and descriptions. This is particularly useful with URIs that contain a non-significant identifier. For instance, the label of http://purl.obolibrary.org/obo/CHEBI_53289 is "donepezil", and its description is "a racemate comprising equimolar amounts of (R)- and (S)-donepezil (...)".
(3) Then the LLM is asked to judge whether the query matches the given question. It is requested to provide a grade between 0 and 10 along with explanations justifying the grade.

The user may then iterate as needed: amend the query based on the grade and insights from the model, test it, have the model judge it, etc. Once a satisfying query is reached, the user can add the question-query pair to a dataset and export it in a variety of formats, catering to different use cases.

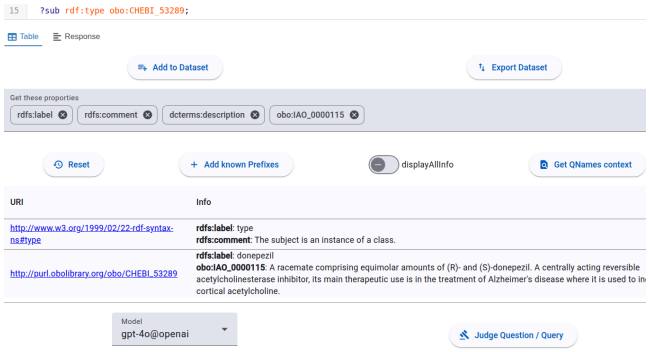

Figure 6: The query refinement interface.

4 Application Use Cases

In this section, we present several application use cases where Q²Forge can be of great help. First, we focus on the creation of Q²sets and the benefits of generating them instead of performing this task manually. Second, we examine how Q²Forge can document existing KGs with multiple competency questions. Finally, we explore its application in creating a golden dataset for benchmarking, training and testing question answering models.

Lowering the entry barrier to query rich KGs: Large public KGs usually provide lay users with user-friendly user interfaces that propose pre-defined queries and exploration options. Yet such interfaces can hardly accommodate complex custom queries where SPARQL expertise is necessary. For instance, with over 111 million chemical substances and extensive bio-activity data, PubChem presents significant navigation challenges for researchers. Metabolomics experts often struggle to formulate complex SPARQL queries that would help them identify relationships between compounds, biological activities, and disease associations. Q²Forge addressed this challenge by allowing chemists to generate natural language questions and automatically convert them to SPARQL queries, thus drastically reducing the time spent on data retrieval. To do so, researchers must provide a textual description of the KG together with additional relevant textual information, such as the abstracts of articles published about PubChem or about research made possible through PubChem. For example, a researcher might ask: "Which compounds have been tested against SARS-CoV-2 Main Protease and reported an IC50 below 1 μM?" or "Which natural product compounds from marine sponges show antimicrobial activities against *Pseudomonas aeruginosa*?". Similarly, environmental health researchers studying the exposome face difficulties extracting meaningful correlations between environmental factors and metabolic responses across heterogeneous datasets. Currently, they rely either on simple predefined queries or must collaborate with knowledge engineering specialists, creating bottlenecks in research workflows. Q²Forge could enable them to independently generate appropriate question-query pairs that bridge environmental exposures and biological outcomes, eliminating technical barriers

to knowledge discovery. A typical researcher might need to ask: "Which air pollutants are known to increase *Nrf2* anti-oxydant protein expression ?" or "What metabolic biomarkers show significant alterations following chronic exposure to per- and polyfluoroalkyl substances (PFAS) in human biomonitoring studies?"

Ground truth and question-query benchmarks: Our community has pioneered challenges and benchmarks for question-answering over linked data [7, 11, 18]. However, each edition of these challenges requires updating Q^2sets for tasks that were proposed in previous editions, and creating new Q^2sets for newly proposed tasks. There does not exist a large number of such readily available Q^2sets, and they are often based on the same KGs (e.g. DBpedia, Wikidata). Setting up a new edition of a challenge therefore requires a significant effort to generate or update the training and test data. Q^2Forge was designed to help produce these Q^2sets and can be used to facilitate the renewal of tasks for challenges and benchmarks on a variety of KGs. For instance, the QALD Challenge [18] has long been centered on DBpedia. In the latest edition, it was extended to Wikidata. Using Q^2Forge, we could extend the challenge with tasks targeting domain-specific graphs such as Uniprot [6] or the aforementioned PubChem graph.

Documenting a KG with competency questions: CQs are commonly used to demonstrate the basic capabilities of a KG. This requires working with domain experts to identify the CQs they may want to ask. In our experience, this is a time-consuming task involving multiple iterations. Q^2Forge can be used to initialize, expand and enhance the scope and variety of CQs by systematically generating hundreds of competency questions. For instance, PubChem's documentation currently provides a valuable foundation of 16 CQs.[8] Q^2Forge could enhance this foundation by showcasing the full scope and complexity of chemical, biological, and pharmacological relationships within this extensive KG. The generated question sets can serve multiple purposes: providing entry points to new users, supporting KGs indexing, benchmarking search capabilities, identifying promising research directions, and accelerating the development of next-generation retrieval systems.

5 Preliminary Experimentations

We have already identified three families of users, that correspond to the three uses cases described in Section 4: (1) the developers and maintainers of question answering systems, chatbots, conversational agents and other natural language search engines over KGs. The methods behind these systems all require to have Q^2sets to train, test and evaluate the system. They are the primary target of Q^2Forge. (2) Events and groups organizing challenges, benchmarking existing solutions and building surveys. These are in constant need of new and renewed Q^2sets to compare the latest methods and establish the state-of-the-art. Here, Q^2Forge facilitates the creation of Q^2sets from any KG in any domain. (3) While it is strongly recommended to document existing datasets and query services with examples of typical questions and queries, this is rarely done and, when it is, rarely extensive. Q^2Forge was designed to help KG publishers and maintainers to generate these examples with quality and quantity in mind.

We have initiated experiments with the first family of users on the IDSM KG related to the chemistry and metabolomics domain. Pharmaceutical researchers developing drug discovery platforms require comprehensive question-query pairs to train intelligent systems for identifying promising molecular candidates across multiple parameters. These researchers benefit from Q^2Forge's ability to generate diverse questions exploring structure-activity relationships, pharmacokinetic properties, and target binding profiles. Metabolomics data scientists integrating multi-omics datasets need sophisticated query templates that traverse complex biochemical pathway knowledge, particularly when correlating mass spectrometry findings with biological outcomes. Based on this experimentation, we believe that academic laboratories focusing on cheminformatics and bioinformatics can utilize Q^2Forge to develop educational materials demonstrating how semantic queries extract meaningful insights from chemical databases. Q^2Forge can significantly reduce technical barriers that have historically prevented domain experts from fully leveraging KG technologies in their specialized fields.

For the third family of users, we have experimented Q^2Forge with outputs of the D2KAB project. D2KAB produced several datasets among which the *Wheat Genomics Scientific Literature Knowledge Graph* [23] that represents the named entities extracted from a corpus of over 8,000 PubMed articles related to wheat genetics and genomics. The NEs include genes, phenotypes, taxon names and varieties in titles and abstracts. During the project, we worked with domain experts to figure out several CQs[9] to document the graph and illustrate its usefulness. When tested with this KG, Q^2Forge was able to automatically generate relevant CQs and translate them into SPARQL queries that were close to the target. After a short refinement step, we managed to get valid question-query pairs. Based on our initial experiments with these two domain-specific KGs, Q^2Forge shows promise for reuse across different contexts, though comprehensive evaluation remains as important future work.

Our experiments on the IDSM and D2KAB KGs allowed us to determine the time required to execute each stage of the pipeline. Table 1 summarises statistics or each KG, as well as the time taken to (1) compute classes' text embeddings, (2) generate 50 CQs, (3) answer one CQ, (4) extract QNs and FQs and obtain one refinement proposal. The experiments were performed using: an Intel Core Ultra 9 185H × 22 CPU with 64GB of RAM and an NVIDIA RTX 2000 Ada Generation Laptop GPU (8GB). We used `nomic-embed-text` embedding model and the FAISS vector store for the embedding task, and DeepSeek-v3 seq2seq model for all LLM calls.

KG	#triple	#cls	#ppt	(1)	(2)	(3)	(4)
D2KAB	27,093,602	590	287	10	132.4	45.6	29.2
IDSM	36,285,192,866	226,809	1,044	2,592.5	146.2	51	40.1

Table 1: IDSM and D2KAB: statistics and time requirements in seconds for Q^2Forge steps to execute.

[8] https://pubchem.ncbi.nlm.nih.gov/docs/rdf-use-cases

[9] https://github.com/Wimmics/WheatGenomicsSLKG/blob/main/SPARQLQueries-JupyterNotebook.ipynb

6 Source Code and Documentation

Q²Forge and accompanying Gen²KGBot integrate several software components that are robust and have been proven effective in various contexts. LangChain and LangGraph[10] are used for LLM workflow orchestration, Spacy[11] for question pre-processing and named entity recognition, rdflib[12] for the manipulation of RDF data, and YasGUI[13] as a SPARQL editor.

Source Code Availability. Q²Forge and Gen²KGBot are provided under the GNU Affero General Public License v3.0 or later (AGPL-3.0-or-later) license. The code is published on public Github repositories, and the versions used at the time of writing are identified by DOIs to ensure the long-term preservation and citability.[14] A prototype is available for public access and has been assigned a W3ID. The API provided by Gen²KGBot is documented according to the OpenAPI format.[15] Table 2 summarises the links to the source code, demonstration videos and online prototype of Q²Forge.

Sustainability Plan. Over the next four years, financial support has been secured through the MetaboLinkAI project.[16] This project aims to transform metabolomics data into actionable insights through the utilisation of AI-powered, knowledge graph-driven solutions. This will provide an opportunity to evaluate the quality, relevance and applicability of Q²Forge in the chemistry domain. Moreover, a fundamental objective of Q²Forge is to provide a generic solution, reusable with a variety of KGs. Consequently, we intend to provide support to communities expressing interest and willingness to experiment with it for their own needs. This support may range from best-effort to more formalized collaboration. To support collaborations, adoptions and contributions we secured two other contributors: the P16 public program[17] that helps open-source project improve their code and diffusion, and the Probabl company[18] whose mission is to develop, maintain the state-of-the-art, sustain, and disseminate a complete suite of open source tools for data science.

License	GNU Affero General Public License v3.0 or later (AGPL-3.0-or-later)
Online prototype	https://www.w3id.org/q2forge/
Q²Forge	**Repo** https://github.com/Wimmics/q2forge **DOI** 10.5281/zenodo.15388693
Gen²KGBot	**Repo** https://github.com/Wimmics/gen2kgbot **DOI** 10.5281/zenodo.15388687
Demo video	**Teaser** https://youtu.be/E9rgCZzWH4k **Full** https://youtu.be/I3w-jmZRJII

Table 2: Source code, demonstration videos and online prototype links of Q²Forge.

7 Conclusion and Perspectives

The present article has highlighted the challenge of creating Q²sets (datasets of question-query pairs) for a KG as part of the ever-growing LOD cloud. To address this challenge, concrete methods and tools have been presented. Utilising robust, industry-proven tools, Q²Forge's pipeline is designed to address the generation of competency questions (CQ) in NL, translate the CQs into SPARQL queries, and help users to refine those queries, and export high-quality QALD-like Q²sets that can be used for benchmarking, training and evaluating text-to-SPARQL models.

This innovative end-to-end pipeline incorporates a variety of dedicated services to achieve these objectives. It emphasizes genericity (it can apply to any KG in any domain), extensibility (the pipeline can easily be modified and extended to account for new needs), and flexibility (the various tasks of the pipeline can be executed as a whole, and some tasks can be used independently of the others). In addition to the interfacing with third-party systems shown in Figure 1, the system is designed to be reused and integrated into other scenarios. In education for instance, when teaching SPARQL, Q²Forge could be modified to serve as a tailored instructor guiding learners to navigate the complexities of SPARQL. Furthermore, since the query refinement task can be accessed independently of the other tasks (its URL takes arguments "question" and "query"), adding the appropriate button to an existing SPARQL editor could seamlessly integrate this task into an existing workflow. Furthermore, although the pipeline does not explicitly address multilingualism, multilingual CQ generation is feasible with minimal modifications (depending on the LLM's capabilities) since the configuration-driven architecture supports language-specific prompt templates. Yet, for for optimal results, ontology descriptions and metadata may need translation into the target language.

Current development of protocols such as MCP (Model Context Protocol),[19] A2A (Agent-to-Agent)[20] and hMAS (Hypermedia Multi-Agent Systems)[21] reflects ongoing efforts to simplify integration, enhance collaboration, and ensure secure and efficient communication between AI agents and external systems. These protocols can potentially be interfaced with Q²Forge to facilitate its incorporation into broader systems. In particular, we plan to implement MCP to expose the three primary functions of Q²Forge as reusable tools, enabling seamless integration of Q²Forge's components into other workflows. Conversely, Q²Forge could be extended to support the invocation of MCP servers providing access to third-party services such as knowledge graphs. Additionally, incorporating A2A would allow Q²Forge to support multi-agent collaboration across diverse ecosystems, fostering coordination between agents of varying frameworks. Finally, aligning with hMAS would leverage semantic hypermedia for uniform interactions among people, devices, and digital services, creating hybrid AI communities that operate transparently and accountably on the Web. These extensions would make Q²Forge even more versatile, facilitating the development of KG applications in different domains.

Besides the implementation of MCP mentioned above, future work will proceed along several directions in the short and medium term. First, we will focus on data quality evaluation. This includes establishing automated validation processes and human evaluation protocols to ensure the relevance and accuracy of generated question-query pairs. Second, we envision integrating existing KG construction and extension tools from unstructured content. Such integration would enable Q²Forge to generate questions about both

[10] https://www.langchain.com/
[11] https://spacy.io/
[12] https://github.com/RDFLib
[13] https://yasgui.triply.cc/
[14] All links are given at the beginning of this article.
[15] http://w3id.org/q2forge/api/docs/
[16] http://www.metabolinkai.net/
[17] https://p16.inria.fr/en/
[18] https://probabl.ai/about
[19] https://modelcontextprotocol.io/
[20] https://google.github.io/A2A/
[21] https://project.hyperagents.org/

structured and unstructured data sources, broadening its applicability across diverse data landscapes. Third, we wish to address several current limitations: (1) Follow-up questions: our plan is to achieve this using short/long memory and conversation summarizing to cope with context explosion; (2) custom and tailor result visualization when a textual rendering of a SPARQL query result is not the optimum way of conveying the results; (3) Federated queries: the current implementation is limited to single SPARQL endpoints and does not support federated queries across multiple KGs, which restricts its applicability in scenarios requiring data integration from diverse sources. This limitation could be addressed by extending Q^{2}Forge architecture to support the generation of queries that span multiple endpoints using the SERVICE clause of the SPARQL 1.1 Federation specification. Finally, extensive dissemination and open-source support activities are planned so that other communities can adopt this work and adapt it to their specific needs. Through community engagement, documentation, and targeted outreach, we aim to foster broader adoption and enable customization of Q^{2}Forge across various domains and research contexts.

Acknowledgments

This work was supported by the French government through the France 2030 investment plan managed by the National Research Agency (ANR), as part of the Initiative of Excellence Université Côte d'Azur (ANR-15-IDEX-01). Additional support came from French Government's France 2030 investment plan (ANR-22-CPJ2-0048-01), through 3IA Cote d'Azur (ANR-23-IACL-0001) as well as the MetaboLinkAI bilateral project (ANR-24-CE93-0012-01 and SNSF 10002786).

References

[1] Reham Alharbi, Valentina Tamma, Floriana Grasso, and Terry R. Payne. 2025. The Role of Generative AI in Competency Question Retrofitting. In *The Semantic Web: ESWC 2024 Satellite Events: Hersonissos, Crete, Greece, May 26–30, 2024, Proceedings, Part I*. Springer-Verlag, 3–13.

[2] Hannah Bast, Johannes Kalmbach, Theresa Klumpp, Florian Kramer, and Niklas Schnelle. 2022. Efficient and Effective SPARQL Autocompletion on Very Large Knowledge Graphs *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 2893–2902.

[3] Jerven Bolleman, Vincent Emonet, Adrian Altenhoff, Amos Bairoch, Marie-Claude Blatter, Alan Bridge, Severine Duvaud, Elisabeth Gasteiger, Dmitry Kuznetsov, Sebastien Moretti, Pierre-Andre Michel, Anne Morgat, Marco Pagni, Nicole Redaschi, Monique Zahn-Zabal, Tarcisio Mendes de Farias, and Ana Claudia Sima. 2024. A large collection of bioinformatics question-query pairs over federated knowledge graphs: methodology and applications.

[4] Fiorela Ciroku, Jacopo de Berardinis, Jongmo Kim, Albert Meroño-Peñuela, Valentina Presutti, and Elena Simperl. 2024. RevOnt: Reverse Engineering of Competency Questions from Knowledge Graphs via Language Models. *Web Semant.* 82, C (Oct. 2024).

[5] K. Bretonnel Cohen and Jin-Dong Kim. 2013. Evaluation of SPARQL Query Generation from Natural Language Questions. *Proceedings of the Joint Workshop on NLP&LOD and SWAIE: Semantic Web, Linked Open Data and Information Extraction* 2013 (Sept. 2013), 3–7.

[6] The UniProt Consortium. 2018. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research* 47, D1 (11 2018), D506–D515.

[7] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia. In *The Semantic Web – ISWC 2019*. Springer International Publishing, Cham, 69–78.

[8] Vincent Emonet, Jerven Bolleman, Severine Duvaud, Tarcisio Mendes de Farias, and Ana Claudia Sima. 2024. LLM-based SPARQL Query Generation from Natural Language over Federated Knowledge Graphs. arXiv:2410.06062 [cs.DB]

[9] Sébastien Ferré. 2017. SPARKLIS: An Expressive Query Builder for SPARQL Endpoints with Guidance in Natural Language. *Open Journal Of Semantic Web* 0 (2017).

[10] Thomas Francart. 2023. Sparnatural: A Visual Knowledge Graph Exploration Tool. In *The Semantic Web: ESWC 2023 Satellite Events*, Catia Pesquita, Hala Skaf-Molli, Vasilis Efthymiou, Sabrina Kirrane, Axel Ngonga, Diego Collarana, Renato Cerqueira, Mehwish Alam, Cassia Trojahn, and Sven Hertling (Eds.). Springer Nature Switzerland, Cham, 11–15.

[11] Ann-Kathrin Hartmann, Tommaso Soru, and Edgard Marx. 2018. Generating a Large Dataset for Neural Question Answering over the DBpedia Knowledge Base. In *Workshop on Linked Data Management, co-located with the W3C WEBBR 2018*.

[12] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. *ACM Computing Surveys (Csur)* 54, 4 (2021), 1–37.

[13] Jens Lehmann, Preetam Gattogi, Dhananjay Bhandiwad, Sébastien Ferré, and Sahar Vahdati. 2023. Language Models as Controlled Natural Language Semantic Parsers for Knowledge Graph Question Answering. In *Frontiers in Artificial Intelligence and Applications (Frontiers in Artificial Intelligence and Applications, Vol. 372)*. IOS Press, Krakow (Cracovie), Poland, 1348–1356.

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 9459–9474.

[15] Xueli Pan, Jacco van Ossenbruggen, Victor de Boer, and Zhisheng Huang. 2025. A RAG Approach for Generating Competency Questions in Ontology Engineering. In *Metadata and Semantic Research*, Michalis Sfakakis, Emmanouel Garoufallou, Matthew Damigos, Athena Salaba, and Christos Papatheodorou (Eds.). Springer Nature Switzerland, 70–81.

[16] Youssra Rebboud, Lionel Tailhardat, Pasquale Lisena, and Raphael Troncy. 2025. Can LLMs Generate Competency Questions?. In *The Semantic Web: ESWC 2024 Satellite Events*, Albert Meroño Peñuela, Oscar Corcho, Paul Groth, Elena Simperl, Valentina Tamma, Andrea Giovanni Nuzzolese, Maria Poveda-Villalón, Marta Sabou, Valentina Presutti, Irene Celino, Artem Revenko, Joe Raad, Bruno Sartini, and Pasquale Lisena (Eds.). Springer Nature Switzerland, 71–80.

[17] Emma Tysinger, Marco Pagni, Olivier Kirchhoffer, Florence Mehl, Fabien Gandon, Jean-Luc Wolfender, and Louis-Félix Nothias. 2023. An Artificial Intelligence Agent for Navigating Knowledge Graph Experimental Metabolomics Data. In *Frontiers in Metabolomics - Book of abstract - (Frontiers in Metabolomics)*. Swiss Metabolomics Socitety Zurich and ETH Zurich, Zurich, Switzerland.

[18] Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, Muhammad Saleem, and Andreas Both. 2024. QALD-10 – The 10th challenge on question answering over linked data: Shifting from DBpedia to Wikidata as a KG for KGQA. *Semantic Web* 15, 6 (2024), 2193–2207.

[19] Paul Warren and Paul Mulholland. 2020. A Comparison of the Cognitive Difficulties Posed by SPARQL Query Constructs. In *Knowledge Engineering and Knowledge Management*, C. Maria Keet and Michel Dumontier (Eds.). Springer International Publishing, 3–19.

[20] Dawid Wiśniewski, Jędrzej Potoniec, and Agnieszka Ławrynowicz. 2022. BigCQ: Generating a Synthetic Set of Competency Questions Formalized into SPARQL-OWL (Student Abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 13079–13080.

[21] Dawid Wiśniewski, Jędrzej Potoniec, and Agnieszka Ławrynowicz. 2021. BigCQ: A large-scale synthetic dataset of competency question patterns formalized into SPARQL-OWL query templates. arXiv:2105.09574 [cs.AI]

[22] Guanming Xiong, Junwei Bao, Wen Zhao, Youzheng Wu, and Xiaodong He. 2022. AutoQGS: Auto-Prompt for Low-Resource Knowledge-based Question Generation from SPARQL. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 2250–2259.

[23] Nadia Yacoubi Ayadi, Stephan Bernard, Robert Bossy, Marine Courtin, Bill Gates Happi Happi, Pierre Larmande, Franck Michel, Claire Nedellec, Catherine Roussey, and Catherine Faron. 2024. A Unified approach to publish semantic annotations of agricultural documents as knowledge graphs. *Smart Agricultural Technology* 8 (Jan. 2024), 43.

[24] Amal Zouaq and Felix Martel. 2020. What Is the Schema of Your Knowledge Graph? Leveraging Knowledge Graph Embeddings and Clustering for Expressive Taxonomy Learning. In *Proceedings of The International Workshop on Semantic Big Data (SBD '20)*. Association for Computing Machinery, 1–6.