The Jedi Approach: Using The Force to Solve Linked Data Incompleteness

Tutor: Maria-Esther Vidal
Task Force Name: Jedis
Students: Valentina Anita Carriero, David Chaves Fraga, Arnaud Grall, Lars Heling, Subhi Issa, Thomas Minier, Alberto Moya Loustaunau

1 - Introduction

Following the Linked Data principles, data providers have made available hundreds of RDF datasets [12]. The standardized approach to query this Linked Data is SPARQL, the W3C recommendation for querying RDF. Public SPARQL endpoints [13, 14] allow any data consumers to query RDF datasets on the Web, and federated SPARQL query engines [5,6,7,8] allow to query multiple datasets at once. The majority of these datasets have been created by integrating multiple, typically heterogeneous sources and exhibit issues concerning Linked Data validity, including data incompleteness. To illustrate, consider the datasets LinkedMDB and DBpedia and query Q1 (c.f. Figure 1) which retrieve all movies with their respective labels. Evaluating Q1 using a state-of-the-art federated SPARQL query engine over the federation only yields a single label for each movie. However, this result is considered incomplete, as not all relevant labels are provided, *i.e.*, no labels from DBpedia are retrieved. This is due to the fact that these engines are not able to detect incomplete answers and leverage the description of the sources to enhance answer completeness.

In this work, we propose a new adaptive approach for federated SPARQL query processing which estimates the answer completeness and uses enhanced source descriptions to complete the answers by taking as few additional sources into account as possible. More precisely, we address the following **research question**:
“Given a SPARQL query and a federation of SPARQL endpoints, how to minimize the number of sources to query during the execution while maximizing answer completeness?”.
Our contributions are as follows:

- We propose a framework, called **extended RDF Molecule Template** (eRDF-MTs), to describe an RDF dataset in terms of the RDF classes, their properties, and the similarity links between classes and properties across the federation. It also allows for detecting incompleteness.
- We propose a relevance-based **cost-model** leveraging eRDF-MT to select sources in order to improve answer completeness without compromising on query execution time.
- We propose a new **physical query operator**, the *Jedi operator*, which dynamically adds new sources during query execution according to the cost-model

The paper is organized as follows. Section 2 presents related work. Section 3 presents the problem statement, while Section 4 describes our main contributions. In Section 5 we experimentally study our approach. Finally, in Section 6, we conclude and outline future works.

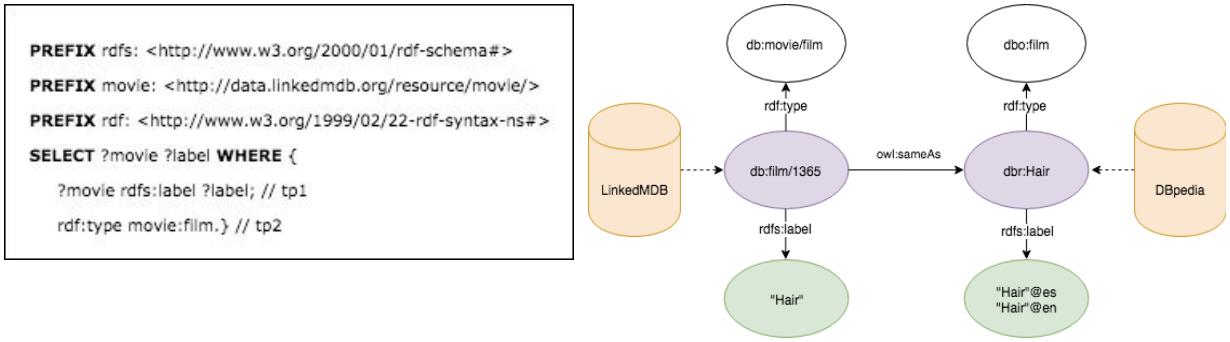

Figure 1: Motivating Example: incompleteness in SPARQL query results. On the left, a query to retrieve movies with their labels. On the right the property graph of the film "Hair"[1] with their respective values for the LinkedMDB dataset and DBpedia dataset. In green, all labels related to the film "Hair" for both datasets. LinkedMDB and DBpedia use different class names for movies resulting in incomplete results when executing a federated query.

2 - Related Work

In the following, we present the work related to our approach. First, we describe how a variety of federated SPARQL query engines select the relevant sources in the federation to minimize the execution time. Next, we present approaches addressing data incompleteness when querying Linked Data.

Federated SPARQL query engines [5, 6, 7, 8] are able to evaluate SPARQL queries over a set of data sources. FedX [5] is a federated SPARQL query engine introduced by Schwarte et al. It performs source selection by dynamically sending *ASK* queries to determine relevant sources and use bind joins to reduce data transfers during query execution. Anapsid [6] is an adaptive approach for federated SPARQL query processing. It adapts query execution based on the information provided by the sources, e.g., their capabilities or the ontology used to describe datasets. Anapsid also proposes a set of novel adaptive physical operators for query processing, which are able to quickly produce answers while adapting to network conditions.

Endris et al. [1] improve the performance of federated SPARQL query processing by describing RDF data sources in form of RDF molecule templates. RDF molecule templates (RDF-MTs) describe properties associated with entities of the same class available in a remote RDF dataset. RDF-MTs are computed for a dataset accessible via a specific web service. They can be linked to the same data set or across datasets accessible via other web services. MULDER [1] is a federated SPARQL query engine that leverages these RDF-MTs in order to improve source selection and reduce query execution time while increasing the answer completeness. MULDER decomposes a query into star-shaped subqueries and associates them with the RDF-MTs to produce an efficient query execution plan.

Finally, Fedra [3] and Lilac [4] leverages replicated RDF data in the context of a federated process. They describe RDF datasets using fragments, which indicates which RDF triples can be fetched from which data source. Using this information, they compute a replication-aware source selection and decompose SPARQL queries in order to reduce redundant data transfers due to data replication.

[1] http://dbpedia.org/page/Hair_%28film%29

However, neither of these approaches are able to detect data incompleteness in a federation. Furthermore, the presented source selection approaches will not be able to overcome semantic heterogeneity to improve answer completeness, as outlined in Section 1.

Acosta et al. [2] propose HARE, a hybrid SPARQL engine which is able to enhance the completeness of query answers using crowdsourcing. It uses a model to estimate the completeness of the RDF dataset. HARE can automatically identify parts of queries that yield incomplete results and retrieves missing values via microtask crowdsourcing. A microtask manager proposes questions to provide specific values to complete the missing results. Thus, HARE relies on the crowd to improve answer completeness and is not able to leverage linked RDF datasets.

We conclude that, to the best of our knowledge, no federated SPARQL query engine is able to tackle the issue of data incompleteness in the presented context.

3 - Proposed Approach

In our work, we rely on the assumptions that the descriptions of RDF datasets are computed and provided by data providers and that Linked RDF datasets are correct but potentially incomplete. Our approach is based on three keys contributions: (1) an extension of the RDF molecule template to detect data incompleteness, (2) a cost model to determine the relevancy of a source, and (3) a physical query operator which leverages the previous contributions to enhance answer completeness during query execution. An overview of the approach is provided in Figure 2.

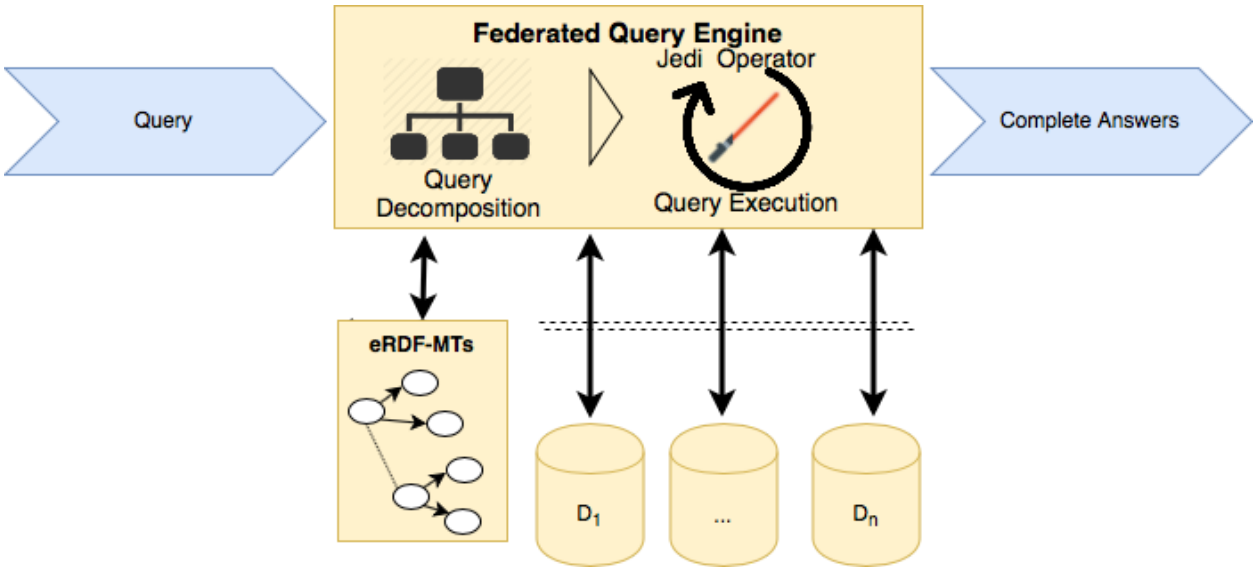

Figure 2: Overview of the approach. The figure depicts the query processing model. The engine gets a query as the input. During query execution, the Jedi operator leverages the eRDF-MTs of the data sources in the federation to increase answer completeness. Finally, the complete answers are returned.

3.1. Problem Statement

First, we formalize the problem of data incompleteness and provide the notion of an oracle as a reference point for our definition.

Given a set of RDF datasets F = {D_1, ..., D_n} and a SPARQL query Q to be evaluated over F, i.e., [[Q]]F. Consider O, the oracle dataset that contains all the data about each entity in the federation. Answer completeness for Q, with respect to O, is defined as [[Q]]F = [[Q]]O.

Research problem:
The problem of evaluating a complete federated SPARQL query over F is:
$\min(|[[Q]]O| - |[[Q]]F^*|)$ s.t. $F^* \subseteq F$ and $\min(|F^*|)$.

In other words, the problem is to find the minimal set of sources in F to use during query execution in order to maximize answer completeness.

3.2 - Extended RDF Molecule template

Next, to tackle the problem of detecting data incompleteness, we rely on the HARE [1] RDF completeness model. We now introduce key notions from this model that we are going to use. HARE is able to estimate that answers to a SPARQL query might be incomplete by leveraging the multiplicity of resources.

Definition 1 (Predicate Multiplicity of an RDF Resource [2])
Given an RDF resource occurring in the data set D, the multiplicity of the predicate p for the resource s in D, denoted MD(s|p), is $MD(s|p) := |\{o \mid (s,p,o) \in D\}|$.

Example 1: Consider the RDF dataset from Figure 1. The predicate multiplicity of the predicate "rdfs:label" for the resource "dbr:Hair" is MD(dbr:Hair | rdfs:label) = 2, because the resource is connected to two labels.

Next, using resource multiplicity, HARE computes the aggregated multiplicity for each RDF class in the dataset.

Definition 2 (Aggregated Predicate Multiplicity of a Class [2])
For each class C occurring in the RDF data set D, the aggregated multiplicity of C over the predicate p, denoted AMD(C|p), is: $AMD(C|p) := f(\{MD(s|p)|(s,p,o) \in D \wedge (s,a,C) \in D\})$ where: f(s,a,C) corresponds to the triple (s,rdf:type,C), which means that the subject s belongs to the class C, and f(.) is an aggregation function.

Example 2: Consider again the RDF dataset from Figure 1, and an aggregation function f that computes the median. The aggregated predicate multiplicity of the class "dbo:film" over the predicate "rdfs:label" is AMD(dbo:film | rdfs:label) = 2.

However, HARE's completeness model is not designed to be used in a federated scenario, as it can only be computed on a single dataset. To address this issue, we introduce a novel source description, called extended RDF Molecule template (eRDF-MT), based on RDF-MTs [2]. An eRDF-MT, defined in Definition 3, describes each dataset of the federation as the set of properties that are associated with each RDF class. It also performs the interlinking of RDF class between datasets, to be able to find equivalent entities across the federation. Finally, eRDF-MTs also capture the equivalence between properties, in order to capture the semantic heterogeneity of datasets.

Definition 3 (Extended RDF Molecule Template (eRDF-MT))

An Extended RDF Molecule Template is a 7-tuple=<W, C, f, DTP, IntraC, InterC, InterP> where:

- **W** is a Web service API that provides access to an RDF dataset G via SPARQL protocol;
- **C** is an RDF class such that the triple pattern (?s rdf:type C) is true in G;
- **f** is an aggregation function;
- **DTP** is a set of pairs (p, T, f(p)) such that p is a property with domain C and range T, and the triple patterns (?s p ?o), (?o rdf:type T) and (?s rdf:type C) are true in G. f(p) is the *aggregated multiplicity* of predicate p for class C;
- **IntraC** is a set of pairs (p, Cj) such that p is an object property with domain C and range Cj, and the triple patterns (?s p ?o) and (?o rdf:type Cj) and (?s rdf:type C) are true in G;
- **InterC** is a set of 3-tuples (p, Ck, SW) such that p is an object property with domain C and range Ck; SW is a Web service API that provides access to an RDF dataset K, and the triple patterns (?s p ?o) and (?s rdf:type C) are true in G, and the triple pattern (?o rdf:type Ck) is true in K.
- **InterP** is a set of 3-tuples (p, p', SW) such that p is a property with domain C and range T, SW is a Web service API that provides access to an RDF dataset K and p' is a property with domain C' and range T' such as the triples (p owl:sameAs p') or (p' owl:sameAs p) exists in G or K.

The idea is to estimate the expected cardinalities of each property for each class in the data set. Thus, if the query engine finds fewer results for an entity of that class and a property than estimated by the eRDF-MT, it would consider the results to be incomplete. In this case, we assume that connected datasets in the eRDF-MT can be used to complete the missing values. Figure 3 provides an example of two eRDF-MTs.

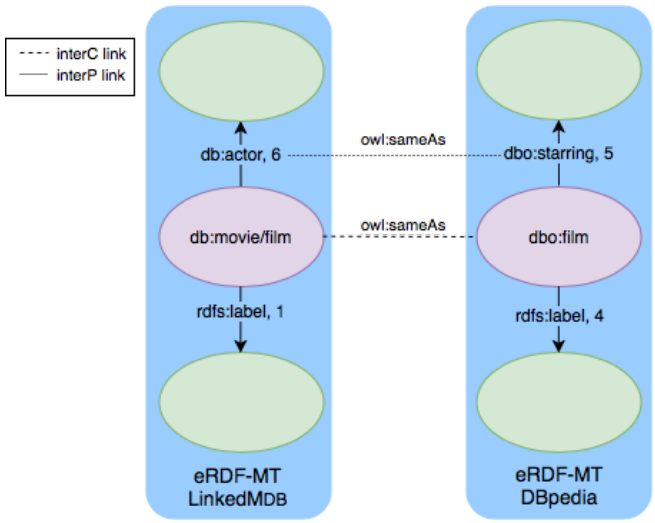

Figure 3: An example of two interlinked eRDF-MT for the data sources LinkedMDB (left) and DBpedia (right). InterC and InterP provide links between the classes and properties in the different data sources. Additionally, the aggregated multiplicity of each predicate is displayed next to the predicates.

3.3 - The Jedi Cost model

We introduce a cost-model which relies on the eRDF-MTs to detect RDF datasets that can be used to complete query results, and estimate the relevance of these RDF datasets. This cost-model aims to solve our research problem, by selecting the minimal number of sources to contact. First, we formalize in Definitions 4 and 5 how to compute the relevant eRDF-MT that can be used to enhance the results when evaluating a given triple pattern in the federation.

Definition 4 (Relevant eRDF-MTs)
Given a triple pattern tp = (s, p, o), a root eRDF-MT r = <W,C,f,DTP,IntraC,InterC,InterP> and a set of eRDF-MTs M = {m_1, …, m_n }, where m_i summarizes the dataset D_i.
The *set of relevant eRDF-MTs* for tp and r are defined as R(tp, r) = { m_i | forall m_i in M, m_i = <W',C',f',DTP',IntraC',InterC',InterP'> such as there exists (p'',C,W') in InterC and there exists (p, W', p') in InterP }

In other word, an eRDF-MT is considered to be relevant with respect to the root eRDF-MT, if it contains the same class (potentially with a different identifier) and the class has the same predicate (potentially also with a different identifier) as the triple tp.

Definition 5 (Relevance of eRDF-MT)
Given a triple pattern tp = (s, p, o) and a eRDF-MT m = <W,C,f,DTP,IntraC,InterC,InterP>, the *relevance* τ of m for tp is τ(tp, m) = f(p) if there exists a (p, T, f(p)) in DTP.

Using these relevant eRDF-MTs, we next devise a strategy to minimize the number of relevant sources to select by ranking sources according to their relevance, formalized in Definition 6.

Definition 6 (Ranking relevant eRDF-MTs)
Given a triple pattern tp = (s, p, o), a root eRDF-MT r = <W,C,f,DTP,IntraC,InterC,InterP> and the set of relevant eRDF-MTs R(tp, r) = {m_1, …, m_k }. The ranking of R(tp, r) is R(tp, r) where eRDF-MTs are sorted by descending relevance.

3.4 - The Jedi operator for Triple Pattern evaluation

Federated SPARQL query engine evaluates SPARQL query for building a plan of physical query operators [8]. We choose to implement our approach as a physical query operator for triple pattern evaluation, named Jedi operator, in order to ease the integration of this operator in an existing federated SPARQL query engine. Thus, it can be used with state of the art physical operator, like Symmetric Hash Join [9] or Bind Join [5,10], to handle query execution.

The Jedi operator follows interlinking between eRDF-MTs using a breadth-first approach to find additional data during query execution. The algorithm of the operator is shown in Figure 4. The inputs are a triple pattern, a root eRDF-MT (from which the computation will start) and a set of eRDF-MTs for the data sources in the federation. Starting with the root eRDF-MT, the Jedi operator first evaluates the triple pattern at the associated data source (Line 1-6). Then, if the results are incomplete according to the aggregated multiplicity, it uses the Jedi cost-model to find relevant datasets to use (Lines 7-8) and selects the more relevant one to continue query execution (Line 14). Next, it performs a *triple pattern mapping* (Line 12) using the property interlinks of eRDF-MTs, to maps the triple pattern to the schema used by the newly found dataset. The operator terminates if there the results are considered complete regarding the expected aggregated multiplicity, or if no more relevant eRDF-MTs to use to improve answer completeness.

Algorithm 1: The JEDI operator

Input: $tp = (s, p, o)$: a triple pattern, $root$: a root eRDF-MT, $\mathcal{D}$: a root RDF dataset, $M = \{m_1, \ldots, m_n\}$: a set of eRDF-MT

Function *EvaluatePattern(tp, root, $\mathcal{D}$, M,)*:
- $\Omega \leftarrow \emptyset$
- $T \leftarrow \{tp\}$
- **while** $|\Omega| < \tau(tp, root)$ **do**
  - $tp' \leftarrow T.pop()$ `// Get next pattern to evaluate`
  - $\Omega \leftarrow \Omega \cup [\![tp']\!]_{\mathcal{D}}$
  - **if** $T = \emptyset$ **then**
    - $R(tp', root) \leftarrow GetRelevantMolecules(tp', M)$
    - `// Found relevant datasets`
    - **if** $R(tp', root) \neq \emptyset$ **then**
      - Rank $R(tp', root)$ using the Jedi cost-model
      - $\xi \leftarrow$ the top ranked eRDF-MT in $R(tp, root)$
      - $T \leftarrow \{(s, p', o) \mid \forall (p, W, p') \in InterP \text{ of } \xi\}$
      - $M \leftarrow M \backslash \{\xi\}$
      - $\mathcal{D} \leftarrow$ the RDF dataset associated with ξ
    - **else**
      - `// No more relevant datasets to use`
      - **break**
- **return** Ω

Figure 4: The Jedi operator algorithm evaluates a triple pattern using eRDF-MTs

4 - Evaluation and Results

In the evaluation, we consider five queries evaluate over two data sets in order to determine the impact of our approach on answer completeness. Each query is associated with a certain domain in order to show that completeness issues are distributed over different parts of the data. The federation contains the data sets DBpedia and Wikidata and we assume Wikidata as a mirror data set of DBpedia. This means that, according to our cost-model, Wikidata is queried only in case the results from DBpedia are estimated to be incomplete. The original queries and the rewritten queries are provided in Appendix A of this work.

For the sake of brevity, we discuss how the evaluation query q1 in the following. In the query, we want to determine the position, date of birth and the team for soccer players. When evaluating the query over DBpedia, we retrieve no results. However, the results are incomplete when considering Wikidata as well. Rewriting the query according to our proposed approach and executing it over the federation of both data sets, we find that there are 42 results. As shown in Table 1, similar results can be observed for the other queries as well.

The results of this first evaluation clearly indicates the potential of our approach to increase answer completeness over a federation of data sets. We expect similar results in other domains and for other data sets as well.

Table 1: Results of our preliminary evaluation. The table shows the number of answers for 5 queries evaluated over the data set DBpedia and the corresponding rewritten queries evaluated over the federation of DBpedia and Wikidata.

Domain	Query	DBpedia	DBpedia + Wikidata
Sport	q1	0	42
Movies	q2	3	6
Culture	q3	0	31
Drugs	q4	0	482
Life Sciences	q5	0	9

5 - Conclusion and Future Work

In this paper, we proposed Jedi, a new adaptive approach for federated SPARQL query processing, which is able to estimate data incompleteness and uses links between classes and properties in different RDF datasets to improve answer completeness. It relies on extended RDF Molecule Templates, which describe the classes, properties as well as the links between data sources. Furthermore, by including the aggregated predicate multiplicity of entities, they allow for detecting incompleteness during query execution. Using these RDF-MTs and a cost-model, the Jedi operator is able to discover new data sources to improve answer completeness.

The results of our evaluation shows that answer incompleteness is presented in various domains of the well-known data sets DBpedia. Furthermore, we show that using our approach to rewritten according to the presented approach will increase the completeness of the results.

Our approach suffers from one main limitation: it assumes that eRDF-MTs are pre-computed and published by data providers. We also suppose that data providers are aware of the interlinking between their datasets. One perspective is to research how these eRDF-MTs can be computed by data consumers instead, in order to reduce the dependence on data providers.
In the future, we also aim to integrate the Jedi operator in a state-of-the-art federated SPARQL query engines, like FedX [5], MULDER [2] or Anapsid [6], in order to conduct a more elaborate experimental study of our approach. According to this study, we will then improve on our approach to maximize the answer completeness.

Appendix A - Evaluated and Rewritten Queries

Domain - Sport

1. Starting Query

```
select distinct ?s ?o1 where {
```

```
        ?s rdf:type dbo:SoccerPlayer.
        ?s <http://dbpedia.org/property/team> ?o1 .
        ?s <http://dbpedia.org/ontology/birthPlace> ?o2 .
        ?s <http://dbpedia.org/ontology/position> ?o4
}
```

2. Rewritten Query

```
select distinct ?s ?o1 where {
        ?s rdf:type dbo:SoccerPlayer.
        ?s <http://dbpedia.org/ontology/birthPlace> ?o2 .
        ?s <http://www.w3.org/2002/07/owl#sameAs> ?o3 .
        ?s <http://dbpedia.org/ontology/position> ?o4 .
        ?o3 <http://www.wikidata.org/prop/direct/P54> ?o1
}
```

Domain - Movies

1. Starting Query

```
select distinct ?s ?o5 where {
        ?s rdf:type dbo:Film .
        ?s <http://dbpedia.org/ontology/director> ?o1 .
        ?s <http://dbpedia.org/ontology/producer> ?o2 .
        ?s <http://dbpedia.org/property/language> ?o3 .
        ?s <http://dbpedia.org/property/editing> ?o5 .
}
```

2. Rewritten Query

```
select distinct ?s ?o5 where {
        ?s rdf:type dbo:Film .
        ?s <http://dbpedia.org/ontology/director> ?o1 .
        ?s <http://dbpedia.org/ontology/producer> ?o2 .
        ?s <http://dbpedia.org/property/language> ?o3 .
        ?s <http://www.w3.org/2002/07/owl#sameAs> ?o4 .
        ?o4 <http://www.wikidata.org/prop/direct/P1040> ?o5 .
}
```

Domain - Culture

1. Starting Query

```
select distinct ?s ?o4 where {
       ?s rdf:type dbo:Artist .
       ?s <http://dbpedia.org/ontology/birthName> ?o1 .
       ?s <http://dbpedia.org/ontology/birthPlace> ?o2 .
       ?s <http://dbpedia.org/ontology/birthDate> ?o3 .
```

```
        ?s <http://dbpedia.org/ontology/occupation> ?o4.

}
```

2. Rewritten Query

```
select distinct ?s ?o5 where {
        ?s rdf:type dbo:Artist .
        ?s <http://dbpedia.org/ontology/birthName> ?o1 .
        ?s <http://dbpedia.org/ontology/birthPlace> ?o2 .
        ?s <http://dbpedia.org/ontology/birthDate> ?o3 .
        ?s <http://www.w3.org/2002/07/owl#sameAs> ?o4.
        ?o4 <http://www.wikidata.org/prop/direct/P106> ?o5
}
```

Domain - Drugs

1. Starting Query

```
select distinct ?s ?o4 where {
        ?s rdf:type dbo:Drug .
        ?s <http://dbpedia.org/ontology/drugbank> ?o1 .
        ?s <http://dbpedia.org/property/routesOfAdministration> ?o2 .
        ?s <http://dbpedia.org/property/molecularWeight> ?o3 .
        ?s <http://dbpedia.org/property/drugName> ?o4 .
}
```

2. Rewritten Query

```
select distinct ?s ?o5 where {
        ?s rdf:type dbo:Drug .
        ?s <http://dbpedia.org/ontology/drugbank> ?o1 .
        ?s <http://dbpedia.org/property/routesOfAdministration> ?o2 .
        ?s <http://dbpedia.org/property/molecularWeight> ?o3 .
        ?s <http://www.w3.org/2002/07/owl#sameAs> ?o4 .
        ?o4 <http://www.w3.org/2000/01/rdf-schema#label> ?o5
}
```

Domain - Life Science

1. Starting Query

```
select distinct ?s ?o4 where {
        ?s rdf:type dbo:Disease .
        ?s <http://dbpedia.org/ontology/icd9> ?o1 .
        ?s <http://dbpedia.org/property/diseasesdb> ?o2.
        ?s <http://dbpedia.org/ontology/icd10> ?o3 .
        ?s <http://dbpedia.org/ontology/meshId> ?o4.
}
```

2. Rewritten Query

```
select distinct ?s ?o5 where {
        ?s rdf:type dbo:Disease .
```

```
        ?s <http://dbpedia.org/ontology/icd9> ?o1 .
        ?s <http://dbpedia.org/property/diseasesdb> ?o2.
        ?s <http://dbpedia.org/ontology/icd10> ?o3 .
        ?s <http://www.w3.org/2002/07/owl#sameAs> ?o4 .
        ?o4 <http://www.wikidata.org/prop/direct/P486> ?o5
}
```

References

[1] Endris, K. M., Galkin, M., Lytra, I., Mami, M. N., Vidal, M. E., Auer, S: MULDER: querying the linked data web by bridging RDF molecule templates. In International Conference on Database and Expert Systems Applications (pp. 3-18). Springer, Cham.

[2] Acosta, M., Simperl, E., Flöck, F., Vidal, M. E.: Enhancing answer completeness of SPARQL queries via crowdsourcing. Web Semantics: Science, Services and Agents on the World Wide Web, 45, 41-62.

[3] Montoya, G., Skaf-Molli, H., Molli, P., Vidal, M.E.: Federated SPARQL queries processing with replicated fragments. In: International Semantic Web Conference. pp. 36–51. Springer (2015)

[4] Montoya, G., Skaf-Molli, H., Molli, P., Vidal, M.E.: Decomposing federated queries in presence of replicated fragments. Web Semantics: Science, Services and Agents on the World Wide Web 42, 1–18 (2017)

[5] Schwarte, A., Haase, P., Hose, K., Schenkel, R., & Schmidt, M. (2011, October). Fedx: Optimization techniques for federated query processing on linked data. In International Semantic Web Conference (pp. 601-616). Springer, Berlin, Heidelberg.

[6] Acosta, M., Vidal, M. E., Lampo, T., Castillo, J., Ruckhaus, E.:ANAPSID: an adaptive query processing engine for SPARQL endpoints. In International Semantic Web Conference (pp. 18-34). Springer, Berlin, Heidelberg.

[7] Görlitz, O., & Staab, S. (2011, October). Splendid: Sparql endpoint federation exploiting void descriptions. In Proceedings of the Second International Conference on Consuming Linked Data-Volume 782 (pp. 13-24).

[8] Görlitz, O., Staab, S.: Federated data management and query optimization for linked open data. In: New Directions in Web Data Management 1, pp. 109–137. Springer (2011)

[9] A. N. Wilschut and P. M. G. Apers, “Dataflow query execution in a parallel main-memory environment,” in PDIS ’91: Proceedings of the First International Conference on Parallel and Distributed Information Systems, pp. 68–77, IEEE Computer Society, (1991).

[10] Haas, L.M., Kossmann, D., Wimmers, E.L., Yang, J.: Optimizing Queries Across Diverse Data Sources. In: Proc. of the 23rd Int. Conf. on Very Large Data Bases (VLDB) (1997)

[12] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. Adoption of the Linked Data best practices in different topical domains. In ISWC, pages 245–260. 2014.

[13] Vrandečić, Denny, and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase." *Communications of the ACM*57.10 (2014): 78-85.

[14] Auer, Sören, et al. "Dbpedia: A nucleus for a web of open data." *The semantic web*. Springer, Berlin, Heidelberg, 2007. 722-735.