Ensuring License Compliance in Linked Data with Query Relaxation

Benjamin Moreau, Patricia Serrano-Alvarado

Ensuring License Compliance in Linked Data with Query Relaxation

Benjamin Moreau[1] and Patricia Serrano-Alvarado[2]

[1] OpenDataSoft {Name.Lastname}@opendatasoft.com
[2] Nantes University, LS2N, CNRS, UMR6004, 44000 Nantes, France
{Name.LastName@}univ-nantes.fr

Abstract. When two or more licensed datasets participate in evaluating a federated query, to be reusable, the query result must be protected by a license compliant with each license of the involved datasets. Due to incompatibilities or contradictions among licenses, such a license does not always exist, leading to a query result that cannot be licensed nor reused on a legal basis. We propose to deal with this issue during the federated query processing by dynamically discarding datasets of conflicting licenses. However, this solution may generate an empty query result. To face this problem, we use query relaxation techniques. Our problem statement is, *given a SPARQL query and a federation of licensed datasets, how to guarantee a relevant and non-empty query result whose license is compliant with each license of involved datasets?* To detect and prevent license conflicts, we propose FLiQue, a license-aware query processing strategy for federated query engines. Our challenge is to limit communication costs when the query relaxation process is necessary. Experiments show that FLiQue guarantees license compliance, and if necessary, can find relevant relaxed federated queries with a limited overhead in terms of execution time.

Keywords: Linked Data, federated queries, licenses, query relaxation, compatibility of licenses.

1 Introduction and Motivation

The Linked Data is a network of distributed and interlinked data sources. Federated query processing allows to query such a network of live and up-to-date datasets. A *federated SPARQL query* can retrieve information from several RDF data sources distributed across the Linked Data. Since the beginning of Linked Data, licensing has been an important issue [1, 36]. To legally facilitate reuse, data owners should systematically associate licenses with resources before sharing or publishing them. There are still several open issues to legally access and reuse linked data, as shown in recent Dagstuhl seminars [2, 12] and surveys [14].

When two or more licensed datasets participate in evaluating of a federated query, the query result must be protected by a license that is compliant with each license of involved datasets. Licenses specify precisely the conditions of reuse of

data, i.e., what actions are permitted, obliged, and prohibited. Machine-readable licenses are necessary to ensure automatic license compliance. The W3C Open Digital Rights Language (ODRL) [17] allows defining machine-readable licenses. The Data Licenses Clearance Center (DALICC) [25], proposes a library of well-known standard machine-readable licenses.

We consider that a license l_j is compliant with a license l_i if a resource licensed under l_i can be licensed under l_j without violating l_i. If l_j is compliant with l_i, then l_i is compatible with l_j. Unfortunately, it is not always possible to find a license compliant with each license of datasets involved in a federated query [22]. If such a license does not exist, the query result cannot be licensed and, thus, should not be reused nor published.

We consider that a query whose result set cannot be licensed should not be executed. Notice that having the rights to query several datasets individually does not mean having the rights to execute a federated query involving these datasets.

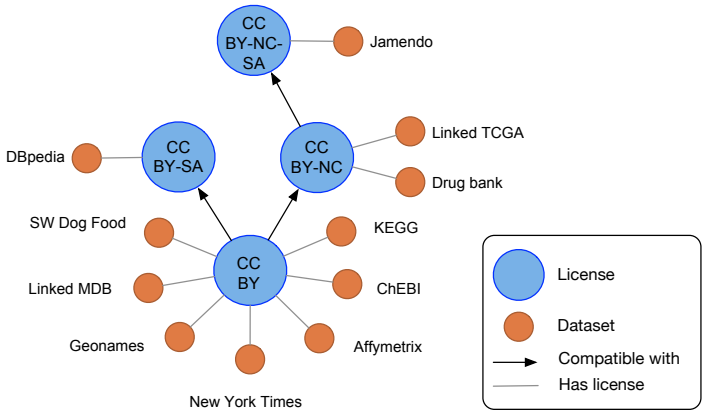

Fig. 1: The compatibility graph of licenses for datasets of LargeRDFBench.

Consider datasets of LargeRDFBench [32], a benchmark for federated query processing. Figure 1 shows the compatibility graph of licenses[3] that protect LargeRDFBench datasets. By transitivity, license CC BY is compatible with itself, with CC BY-SA, CC BY-NC, and CC BY-NC-SA. Thus, datasets protected by CC BY can be queried along with other datasets protected by these licenses. However, the whole set of datasets of Figure 1 cannot be queried together because there is no license compliant with the fourth licenses. For instance, there is no license with which CC BY-SA and CC BY-NC-SA are both compatible.

[3] This compatibility graph conforms to the license compatibility chart shown in `https://wiki.creativecommons.org/wiki/Wiki/cc_license_compatibility`.

One solution to the incompatibility of licenses is negotiating with data providers to change a conflicting license, e.g., to ask DBpedia to change its license to CC BY or CC BY-NC. Nevertheless, negotiation takes time and is not always possible. A second solution is to discard datasets that are protected by conflicting licenses. However, this solution can lead to a query with an empty result set. To face this problem, we use query relaxation techniques. That is, we use *relaxation rules* to relax the query constraints to match triples of other datasets.

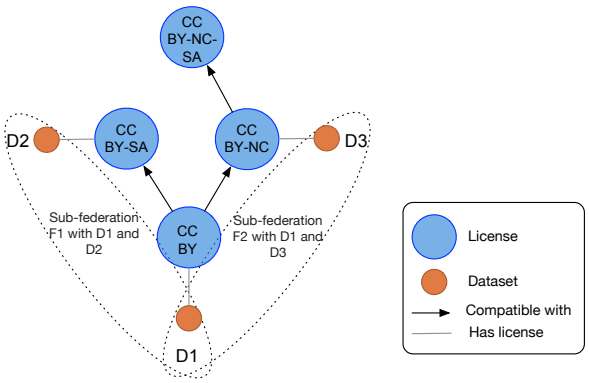

Fig. 2: Compatibility graph of licenses for datasets D1, D2, and D3. As there is no license compatible with licenses of these datasets, sub-federations F1 and F2 should be created. Queries (or relaxed queries) can be evaluated over these sub-federations to produce licensable results.

```
SELECT ?student WHERE {
  ?student  rdf:type  ex:Student .                  #tp1@{D3} CC BY-NC
  ?student  ex:enrolledIn  ?course .                #tp2@{D3} CC BY-NC
  ?course   ex:heldAt  ex:UniversityOfNantes .      #tp3@{D1} CC BY
  ex:Ben    ex:teaches  ?course .                   #tp4@{D2} CC BY-SA
}
```

Listing 1.1: A SPARQL query Q.

Consider the compatibility graph of licenses for datasets D1, D2, and D3 of Figure 2[4] and the federated query Q of Listing 1.1, which asks for students enrolled in a course held at the University of Nantes taught by *ex:Ben*. The query is annotated with the datasets over which each triple pattern can be evaluated. $D2$ and $D3$ should not be queried together because their licenses are respectively CC BY-SA and CC BY-NC, and there is no license compliant with both. Thus,

[4] To simplify, we show the same licenses as in Figure 1. However, a compatibility graph of licenses can contain many more licenses and not limited to Creative Commons ones.

the result set of the query cannot be licensed. Creating a sub-federation of sources with compatible licenses without $D2$ makes the result set licensable with CC BY-NC or with another license compliant with CC BY-NC (e.g., CC BY-NC-SA). The problem is that the query gives no result because there is no more dataset to evaluate $tp4$. The case is similar with a sub-federation with compatible licenses discarding $D3$ because there is no more dataset to evaluate $tp1$ and $tp2$.

As none subset of licence compatible sources can produce a non-empty result set for Q, we propose using query relaxation techniques. For instance, in such relaxation, instead of asking for students in $tp1$, the federated query could ask for persons, or instead of asking for courses taught by *ex:Ben* in $tp4$, the federated query could ask for courses taught by anybody. The number of possible relaxed queries can be huge. To find the most relevant relaxed federated queries efficiently, we use approaches that compute relaxed queries from the most to the least *similar* to the original query [7, 8, 15, 16]. Though, the most similar queries may produce no results. In a distributed environment, verifying each relaxed federated query is not feasible. So the challenge is to find the most similar relaxed queries that return a non-empty result from datasets with compatible licenses while limiting communication costs.

Our research question is, *given a SPARQL query and a federation of licensed datasets, how to guarantee a relevant and non-empty query result whose license is compliant with each license of involved datasets?* The challenge is to limit the communication cost when the relaxation process is necessary.

We propose FLiQue[5], a Federated License-aware Query processing strategy. FLiQue is designed to detect and prevent license conflicts and gives informed feedback with licenses able to protect a result set of a federated query. If necessary, it applies distributed query relaxation to propose a set of most similar relaxed federated queries whose result set can be licensed. Our solution combines existing approaches, namely, CaLi [22] to maintain a compatibility graph of licenses dynamically and to detect license conflicts, OMBS [8] to efficiently find relaxed queries and data summaries of CostFed [34] to limit communication overhead during the federated query relaxation. Our contributions are:

- an efficient license-aware federated query processing strategy able to relax federated queries,
- an implementation of a license-aware federated query engine, and
- an experimental evaluation of our solution.

In the next, Section 2 overviews related works, Section 3 introduces FLiQue, Section 4 shows experimental results, and Section 5 concludes.

2 Related Work

To our knowledge, there is no federated query engine that ensures license compliance with all licenses involved in query execution.

[5] In French, FLiQue is a homophone of *flic*, which means *cop*.

Many works focus on access control over linked data using policy-based [5, 19, 20, 28], view-based [9], or query-rewriting [24] approaches. In these works, datasets are protected by *access control rules* that prevent non-authorized users from querying data of each dataset. These approaches do not resolve our problem statement because having the right to query datasets individually does not mean that it should be possible to execute a federated query involving these datasets.

2.1 Compatibility graph of licenses

To know if a result set can be licensed, we need to know the license(s) with whom all licenses of datasets involved in a federated query are compatible. Automatic license compatibility requires machine-readable licenses. License expression languages such as CC REL[6], ODRL, or L4LOD[7] enable fine-grained RDF description of licenses. Works like [31] and [3] use natural language processing to automatically generate RDF licenses from licenses described in natural language. Other works such as [13, 30] propose a set of well-known licenses in RDF described in CC REL and ODRL[8]. Thereby, we suppose that there exist consistent licenses described in RDF.

A compatibility graph of licenses contains a set of licenses partially ordered by compatibility. It can be defined by hand using, for instance, the license compatibility chart of Creative Commons. But licenses used in the Linked Data are not limited to Creative Commons licenses. For instance, the license library of DALICC has at least 60 licenses[9].

Works like [37] address the problem of license compatibility and license combination. If licenses are compatible, a new license compliant with combined ones is generated. This approach allows defining the compatibility graph of licenses progressively. However, it does not allow us to know all the compliant licenses that can be used to protect a query result set. That is, all licenses that are compliant to all licenses involved in a query.

In the context of Free Open Source Software (FOSS), [18, 38] propose compatibility graphs of well-known licenses. Based on a directed acyclic graph, they propose to detect license violations in existing software packages. They consider that license l_i is compatible with l_j if the graph contains a path from l_i to l_j. The combined software can be protected by the license l_j, possibly with additions from l_i. However, as such a graph is built from a manual interpretation of each license, its generalisation and automation is not possible. In particular, with these approaches it is not possible to add automatically a new license to the compatibility graph of licenses.

CaLi [21, 22], is a lattice-based model for license orderings. It automatically positions a license over a set of licenses in terms of compatibility and compliance.

[6] `https://creativecommons.org/ns`
[7] `https://ns.inria.fr/l4lod/`
[8] Creative Commons also proposes its licences in RDF `https://github.com/creativecommons/cc.licenserdf/tree/master/cc/licenserdf/licenses`
[9] `https://www.dalicc.net/license-library`

The originality of CaLi is to pass through a restrictiveness relation to partially order[10] licenses in terms of compatibility and compliance. In a license, actions (e.g., *read, modify, distribute*, etc.) can be distributed in *status* (e.g., *permissions, duties,* and *prohibitions*). To decide if a license l_i is less restrictive than l_j, it is necessary to know if an action in a status is considered less restrictive than the same action in another status. l_i is said to be less restrictive than l_j if for all actions $a \in A$, the status of a in l_i is less restrictive than the status of a in l_j. The restrictiveness relation between licenses can be obtained automatically, according to the status of actions. Thus, based on lattice-ordered sets, CaLi defines a restrictiveness relation among licenses. If two licenses have a restrictiveness relation, then they may have a compatibility relation too.

To identify the compatibility among licenses, CaLi refines the restrictiveness relation with two types of constraints (license constraints and compatibility constraints). The goal is to take into account the semantics of actions.

- *License constraints* allow to identify non-valid licenses and can be defined as a set of conditions. A condition of a license constraint defines if an action should exist or not in a status. For instance the condition (*cc:CommercialUse* $\notin$ *Duty*) means that a valid license should not have the action *cc:CommercialUse* as a duty. The condition (*cc:ShareAlike* $\notin$ *Prohibition*) means that a valid license should not prohibit the *cc:ShareAlike* action.
- A *compatibility constraint* concerns two licenses where one is more restrictive than another. For instance, consider the action *cc:ShareAlike* which requires that the distribution of derivative works be under the same license only. The compatibility constraint (*cc:ShareAlike* $\notin$ *Duty*) $\notin l_i$ means that l_i is compatible with l_j if the action *cc:ShareAlike* is not a duty in l_i. In another example, the compatibility constraint (*cc:DerivativeWorks* $\notin$ *Prohibition*) $\notin l_i$ means that l_i is compatible with l_j if l_i does not prohibit the distribution of a derivative resource, regardless of the license.

CaLi is able to define all the licenses that can be expressed with a set of actions over a lattice of status[11]. For instance, the CaLi ordering for the set of 7 actions used by Creative Commons has 972 licenses[12]. CaLi can provide all the licenses that can protect a result set ordered by restrictiveness. It can also identify which licenses are in conflict. Knowing the compatibility of a license allows estimating the reusability of the protected resource. On the other hand, knowing the compliance of a license allows knowing to which extent other licensed resources can be reused[13].

[10] A partial order is any binary relation that is reflexive, antisymmetric, and transitive.

[11] A demonstration tool to define, step by step, a compatibility graph of licenses with the CaLi approach can be found here `https://saas.ls2n.fr/cali/`.

[12] That is $|S|^{|A|}$ minus the licenses discarded by constraints, where S is a set of status and A a set of actions. The three status considered by Creative Commons licenses are: *permissions, duties,* and *prohibitions*.

[13] Next compatibility graphs of licenses illustrate the CaLi approach: `http://cali.priloo.univ-nantes.fr/ld/graph`, `http://cali.priloo.univ-nantes.fr/rep/graph`

In this work, we use CaLi to verify license compliance thanks to its facilities to add dynamically new licenses to the compatibility graph of licenses.

When the result set of a federated query cannot be licensed, we propose to define sub-federations that avoid license conflicts. If there is no sub-federation able to produce a licensable and non-empty result set for the user query, we propose alternative queries through query relaxation.

2.2 Query relaxation

Our goal is to provide users with a means to automatically identify new queries that are similar to the user query and whose result set can be licensed.

The OPTIONAL clause of SPARQL[14] was defined to allow query users to add supplementary information to a query solution. That is, if available, the solution of optional triple patterns is added to the query solution. This clause allows users to relax some query's conditions because triple patterns defined as optional extend the query result of the non-optional triple patterns. As this clause is not added dynamically (query users define which triple patterns are OPTIONAL), it cannot be useful when a license compatibility problem arises during the source selection phase of the query execution.

Query relaxation techniques are used to provide an alternative for queries producing no result. Transformations are applied to user queries to relax constraints in order to generalize the query so that it can produce more answers. The solution espace grows in a combinatorial way with the number of relaxation steps and the size of the query. Existing works propose techniques to reduce this espace producing the most *similar* relaxed queries. We are interested in query relaxation techniques that could be used to relax efficiently a federated query.

There are works that focus on symbolic forms of semantic similarity that can be represented with graph patterns [7]. They identify similar entities based on common graph patterns. Their approach is a symbolic form of the k-nearest neighbours where numerical distances are replaced by graph patterns that provide an intelligible representation of how similar two nodes are. The drawback of this approach is that it needs all possible answers of relaxed queries (*i.e.*, the entities). As we do not consider a centralized RDF graph with all datasources, this solution applied to a distributed environnement would be very expensive in communication and execution time.

Other works focus on ontology-based similarity measures to retrieve additional answers of possible relevance [8, 15, 16]. They use logical relaxation of the query conditions based on RDFS entailment and RDFS ontologies.

[16] proposes a RELAX clause as a generalization of the OPTIONAL clause for the conjunctive part of a SPARQL query. Their goal is to relax the query without simply dropping the optional triple pattern. The idea consists of relaxation rules that use information from the ontology; these include relaxing a class to its super-class, relaxing a property to its super-property, etc. Other relaxations can be entailed without an ontology, which include replacing constants

[14] `https://www.w3.org/TR/rdf-sparql-query/#OptionalMatching`

with variables, suppressing join dependencies and dropping triple patterns. All possible relaxed queries are organized in a lattice called *relaxation graph*. They propose to rank the results of a query based on how closely they satisfy the query. Their ranking is based on the relaxation graph, in which relaxed versions of the original query are ordered from less to more general from a logical standpoint. Given two relaxed queries of the user query, if one is subsumed by the other, then the former relaxed query is better. Subsumption is based on the hypothesis that if c_1 is a subclass of c_2, c_2 is the class that subsumes c_1 and c_2 (idem for subproperties). The size of the relaxation graph grows combinatorially with the number of relaxation rules, the richness of the ontology, and the relaxation possibilities of each triple pattern in the original query.

[8, 15] focus on obtaining a certain number of alternative results (top-k) by relaxing a query that produces no results. Their challenge is to execute as less as possible relaxed queries to obtain the top-k results. Relaxed queries are executed in a similarity-based rank order to avoid executing all relaxed queries in the relaxation graph. *Information content* [29] is used to measure the *similarity* between a relaxed query and the original query. That is, statistical information about the concerned dataset, like the number of entities per class and the number of triples per property. Nevertheless, the number of failing relaxed queries executed before obtaining the top-k results can be large. Thus, it is necessary to identify unnecessary relaxations that do not generate new answers. Relaxed queries containing unnecessary relaxation should not be executed.

[15] proposes OBFS (Optimized Best First Search algorithm) to identify unnecessary relaxations in a similarity-based relaxation graph. It is based on the *selectivity* of relaxations using the number of entities per class or the number of triples per property. If the selectivity is the same before and after the relaxation, the relaxation is considered unnecessary. That is, if the number of entities of a class is equal to the number of entities of its super-class, then the class relaxation does not generate new answers. The same idea is used for property relaxation.

[8] proposes OMBS (Optimized Minimal-failing-sub-queries-Based Search algorithm) as an improvement to OBFS. The contribution of OMBS is to identify the minimal sets of triple patterns in failing queries that fail to return answers. These failing sets of triple patterns are called Minimal Failing Sub-queries (MFS). MFS existing in a query must be relaxed, otherwise, the query fails in producing results. Relaxed queries where the MFS are not relaxed are considered unnecessary. OMBS defines optimal similarity-based relaxation graphs where relaxed queries producing no results (based on MFS), or not new results (based on selectivity) are not executed.

Table 1 overviews these three approaches from four dimensions: (1) RDFS properties used, (2) similarity definition, (3) information needed for the query relaxation process and (4) methods used to prune the relaxation graph.

1. The **RDFS properties** used in the relaxation rules proposed by [16], concern the domain, the range, the subproperties and the subclasses of a class used in a triple pattern. [15] and [8], base their relaxation rules only on the subproperties and subclasses of a class used in a triple pattern.

	RDFS properties	Similarity definition	Information needed	Pruning method
[16]	*rdfs:domain* *rdfs:range* *rdfs:subPropertyOf* *rdfs:subClassOf*	Subsumption relation among relaxed queries	Ontology	Discards indirect ontology relations
[15]	*rdfs:subPropertyOf* *rdfs:subClassOf*	Based on information content measures between the original query and the relaxed queries	Ontology and dataset statistics	Based on the number of instances Based on join dependency
[8]	*rdfs:subPropertyOf* *rdfs:subClassOf*	Based on information content measures between the original query and the relaxed queries	Ontology and dataset statistics	Based on failure causes of triple patterns

Table 1: An analysis of existing approaches about ontology-based query relaxation.

2. The **similarity definition** of [16], is only based on the relations of the relaxation graph, where relaxed queries are ordered based on the subsumption relation of their result sets. This partial order makes that it is not always possible to compare two relaxed queries. [15] and [8] use information content measures to obtain a total order between the original query and the relaxed queries. Such similarity is computed using statistical information of the concerned dataset, e.g., the number of entities per class and the number of triples per property.
3. Concerning the **information needed**, these works use the dataset ontology. [15] and [8] also use dataset statistics to calculate the information content of relaxed queries.
4. For **pruning the relaxation graph**, [16] avoids redundant query relaxation by reducing ontologies, e.g., it uses the ontology without saturation. [15] uses the number of entities per classes and the number of triples per properties to identify query relaxation that does not generate new results. Finally, [8] identifies the set of triple patterns that fail to return results. Thus, it only keeps relaxed queries that do not contain failing triple patterns.

We consider that these three approaches [8,15,16] can be used to relax federated queries because they can produce relaxed queries using the dataset ontology and statistical information (to calculate selectivity) without needing the RDF graph (i.e., the instances). We use OMBS to find relaxed queries because it optimizes obtaining the number of relaxed queries that potentially give non-empty results.

2.3 Data summaries

In this work, we need *data summaries* to calculate similarity measures (based on information content), selectivity, but also to limit the communication overhead, during the *distributed query relaxation* process. A very complet survey of the state-of-the about summarization methods for semantic RDF graphs is proposed in [4]. In federated query processing, some federated query engines, use statistics to reduce the number of requests sent to data sources during the source selection and the query optimization steps [10, 27, 33, 34]. Analysis of state-of-the-art federated query engines can be found in [23, 26, 32].

VoID[15], is the Vocabulary of Interlinked Datasets [6]. It allows to formally describe linked RDF datasets with metadata like contact, topic, licenses, SPARQL endpoint, url of data dumps, basic statistics, etc. In particular, with VoID descriptions is possible to describe dataset instances, i.e., the number of instances of a given class and the number of triples that have a certain predicate.

DARQ [27] (from Distributed ARQ)[16] is the first query engine that allows querying multiple, distributed SPARQL endpoints. DARQ introduced service descriptions which provide a declarative description of the data available from an SPARQL endpoint[17]. Service descriptions include statistical information used for query optimization represented in RDF. A service description describes the data available from a data source in form of *capabilities*. Capabilities define what kind of triple patterns can be answered by the data source. The definition of capabilities is based on predicates. Statistical information includes the total number of triples in data sources and average selectivity estimates for combinations of subject, predicate, and object.

SPLENDID [10], a query optimization strategy for federating SPARQL endpoints, uses VoID descriptions of datasets to speed-up query processing. The statistical information for every predicate and type are organized in inverted indexes which map predicates and types to a set of tuples containing the data source and the number of occurrences in the data source. For triple patterns with bound variables which are not covered in the VoID statistics, SPLENDID uses SPARQL ASK queries including the triple pattern to all pre-selected data sources and remove failing sources. This improves the source selection efficiency.

HIBISCuS [33], a join-aware source selection algorithm, discards dataset that are relevant for a triple pattern, but that do not contribute to a query result. It proposes detailed data summaries, *dataset capabilities*, containing all the distinct properties with all the URI authorities of their subjects and objects. HIBISCuS has been implemented over the federated query engines FedX [35] and SPLENDID.

CostFed [34], an index-assisted federated engine for SPARQL endpoints, extends and improves the join-aware source selection of HIBISCuS by considering URI prefixes instead of URI authorities. It uses such prefixes to prune irrelevant data sources more effectively than the state-of-the-art approaches. In particular, HIBISCuS fails to prune the data sources that share the same URI authority. CostFed overcomes this problem by using source specific sets of strings that many URIs in the data source begin with (these strings are the prefixes of the URI strings). The CostFed query planner also considers the skew distribution of subjects and objects per predicate in each data source. In addition, separate cardinality estimation is used for multi-valued predicates. The dataset capabilities calculated by CostFed are more efficient than other state-of-the-art approaches, its source selection chooses, in general, more pertinently the data sources for each query.

[15] `https://www.w3.org/TR/void/`
[16] DARQ is an extension of ARQ `http://jena.sourceforge.net/ARQ/`
[17] The VoID vocabulary was proposed after DARQ.

In our work we use VoID descriptions to calculate similarity measures and the join-aware source selection of CostFed to limit communication costs during the distributed query relaxation process.

3 A Federated License-Aware Query Processing Strategy

To legally facilitate reuse of query results when retrieving information from several RDF data sources distributed across the Linked Data, we propose FLiQue, a federated license-aware query processing strategy to detect and prevent license conflicts. The goal of FLiQue is to empower federated query engines to produce licensable query results.

A federation is a set of SPARQL endpoints. We consider that a sub-federation is a subset of endpoints of a federation.

FLiQue gives informed feedback with the set of licenses that can protect a result set of a federated query. When the result set of a federated query cannot be licensed, FLiQue defines sub-federations that avoid license conflicts. If there is no sub-federation able to produce a licensable and non-empty result set, FLiQue proposes alternative relaxed federated queries.

Figure 3 shows the global architecture of a license-aware federated query engine using FLiQue. We consider that there exist a federation of endpoints whose datasetes are associated to licenses[18]. FLiQue is located between the source selection and the query optimization functions of a federated query engine. A join-aware source selection [34], selects the *capable datasets* for each triple pattern of a query. Using a compatibility graph of licenses [22], we search for licenses compliant with each license of the chosen capable datasets. Then, the query is executed and the result set returned with the licenses that can protect it. If no compliant license exists, we identify the license conflicts and define sub-federations that avoid these conflicts. If one sub-federation can produce a licensable and non-empty result, the query is executed. Otherwise, based on the OMBS approach [8], we propose to the query issuer a set of relaxed queries whose result sets are licensable and non-empty.

Several sub-federations may produce a licensable and non-empty result. In that case, it is possible to choose the sub-federation that produces a result set licensable by the least restrictive license[19].

Consider the query Q of Listing 1.1, and the federation containing datasets $D1$, $D2$, and $D3$ shown in Tables 2-4. As there is no license compliant with the licenses of $D2$ and $D3$, the result set of Q cannot be licensed. Thus, our strategy defines the sub-federations $F1$=$\{D1, D2\}$ and $F2$=$\{D1, D3\}$ that avoid license conflicts (cf. Figure 2). The source selection for Q over $F1$ and $F2$ fails to obtain a data source for each triple pattern. This launches a process of federated query relaxation for each sub-federation.

[18] Datasets without licenses can be associated with the most permissive license (e.g., CC Zero, and ODbL) or can be discarded from the federation.

[19] Other choices could be defined, for example, based on the cardinality estimations of result sets or based on the number of involved data sources.

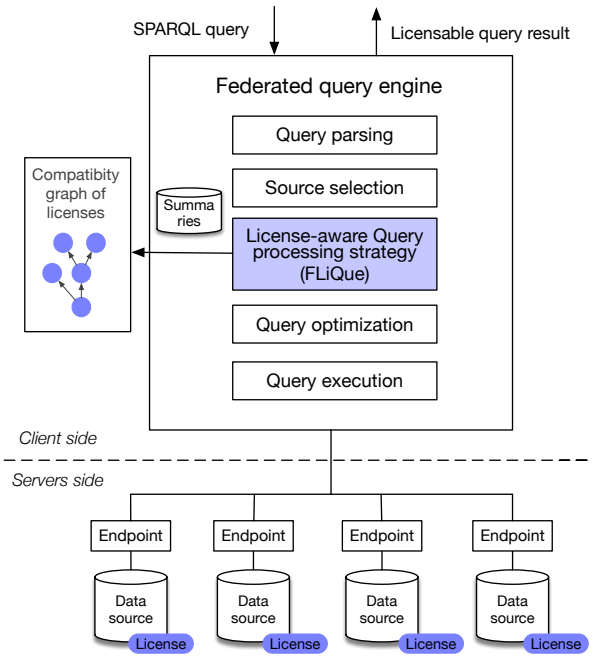

Fig. 3: Global architecture of a federated license-aware query engine using FLiQue.

To avoid verifying that the result set of an important number of relaxed queries is not empty, our strategy defines, by sub-federation, an optimal similarity-based relaxation graph. When we find one licensable and non-empty relaxed query, we stop the relaxation process.

Figure 4 shows Q and three relaxed queries. Figure 5 shows the ontology used in our example. As we explain next, $Q'4b$ and $Q'4d$ are the most similar licensable, and non-empty relaxed query for the sub-federations $F1$ and $F2$ respectively. Identifying sub-federations with compatible licenses and queries (or relaxed queries) that produce non-empty results are the contributions of FLiQue.

In the next, Section 3.1 shows the relaxation techniques we use. Section 3.2 presents the information content measures that allow us to rank relaxed queries. Section 3.3 shows the data summaries that allow limiting communication costs. Finally, Section 3.4 explains the global algorithm of FLiQue and how we define the similarity-based relaxation graph.

3.1 Query relaxation techniques used in FLiQue

In this work, we use query relaxation using RDFS entailment and RDFS ontologies. We consider that ontologies of datasets are accessible and that SPARQL

Subject	Predicate	Object
ex:UniversityOfNantes	rdf:type	ex:University
ex:SemanticWeb	rdf:type	ex:Course
ex:SemanticWeb	ex:heldAt	ex:UniversityOfNantes
ex:Databases	rdf:type	ex:Course
ex:Databases	ex:heldAt	ex:UniversityOfNantes

Table 2: Dataset D1 containing courses. D1 has licence CC BY.

Subject	Predicate	Object
ex:Ben	rdf:type	ex:Teacher
ex:Ben	rdf:type	ex:Person
ex:Ben	ex:attends	ex:SemanticWeb
ex:Ben	ex:teaches	ex:SemanticWeb
ex:Mary	rdf:type	ex:Teacher
ex:Mary	rdf:type	ex:Person
ex:Mary	ex:attends	ex:Databases
ex:Mary	ex:teaches	ex:Databases
my:William	rdf:type	ex:Student
my:William	rdf:type	ex:Person
my:William	ex:attends	ex:Databases
my:William	ex:enrolledIn	ex:Databases

Table 3: Dataset D2 containing teachers and students. D2 has licence CC BY-SA.

Subject	Predicate	Object
ex:Elsa	rdf:type	ex:Student
ex:Elsa	rdf:type	ex:Person
ex:Elsa	ex:attends	ex:SemanticWeb
ex:Elsa	ex:enrolledIn	ex:SemanticWeb

Table 4: Dataset D3 containing students. D3 has licence CC BY-NC.

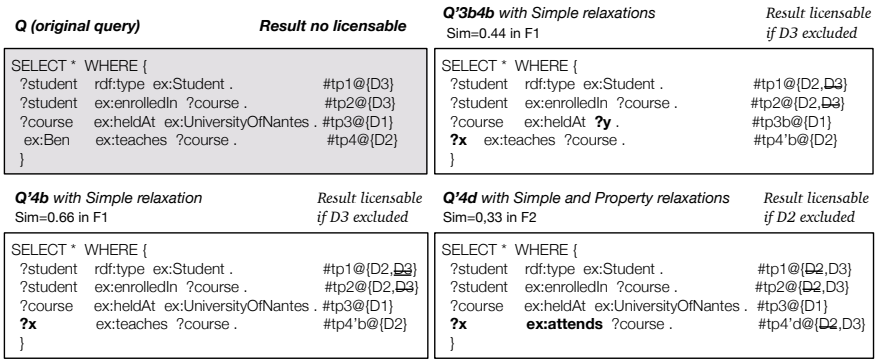

Fig. 4: Example of SPARQL query Q and some relaxed queries Q'.

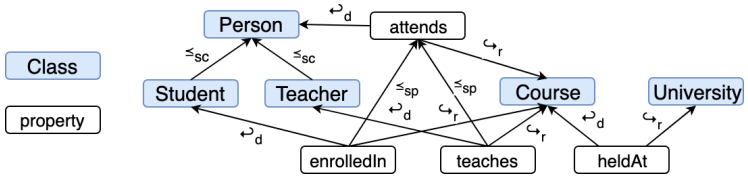

Fig. 5: Ontology representing courses in a university. $\preceq_{sc}$ is rdfs:subClassOf, $\preceq_{sp}$ is rdfs:subPropertyOf, $\hookleftarrow_d$ is rdfs:domain, and $\hookrightarrow_r$ is rdfs:range.

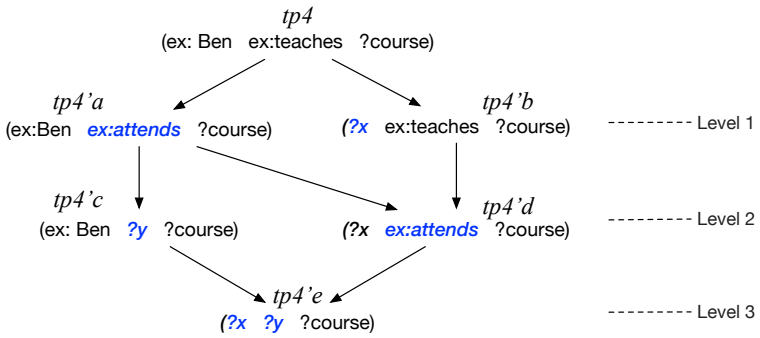

Fig. 6: Relaxation lattice of triple pattern $tp4$ of query Q.

endpoints expose saturated RDF data (or support on-the-fly entailment) according to the RDFS entailment rules rdfs7 and rdfs9. We use the relaxations of triple patterns and queries as proposed in [15].

Triple Pattern Relaxation. Given two triple patterns tp and tp', tp' is a relaxed triple pattern obtained from tp, denoted $tp \prec tp'$, by applying one or more triple pattern relaxations. We use the three following triple pattern relaxations:

- *Simple relaxation* replaces a constant of a triple pattern by a variable. For example, $tp_4 = \langle ex{:}Ben,\ ex{:}teaches,\ ?course \rangle$, can be relaxed to $tp'_4 = \langle ?x,\ ex{:}teaches,\ ?course \rangle$, thus $tp_4 \prec_s tp'_4$.
- *Type relaxation* replaces a class C of a triple pattern with its super-class C'. It is based on the rdfs9 rule (rdfs:subClassOf). For example, $tp_1 = \langle ?student,\ rdf{:}type,\ ex{:}Student \rangle$, can be relaxed to $tp'_1 = \langle ?student,\ rdf{:}type,\ ex{:}Person \rangle$, thus $tp_1 \prec_{sc} tp'_1$.
- *Property relaxation* replaces a property P of a triple pattern with its super-property P'. It is based on the rdfs7 rule (rdfs:subPropertyOf). For example, $tp_1 = \langle ?student,\ ex{:}enrolledIn,\ ?course \rangle$, can be relaxed to $tp'_1 = \langle ?student,\ ex{:}attends,\ ?course \rangle$, thus $tp_1 \prec_{sp} tp'_1$.

The set of all possible relaxed triple patterns of tp can be represented as a lattice called a *relaxation lattice of a triple pattern*. Figure 6 shows this lattice for triple pattern $tp4$ of Q. $tp4'b$, $tp4'c$ and $tp4'e$ show simple relaxations. $tp4'a$ shows a property relaxation. This lattice has three levels of relaxation.

Query Relaxation. Given two queries Q and Q', Q' is a relaxed query obtained from Q, denoted $Q \prec Q'$, by applying one or more triple pattern relaxations to triple patterns of Q. $\prec$ is a partial order over the set of all possible relaxed queries of Q. This order can be represented as a lattice, called a *relaxation lattice*

of a query (or relaxation graph). Figure 4 shows the query Q and three relaxed queries of its relaxation graph where, $Q \prec Q'4b \prec Q'4d$ and $Q \prec Q'4b \prec Q'3b4b$.

3.2 Information content measures used in FLiQue

Analyzing all relaxed queries is time-consuming and unnecessary. We use *information content measures* to compute the similarity of relaxed queries to the original query. To avoid the analysis of an important number relaxed queries, our approach generates and executes relaxed queries from the most to the least similar. This execution allows to verify that the result set is not empty. It is stopped when the first result is returned. We use the *similarity mesures* proposed in [15], and explained in the following.

Similarity between terms. FLiQue uses three similarity measures for terms in a triple pattern. They correspond to the three relaxations described in Section 3.1.

- *Similarity between classes* is $Sim(C, C') = \frac{IC(C')}{IC(C)}$ where $IC(C)$ is the information content of C: $-logPr(C)$, where $Pr(C) = \frac{|Instances(C)|}{|Instances|}$ is the probability of finding an instance of class C in the RDF dataset.
  For example, if the subject or object of a triple pattern is a class c_1 and is relaxed to its super class c_2 using type relaxation, the similarity between c_1 and c_2 is $Sim(c_1, c_2)$.
  Notice that, the similarity between classes is zero when all the instances in the RDF dataset belong to the super class C', i.e., $Pr(C') = 1$ and thus $-logPr(C') = 0$. Notice also that the similarity between classes is undefined when all the instances belong to the super class C', and none to C, i.e., $Pr(C) = 0$ and thus $-logPr(C') = undefined$.
- *Similarity between properties* is $Sim(P, P') = \frac{IC(P')}{IC(P)}$ where $IC(P)$ is the information content of P: $-logPr(P)$, where $Pr(P) = \frac{|Triples(P)|}{|Triples|}$ is the probability of finding a property of P in triples of the RDF dataset.
  For example, if the predicate of a triple pattern is a property p_1 and is relaxed to its super property p_2 using property relaxation, the similarity between p_1 and p_2 is $Sim(p_1, p_2)$.
  Notice that as the similarity between classes, the similarity between properties is zero when all the triples in the RDF dataset belong to the super property P', and the similarity between properties is undefined when all the triples belong to the super property P', and none to P.
- *Similarity between constants and variables* is $Sim(T_{const}, T_{var}) = 0$.
  For example, if the object of a triple pattern t_{const} is a class and is relaxed to a variable t_{var} using simple relaxation, the similarity between t_{const} and t_{var} is 0.

Similarity between triple patterns Given two triple patterns tp and tp', such that $tp \prec tp'$, the similarity of the triple pattern tp' to the original triple pattern tp, denoted $Sim(tp, tp')$, is the average of the similarities between the terms of the triple patterns:

$$Sim(tp, tp') = \frac{1}{3}.Sim(s, s') + \frac{1}{3}.Sim(p, p') + \frac{1}{3}.Sim(o, o')$$

where s, p, o, s', p' and o' are respectively the subject, predicate and object of the triple pattern tp and the relaxed triple pattern tp'. If tp' and tp'' are two relaxations obtained from tp and $tp' \prec tp''$ then $Sim(tp, tp') \geq Sim(tp, tp'')$.

Similarity between queries. Given two queries Q and Q', such that $Q \prec Q'$, the similarity of the original query Q' to the original query Q, denoted $Sim(Q, Q')$, is the product of the similarity between triple patterns of the query:

$$Sim(Q, Q') = \prod_{i=1}^{n} w_i.Sim(tp_i, tp'_i)$$

Where tp_i is a triple pattern of Q, tp'_i a triple pattern of Q', $tp_i \prec tp'_i$, and $w_i \in [0, 1]$ is the weight of triple patterns tp_i. Weight can be specified by the user to take into account the importance of a triple pattern tp_i in query Q. Thus $Sim(Q, Q') \in [0, 1]$ is a function that defines a total order among relaxed queries.

This similarity function is monotone, i.e., given two relaxed queries $Q'(tp'_1, ..., tp'_n)$ and $Q''(tp''_1, ..., tp''_n)$ of the user query Q, if $Q' \prec Q''$ then $Sim(Q, Q') \geq Sim(Q, Q'')$.

Considering the query Q and datasets D1 and D2, $Sim(Q, Q'4b) = 0.66$ is greater than $Sim(Q, Q'3b4b) = 0.44$. This verifies the ordering of these relaxed queries, $Q'4b \prec Q'3b4b$, where $Q'4b$ is analyzed first to see if it returns some results.

3.3 Data summaries used in FLiQue

FLiQue collects dataset summaries and ontologies before executing queries. A data summary is a compact structure that represents an RDF dataset. Using dataset statistics and dataset capabilities as in [34], allow us to limit communication cost in the similarity calculation and the source selection process.

Dataset statistics contain VoID descriptions, such as the number of entities per class and the number of triples per property. Having dataset statistics is twofold; they allow computing similarities, and they help in the source selection process. Tables 5 and 6 show respectively statistics about properties and classes for subfederations F1 and F2. In Table 5, the property *ex:teaches* has no triples in F2. So there is no data source for $tp4$ and $tp4'b$ (cf. Figure 6). That allows to identify whatever query including these triple patterns as failing queries if executed over F2.

Dataset capabilities contain the properties of a dataset with the common prefixes of their subjects and objects. We recall that prefixes are strings that many URIs in the data source begin with. The *rdf:type* property, is treated differently. The prefixes of its objects are replaced by all the classes used in the dataset. Dataset capabilities are used in the source selection process. The goal is to discard datasets that individually return results for a triple pattern, but that fail to perform joins with other triple patterns of the query. For multiple triple patterns of a query sharing a variable, the dataset capabilities allow identifying data sources that do not share the same URIs prefixes and thus whose joins yield empty results. This information allows performing an optimal source selection by limiting the communication with the data sources. Table 7 shows the capabilities of F1 and F2.

Consider $tp4'a$: $\langle$*ex:Ben ex:attends ?cours*$\rangle$

Statistics in Table 5 show one triple for *ex:attends* in F2 but capabilities of this property in F2 show only one subject prefix that is *ex: Elsa*, not *ex:Ben*. Thus, capabilities of F2 allow to identify $tp4'a$ as failing if executed over F2.

Consider also the join $tp3$. $tp4'c$, that is a subject-object join:
{?course ex:heldAt ex:UniversityOfNantes . ex:Ben ?y ?course }

Statistics in Table 5 show two triples for *ex:heldAt* in both sub-federations. But, analyzing the subject and object capabilities of whatever property (the predicate of $tp4'c$ is a variable) of F2, we notice that when there exists *ex:SemanticWeb* in the object, the subject contains *ex: Elsa*, not *ex:Ben*, so the join dependency on *?course* cannot be satisfied. Thus, thanks to the dataset capabilities of F2, we identify that whatever query with this join will be identified as failing query. Notice that this join over F1 cannot be discarded with the statistiques and dataset capabilities. Indeed, this join returns results over F1.

3.4 Global FLiQue algorithm and the similarity-based relaxation graph

Algorithm 1 shows the global approach of FLiQue. After a process of source selection for a federated query Q, FLiQue checks if the result set can be licensed (Line 5). If that is the case ($\mathcal{L} == \emptyset$ is false), the query plan as long as the set of licences that can protect the result set are returned (Line 18). Then the query is optimized and executed as usual.

When the result set cannot be licensed because there is no license compliant to every license of selected datasets, FLiQue calls for a source selection over each sub-federation (Line 6).Each sub-federation that can evaluate the query is returned as long as with the corresponding set of compliant licenses (Line 10).

If all source selections fail (Line 11), FLiQue relaxes the query conditions and proposes alternative relaxed federated queries. We consider that a source selection process fails if it does not identify at least one data source to evaluate each triple pattern of the federated query. Based on techniques of query relaxation in

	Number of triples	
Property	F1 = {D1, D2}	F2 = {D1, D3}
ex:enrolledIn	1	1
ex:teaches	2	0
ex:heldAt	2	2
ex:attends	3	1
rdf:type	9	5
Total	17	9

Table 5: Statistics of properties in federations F1 and F2.

	Number of entities	
Class	F1 = {D1, D2}	F2 = {D1, D3}
ex:University	1	1
ex:Student	1	1
ex:Teacher	2	0
ex:Course	2	2
ex:Person	3	1
Total	6	4

Table 6: Statistics of classes in federations F1 and F2.

	F1 = {D1, D2}		F2 = {D1, D3}	
Property	**subjPrefixes**	**objPrefixes**	**subjPrefixes**	**objPrefixes**
rdf:type	ex: my: William	ex:Person ex:Student ex:Teacher	ex:	ex:University ex:Course ex:Student ex:Person
ex:heldAt	ex:	ex:UniversityOfNantes	ex:	ex:UniversityOfNantes
ex:attends	ex: my: William	ex:	ex: Elsa	ex:SemanticWeb
ex:teaches	ex:	ex:		
ex:enrolledIn	my: William	ex:Database	ex: Elsa	ex:SemanticWeb

Table 7: Capabilities of sub-federations F1 and F2.

RDF [8] (Line 13), FLiQue proposes to the query issuer a relaxed query for each sub-federation whose result set is licensable and non-empty (line 16).

The result of the algorithm is a set of triplets representing what we call *candidate queries*. A triplet $\langle Q, E, \mathcal{L} \rangle$ is a query Q that returns a non-empty result set when executed on a sub-federation of endpoints E that can be protected by a set of licenses $\mathcal{L}$.

When the distributed query relaxation is necessary, we define an optimal similarity-based relaxation graph by sub-federation. The goal is to avoid verifying that the result set of an important number of relaxed queries is not empty. Relaxed queries are generated and executed from the most to the least similar. When we find one licensable and non-empty candidate query, we stop the relaxation process. OMBS [8] guarantees that a candidate query is the most similar to Q for a sub-federation.

In the following, we explain how FLiQue finds the candidate query for the sub-federation F2. First, the algorithm computes the MFS of the original query, MFS(Q)={ex:Ben ex:teaches ?course}. It contains only $tp4$ because $F2$ does not contain a data source to evaluate $tp4$. Using the MFS, the algorithm considers only relaxed queries that contain a relaxation of $tp4$.

Algorithm 1: The global approach of FLiQue.

```
1  Function FLiQue(Q, F, E, S, C):
       Data: Q: RDF Query,
       F: Federation of endpoints,
       E ⊆ F: Set of pertinent endpoints for Q,
       S: Dataset summaries,
       C: Compatibility graph of licenses.
       Result: A set of tuples ⟨Q, E, L⟩ representing candidate queries with
               corresponding pertinent endpoints and compliant licenses.
2      Fs = {F' ⊆ F | compliantLicenses(F', C) ≠ ∅}
3      Candidates = ∅
4      L = compliantLicenses(E,C)
5      if L == ∅ then
6          for F' ∈ Fs do
7              E'=sourceSelection(Q, F', S)
8              if E' can evaluate Q then
                   // The original query can be executed on E'.
9                  L = compliantLicenses(E',C)
10                 Candidates = Candidates ∪ ⟨Q, E', L⟩
11         if Candidates == ∅ then
               // Compute most similar queries.
12             for F' ∈ Fs do
13                 Q' = queryRelaxation(Q, F', S)
14                 E'=sourceSelection(Q', F', S)
                   // The relaxed query Q' can be executed on E'.
15                 L = compliantLicenses(E',C)
16                 Candidates = Candidates ∪ ⟨Q', E', L⟩
17     else
           // The original query can be executed on E.
18         Candidates = {⟨Q, E, L⟩}
19     return Candidates
```

The relaxation algorithm uses a query queue ordered by similarity. This query queue gives the analysis order of relaxed queries. Figure 7(a) shows a relaxation sub-graph where $tp4$, that is the MFS, is relaxed, and Figure 7(b) shows the analysis process of relaxed queries with the query queue (failing relaxed queries are in gray).

Relaxed queries of the first level, $Q'4a$ and $Q'4b$, are generated and inserted in the queue, see (1) in Figure 7(b). The most similar relaxed query $Q'4a$ is analyzed. It is identified as a failing query (cf. Section 3.3), thus it is relaxed, so $Q'4c$, and $Q'4d$ are generated and inserted in the query queue (2). Then, the first relaxed query in the queue, now $Q'4b$, is analyzed. It is also identified as a failing query so it is relaxed in $Q'4d$, which is already in the queue (3). Then, the first relaxed query in the queue, now $Q'4c$, is analyzed and identified as a

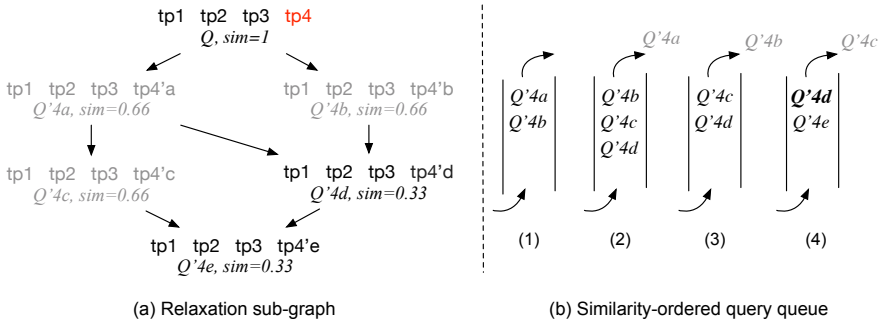

Fig. 7: Relaxation sub-graph of Q over $F2$ with relaxations of $tp4$.

failing query, so it is relaxed into $Q'4e$, which is inserted in the queue (4). Then, the first relaxed query in the queue, now $Q'4d$, is analyzed. It is not identified as a failing query. It is executed, and it returns a non-empty result set. Thus, $Q'4d$ (in bold) is the candidate query of query Q for the federation $F2$, and the relaxation process stops.

The MFS and the failing relaxed queries of this example are identified thanks to data summaries without making requests to data sources (cf. Section 3.3).

In this example, we found a candidate query only with the relaxation of $tp4$. But the relaxation may continue until all triple patterns are composed of variables. A threshold of similarity can be used to avoid such a worst case.

Figure 4 shows the candidate query $Q'4d$, whose similarity with Q is 0.33. This query asks for students attending a course held at the University of Nantes.

The candidate query for federation $F1$ is $Q'4b$, whose similarity with Q is 0.66. Figure 4 shows $Q'4b$, this query asks for students enrolled in a course held at the University of Nantes and taught by someone.

CC BY-SA can protect relaxed queries for $F1$. Relaxed queries for $F2$ can be protected by CC BY-NC but also by CC BY-NC-SA because both licenses are compliant with licenses of $D1$ and $D3$. Table 8 shows the feedback returned to the query issuer so that she can choose which query to execute.

Sub-federation	Query	Similarity	Compliant licenses
F1={D1, D2}	Q'4b	0.66	CC BY-SA
F2={D1, D3}	Q'4d	0.33	CC BY-NC, CC BY-NC-SA

Table 8: FLiQue feedback with candidate queries for the user query Q.

4 Experimental evaluation

The goal of our experimental evaluation is to measure the overhead produced by the implementation of our proposal. In particular, (a) when the result set of the original query is licensable, and (b) when the original query is relaxed.

4.1 Setup and implementation

FLiQue is implemented over CostFed, which relies on a join-aware triple-wise source selection. Recent studies show that the source selection of CostFed least overestimates the set of capable data sources, with a small number of ASK requests [32,34]. These performances make CostFed a good choice for our license-aware query processing strategy. The join ordering of CostFed is based on left-deep join trees. It implements bind and symmetric hash joins.

Our test environment uses LargeRDFBench [32]. A benchmark with more than 1 billion triples for SPARQL endpoints. This benchmark contains 32 federated queries that are executed over a federation of 11 interlinked data sources (we consider the three Linked TCGA datasets as only one). We identified the license of each dataset (cf. Figure 1). We use a Creative Commons CaLi ordering [22] to verify compatibility and compliance among licenses.

LargeRDFBench was defined to evaluate and compare federated query engines. It is composed of 14 simple queries (S1-S14), 10 complex queries (C1-C10), and 8 large queries (L1-L8). The number of triple patterns in queries varies from 2 to 12 (the average is 7). Simple queries range from 2 to 7 triple patterns and their execution time is small (few seconds). Complex queries add constraints to simple queries. They range from 8 to 12 triple patterns and their evaluation is costly in time (few minutes). Large queries were defined to be evaluated over large datasets and involve large intermediate result sets. They range from 5 to 11 triple patterns and their evaluation is very costly (it can exceeds one hour).

In our experiments we are not interested in obtaining the complete query results of all these queries but in showing that preserving licences during federated query processing is possible. Thus, queries were executed until obtaining the first result (LIMIT 1 was added to all queries).

Our experiment runs on a single machine with a 160xIntel(R) Xeon(R) CPU E7-8870 v4 2.10GHz 1,5 Tb RAM. Each dataset of LargeRDFBench is saturated and made available using a single-threaded Virtuoso endpoint in a docker container with 4 Gb RAM. Between each query execution, caches are reset.

4.2 Performance of FLiQue vs CostFed

To measure the overhead produced by FLiQue, we compare two different federated query engines: CostFed and CostFed+FLiQue (that we call FLiQue to simplify). They correspond to the original implementation of CostFed[20] and our

[20] https://github.com/dice-group/CostFed

Conflicting sources	Conflicting licenses	Queries
DBP, DB	CC BY-SA, CC BY-NC	S1, **S10, C9**
DBP, TCGA	CC BY-SA, CC BY-NC	**L7**
DBP, JA	CC BY-SA, CC BY-NC-SA	**L6**
DBP, DB, TCGA	CC BY-SA, CC BY-NC	**C10**
DBP, DB, TCGA, JA	CC BY-SA, CC BY-NC, CC BY-NC-SA	S6, **S8**, S9, C3, **C5, C8**, L1, L3, **L5, L8**

Table 9: The 16 queries of LargeRDFBench whose result set cannot be licensed. DBP (DBpedia), DB (Drug bank), TCGA (Linked TCGA), JA (Jamendo).

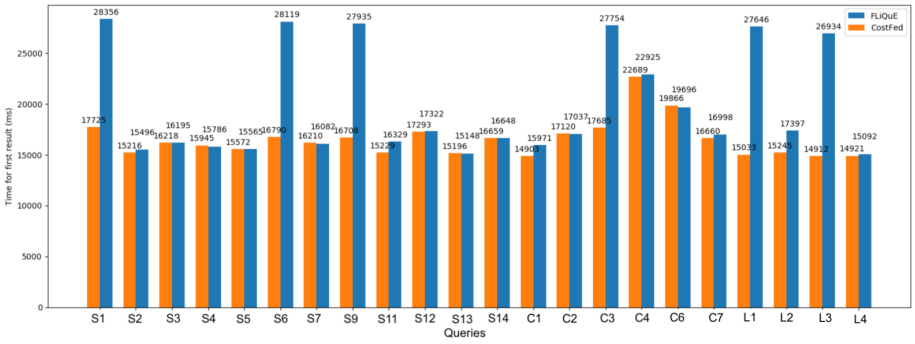

Fig. 8: Average time to get the first result of the 22 queries of LargeRDFBench that can produce a licensable result set without relaxation.

extension of CostFed that includes FLiQue[21]. CostFed executes a query without considering licenses while FLiQue guarantees license compliance of the result set.

We executed all queries 5 times with each federated query engine. We measured the time in milliseconds to return the first result of each query.

Using the capable data sources by query, and the compatibility graph of licenses, we identified 16 queries whose result set cannot be licensed. Table 9 shows these queries, their conflicting capable data sources and conflicting licenses. 10 queries need to be relaxed, they are shown in bold. We recall that the DBpedia license (CC BY-SA) is not compliant with the licenses of Jamendo (CC BY-NC-SA), Linked TCGA and Drug bank (CC BY-NC). The average time to check license conflicts is 296 milliseconds which is negligible.

Evaluation of queries that do not need relaxation. Figure 8 presents the execution time of the 22 queries of LargeRDFBench that do not need relaxation. We identify two sets of queries.

– For 16 queries, {S2, S3, S4, S5, S7, S11, S12, S13, S14, C1, C2, C4, C6, C7, L2, L4}, FLiQue finds a license that can protect the result set when the query is executed on the complete federation. For these queries, the overhead of FLiQue is negligible and corresponds to the time to check license

[21] https://github.com/benjimor/FLiQuE

conflicts among the capable datasets. This overhead depends on the number of distinct licenses that protect the capable datasets.

– For 6 queries, {S1, S6, S9, C3, L1, L3}, FLiQue does not find a license that can protect the result set when the query is executed over the complete federation. However, it finds a sub-federation such that the original query returns a non-empty result set that is licensable. In this case, the overhead of FLiQue corresponds to the time to check license conflicts, to compute sub-federations, and to execute the original query on these sub-federations until the first result is returned. This overhead depends on the number of tested sub-federations. The number of sub-federations depends on the number of distinct conflicting licenses by query. In our test environment, this number is always 2. For instance, conflicting licenses CC BY-SA, CC BY-NC, and CC BY-NC-SA can be separated into two non-conflicting sets {CC BY-SA} and {CC BY-NC, and CC BY-NC-SA}. These sub-federations are ordered by the number of datasets in the federation. In the benchmark, the average time to generate the sub-federations and find a non-empty result set is 11020 milliseconds. For these 6 queries, we remark that this overhead is almost constant. That is because, a non-empty result set is found when FLiQue executes the original query on the second sub-federation.

Evaluation of queries that are relaxed. Figure 9 presents the execution of the 10 queries of LargeRDFBench that need relaxation to return a non-empty result set that can be protected by a license. For each query, we compare the time to get the first result of the original query for CostFed, and the time to get the first result of the first candidate query found by FLiQue.

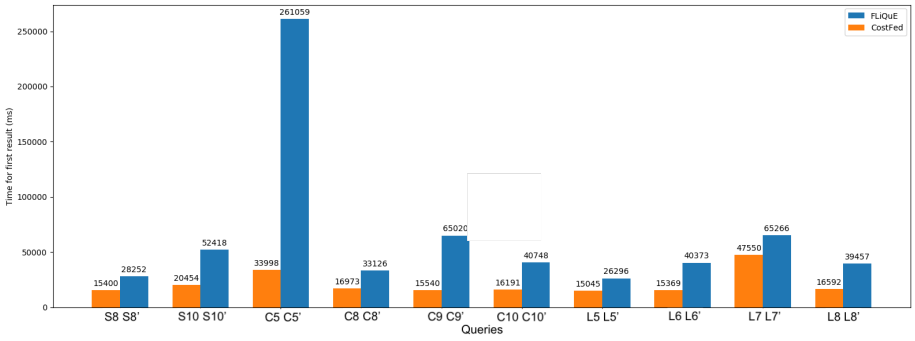

Fig. 9: Average time to get the first result of the 10 queries of LargeRDFBench that need relaxation to produce a licensable result set.

The FLiQue overhead corresponds to the time to check license conflicts to compute sub-federations, and to find the first candidate query.

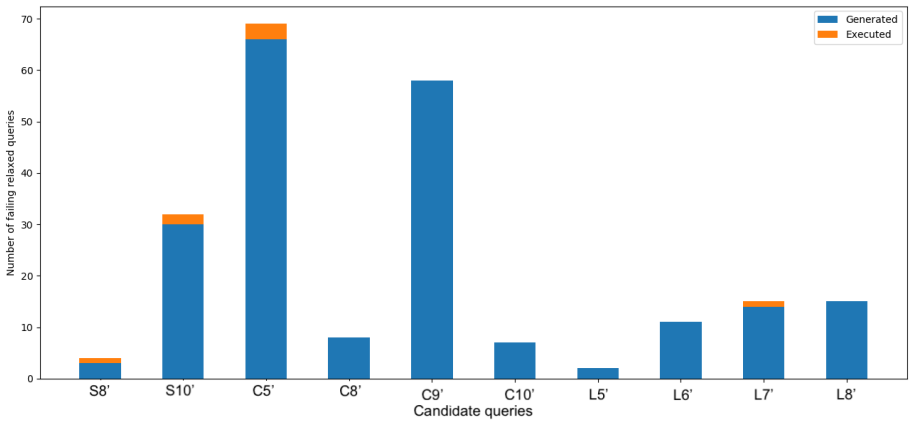

Fig. 10: Number of generated and executed failing relaxed queries until finding each candidate query.

We remark that the execution time of an original query and a candidate query is not comparable. They are not the same query, and they are not executed on the same number of data sources. The candidate queries are more general. To have an idea (non-representative) of the similarities, the maximum is 0.811 (L5'), the minimum is 0.077 (C8'), the average is 0.487, and the median is 0.603. Candidate queries and their similarities are included in Annex A.

Overhead varies a lot depending on the queries. It depends on the number of generated and executed failing relaxed queries, before finding the first candidate query. Figure 10 shows the number of failing relaxed queries, (1) generated, and (2) executed before finding each candidate query. Most of the relaxed queries generated are identified as failing thanks to data summaries. The candidate query $C5'$, is found after generating 69 failing relaxed queries, but only 3 were executed until finding the first one with non-empty result. In contrast, candidate query $S8'$ is found after generating 3 failing relaxed queries but executing only one. For 6 out of 10 relaxed queries, FLiQue does not need to execute any generated relaxed query to identify them as failing.

With this benchmark, on average FLiQue generates 21.4 failing relaxed queries, and executes 1.75 failing relaxed queries. Thus, we consider that FLiQue succeeds in limiting communication costs during the relaxation of queries whose result set cannot be licensed.

4.3 Discussion

[8, 15] used in their experiments the datasets generated with the LUBM benchmark [11]. LUBM generates synthetic data as one data source and was developed to evaluate systems with different reasoning capabilities. [15] used 7 queries with a number of triple patterns ranging from 2 to 5. [8] modified these 7 queries to added constraints. In their modified queries, triple patterns range from 2 to 15,

the average is 7. Their experimental goal was to show the number of relaxed queries executed until obtaining top-k results.

Our goal is different. We aim to show that preserving licenses during federated query processing is possible. LargeRDFBench is the only benchmark with federated queries. It is composed of several real-world interlinked datasets whose federated queries were designed by domain specialists. This benchmark was not defined for reasoning. As it does not include ontologies[22] we searched them[23]. Several properties or classes of some datasets are not defined in the found ontologies, and hierarchies in concept ontologies are frequently undefined.

As LargeRDFBench was defined to evaluate federated query processing, its federated queries have many variables. Very few triple patterns have classes in objects or subjects. That is why query relaxation was mainly based on property relaxation. Only two classes in objects were relaxed (dbo:Drug in S10 and dbpedia:Drug in C9).

LargeRDFBench datasets are of multiple domains. They were chosen because they are interlinked, i.e., they have *owl:sameAs* relationships, the subject/object of one dataset exists in another dataset, or some predicates (like *title* and *genericName*) have the same literal values. The choice of these datasets was not focused on common classes or predicates, but on instances. That is why almost all relaxations ended in simple relaxations (except for L5 and L8 where *foaf:name* was relaxed to *rdfs:label*).

Although LargeRDFBench is not the ideal benchmark for evaluating FLiQue, we were able to demonstrate that federated query processing can ensure license compliance using query relaxation.

5 Conclusion and perspectives

In this work, we propose FLiQue, a federated license-aware query processing strategy that guarantees that a license protects the result set of a federated SPARQL query. FLiQue is designed to detect and prevent license conflicts and gives informed feedback with licenses able to protect a result set of a federated query. If necessary, it applies distributed query relaxation to propose a set of most similar relaxed federated queries whose result set can be licensed. To our knowledge, this is the first work that uses query relaxation in a distributed environment. Our implementation extends an existing federated query engine with our license-aware query processing strategy. Our prototype demonstrates the feasibility of our approach. Experimental evaluation shows that FLiQue ensures license compliance with a limited overhead in terms of execution time. FLiQue is a step towards facilitating and encouraging the publication and reuse of licensed resources in the Web of Data. FLiQue is not a data access control strategy.

[22] https://github.com/dice-group/LargeRDFBench

[23] This is a compilation of all ontologies we found for LargeRDFFech datasets: `https://raw.githubusercontent.com/benj-moreau/FLiQue/master/flique/ontologies/ontology.n3`.

Instead, it empowers well-intentioned data users in respecting the licenses of datasets involved in a federated query.

This work has several perspectives. One perspective considers other aspects of licenses related to usage contexts like jurisdiction, dates of reuse, etc. Another perspective is about estimating the selectivity of relaxed queries and use it to help users choose the best-relaxed query for their purposes. Information content measures are currently used to define similarity between queries; nevertheless, semantic similarity may also be used, for instance, *owl:sameAs* relationships.

A perspective not directly related to FLiQue but to the used benchmark, is to analyze the ontologies of datasets composing LargeRDFBench and complete them. Furthermore, it will be interesting to modify the federated queries to include more classes in subjects and objects.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data: The Story so Far. In: Semantic services, interoperability and web applications: emerging concepts. IGI global (2011)
2. Bonatti, P.A., Decker, S., Polleres, A., Presutti, V.: Knowledge graphs: New Directions for Knowledge Representation on the Semantic Web (Gagstuhl seminar 18371). In: Dagstuhl Reports (2019)
3. Cabrio, E., Aprosio, A.P., Villata, S.: These Are Your Rights. In: European Semantic Web Conference (ESWC) (2014)
4. Čebirić, Š., Goasdoué, F., Kondylakis, H., Kotzinos, D., Manolescu, I., Troullinou, G., Zneika, M.: Summarizing Semantic Graphs: a Survey. The VLDB Journal **28**(3) (2019)
5. Costabello, L., Villata, S., Gandon, F.: Context-Aware Access Control for RDF Graph Stores. In: European Conference on Artificial Intelligence (ECAI) (2012)
6. Cyganiak, R., Hausenblas, M.: Describing Linked Datasets - On the Design and Usage of voiD, the "Vocabulary of Interlinked Datasets". In: Linked Data on the Web Workshop (LDOW) (2009)
7. Ferré, S.: Answers Partitioning and Lazy Joins for Efficient Query Relaxation and Application to Similarity Search. In: Extended Semantic Web Conference (ESWC) (2018)
8. Fokou, G., Jean, S., Hadjali, A., Baron, M.: RDF Query Relaxation Strategies Based on Failure Causes. In: Extended Semantic Web Conference (ESWC) (2016)
9. Gabillon, A., Letouzey, L.: A View Based Access Control Model for SPARQL. In: International Conference on Network and System Security (NSS) (2010)
10. Görlitz, O., Staab, S.: SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions. In: Workshop Consuming Linked Data (COLD) collocated with ISWC (2011)
11. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. Journal of Web Semantics **3**(2-3) (2005)
12. Hartig, O., Vidal, M.E., Freytag, J.C.: Federated Semantic Data Management (Dagstuhl Seminar 17262). In: Dagstuhl Reports (2017)
13. Havur, G., Steyskal, S., Panasiuk, O., Fensel, A., Mireles, V., Pellegrini, T., Thurner, T., Polleres, A., Kirrane, S.: DALICC: A Framework for Publishing and Consuming Data Assets Legally. In: International Conference on Semantic Systems (SEMANTICS),Poster&Demo (2018)

14. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Gayo, J.E.L., Kirrane, S., Neumaier, S., Polleres, A., Navigli, R., Ngomo, A.N., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J.F., Staab, S., Zimmermann, A.: Knowledge Graphs. CoRR **abs/2003.02320** (2020)
15. Huang, H., Liu, C., Zhou, X.: Approximating Query Answering on RDF Databases. Journal of World Wide Web **15** (2012)
16. Hurtado, C.A., Poulovassilis, A., Wood, P.T.: Query Relaxation in RDF. Journal on Data Semantics X (2008)
17. Iannella, R., Villata, S.: ODRL Information Model 2.2. W3C Recommendation (2018)
18. Kapitsaki, G.M., Kramer, F., Tselikas, N.D.: Automating the License Compatibility Process in Open Source Software With SPDX. Journal of Systems and Software **131** (2017)
19. Khan, Y., Saleem, M., Iqbal, A., Mehdi, M., Hogan, A., Ngomo, A.C.N., Decker, S., Sahay, R.: SAFE: Policy Aware SPARQL Query Federation Over RDF Data Cubes. In: Semantic Web Applications and Tools for Life Sciences (SWAT4LS) (2014)
20. Kirrane, S., Abdelrahman, A., Mileo, A., Decker, S.: Secure Manipulation of Linked Data. In: International Semantic Web Conference (ISWC) (2013)
21. Moreau, B., Serrano-Alvarado, P., Perrin, M., Desmontils, E.: A License-Based Search Engine. In: Extended Semantic Web Conference (ESWC), Demo (2019)
22. Moreau, B., Serrano-Alvarado, P., Perrin, M., Desmontils, E.: Modelling the Compatibility of Licenses. In: Extended Semantic Web Conference (ESWC) (2019)
23. Oguz, D., Ergenc, B., Yin, S., Dikenelli, O., Hameurlain, A.: Federated Query Processing on Linked Data: a Qualitative Survey and Open Challenges. The Knowledge Engineering Review **30**(5) (2015)
24. Oulmakhzoune, S., Cuppens-Boulahia, N., Cuppens, F., Morucci, S., Barhamgi, M., Benslimane, D.: Privacy Query Rewriting Algorithm Instrumented by a Privacy-Aware Access Control Model. Annals of Telecommunications **69** (2014)
25. Pellegrini, T., Havur, G., Steyskal, S., Panasiuk, O., Fensel, A., Mireles, V., Thurner, T., Polleres, A., Kirrane, S., Schönhofer, A.: DALICC: A License Management Framework for Digital Assets. Proceedings of the Internationales Rechtsinformatik Symposion (IRIS) **10** (2019)
26. Qudus, U., Saleem, M., Ngomo, A.N., Lee, Y.: An empirical evaluation of cost-based federated SPARQL query processing engines. CoRR **abs/2104.00984** (2021), `https://arxiv.org/abs/2104.00984`
27. Quilitz, B., Leser, U.: Querying Distributed RDF Data Sources with SPARQL. In: European Semantic Web Conference (ESWC) (2008)
28. Reddivari, P., Finin, T., Joshi, A., et al.: Policy-Based Access Control for an RDF Store. In: Workshop Semantic Web for Collaborative Knowledge Acquisition (SWeCKa) collocated with IJCAI (2007)
29. Resnik, P.: Using Information Content To Evaluate Semantic Similarity in a Taxonomy. International Joint Conference on Artificial Intelligence (IJCAI) (1995)
30. Rodríguez Doncel, V., Gómez-Pérez, A., Villata, S.: A Dataset of RDF Licenses. In: Legal Knowledge and Information Systems Conference (ICLKIS) (2014)
31. Sadeh, N., Acquisti, A., Breaux, T.D., Cranor, L.F., et.al.: Towards Usable Privacy Policies: Semi-Automatically Extracting Data Practices from Websites' Privacy Policies. In: Symposium on Usable Privacy and Security (SOUPS) (2014), poster
32. Saleem, M., Hasnain, A., Ngomo, A.C.N.: LargeRDFBench: a Billion Triples Benchmark for SPARQL Endpoint Federation. Journal of Semantic Web **48** (2018)

33. Saleem, M., Ngomo, A.C.N.: HIBISCuS: Hypergraph-Based Source Selection For SPARQL Endpoint Federation. In: Extended Semantic Web Conference (ESWC) (2014)
34. Saleem, M., Potocki, A., Soru, T., Hartig, O., Ngomo, A.N.: CostFed: Cost-Based Query Optimization for SPARQL Endpoint Federation. In: International Conference on Semantic Systems (SEMANTICS) (2018)
35. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: Optimization Techniques for Federated Query Processing on Linked Data. In: International Semantic Web Conference (ISWC) (2011)
36. Seneviratne, O., Kagal, L., Berners-Lee, T.: Policy-Aware Content Reuse on the Web. In: International Semantic Web Conference (ISWC) (2009)
37. Villata, S., Gandon, F.: Licenses Compatibility and Composition in the Web of Data. In: Workshop Consuming Linked Data (COLD) collocated with ISWC (2012)
38. Wheeler, D.A.: The Free-Libre/Open Source Software (FLOSS) License Slide. `https://www.dwheeler.com/essays/floss-license-slide.pdf` (2007)

A Supplemental material

Listing 1.2: Query S8 followed by the candidate query S8'.

```
PREFIX drugbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/
    drugbank/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?drug ?melt WHERE {
 { ?drug drugbank:meltingPoint ?melt. }
 UNION
 { ?drug dbo:meltingPoint ?melt . }
}


SELECT ?drug ?melt WHERE {
 { ?drug drugbank:meltingPoint ?melt. }
 UNION
 { ?drug ?1HmoLC ?melt . }
}

# Similarity: 0.666666666666666
```

Listing 1.3: Query S10 followed by the candidate query S10'.

```
PREFIX drugbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/
    drugbank/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?Drug ?IntDrug ?IntEffect WHERE {
 ?Drug a dbo:Drug .
 ?y owl:sameAs ?Drug .
 ?Int drugbank:interactionDrug1 ?y .
 ?Int drugbank:interactionDrug2 ?IntDrug .
 ?Int drugbank:text ?IntEffect .
}


SELECT ?Drug ?IntDrug ?IntEffect WHERE {
 ?Drug ?zkB8o2 ?OKS9kY .
 ?y ?Y2df3t ?Drug .
 ?Int drugbank:interactionDrug1 ?y .
 ?Int ?Q6kLIS ?IntDrug .
 ?Int drugbank:text ?IntEffect .
}

# Similarity: 0.11111111111
```

Listing 1.4: Query C5 followed by the candidate query C5'.

```
PREFIX linkedmdb: <http://data.linkedmdb.org/resource/movie/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?actor ?movie ?movieTitle ?movieDate ?birthDate ?spouseName
{
 ?actor rdfs:label ?actor_name_en .
 ?actor dbpedia:birthDate ?birthDate .
 ?actor dbpedia:spouse ?spouseURI .
 ?spouseURI rdfs:label ?spouseName .
 ?imdbactor linkedmdb:actor_name ?actor_name.
 ?movie linkedmdb:actor ?imdbactor .
 ?movie dcterms:title ?movieTitle .
 ?movie dcterms:date ?movieDate .
 FILTER(STR(?actor_name_en ) = STR(?actor_name))
}

SELECT ?actor ?movie ?movieTitle ?movieDate ?birthDate ?spouseName
{
 ?actor rdfs:label ?actor_name_en .
 ?actor ?SxzR4W ?birthDate .
 ?actor ?1OCiE4 ?spouseURI .
 ?spouseURI rdfs:label ?spouseName .
 ?imdbactor linkedmdb:actor_name ?actor_name .
 ?movie linkedmdb:actor ?imdbactor .
 ?movie dcterms:title ?movieTitle .
 ?movie dcterms:date ?movieDate .
 FILTER(STR(?actor_name_en ) = STR(?actor_name))
}
# Similarity: 0.44444444444
```

Listing 1.5: Query C8 followed by the candidate query C8'.

```
PREFIX swc: <http://data.semanticweb.org/ns/swc/ontology#>
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
PREFIX eswc: <http://data.semanticweb.org/conference/eswc/>
PREFIX iswc:  <http://data.semanticweb.org/conference/iswc/2009/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX purl: <http://purl.org/ontology/bibo/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT * WHERE
{
 ?paper swc:isPartOf iswc:proceedings .
 iswc:proceedings swrc:address ?proceedingAddress.
 ?paper swrc:author ?author .
 ?author swrc:affiliation ?affiliation ;
 ?author rdfs:label ?fullnames ;
 ?author foaf:based_near ?place.
 ?place dbpedia:capital ?capital .
 ?place dbpedia:populationDensity ?populationDensity .
 ?place dbpedia:governmentType ?governmentType .
 ?place dbpedia:language ?language .
 ?place dbpedia:leaderTitle ?leaderTitle .
}


SELECT DISTINCT * WHERE
{
 ?paper swc:isPartOf iswc:proceedings .
 iswc:proceedings swrc:address ?proceedingAddress.
 ?paper swrc:author ?author .
 ?author swrc:affiliation ?affiliation .
 ?author rdfs:label ?fullnames .
 ?author foaf:based_near ?place.
 ?place ?mM9RIT ?capital .
 ?place ?cZP8iP ?populationDensity .
 ?place ?Pp7c1t ?governmentType .
 ?place ?z2uYJB ?language .
 ?place ?de7OQZ ?leaderTitle .
}

# Similarity: 0.07777777777
```

Listing 1.6: Query C9 followed by the candidate query C9'.

```
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX drugbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/
    drugbank/>

SELECT * WHERE
{
 ?Drug rdf:type dbpedia:Drug .
 ?drugbankDrug owl:sameAs ?Drug .
 ?InteractionName drugbank:interactionDrug1 ?drugbankDrug .
 ?InteractionName drugbank:interactionDrug2 ?drugbankDrug2 .
 ?InteractionName drugbank:text ?IntEffect .
 OPTIONAL
 {
  ?drugbankDrug  drugbank:affectedOrganism 'Humans and other mammals' .
  ?drugbankDrug drugbank:description ?description .
  ?drugbankDrug drugbank:structure ?structure .
  ?drugbankDrug drugbank:casRegistryNumber ?casRegistryNumber .
 }
}
ORDER BY (?drugbankDrug)


SELECT * WHERE
{
 ?Drug ?0OgzMk ?Cvqg5H .
 ?drugbankDrug ?c5DqMr ?Drug .
 ?InteractionName drugbank:interactionDrug1 ?drugbankDrug .
 ?InteractionName drugbank:interactionDrug2 ?drugbankDrug2 .
 ?InteractionName drugbank:text ?IntEffect .
 OPTIONAL
 {
  ?drugbankDrug  drugbank:affectedOrganism 'Humans and other mammals' .
  drugbank:description ?description .
  drugbank:structure ?structure .
  drugbank:casRegistryNumber ?casRegistryNumber .
 }
}
ORDER BY (?drugbankDrug)

# Similarity: 0.222222222222
```

Listing 1.7: Query C10 followed by the candidate query C10'.

```
PREFIX tcga: <http://tcga.deri.ie/schema/>
PREFIX kegg: <http://bio2rdf.org/ns/kegg#>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX drugbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/>
PREFIX purl: <http://purl.org/dc/terms/>

SELECT DISTINCT ?patient ?gender ?country ?popDensity ?drugName ?indication ?formula ?compound
WHERE
{
 ?uri tcga:bcr_patient_barcode ?patient .
 ?patient tcga:gender ?gender .
 ?patient dbpedia:country ?country .
 ?country dbpedia:populationDensity ?popDensity.
 ?patient tcga:bcr_drug_barcode ?drugbcr.
 ?drugbcr tcga:drug_name ?drugName.
 ?drgBnkDrg drugbank:genericName ?drugName.
 ?drgBnkDrg drugbank:indication ?indication.
 ?drgBnkDrg drugbank:chemicalFormula ?formula.
 ?drgBnkDrg drugbank:keggCompoundId ?compound .
}


SELECT DISTINCT ?patient ?gender ?country ?popDensity ?drugName ?indication ?formula ?compound
WHERE
{
 ?uri tcga:bcr_patient_barcode ?patient .
 ?patient tcga:gender ?gender.
 ?patient dbpedia:country ?country.
 ?country ?7sqC60 ?popDensity.
 ?patient tcga:bcr_drug_barcode ?drugbcr.
 ?drugbcr tcga:drug_name ?drugName.
 ?drgBnkDrg drugbank:genericName ?drugName.
 ?drgBnkDrg drugbank:indication ?indication.
 ?drgBnkDrg drugbank:chemicalFormula ?formula.
 ?drgBnkDrg drugbank:keggCompoundId ?compound .
}

# Similarity: 0.66666666666
```

Listing 1.8: Query L5 followed by the candidate query L5'.

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbprop: <http://dbpedia.org/property/>
PREFIX dbowl: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX factbook: <http://www4.wiwiss.fu-berlin.de/factbook/ns#>
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX fb: <http://rdf.freebase.com/ns/>

SELECT * WHERE {
 ?a dbowl:artist dbpedia:Michael_Jackson .
 ?a rdf:type dbowl:Album .
 ?a foaf:name ?n .
}


SELECT * WHERE {
 ?a dbowl:artist dbpedia:Michael_Jackson .
 ?a rdf:type dbowl:Album .
 ?a rdfs:label ?n .
}

# Similarity: 0.8115399282213058
```

Listing 1.9: Query L6 followed by the candidate query L6'.

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbowl: <http://dbpedia.org/ontology/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX linkedMDB: <http://data.linkedmdb.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX geo: <http://www.geonames.org/ontology#>

SELECT * WHERE {
 ?director dbowl:nationality dbpedia:Italy .
 ?film dbowl:director ?director.
 ?x owl:sameAs ?film .
 ?x foaf:based_near ?y .
 ?y geo:officialName ?n .
}


SELECT * WHERE {
 ?director dbowl:nationality dbpedia:Italy .
 ?film dbowl:director ?director.
 ?x owl:sameAs ?film .
 ?x ?6GKJwd ?y .
 ?y geo:officialName ?n .
}

# Similarity: 0.66666666666
```

Listing 1.10: Query L7 followed by the candidate query L7'.

```
PREFIX tcga: <http://tcga.deri.ie/schema/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?patient ?p ?o
WHERE
{
 ?uri tcga:bcr_patient_barcode ?patient .
 ?patient dbpedia:country ?country.
 ?country dbpedia:populationDensity ?popDensity.
 ?patient tcga:bcr_aliquot_barcode ?aliquot.
 ?aliquot ?p ?o.
}


SELECT DISTINCT ?patient ?p ?o
WHERE
{
 ?uri tcga:bcr_patient_barcode ?patient .
 ?patient dbpedia:country ?country .
 ?country ?cG4icP ?popDensity.
 ?patient tcga:bcr_aliquot_barcode ?aliquot .
 ?aliquot ?p ?o .
}

# Similarity: 0.66666666666
```

Listing 1.11: Query L8 followed by the candidate query L8'.

```
PREFIX kegg: <http://bio2rdf.org/ns/kegg#>
PREFIX drugbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/
    drugbank/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT * WHERE {
 ?drug drugbank:drugCategory drugbank:micronutrient .
 ?drug drugbank:casRegistryNumber ?id .
 ?drug owl:sameAs ?s .
 ?s foaf:name ?o .
 ?s skos:subject ?sub .
}


SELECT * WHERE {
 ?drug drugbank:drugCategory drugbank:micronutrient .
 ?drug drugbank:casRegistryNumber ?id .
 ?drug ?XKeC36 ?s .
 ?s rdfs:label ?o .
 ?s skos:subject ?sub .
}

# Similarity: 0.5410266188142039
```