Querying the Web of Interlinked Datasets using VOID Descriptions

Ziya Akar
Department of Computer Engineering, Ege University
35100 Bornova, Izmir, Turkey
ziya.seagent@gmail.com

Tayfun Gökmen Halaç
Department of Computer Engineering, Ege University
35100 Bornova, Izmir, Turkey
tayfunhalac@gmail.com

Erdem Eser Ekinci
Department of Computer Engineering, Ege University
35100 Bornova, Izmir, Turkey
erdemeserekinci@gmail.com

Oguz Dikenelli
Department of Computer Engineering, Ege University
35100 Bornova, Izmir, Turkey
oguz.dikenelli@ege.edu.tr

ABSTRACT

Query processing is an important way of accessing data on the Semantic Web. Today, the Semantic Web is characterized as a web of interlinked datasets, and thus querying the web can be seen as dataset integration on the web. Also, this dataset integration must be transparent from the data consumer as if she is querying the whole web. To decide which datasets should be selected and integrated for a query, one requires a metadata of the web of data. In this paper, to enable this transparency, we introduce a federated query engine called WoDQA (Web of Data Query Analyzer) which discovers datasets relevant with a query in an automated manner using VOID documents as metadata. WoDQA focuses on powerful dataset elimination by analyzing query structure with respect to the metadata of datasets. Dataset and linkset descriptions in VOID documents are analyzed for a SPARQL query and a federated query is constructed. By means of linkset concept of VOID, links between datasets are incorporated into selection of federated data sources. Current version of WoDQA is available as a SPARQL endpoint.

1. INTRODUCTION

While the web is evolving through a structured data space, many applications are publishing and linking their data, and the cloud of this linked and open data will be gigantic as times go. In this interlinked and structured data space, query execution becomes one of the most important research problems and different query execution approaches and tools have been proposed in the literature [11, 7]. The query execution on the web of data is basically depends on searching for resources that satisfy our needs, but we need to discover which parts of linked open data cloud may have such resources. To make this discovery effectively, which resources and vocabularies reside in a dataset and which datasets are interlinked to others via interested links should be taken into account. If dataset publishers provide such information by describing metadata of their datasets, relevant datasets can be selected effectively in an automated manner. To enable this automation, Vocabulary of Interlinked Datasets (VOID) [1] is published as W3C Semantic Web Interest Group note[1]. VOID is an RDF vocabulary and is used to describe metadata of RDF datasets, in a sense, metadata of the web of data. Linked open data cloud is represented as a graph of datasets in which datasets are represented as nodes and sets of links between datasets are represented as edges. Since VOID bases on graph based soul of web of data, it provides a strong way of describing metadata that allows to discover datasets which queries are distributed over.

In this paper, we present a federated query engine called WoDQA (Web of Data Query Analyzer) which is developed to execute a query on distributed datasets without missing answers using VOID metadata of datasets in linked open data cloud. WoDQA focuses on effective dataset selection for a query and analyzes query structure to eliminate irrelevant datasets. Relevant datasets are selected by analyzing VOID documents and considering which dataset includes a resource related with the query and which links between datasets allow to find a result to the query. VOID metadata provides dataset descriptions representing content of a dataset and linkset descriptions representing relationships between datasets which are used by WoDQA for effective dataset selection.

There are two main approaches which enable automated query processing on the web of data and prevent data consumers from searching for relevant datasets. The first approach called *follow-your-nose* (link traversal) [11] is based on following links between data to discover potentially relevant data, and the second one is *query federation* [7] which is based on dividing a query into sub-queries and distributing sub-queries to relevant datasets which are selected using metadata about datasets.

Follow-your-nose approach conceptualizes the web as a graph of documents which contains dereferenceable URIs. This approach is based on executing queries on relevant documents which are retrieved by following links between resources in different documents. But, this method raises completeness and performance issues. Although some heuristic query planning methods can be used to answer different kinds of queries [10], this approach cannot guarantee finding all results because relevant documents vary according

LDOW2012, April 16, 2012, Lyon, France.

[1] http://www.w3.org/TR/void/

to the starting point and the path. Also, although follow-your-nose requires nothing other than linked data principles to process a query, another disadvantage is that encountering large documents causes retrieval problems. The other approach, query federation, has raised from database literature, and is composed of two main steps before performing a query. Firstly, query is divided into sub-queries and datasets relevant with sub-queries are selected using some metadata which reflects dataset content. Then, the query evaluation plan is changed using statistics about datasets in the query optimization step. For the purpose of executing sub-queries on distributed data sources, query federation requires accessing datasets via SPARQL endpoints. Contrary to follow-your-nose approach, in this approach, all results can be found under the assumption of metadata of all datasets is complete and accurate, and queries can be optimized before execution by estimating execution using dataset metadata. To find all results in an effective way, query federation determines relevant datasets before execution using well-defined dataset metadata such as VOID documents.

In the light of these ideas, WoDQA executes queries by analyzing VOID documents which constitute a projection of the web of data and incorporates follow-your-nose approach into query federation by considering links between datasets in metadata. WoDQA does not change the evaluation order of a query because the main focus of this initial version of WoDQA is only eliminating much more irrelevant datasets in dataset selection without query optimization. Current RDF federation implementations select relevant datasets by considering only predicate and type indexes. Since vocabularies in the Semantic Web should be common, there can be a lot of datasets which use a specific property or class. Therefore, using such indexes causes selection of redundant datasets. The main contribution of WoDQA is incorporating both links between data through linkset concept and relationships between triple patterns of a query into dataset selection to eliminate irrelevant datasets effectively. We serve WoDQA as a SPARQL endpoint and a simple web form[2] to execute raw queries by analyzing datasets in the VOID stores.

Remaining sections are organized as follows. In Section 2, related work is discussed. Section 3 introduces general architecture of WoDQA and details dataset selection approach. In Section 4, usage of WoDQA is shown with a working example. Finally, Section 5 concludes the paper.

2. RELATED WORK

The Semantic Web querying approaches can be classified as centralized and distributed. Centralized querying is based on collecting linked data into a single central data store, and querying the data from this store. This approach includes data warehousing which collects pre-selected data sources and search engines which crawl the Web by following RDF links and index discovered data [12]. But, the main disadvantage of this approach is that queried data is not live, i.e. duplicate of original sources. On the other hand, search engines cannot crawl all the web and cannot answer complete structured queries.

[2]The simple web form and up-to-date endpoint address can be found on http://seagent.ege.edu.tr/etmen/wodqa.html page.

On the other hand, distributed querying depends on processing query parts directly on original data and managing results retrieved from distributed data. Query federation [9, 7] and follow-your-nose [11] are mainstream distributed querying approaches. DARQ [14], FedX [15] and SPLENDID [8] are the example implementations of the query federation approach. DARQ distributes a query using dataset metadata called Service Descriptions[3] which are constructed manually by query developer, and benefits from triple and entity counts and selectivity estimates to optimize the query plan. Since DARQ uses predicates to select relevant datasets, the success of the query execution depends on associating datasets with predicates, and triple patterns which have unbound predicates cannot be handled[4]. On the other hand, FedX is an extended version of Federation SAIL provided by AliBaba[5]. Datasets which will be queried are given to FedX, and it checks each triple pattern existence on each dataset by ASK queries to decide about which triple pattern will be queried on which datasets [16]. These two query federation implementations also stand up to self-descriptive nature of linked data since metadata of datasets should be described by data publishers as is in describing and linking their data. The last query federation implementation is SPLENDID which indexes dataset using VOID descriptions, eliminates datasets by ASK queries for triple patterns, and benefits from statistical data in VOID to optimize federated queries.

Although aforementioned query federation implementations aim to query linked datasets, they do not consider links between data for dataset selection. For this reason, there are some shortcomings of these implementations from querying web of data perspective. The first one is that deciding datasets via only predicate indexes causes inability to select datasets effectively for triple patterns which have unbounded predicates or have so general predicates such as owl:sameAs and foaf:page that are extensively used in datasets[6]. The second shortcoming is that so many datasets may be selected for triple patterns, and executing ASK queries in such a case increases the cost notably. One need to take the structure of the query into account to eliminate right irrelevant datasets in the web of data context.

The second distributed querying approach is follow-your-nose [11] whose basic idea is traversing RDF links between data to discover relevant datasets. There is no need to any prior metadata about datasets in advance as in query federation, but it needs initial URIs in some triple patterns to start exploring datasets. The main disadvantages of this approach are infinite link discovery, trying to retrieve large RDF graphs, failing to discover relevant data for queries with only bound predicates (?s foaf:friend ?o) or type statements (?s rdf:type foaf:Person). These restrictions cause less comprehensive result sets. One of well-known follow-your-nose implementations is SQUIN [11] that traverse RDF links on the fly, i.e. during query execution. Hartig et al. improve this work using some heuristic methods that modify query

[3]Service Description introduced in that paper contains information about triples in the dataset, limitations on access patterns, and statistical information about dataset.
[4]http://darq.sourceforge.net/#Limitations_and_known_issues
[5]http://www.openrdf.org/doc/alibaba/2.0-beta6/alibaba-sail-federation/
[6]SPLENDID also uses type indexes, but it is still not enough since vocabularies can be used frequently.

evaluation order to reduce execution cost and to provide more comprehensive results [10], but the results strictly depend on the starting point and the evaluation order. On the other hand, Bouquet et al. formalize the web of data and suggest three different querying methods exploiting their web of data formalization [4]. These methods based on merging relevant graphs to execute queries on them. One of these methods uses follow-your-nose approach which specifies and merges relevant graphs by looking up URIs before query execution.

WoDQA aims to query the web of interlinked datasets using VOID dataset and linkset descriptions to decide relevant datasets for a query. At first, it assumes that all datasets are relevant with a query, then irrelevant datasets are eliminated by analyzing query structure in the light of metadata of datasets. Its novelty is considering query structure and links between datasets to select relevant datasets before query execution, and thus it incorporates follow-your-nose approach into the query federation. To the best of our knowledge, WoDQA is the first query engine which uses datasets and linksets together that are critical elements of VOID to describe dataset metadata.

3. WODQA INTERNAL ARCHITECTURE

In this section, query processing architecture of WoDQA is explained in detail. Since it is impractical to perform a query on all published datasets on the web, WoDQA aims to transform a query into a federated query which is evaluated only on relevant datasets. In this direction, to process a query on the linked data cloud, WoDQA contains three main modules as seen in Figure 3.1: *DatasetAnalyzer, QueryReorganizer* and *Jena ARQ*[7].

Dataset publishers construct the VOID documents of their datasets and the Semantic Web programmers can access these documents through services called VOID store such as voiD Browser[8], CKAN[9] and voiDStore[10]. A VOID store generates a projection of Linked Open Data, and thus this structure obliges dataset publishers to create well-defined VOID document which reflects actual content of the dataset to enable including the dataset in relevant queries. DatasetAnalyzer is the module which is responsible for discovering relevant datasets and eliminating irrelevant ones using VOID documents of datasets in the VOID stores. We assume that dataset publishers update the description documents in the VOID stores to make VOID stores up-to-date for dataset selection when datasets are changed. In the current version of WoDQA, DatasetAnalyzer discovers the VOID documents from the CKAN net, and analyzes dataset and linkset descriptions for each triple pattern in the query. This analysis eliminates irrelevant datasets which definitely do not contain any result contributing to the result of the query by assuming that accurate and complete VOID documents of datasets are available. Dataset analysis is achieved by a rule-based approach. We explain the rules which discovers relevant datasets Subsection 3.1.

The second module is QueryReorganizer which rewrites queries depending on results of DatasetAnalyzer. This rewriting process constructs federated SPARQL queries including

[7]http://jena.sourceforge.net/ARQ/
[8]http://kwijibo.talis.com/voiD/
[9]http://ckan.net/
[10]http://void.rkbexplorer.com/

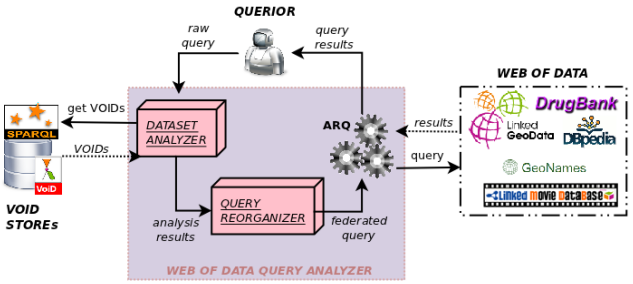

Figure 3.1: WoDQA internal architecture

SERVICE expressions[11]. Details of QueryReorganizer are given in Subsection 3.2.

The last module is query executor which directly uses Jena ARQ to execute SPARQL queries including the SERVICE expressions inserted by the QueryReorganizer. The federated query constructed by QueryReorganizer is passed to ARQ to be executed. Results of query execution are returned to the querior. In the following subsections, the first two modules which implement WoDQA analysis and reorganization phases are explained.

3.1 Dataset Analyzer

This section introduces the details of the DatasetAnalyzer module which is the core and the innovative part of the current version of WoDQA. Unlike other query federation approaches, WoDQA considers triple pattern relations and links between datasets while selecting datasets. Thanks to dataset analysis of WoDQA, relevant datasets are specified while plenty of irrelevant ones are excluded. Output of dataset analysis is a subset of all published datasets on the web of data, and thus the query is performed only on this subset including related ones. For the purpose of explaining how this subset is constructed, we give a formalization in this section.

We firstly give a definition of the web of data to formalize our dataset selection approach. In summary, the web of data is an RDF graph which is constructed by typed links between data from different sources. Basically an RDF graph ($\mathcal{G}$) is formally represented as a set of triples in the form of $\langle s, p, o\rangle$: $\mathcal{G} = \{\langle s, p, o\rangle \mid \langle s, p, o\rangle \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})\}$ where $\mathcal{I}$ is the set of IRIs[6], $\mathcal{B}$ is the set of blank nodes, $\mathcal{L}$ is the set of literals, and all are RDF terms $\mathcal{T} = \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$. In this direction, web of data is the global graph ($\mathcal{G}_{wod}$) which consists of the triples constructed from IRIs, blank nodes and literals on the web. $\mathcal{G}_{wod}$ is a model of mathematical RDF construct for the web of data.

From another perspective, the web of data means web of interlinked datasets [3]. A dataset (δ) is a meaningful set of RDF triples [1] which decreases granularity of the web. Rather than publishing information only as single resources and connecting these resources, datasets are the way of publishing information as sub-graphs of $\mathcal{G}_{wod}$. These subgraphs, i.e. datasets, are connected via RDF triples which connect resources in different datasets [13]. The publishers create their resources and deploy them into the datasets on the web, and consumers use these resources while creating their datasets. With regard to this, to formalize a dataset, we use $subj\,(\mathcal{G})$ which represents the set of resources which

[11]http://www.w3.org/TR/sparql11-federated-query/

are the subjects of triples in a graph.

Definition 1. A dataset is a sub-graph of web of data, $\delta_x \subset \mathcal{G}_{wod}$, and the resources which are included by δ_x are specified as follows: $\forall r, \delta_i \left(r \in subj\left(\delta_x\right) \rightarrow Owner\left(\delta_x, r\right)\right)$.

VOID describes a dataset with well-defined properties[12], and we formalize a VOID dataset description as a tuple $\langle \mathcal{L}^{space}, \mathcal{I}^{voc} \rangle \in \mathcal{L} \times \mathcal{I}$. The first dataset property $\mathcal{L}^{space}$ corresponding to void:uriSpace set which contains string literals that all entity IRIs in a dataset start with. The other one is $\mathcal{I}^{voc}$ corresponding to void:vocabulary which denotes the set of vocabularies used by the dataset[13].

The triples whose object is a resource in another dataset make the web of data a graph of interlinked datasets. We call such triples *link triples*, and define the set of link triples as $\mathcal{LT} = \{(s,p,o) \,|owner\left(s\right) \neq owner\left(o\right)\}$ where $s, o \in \mathcal{I}$. This definition leads us to define link predicate (p^{link}) concept which corresponds to void:linkPredicate used to define a linkset which is an important contribution of VOID effort. Set of link predicates $\left(\mathcal{P}^{link}\right)$ includes all the predicates which are used in a link triple: $\mathcal{P}^{link} = \{p^{link} | \exists \left\langle s, p^{link}, o \right\rangle \in \mathcal{LT}\}$. A linkset represents link triples which connect resources in different datasets using the same link predicate. We formalize the linkset, λ, as a tuple $\left\langle \delta_\lambda^{from}, \delta_\lambda^{to}, p_\lambda^{link} \right\rangle \in \Delta \times \Delta \times \mathcal{P}^{link}$ where Δ is the set of all datasets on the web. In this definition, δ_λ^{from} is the referrer dataset which is the owner of subject of link triples in the linkset, δ_λ^{to} is the referenced dataset which is the owner of object of link triples in the linkset, and p_λ^{link} is the link predicate of all link triples in the linkset.

DatasetAnalyzer uses both dataset and linkset descriptions of VOID metadata to select the relevant datasets, in other words sub-graphs, which may contain the results of the query. By this means, the query is transformed into a federated query and executed on the relevant datasets on the web. Accordingly, we exclude the datasets which do not contain any result for the query while querying the global graph, $\mathcal{G}_{wod}$. Beside analyzing relationships between VOID descriptions (datasets, linksets) and triple pattern, relationships between triple patterns in a query are considered. For this reason, we need to give a formal definition of SPARQL queries.

We consider a subset of SPARQL queries which corresponds to a basic graph pattern for formalization [5]. A basic graph pattern consists of triple patterns, $BGP = \{tp_i | \langle s_{tp_i}, p_{tp_i}, o_{tp_i} \rangle \in (\mathcal{I} \cup \mathcal{V}) \times (\mathcal{I} \cup \mathcal{V}) \times (\mathcal{I} \cup \mathcal{L} \cup \mathcal{V})\}$. A triple pattern is slightly different from RDF triple since it contains at least one and at most three *variables* that are elements of infinite set $\mathcal{V}$[14]. While performing a query, its variables are replaced by RDF terms. Thus, according to semantics of SPARQL, a query result is a set of solution mappings, $\{\mu | \mu : \mathcal{V} \rightarrow \mathcal{T}\}$, where a solution mapping, μ, is a partial function from variables to RDF terms.

To perform a query on the web of data, in the worst case, each triple pattern has to be queried on all published datasets. Since this is impractical, our purpose is eliminating irrelevant datasets for each triple pattern. By elimination of irrelevant datasets, we construct a federated query which distributes sub-queries to only relevant ones. In order to eliminate irrelevant datasets, we introduce a set of rules which discover the relevant datasets in dataset analysis, called relevant dataset discovery rules. A relevant dataset for a triple pattern is formalized as an assertion $\rho(\delta_x, tp_i)$ which denotes that the dataset δ_x may contain a result for the triple pattern tp_i. We need to discuss how relevant dataset assertions (ρ) inferred by discovery rules used for eliminating irrelevant datasets. To explain the method of eliminating irrelevant datasets, assume that $\mathcal{Q}_{tp_i}$ is the set of datasets selected to be queried for tp_i which we call *selected set*, and this set initially contains all datasets on the web, $Q_{tp_i}^{init} \equiv \Delta$ (all datasets in a VOID store in our case). Each rule analyzes datasets in Q_{tp_i} of each triple pattern. After applying a rule, elements of Q_{tp_i} which the rule does not infer relevant dataset assertion about are removed from Q_{tp_i}. But, if the rule does not imply any relevant dataset assertion, Q_{tp_i} remains the same. *Irrelevant dataset elimination method* is formalized as $Q_{tp_i}^{new}$ below to denote such an update of selected set subsequent to executing a rule.

$$\mathcal{Q}_{tp_i}^{new} = \left\{ \begin{array}{ll} \exists \delta_a \left(\rho\left(\delta_a, tp_i\right)\right); & \left\{\delta_x | \left(\delta_x \in \mathcal{Q}_{tp_i}\right) \wedge \rho(tp_i, \delta_x)\right\} \\ \exists^0 \delta_a \left(\rho\left(\delta_a, tp_i\right)\right); & \mathcal{Q}_{tp_i} \end{array} \right\}$$

In the following subsection we give relevant dataset discovery rules in detail and we use some queries to exemplify application of rules. Figure 3.2 shows VOID models of a set of datasets which are used in these examples. This model contains simplified VOID descriptions of five datasets and the linksets connecting these datasets by link predicates. Link predicates are showed by arrows between dataset descriptions which are represented with squares. Also we create a sample dataset called Facebook which keeps the data about Facebook users in our local store. This data is about that movie resources located in LinkedMDB dataset are liked by which users. Although there can be a lot of linksets, in Figure 3.2, only a few linksets are taken into account to explain our rules are depicted.

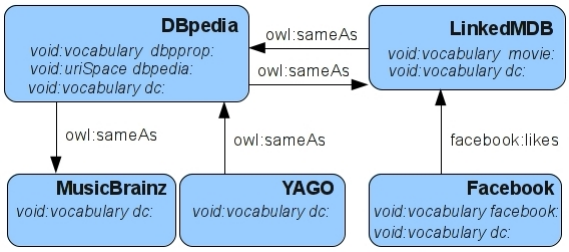

Figure 3.2: Example VOID Models

3.1.1 Relevant Dataset Discovery Rules

This subsection presents a set of rules each of which represents an analysis method of the DatasetAnalyzer module to discover relevant datasets effectively. Each relevant dataset discovery rule aims to analyze datasets from different perspectives and combinations of them to infer relevant dataset (ρ) assertions for triple patterns. These perspectives can be classified into three groups. The first one is analyzing

[12]DatasetAnalyzer of the current version of WoDQA considers only these properties for the sake of simplicity. We plan to integrate other properties such as statistics in the future to make optimized queries.

[13]Note that schemas of the Semantic Web languages such as RDFS and OWL are not specified in $\mathcal{I}^{voc}$.

[14]We exclude blank nodes in query.

IRIs in the triple patterns. We call this perspective *IRI-based Analysis* where namespaces of IRIs and vocabularies in the VOID documents are considered to determine relevant datasets. The second one is considering linked resources. We call this perspective *Linking Analysis* which considers whether a triple links two resources in the same dataset (internal) or in different datasets (external) to eliminate irrelevant datasets. The last perspective is *Shared Variable Analysis*. Since triple patterns share some variables, each triple pattern affects relevant datasets of other triple patterns that include same variables. Relevant dataset discovery rules of all perspectives are introduced in this section.

The first two rules are under the IRI-based analysis perspective each of which considers vocabularies $\mathcal{I}^{voc}$ of VOID metadata. The first discovery rule checks whether IRIs in triple patterns are RDFS (or OWL) classes or properties in the vocabulary set ($\mathcal{I}^{voc}$) of VOID documents. To give the rule, we define $has\left(\mathcal{I}^{voc}_{\delta_x}, r\right)$ function which represents that a resource $r \in \mathcal{I}$ (a property or a class IRI) is included by one of the vocabularies in $\mathcal{I}^{voc}_{\delta_x}$. Using *has* expression, relevance of a dataset δ_x to an IRI r is represented with $VocMatch$ as shown in Definition 2.

Definition 2. $\forall \delta_x(has(\mathcal{I}^{voc}_{\delta_x}, r) \rightarrow VocMatch(\delta_x, r))$ where $r \in \mathcal{I}$

For a triple pattern such as ?s dbpprop:name[15] "Nikola Tesla", dbpprop:name RDFS property in the predicate position obliges that matching triple patterns can only be in datasets which uses dbpprop vocabulary. Therefore, we can eliminate datasets which do not use dbpprop vocabulary. This situation is handled by Rule 1 which is similar to predicate indexes.

RULE 1. *If there is a dataset in which one of its vocabularies includes the predicate of a triple pattern, then it is relevant for the triple pattern.*

$$\forall tp_i, \delta_x \left(VocMatch\left(\delta_x, p_{tp_i}\right) \rightarrow \rho\left(\delta_x, tp_i\right)\right)$$

According to Figure 3.2, DBpedia uses dbpprop vocabulary, and the rule decides that it is relevant for such a triple pattern. In web of data, lots of datasets which uses dbpprop vocabulary can be found, and they can be eliminated using outputs of other discovery rules.

To introduce Rule 2, consider another triple pattern, ?producer rdf:type linkedMDB:producer, which contains a type definition for the variable ?producer. In such cases, the object of the triple pattern is a class definition, it makes sense to eliminate the datasets which do not use the vocabulary of this class. Rule 2 resembling type indexes is used to specify relevant datasets for such triple patterns. The example triple pattern is queried from the datasets that use linkedMDB vocabulary, i.e. LinkedMDB dataset for our example model. Other datasets do not include a resource which is an instance of linkedMDB:producer class, and therefore they are eliminated by output of this rule.

RULE 2. *If there is a dataset in which one of its vocabularies includes the object of a triple pattern when the predicate of the triple pattern is* rdf:type*, then the dataset is relevant for the triple pattern.*

$$\forall tp_i, \delta_x(VocMatch(\delta_x, o_{tp_i}) \wedge (p_{tp_i} = \text{rdf:type}) \rightarrow \rho(\delta_x, tp_i))$$

[15]All prefixes used in the paper are defined in Table 1.

Another perspective to discover relevant datasets is Linking Analysis. Since a triple links two resources in the same dataset or in different datasets, this perspective is separated into two kinds of analyses, each of which considers different kind of triples. The first one is Internal Linking Analysis which considers triples linking resources in the same dataset. A relevant dataset found by this analysis is called internal relevant dataset, and is represented with $\rho^{int}\left(\delta_x, tp_i\right)$. On the other hand, External Linking Analysis considers link triples which connect resources in different datasets. In this case, linkset descriptions of VOID documents are taken into account to discover relevant datasets, and a dataset found by this analysis is called external relevant dataset which is represented as $\rho^{ext}\left(\delta_x, tp_i\right)$. Internal and External Linking analyses are both executed for a triple pattern in whole Linking Analysis process, and then produced internal and external relevant datasets for a triple pattern are unified as relevant datasets for the triple pattern as shown in Rule 3.

RULE 3. *Union of external and internal datasets for a triple pattern constitutes relevant datasets for the triple pattern.*

$$\forall \delta_x, tp_i \left(\rho^{int}\left(\delta_x, tp_i\right) \vee \rho^{ext}\left(\delta_x, tp_i\right) \rightarrow \rho\left(\delta_x, tp_i\right)\right)$$

Irrelevant dataset elimination method considers relevant datasets which are specified by Rule 3 to eliminate irrelevant datasets from selected set of a triple pattern (Q_{tp_i}). Internal and external datasets are intermediate results to infer relevant datasets in a Linking Analysis.

The first two rules under Linking Analysis perspective are linking-to-IRI discovery rules. Consider the example triple pattern, ?film owl:sameAs dbpedia:A_Fistful_of_Dollars, which can be matched with a triple that links a resource to dbpedia:A_Fistful_of_Dollars resource. Triple patterns whose object is an IRI are analyzed by these rules. Since owners of the linked IRI must be known in these rules, we give a definition that depicts the owners of any resource on the basis of IRI Analysis. We formalize inclusion of a resource ($r \in \mathcal{I}$) by a dataset (δ_x) in Definition 3 by using the urispaces ($\mathcal{L}^{space}$) property of the VOID description and $startsWith\left(r, \mathcal{L}^{space}_{\delta_x}\right)$ function which represents that r starts with one of the urispaces in $\mathcal{L}^{space}_{\delta_x}$.

Definition 3. $\forall \delta_x(startsWith(r, \mathcal{L}^{space}_{\delta_x}) \rightarrow Owner(\delta_x, r))$

Rule 4 is linking-to-IRI internal discovery rule from the internal linking point of view. According to the example query ?film should be in the same dataset with dbpedia:A_Fistful_of_Dollars, i.e. owner of the dbpedia:A_Fistful_of_Dollars resource. Therefore appropriate triple patterns can be found in DBpedia dataset.

RULE 4. *If there is a triple pattern whose object is an IRI, then owner datasets of the IRI are internal relevant for the triple pattern.*

$$\forall tp_i, \delta_x \left(\left(\delta_x \in \mathcal{Q}_{tp_i}\right) \wedge Owner\left(\delta_x, o_{tp_i}\right) \rightarrow \rho^{int}\left(\delta_x, tp_i\right)\right)$$
where $o_{tp_i} \in \mathcal{I}, s_{tp_i} \in \mathcal{V}$

On the other hand, from the external linking point of view, appropriate triples can be found in datasets which are linked to the owner datasets of object IRI. For our example triple pattern, ?film should be in datasets which contain

link triples whose object resource is defined in DBpedia. For this analysis, linkset descriptions of VOID documents are used. To discover relevant datasets for a triple pattern by using linking-to-IRI external discovery rule, the triple pattern should have a bound predicate. We define *Compatible* expression in Definition 4 to represent that which linkset description is appropriate to use for determining relevant datasets for a triple pattern.

Definition 4. If selected set of a triple pattern has the referrer dataset of a linkset description and link predicate of the linkset description is same with the triple pattern's predicate, then the linkset description is compatible with the triple pattern.

$\forall \lambda_m, tp_i((\delta^{from}_{\lambda_m} \in \mathcal{Q}_{tp_i}) \wedge (p^{link}_{\lambda_m} = p_{tp_i}) \rightarrow Compatible(\lambda_m, tp_i))$

Considering ?film owl:sameAs dbpedia:A_Fistful_of_Dollars triple pattern, and remembering the linkset description of our example model in Figure 3.2, there are linksets from LinkedMDB and YAGO datasets to DBpedia dataset whose link predicates are owl:sameAs. Rule 5 which is under the Linking Analysis perspective gives these two datasets as external relevant datasets for this triple pattern. If a dataset is not linked to DBpedia by owl:sameAs predicate then one can conclude that this dataset is irrelevant with the triple pattern.

RULE 5. *If there is a linkset description that is compatible with the triple pattern and whose referenced dataset is an owner dataset of the triple pattern's object, then the referrer dataset of the linkset description is external relevant for the triple pattern.*

$\forall tp_i, \lambda_m, \delta_x(Compatible(\lambda_m, tp_i) \wedge Owner(\delta_x, o_{tp_i}) \wedge (\delta_x = \delta^{to}_{\lambda_m}) \rightarrow \rho^{ext}(\delta^{from}_{\lambda_m}, tp_i))$ where $o_{tp_i} \in \mathcal{I}, s_{tp_i} \in \mathcal{V}$

Recall that internal and external relevant datasets are unified by Rule 3 after applying rules in Rule 4 and Rule 5. Thus, the final relevant datasets which are selected by the linking-to-IRI rules are DBpedia, LinkedMDB and YAGO.

Another couple of rules under the Linking Analysis perspective are IRI-links-to rules. These rules are applied to triple patterns whose subject is an IRI and object is a variable to determine relevant datasets from internal and external linking point of view. Rule 6 is IRI-links-to internal discovery rule and it finds the triples that link resources in the same dataset. One can conclude that if the subject of a triple pattern is an IRI, then triples matching with this triple pattern are in the owner dataset of this IRI.

RULE 6. *If a dataset is an owner of the subject of a triple pattern, then this dataset is internal relevant dataset for the triple pattern.*

$\forall tp_i, \delta_x \left((\delta_x \in \mathcal{Q}_{tp_i}) \wedge Owner(\delta_x, s_{tp_i}) \rightarrow \rho^{int}(\delta_x, tp_i)\right)$ where $o_{tp_i} \in \mathcal{V}, s_{tp_i} \in \mathcal{I}$

Consider the triple pattern dbpedia:Ennio_Morricone owl:sameAs ?person. Subject of this triple pattern is an IRI whose namespace is dbpedia, and therefore an internal relevant dataset is DBpedia whose VOID metadata contains dbpedia as value of urispace property.

Rule 7 is IRI-links-to external discovery rule and it finds the triples that connect resources in different datasets. According to this rule an owner dataset of the subject IRI of the triple pattern is external relevant only when there is a linkset definition that includes owner dataset as referrer dataset compatible with the triple pattern.

RULE 7. *If there is a linkset description that is compatible with the triple pattern and whose referrer dataset is an owner dataset of the triple pattern's subject, then the referrer dataset of the linkset description is external relevant for the triple pattern.*

$\forall tp_i, \lambda_m, \delta_x(Compatible(\lambda_m, tp_i) \wedge Owner(\delta_x, s_{tp_i}) \wedge (\delta_x = \delta^{from}_{\lambda_m}) \rightarrow \rho^{ext}(\delta_x, tp_i))$ where $o_{tp_i} \in \mathcal{V}, s_{tp_i} \in \mathcal{I}$

According to the example, DBpedia is the owner dataset of dbpedia:Ennio_Morricone and also there is a linkset description whose referrer dataset is DBpedia and whose link predicate is owl:sameAs.

Other discovery rules in Linking Analysis are combined with Shared Variable Analysis. The first discovery rule which conforms to this combined analysis is Chaining Triple Patterns Analysis. This rule considers two triple patterns together to discover relevant datasets. This is a characteristic of Shared Variables Analysis, since it depends on analyzing more than one triple pattern that have same variable. Triple patterns below are example of chaining triple patterns:
?s owl:sameAs ?film.
?film linkedMDB:producer_name "Sergio Leone"

Notice that the second triple pattern is used for querying films whose producer name is "Sergio Leone". Thus, the second triple pattern affects the relevant datasets of the first triple pattern. From internal linking point of view, the first triple pattern can be found in datasets which satisfy the second triple pattern because ?s and ?film should be in the same dataset. In this direction, Internal Chaining Triple Pattern Analysis formalized in Rule 8 is used to discover internal relevant datasets for triple patterns.

RULE 8. *If there is a triple pattern whose object is same with the subject of another triple pattern, then datasets included by selected sets of both triple patterns are internal relevant.*

$\forall \delta_x, tp_i, tp_j((o_{tp_i} = s_{tp_j}) \wedge (\delta_x \in \mathcal{Q}_{tp_i}) \wedge (\delta_x \in \mathcal{Q}_{tp_j}) \rightarrow \rho^{int}(\delta_x, tp_i) \wedge \rho^{int}(\delta_x, tp_j))$ where $o_{tp_i}, s_{tp_i} \in \mathcal{V}$

To execute Chaining Triple Pattern Analysis, execution order of rules becomes important. To exemplify this situation according to Chaining Triple Patterns query, assume that IRI-based analysis is applied before, and LinkedMDB is the relevant dataset for the second triple pattern since LinkedMDB VOID includes linkedMDB as the value of vocabulary. Then, this rule can specify that the internal relevant dataset for the first triple pattern is LinkedMDB. It is clearly seen from this example, Shared Variable Analysis should be performed after the execution of IRI-based Analysis rules to eliminate more datasets. Hence, in the Subsection 3.1.2, we give an overview of analysis process which specifies an execution order for these rules.

After the IRI-based analysis, if we apply Rule 8 for the example triple patterns, relevant datasets of the second triple pattern is shown as $Q_{tp_2} = \{\delta_{LinkedMDB}\}$, and of the first

one is shown as $Q_{tp_1} \equiv \Delta$. For this case, this rule asserts that $\rho^{int}(\delta_{LinkedMDB}, tp_1)$ and $\rho^{int}(\delta_{LinkedMDB}, tp_2)$.

On the other hand, External Chaining Triple Patterns analysis uses linkset descriptions while considering two triple patterns. While internal one can be used for all triple patterns without considering the predicate, external rule is applied for a triple pattern which has a link predicate. Rule 9 introduces this rule.

RULE 9. *If there is a triple pattern (tp_i) whose object is same with the subject of another one (tp_j), and there is a linkset description which is compatible with (tp_i) and its referenced dataset is included by the selected set of tp_j, then referrer dataset is external relevant for tp_i and referenced dataset is external relevant for tp_j.*

$\forall \lambda_m, tp_i, tp_j((o_{tp_i} = s_{tp_j}) \wedge Compatible(\lambda_m, tp_i) \wedge (\delta^{to}_{\lambda_m} \in \mathcal{Q}_{tp_j}) \rightarrow \rho^{ext}(\delta^{from}_{\lambda_m}, tp_i) \wedge \rho^{ext}(\delta^{to}_{\lambda_m}, tp_j))$ *where* $o_{tp_i} \in \mathcal{V}$

With respect to the example of chaining triple patterns, assume that selected set for the first triple pattern is $\mathcal{Q}_{tp_1} \equiv \Delta$, and for the second one is $\mathcal{Q}_{tp_2} = \{\delta_{LinkedMDB}\}$. Rule 9 determines that $\delta_{DBpedia}$ is external relevant for tp_1 since resources ?film can be found in $\delta_{LinkedMDB}$ and there is a linkset between these two datasets with owl:sameAs predicate. It is clear that no other dataset can contain an appropriate triple if it is not linked to LinkedMDB by owl:sameAs. At the end of Chaining Triple Pattern Analysis, internal and external relevant datasets specified by Rule 8 and 9 are unified according to Rule 3.

Another analysis which uses both Linking Analysis and Shared Variable Analysis is Object Sharing Triple Patterns Analysis. For triple patterns which have the same object variable, only the datasets can include triples which satisfy the object variable of both triple patterns. To simplify the explanation, we use the following example for Object Sharing Triple Pattern Analysis below:

?person facebook:likes ?movie.
?film owl:sameAs ?movie.

From internal linking point of view, ?person and ?film should be in the same dataset with ?movie. Rule 10 specifies the internal relevant datasets for triple patterns which have the same object. Assume that $\mathcal{Q}_{tp_1} = \{\delta_{Facebook}\}$ due to value of vocabulary property of Facebook VOID and $\mathcal{Q}_{tp_2} \equiv \Delta$. This rule determines that only $\delta_{Facebook}$ can contain internal triples that satisfy triple patterns together.

RULE 10. *If there is a triple pattern whose object is same with the object of another one, then the datasets included by selected sets of both triple patterns are internal relevant.*

$\forall \delta_x, tp_i, tp_j((o_{tp_i} = o_{tp_j}) \wedge (\delta_x \in \mathcal{Q}_{tp_i}) \wedge (\delta_x \in \mathcal{Q}_{tp_j}) \rightarrow \rho^{int}(\delta_x, tp_i) \wedge \rho^{int}(\delta_x, tp_j))$ where $o_{tp_i} \in \mathcal{V}$

To execute Object Sharing Analysis from external point of view, we benefit from the linkset descriptions. To find appropriate link triples, ?person and ?film should be in different datasets. Rule 11 determines external relevant dataset for triple patterns which have the same object. This rule considers two linkset descriptions together for two triple patterns.

RULE 11. *If two triple patterns have the same object, and there are two linkset descriptions which have the same referenced dataset each of which is compatible with one of the triple patterns, then referrer datasets of the linkset descriptions are external relevant for the triple patterns.*

$\forall \lambda_m, \lambda_n, tp_i, tp_j((o_{tp_i} = o_{tp_j}) \wedge Compatible(\lambda_m, tp_i) \wedge Compatible(\lambda_n, tp_j) \wedge (\delta^{to}_{\lambda_m} = \delta^{to}_{\lambda_n}) \rightarrow \rho^{ext}(\delta^{from}_{\lambda_m}, tp_i) \wedge \rho^{ext}(\delta^{from}_{\lambda_n}, tp_j))$ where $o_{tp_i} \in \mathcal{V}$

For the example of Object Sharing Triple Pattern Analysis, assume that no rule is applied before this analysis and selected sets are $\mathcal{Q}_{tp_1} \equiv \Delta$ and $\mathcal{Q}_{tp_2} \equiv \Delta$. In Figure 5, $\delta_{DBpedia}$, δ_{YAGO}, and $\delta_{LinkedMDB}$ have link triples with predicate owl:sameAs. But, only $\delta_{Facebook}$ has a linkset to $\delta_{LinkedMDB}$ with predicate facebook:likes, and therefore $Q^{new}_{tp_1} = \{\delta_{Facebook}\}$. In this case, tp_2 can only be queried on the datasets which is linked to $\delta_{LinkedMDB}$ with predicate owl:sameAs, and thus $Q^{new}_{tp_2} = \{\delta_{DBpedia}\}$. Other triples whose subject corresponding to ?film in other datasets according to tp_2 cannot include objects that satisfy object of tp_1. As done in other Linking Analysis methods, internal and external relevant datasets which are inferred by Rule 10 and Rule 11 are unified according to Rule 3.

The last analysis from the Shared Variable Analysis perspective considers the triple patterns which have the same subject called Subject Sharing Triple Patterns Analysis. This rule does not use Linking Analysis perspective, and thus it does not contain Internal and External Linking Analysis. Consider the following example triple patterns for Subject Sharing Triple Pattern Analysis:

?city dbpprop:name "Izmir" .
?city dc:terms ?subject.

According to our dataset definition, triples which have the same subject are included by the same dataset. Based on this, Rule 12 infers datasets which are relevant for triple patterns by taking the intersection of selected sets into account.

Assume that selected sets for triple patterns are $\mathcal{Q}_{tp_1} = \{\delta_{DBpedia}\}$ and $\mathcal{Q}_{tp_2} \equiv \Delta$. According to the rule, the final datasets according to this rule are $\mathcal{Q}^{new}_{tp_1} \equiv \mathcal{Q}^{new}_{tp_2} = \{\delta_{DBpedia}\}$.

RULE 12. *If there is a triple pattern whose subject is same with the subject of another triple pattern, then the datasets included by selected sets of both triple patterns are relevant.*

$\forall \delta_x, tp_i, tp_j((s_{tp_i} = s_{tp_j}) \wedge (\delta_x \in (\mathcal{Q}_{tp_i} \cap \mathcal{Q}_{tp_j})) \rightarrow \rho(\delta_x, tp_i) \wedge \rho(\delta_x, tp_j))$ where $s_{tp_i} \in \mathcal{V}$

Up to this point, we have given the relevant dataset discovery rules which are used to determine relevant datasets from different perspectives. These rules are executed together for a query to make a complete analysis. Next section introduces the analysis process which specifies the execution order for rules.

3.1.2 Analysis Process

The rules introduced above should be executed together to provide effective dataset selection. In this section, execution of rules are explained in the process of DatasetAnalyzer which is shown Figure 3.3. In the figure, $\mathcal{Q}_{BGP} = \{\mathcal{Q}_{tp_i} | tp_i \in BGP\}$ is the set of selected sets of all triple patterns in a query. We use $\mathcal{Q}^{init}_{BGP}$ to represent the initial state where selected set of each triple pattern contains the whole web of data, $\forall \mathcal{Q}_{tp_i} \in \mathcal{Q}^{init}_{BGP}(\mathcal{Q}_{tp_i} \equiv \Delta)$. This set is the input of the single step analysis, and selected sets in this set are constrained by execution of the rules includes the IRI-based

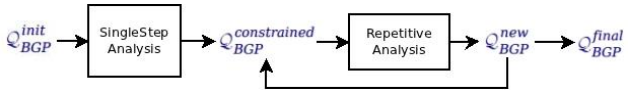

Figure 3.3: WoDQA Analysis Process

analyses. Single step analysis includes vocabulary match, linking-to-IRI and IRI-links-to rules which are executed only once. The reason is that the rules based on IRI-based analysis produce the same result for every execution because they do not depend on current $\mathcal{Q}_{tp_i}$. Although the rules in single step analysis do not have a specific order, using output of each rule in dataset elimination method reduces current datasets set of the triple patterns. Then, $\mathcal{Q}^{constrained}_{BGP}$ is given to the repetitive analysis phase.

On the other hand, rules based on Shared Variable Analysis take more than one triple pattern into consideration. Since different combinations of triple patterns affect the selected sets of each other, these rules depend on current selected sets of triple patterns to discover the datasets relevant to the triple patterns. For this reason, they are executed repetitively until no dataset is eliminated from any $\mathcal{Q}_{tp_i}$. The repetitive analysis phase produces $\mathcal{Q}^{new}_{BGP}$ by executing Shared Variable Analysis rules. After the phase is completed once, if an elimination has been done, $\mathcal{Q}^{new}_{BGP}$ is given to repetitive analysis as $\mathcal{Q}^{constrained}_{BGP}$, and the phase is repeated. On the other hand, when the rules do not change any selected set, i.e. $\mathcal{Q}^{new}_{BGP} \equiv \mathcal{Q}^{constrained}_{BGP}$, the analysis is finished and the result is produced as $\mathcal{Q}^{final}_{BGP}$.

3.2 Query Reorganizer

The QueryReorganizer module is responsible for rewriting a query using final selected set of each triple pattern $\left(\mathcal{Q}^{final}_{BGP}\right)$ decided by the DatasetAnalyzer. While rewriting a federated query, Query Reorganizer conforms to SPARQL 1.1 federation extension[16].

While the initial query is a set of triple patterns, QueryReorganizer divides the query into sub-queries and makes it a set of service graph patterns each of which is represented with a tuple $sgp_\alpha = \langle Srv_\alpha, SubTp_\alpha \rangle$. A service graph pattern consists of a Srv set which includes datasets[17] to send the sub-query, and a $SubTp$ set which is the subset of the triple patterns of the initial query, i.e. a sub-query. Performing a service graph pattern is unifying the results of the sub-query in each dataset in Srv_α.

Triple patterns which have the same selected set ($\mathcal{Q}_{tp_i}$) are added to the same service graph pattern to decrease networking cost. But, only consecutive triple patterns can be in the same sub-query because WoDQA does not change the evaluation order of the query. In this direction, elements of $SubTp$ sets of service graph patterns are found by Algorithm 1.

Datasets of a sub-query is formalized as $Srv_\alpha = \{\delta_x | \ \delta_x \in \mathcal{Q}_{tp_i}, \ tp_i \in SubTp_\alpha\}$, and service endpoint URLs are procured from VOID documents of the datasets. The output of the Query Reorganizer is shown as $ReorganizedBGP = \langle sgp_1, \ldots, sgp_m \rangle$ where $1 \leq m \leq n$ and n is the number of triple patterns of the initial query. $ReorganizedBGP$ represents the federated form of the initial query which contains ordered service graph patterns. WoDQA executes the reorganized query using Jena ARQ query engine.

Algorithm 1 This algorithm divides query into sub-triples

```
FUNCTION DivideTriples()
INPUT bgp = {tp_1, ..., tp_n} including n triple patterns;
LET i := 1, α := 1;
LET SubTriples_α := {tp_i};
WHILE i < n DO
    IF Q_{tp_{i+1}} = Q_{tp_i} THEN
        LET SubTriples_α := SubTriples_α ∪ {tp_{i+1}};
    ELSE
        LET SubTriples_{α+1} := {tp_{i+1}};
        LET α := α + 1;
    LET i := i + 1;
```

Besides grouping triple patterns, although WoDQA does not include query optimization phase of query federation approach, only moving up FILTER expression[2] optimization technique is used. Thus intermediate results are filtered as early as possible. Furthermore, WoDQA supports queries include UNION and OPTIONAL keywords but queries include GRAPH keyword and blank nodes are not supported.

4. USAGE SCENARIO

There are two ways for users to benefit from WoDQA. The first one is the SPARQL endpoint of WoDQA[18] which can be used to redirect raw queries. One can construct a SPARQL query with a SERVICE block including the raw query, and use WoDQA SPARQL endpoint as the remote service. When this query is executed, the WoDQA endpoint is invoked, and this endpoint transforms the query into a federated form by means WoDQA and executes this federated query on the relevant dataset transparently to the user.

The other way is using the web form of the WoDQA[19]. In this section, a sample query execution on the web form of WoDQA is explained. Reorganized form of the query, results of select and construct queries and execution time can be observed in this form.

The example query seen in the WoDQA web form in Figure 4.1 searches for an answer to "Which facebook users like movies which are produced by a German producer?". This query is represented as $BGP = \langle tp_1, \ldots, tp_5 \rangle$ where
$tp_1 = \langle$?faceUser,facebook:likes,?movie$\rangle$,
$tp_2 = \langle$?movie,linkedMDB:producer,?producer$\rangle$,
$tp_3 = \langle$?dbProducer,owl:sameAs,?producer$\rangle$,
$tp_4 = \langle$?anyMovie, dbpo:producer, ?dbProducer$\rangle$,
$tp_5 = \langle$?dbProducer, dbpo:birthPlace, dbpedia:Germany$\rangle$.

We explain how relevant datasets are found according to the WoDQA Analysis process introduced in Figure 3.3. Initially, single step analysis phase is performed for this query. Selected sets of tp_1 and tp_2 are eliminated via output of predicate vocabulary match, and in the consequence of single step analysis they are $\mathcal{Q}_{tp_1} = \{\delta_{Facebook}\}$ and $\mathcal{Q}_{tp_2} = \{\delta_{LinkedMDB}\}$. No relevant dataset is found for tp_3 in the single step analysis, because owl:sameAs is a generic

[16]http://www.w3.org/TR/sparql11-federated-query/
[17]In the implementation, SPARQL endpoints of datasets are used in SERVICE expressions.
[18]Up-to-date WoDQA SPARQL endpoint address can be found on http://seagent.ege.edu.tr/etmen/wodqa.html page.
[19]http://seagent.ege.edu.tr/etmen/wodqa.html

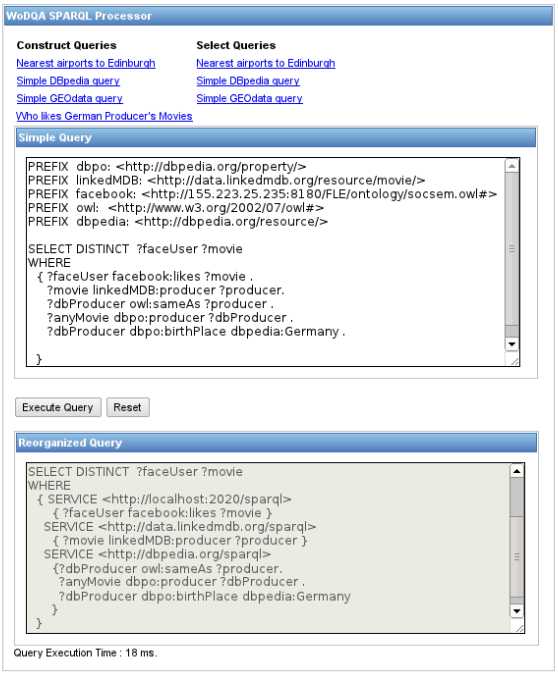

Figure 4.1: Example query execution with WoDQA

property and owl is not defined as vocabulary property in VOIDs. Thus, $\mathcal{Q}_{tp_3}$ still includes all datasets (Δ). Predicate vocabulary match discovers relevant datasets for tp_4 and tp_5, and their selected sets are $\mathcal{Q}_{tp_4} = \mathcal{Q}_{tp_5} = \{\delta_{DBpedia}\}$.

After the single step analysis is applied to all triple patterns, the repetitive analysis phase is performed, and selected set of tp_3 is eliminated in this phase. Subject Sharing Triple Patterns Analysis discovers relevant datasets for tp_3 since tp_3 and tp_5 have the same subject variable, and thus its selected set becomes $\mathcal{Q}_{tp_3} = \{\delta_{DBpedia}\}$.

The reorganized query shown in Figure 4.1 is rendered with these analysis results by QueryReorganizer and formalized as $ReorganizedBGP = \langle sgp_1, sgp_2, sgp_3 \rangle$ where

$sgp_1 = \langle \{\delta_{Facebook}\}, \{tp_1\} \rangle$,
$sgp_2 = \langle \{\delta_{LinkedMDB}\}, \{tp_2\} \rangle$,
$sgp_3 = \langle \{\delta_{DBpedia}\}, \{tp_3, tp_4, tp_5\} \rangle$.

After the reorganizing process, the triple patterns in the service graph patterns are executed on related endpoints by means of Jena ARQ, and query results are incrementally collected. In conclusion, results related with the query are listed at the bottom of the web form page as seen in Figure 4.1.

5. CONCLUSION

In this paper, we have introduced a query federation engine called WoDQA that discovers related datasets in a VOID store for a query and distributes the query over these datasets. The novelty of our approach is exhaustive dataset selection mechanism which includes analysis of triple pattern relations and links between datasets besides analyzing datasets for each triple pattern. WoDQA focuses on discovering relevant datasets and eliminating irrelevant ones using a rule-based approach introduced in this paper. Our approach requires VOID descriptions which include a SPARQL endpoint to query the dataset, reflect actual content of the dataset completely and accurately, and include linksets between datasets to select datasets effectively. WoDQA allows users to construct raw queries without the need to know how query will divide into sub-queries and where sub-queries are executed. Query results are complete under the assumption of available, accurate and complete VOID descriptions of datasets.

The initial version of WoDQA which is introduced in this paper has some disadvantages arising from query federation approach which WoDQA builds upon. As mentioned previously, follow-your-nose has some problems such as missing results and large document retrieval. Similar problems may occur for query federation. Firstly, to find complete results to queries, it is required that metadata of all datasets must be well-defined and accurate. But, to provide such an accurate dataset metadata an automated mechanism which continuously updates the metadata is required. However, even there would be a tool which implements this requirement, providing accurate dataset metadata via such a tool is the responsibility of dataset publishers.

Another problems of query federation are high latency and low selectivity of datasets which are similar to retrieval of large documents in follow-your-nose. Query optimization can be a solution for these problems of query federation. Grouping triple patterns to filter more triples on an endpoint can prevent high latency (required processing time) and changing query evaluation order according to dataset selectivity statistics can prevent retrieving large result sets. To make WoDQA functioning in the wild, optimization step of query federation is required to be implemented. We plan to incorporate triple pattern selectivity into query reorganization using VOID properties about statistics.

On the other hand, we could not make an evaluation of our approach in this paper, since VOID documents in current VOID stores are not well-defined. Since SPARQL endpoint definitions, linkset descriptions or vocabularies are missing in most of VOID documents, we could not find a chance to execute comprehensive scenarios. Developing a tool which extracts well-defined VOID descriptions of datasets, and by this means evaluating our approach is a required future work to confirm applicability of WoDQA on linked open data. Also, evaluating the analysis cost of WoDQA for a large VOID store will be possible when well-defined VOIDs are constructed.

6. REFERENCES

[1] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. Describing Linked Datasets - On the Design and Usage of voiD, the 'Vocabulary of Interlinked Datasets'. In *WWW 2009 Workshop: Linked Data on the Web (LDOW2009)*, Madrid, Spain, 2009.

[2] Abraham Bernstein, Christoph Kiefer, and Markus Stocker. OptARQ: A SPARQL Optimization Approach based on Triple Pattern Selectivity Estimation. Technical Report ifi-2007.03, Department of Informatics, University of Zurich, 2007.

[3] Chris Bizer, Tom Heath, Danny Ayers, and Yves Raimond. Interlinking open data on the web. www4.wiwiss.fu-berlin.de/bizer/pub/LinkingOpenData.pdf, 2007. Stand 12.5.2009.

Table 1: Prefix definitions

rdf:	<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
dbpedia:	<http://dbpedia.org/resource/>
dbpprop:	<http://dbpedia.org/property/>
linkedMDB:	<http://data.linkedmdb.org/resource/movie/>
owl:	<http://www.w3.org/2002/07/owl#>
dc:	<http://purl.org/dc/terms/>
foaf:	<http://xmlns.com/foaf/0.1/>
facebook:	<http://155.223.25.235:8180/FLE/ontology/socsem.owl#>

[4] Paolo Bouquet, Chiara Ghidini, and Luciano Serafini. Querying the web of data: A formal approach. In *Proceedings of the 4th Asian Conference on The Semantic Web*, ASWC '09, pages 291–305, Berlin, Heidelberg, 2009. Springer-Verlag.

[5] Carlos Buil, Marcelo Arenas, and Óscar Corcho. Semantics and optimization of the SPARQL 1.1 Federation Extension. In *Proc. of 8th Extended Semantic Web Conference (ESWC 2011), Heraklion, Crete, Greece*, volume 6644 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2011.

[6] M. Duerst and M. Suignard. Internationalized Resource Identifiers (IRIs). RFC 3987 (Proposed Standard), January 2005.

[7] Olaf Görlitz and Steffen Staab. Federated data management and query optimization for linked open data. In Athena Vakali and Lakhmi Jain, editors, *New Directions in Web Data Management 1*, volume 331 of *Studies in Computational Intelligence*, pages 109–137. Springer Berlin / Heidelberg, 2011.

[8] Olaf Görlitz and Steffen Staab. SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions. In *Proceedings of the 2nd International Workshop on Consuming Linked Data*, Bonn, Germany, 2011.

[9] Peter Haase, Tobias Mathäß, and Michael Ziller. An evaluation of approaches to federated query processing over linked data. In *Proceedings of the 6th International Conference on Semantic Systems*, I-SEMANTICS '10, pages 5:1–5:9, New York, NY, USA, 2010. ACM.

[10] Olaf Hartig. Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In Grigoris Antoniou, Marko Grobelnik, Elena Paslaru Bontas Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Pan, editors, *ESWC (1)*, volume 6643 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 2011.

[11] Olaf Hartig, Christian Bizer, and Johann Christoph Freytag. Executing sparql queries over the web of linked data. In *International Semantic Web Conference*, pages 293–309, 2009.

[12] Olaf Hartig and Andreas Langegger. A Database Perspective on Consuming Linked Data on the Web. *Datenbankspektrum, Semantic Web Special Issue*, July 2010.

[13] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, San Rafael, CA, 1 edition, 2011.

[14] Bastian Quilitz and Ulf Leser. Querying Distributed RDF Data Sources with SPARQL. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, chapter 39, pages 524–538. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2008.

[15] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. Fedx: A federation layer for distributed query processing on linked open data. In Grigoris Antoniou, Marko Grobelnik, Elena Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Pan, editors, *The Semanic Web: Research and Applications*, volume 6644 of *Lecture Notes in Computer Science*, pages 481–486. Springer Berlin / Heidelberg, 2011.

[16] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. Fedx: Optimization techniques for federated query processing on linked data. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist, editors, *The Semantic Web - ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 601–616. Springer Berlin / Heidelberg, 2011.