Estimating query rewriting quality over LOD

Ana I. Torre-Bastida [a,*], Jesús Bermúdez [b] and Arantza Illarramendi [b]

[a] *Optima, Tecnalia Research & Innovation, Pais Vasco, Spain*
E-mail: isabel.torre@tecnalia.com
[b] *Computer Languages and Systems, Basque Country University UPV-EHU, Pais Vasco, Spain*
E-mails: jesus.bermudez@ehu.es, a.illarramendi@ehu.es

Editors: Dhaval Thakker, University of Bradford, United Kingdom; Daniel Schwabe, Catholic University in Rio de Janeiro (PUC-Rio), Brazil; Kouji Kozaki, Osaka University, Japan; Roberto García, Universitat de Lleida, Spain; Marco Brambilla, Politecnico di Milano, Italy; Vania Dimitrova, University of Leeds, United Kingdom
Solicited reviews: Roberto García, Universitat de Lleida, Spain; Luiz André Portes Pais Leme, Universidade Federal Fluminense, Brazil; Kouji Kozaki, Osaka University, Japan; Martin Rezk, Rakuten Inc.; Jefferson Santos, Federal Institute of Science and Technology (IFRS), Brazil; one anonymous reviewer

Abstract. Nowadays it is becoming increasingly necessary to query data stored in different datasets of public access, such as those included in the Linked Data environment, in order to get as much information as possible on distinct topics. However, users have difficulty to query those datasets with different vocabularies and data structures. For this reason it is interesting to develop systems that can produce on demand rewritings of queries. Moreover, a semantics preserving rewriting cannot often be guaranteed by those systems due to heterogeneity of the vocabularies. It is at this point where the quality estimation of the produced rewriting becomes crucial. In this paper we present a novel framework that, given a query written in the vocabulary the user is more familiar with, the system rewrites the query in terms of the vocabulary of a target dataset. Moreover, it informs about the quality of the rewritten query with two scores: a similarity factor which is based on the rewriting process itself, and a quality score offered by a predictive model. This Machine Learning based model learns from a set of queries and their intended (gold standard) rewritings. The feasibility of the framework has been validated in a real scenario.

Keywords:
Semantic web, Linked Open Data, SPARQL, Query rewriting, Similarity

1. Introduction

The increasing adoption of the Linked Open Data (LOD) paradigm has generated a distributed space of globally interlinked data, usually known as the Web of Data. This new space opens up the possibility of querying over a huge set of updated data. However, many users find difficulties when formulating queries over it, due to the fact that they are not familiar with the data, links and vocabularies of many heterogeneous datasets that constitute the Web of Data. In this scenario it becomes necessary to provide the users with tools and mechanisms that help them to exploit the vast amount of available data.

We can find in the specialized literature different proposals that have considered the goal of facilitating the task of querying heterogeneous datasets. We can highlight three main approaches among those proposals: 1) those that generate a kind of *centralized repository* that contains all the data of different datasets and then queries are formulated over that repository (e.g. [1]); 2) those that follow the *federated query* processing approach (e.g. [2]) in which a query against a federation of datasets is split into sub-queries that can be answered in the individual nodes where datasets are stored; and 3) those that follow the *exploratory query* processing approach (e.g. [3]), which take advantage

[*] Corresponding author. E-mail: isabel.torre@tecnalia.com.

of the dereferenceable IRIs principle promoted by Linked Data. In this approach, query execution begins in a source dataset and is intertwined with the traversal of the HTTP dereferenceable IRIs to retrieve more data, from different nodes, that incorporate additional data for answering parts of the query and include more IRIs that can be successively dereferended to augment the queried dataset until the initial query is sufficiently answered. The two first approaches require a costly preparation task and the last one is mainly oriented to leverage the dereferencing architecture of Linked Data.

In the centralized and federated approaches, users pose queries using the vocabulary chosen for the global schema and can only expect answers from the centralized repository or from federated datasets. However, it is very common that several datasets offer data on the same or overlapped domains. For example, GeoData and Geo Linked Data in the geographic domain, BNE (Biblioteca Nacional de España) and BNF (Bibliothèque National de France) in the bibliographic domain, MusicBrainz and Jamendo in the music domain, or Drugbank and Diseasome in the bio domain. Each centralized repository or a datasets federation only considers a limited collection of datasets and, therefore, cannot help a user with datasets that are out of the collection. Moreover, it seems interesting for the users to pose queries to a preferred dataset whose schema and vocabulary is sufficiently known by them and then a system could help those users enriching the answers to the query with data taken from a domain sharing dataset although with different vocabulary and schema. Our approach considers that type of systems. Notice that a proper rewriting of the query must be eventually managed by those systems.

In general, those kind of systems can be very useful in different scenarios. For example, an ordinary user posing a keyword-based query to a question answering system, which constructs a SPARQL query to be run on a source dataset, and then demanding for more answers from a dataset with different vocabulary. Another scenario can be that of scientists formulating queries over source datasets they are familiar with, and then, demanding more answers by accessing other different datasets, not requiring strict query equivalence but giving a chance to serendipity (notice that scientists need not be aware of the internal structure/vocabulary of the new target datasets). A third scenario can be that of an application programmer trying to query the English DBpedia using terms extracted from the user defined Spanish Wikipedia infoboxes (or whatever language Wikipedia), which are not mapped with official DBpedia terms. Then, in order to get some answers, a transformation of the source query is needed in order to be adequately expressed for the English DBpedia. The relevant common feature of all these scenarios is the need to cope with the vocabulary and schema heterogeneity of the stored data. Notice that such heterogeneity may reach the conceptual level leading to different granularity knowledge and to the point that some notions are conceptualized in one dataset but not in the others.

Our system deals with a query rewriting process where the preservation of the semantics is not a strong requirement and therefore it considers semantics-preserving and non-semantics-preserving rewritings in order to increase the opportunities of getting results. When a non-semantics-preserving scenario is considered, the definition of a quality estimation of the rewritten query becomes crucial because the user needs to be aware of the confidence that can be deposited on the results obtained from the new dataset.

As a motivating example, let us imagine a user that is only familiar with the LinkedMDB vocabulary (a dataset about movies and their related people). This user asks for the films and the names of art directors working on those films directed by Woody Allen and performed by Sean Penn. The SPARQL query constructed by the user could be the following one:

```
PREFIX mdb:<http://data.linkedmdb.org/
    resource/movie/>
SELECT  DISTINCT ?movie ?name
WHERE {
  ?woody  mdb:director_name "Woody Allen".
  ?movie  mdb:director ?woody;
          mdb:actor ?actor;
          mdb:film_art_director ?art.
  ?actor  mdb:actor_name "Sean Penn".
  ?art mdb:film_art_director_name ?name. }
```

Listing 1: Films and names of art directors working on those films directed by Woody Allen and performed by Sean Penn.

and the obtained results are listed on table 1:

?movie	?name
db:film/38778	“Tom Warren"

Table 1
Query results from LinkedMDB.

Given the scarcity of the response or its inadequacy, the user would find useful to execute the same query in other datasets, perhaps more recognized ones or more active ones, trying to obtain more results. A good example of those datasets may be DBpedia. Using our system the user could obtain the following reformulation of the query, according to the DBpedia vocabulary:

```
PREFIX dbo:<http://dbpedia.org/ontology/>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?movie ?name
WHERE {
  ?woody   foaf:name "Woody Allen"@en.
  ?movie   dbo:director ?woody;
           dbo:starring ?actor.
           dbo:cinematography ?art.
  ?actor   foaf:name "Sean Penn"@en.
  ?art  foaf:name ?name. }
```

Listing 2: Films and names of cinematographers working on those films directed by Woody Allen and performed by Sean Penn.

This reformulation was based on some declared mapppings. In particular, mdb:director was declared an equivalent property to dbo:director in a set of RDF triples published as an Open Linked Data file[1], captured from the web, and incorporated into a local Open Link Virtuoso RDF Store which allows to access them through a local SPARQL endpoint. Moreover, properties mdb:director_name, mdb:actor_name and mdb:film_art_director_name were declared as subproperties of foaf:name in the form of mappings by the own LinkedMDB dataset[2]. Although no declared mapping for mdb:film_art_director was found, the system proposed the term dbo:cinematography as an approximation to the original one, the same happened with the properties: mdb:actor and dbo:starring (in section 4 we will show how this kind of proposed approximations can be discovered). The results obtained by the query in listing 2 are presented on table 2:

?movie	?name
dbr:Sweet_and_Lowdown	“Zhao Fei"
dbr:Sweet_and_Lowdown	赵非

Table 2

Query results from DBpedia. dbr:<http://dbpedia.org/resources/>

Notice that new results appeared when querying the DBpedia dataset, which may be of interest to the user. Moreover, which further enriches the answer is the provision of some quality estimation of the reformulated query. It is at this point where a main contribution of this paper plays a relevant role. In this particular case, our system offered a *similarity factor* of 0.86 and a *quality score* of $0, 79$. The meaning of these features will be explained in subsequent sections.

In general, quality estimation can be defined in terms of a query similarity measure between the source and target queries (source query, formulated over the initial dataset, and target query, formulated over another dataset of the Web of Data indicated as target). However, when comparing two queries, different similarity dimensions can be considered [4]: (1) the query structure, expressed as a string or a graph structure; (2) the query content, its triple patterns and ontological terms and literal values; (3) the language features, such as query operators and modifiers; and (4) the results set retrieved by the query. Queries may be assessed with respect to one or several of that considered dimensions. And it is widely accepted that the application context heavily determines the choice for a similarity measure. We think that the more relevant dimensions to be taken into account in the scenario considered in this paper are (2) the content and (4) the result set. After all, structure and language features of a target query seem to be of less relevance to inform the user about the intended similarity spirit to the formulated source query. Therefore, (1) structure and (3) language dimensions are not considered in the proposed assessment. Although the result set is what matters to the user, it is crucial to notice that the intention of issuing the query to a target dataset is to look for more or different results than those obtained by the source query. Therefore, the intended result set of the target query cannot be compared with that of the source query in terms of exact matching and the query similarity measure must take into account this distinctive feature.

In summary, a main contribution of this paper is an approach that estimates the quality of rewritten queries, including non-semantics-preserving rewriting, to search the distributed space of globally interlinked data, that is different from the *centralized repository*, *federated*, and *exploratory* aforementioned approaches. Terms for the source query can be selected from a preferred schema or vocabulary instead of that offered by the centralized approach or federated schema. Moreover, target datasets are not limited to those comprising the centralized repository or the federation. Technical contributions to support the proposed approach are: (1) A proposal of a general frame-

[1] http://wifo5-03.informatik.uni-mannheim.de/bizer/r2r/examples/mappings.ttl

[2] http://wiki.linkedmdb.org/Main/Interlinking

work for the deployment and management of a query rewriting system concerning the scenario previously explained. And, (2) the validation of the framework in a real context, including the machine learning and optimization techniques used to estimate the quality of the rewriting outcome.

Proposal of a general Framework. We propose a new framework that groups the following components: query rewriting rules, an algorithm to manage the rules, a set of similarity measures, and a predictive model. This framework would be oriented to two types of users:

- *End users*, who formulate a query over a source dataset and then the system issues a new query, which mimics the original one, over a target dataset (of the Web of Data) whose results further enrich the answer. The new issued query is annotated with a similarity factor between queries and a quality score.
- *Expert/technical users* who, in addition to benefiting from the functionalities provided for end users, can also include in the framework: new rewriting rules, algorithms to process them, and similarity measures to qualify the query rewriting. The framework would also provide them with facilities to tune the introduced similarity measures by means of optimization techniques. The rewriting rules, similarity measures and training queries introduced could be stored in the log of the framework with the idea of serving as experimental and comparison benchmark.

Validation of the Framework. The framework has been tested in a real context. For that, we have instantiated the framework with the following elements:

- Query rewriting rules. Apart from some rules dealing with the rewriting of terms by their specified equivalents, via synonym mappings or EDOAL (Expressive and Declarative Ontology Alignment Language) [5] alignment rules, the framework also deals with some other heuristic based rules which conform a carefully controlled set of cases.
- An algorithm to schedule the application of the rules.
- Similarity measures. The computation of the similarity factor takes into account different similarity measures depending on the motif of the rule being applied. Those motifs range from relational to ontological structure, and from language based to context based similarity.
- Queries. 100 queries were formulated over the previously selected datasets. Three domain areas were considered for the datasets: media-domain, bibliographic, and life science. From the media-domain six datasets were selected, five datasets from the bibliographic domain, and five more from the life science domain, respectively. When selecting the queries, our aim was to get a set that would contain a broad spectrum of SPARQL query types [6]. Concerning provenance we selected queries that appeared in well known benchmarks such as QALD4 or FedBench, and we also considered queries that belonged to LOD SPARQL endpoints logs from the selected datasets.
- Predictive model. A model has been created using a supervised machine learning method applied to the considered experimental scenario to predict the F1 score of each rewritten query.

The rest of the paper is organized as follows. Section 2 presents some related work in the scope of resource matching and query rewriting. Section 3 introduces a description of the main features of the proposed framework. Section 4 shows a framework embodiment. Section 5 describes the framework validation results. Finally, some conclusions are presented.

2. Related work

The impressive growth of the Web of Data has pushed the research on Data Linking [7]: “the task of determining whether two object descriptions can be linked one to the other to represent the fact that they refer to the same real-world object in a given domain or the fact that some kind of relation holds between them". Those object descriptions can be expressed with diverse structural relationships, depending on different contexts, using classes and properties from different ontologies. Research on object similarity and class matching has issued a considerable amount of techniques and systems in the field of Ontology Matching [8], although less work has been devised for property alignment [9, 10]. The work in [11] presents an unsupervised learning process for instance matching between entities. Queries considered in this paper involve terms for classes, properties and individuals. Therefore, techniques for discovering similarity for any of them are relevant. However, the topic of this paper regards query similarity, which can be recognized as a different problem. As has been noticed in [4], the appropriate notion of query similarity depends on the goal of the task. In our case, the task is to estimate the similarity of the intended semantics between a query designed for a source dataset and a rewriting to a different vocabulary, to be evaluated in a different target dataset.

Some works, for example [12, 13], have approached a restricted version of the task carried out in our case. They restrict themselves to produce semantic preserving translations (i.e. total similarity) and so they assume that enough equivalent correspondences exist among entities in datasets. Taking into account that such an assumption is too strong in real scenarios we consider situations where different types of correspondences exist (not only of equivalence type) and even more, situations where some correspondences are missing. This consideration implies that query semantics is sometimes not preserved in the rewriting process and therefore the estimation of similarity of the produced rewriting becomes crucial.

The aim of our considered rewriting is to look for more answers in a target dataset than those obtained from the source dataset. Some other works have the goal of obtaining more answers (including approximate ones) for an original query; however, all of them restrict their scope to a single source dataset. In [14] they propose a logical relaxation of conditions in conjunctive queries based on RDFS semantics. Those conditions are successively turned more general and a ranking in the successively obtained answers is generated. [15, 16] use the same kind of relaxations as [14], but propose different ranking models. In [15], similarity of relaxed queries is measured with a model based on the distance between nodes in the ontology hierarchy. In [16], they use an information content based model to measure similarity of relaxed queries. The work in [17] addresses the query relaxation problem by broadening or reformulating triple patterns of the queries. Their framework admits replacement of terms by other terms or by variables and also removal of entire triple patterns. In that work, generation and ranking of relaxed queries is guided by statistical techniques: a distance between the language models associated to entity documents is defined. All those works can be situated under the topic of query relaxation.

With different use cases in mind, the papers [18–20] present different possibilities for approaching the querying of Linked Data. In [19] a framework for relaxation of star-shaped SPARQL queries is proposed. They present different *matchers* (functions that map pairs of values to a relaxation score) for different kinds of attributes (numeric, lexical or categorical). The framework may involve multiple matchers. The matchers generate a tuple of numeric distances between a query and an entity (answer for the query). Notice that the distance is defined between an entity and a query, not between two queries as in our approach. [18] proposes a measure to evaluate the similarity between a graph representing a query and a graph representing the dataset. With a suitable relaxation of the notion of alignment between query graph paths and dataset graph paths they generate approximate answers to queries. In [20] a method for query approximation, query relaxation, and their combination is proposed for providing flexible querying capabilities that assist users in formulating queries. Query answers are ranked in order of increasing distance from the user's original query.

In summary, cited works that transform the query or reformulate the notion of answer in order to provide users with more answers from the source dataset, do not try to reformulate the query in a different dataset with different vocabulary and data structure; and this is a distinguishing feature of our use case.

It is worth mentioning another data access paradigm that uses query rewriting. In the Ontology Based Data Access (OBDA) paradigm, an ontology provides a conceptual view of the data and a vocabulary for user queries [21]. Users pose queries in terms of a convenient conceptualization and familiar vocabulary, without being aware of the details of the structure of data sources. SPARQL can be considered as a query language in this paradigm [22], and the SPARQL query must be rewritten in an appropriate query language for the underlying data source which, for instance, could be SQL for relational databases. Such rewritting is based on mappings between terms in the ontology and (in case of relational databases) views of the relational schema. R2RML [23] is a W3C standard language for specifying those mappings. A sufficiently complete set of mappings must be specified in order to rewrite the query, since OBDA paradigm intends to process a query, over the underlying data source, which is semantically equivalent to the query posed by the user with the ontology vocabulary. Notwithstanding the relevance of OBDA paradigm, we point out that it tackles with a different problem to the stated one in this paper. OBDA rewrites a query to adapt it to another data model. The problem tackled in this paper is to rewrite a SPARQL query to adapt it to another vocabulary without considering complete mappings between the respective vocabularies.

3. Abstract framework

An abstract representation of the proposed framework for rewriting a query and estimating the quality of the rewritten query, can be expressed as a structure ($\mathcal{R}$, $\mathcal{A}$, $\mathcal{Q}$, $\mathcal{P}$) where

- $\mathcal{R}$ is a set of SPARQL query rewriting rules,
- $\mathcal{A}$ is the algorithm for applying the rules,
- $\mathcal{Q}$ is a rewriting quality estimation system, composed of three elements ($\mathcal{M}$, $\mathcal{V}$, $\mathcal{SF}$) such that
  - ∗ $\mathcal{M}$ is a set of similarity measures between fragments of query expressions,
  - ∗ $\mathcal{V} : \mathcal{R} \to \mathcal{M}$ is an application that associates a similarity measure to each rule, and
  - ∗ $\mathcal{SF} : \mathcal{R}^* \to [0, 1]$ associates each sequence of applied rewriting rules with a similarity factor from the $[0, 1]$ real interval,
- $\mathcal{P}$ is a predictive model which estimates a quality score for the target query.

The part of a SPARQL query to be rewritten by rules in $\mathcal{R}$ is the graph pattern in the WHERE clause of the query. A graph pattern consists of a set of *triple patterns*. A *triple pattern* is a triple (s, p, o) where s is the subject, p is the predicate, and o is the object. The three of them represent resources and any of them can be a *variable* (denoted by prefixing it with a question mark, for instance $?x$). The rule language is a variation of the CONSTRUCT query form of SPARQL 1.1, as follows:

```
REPLACE   template
BY        template
WHEN {
   graph pattern
}
```

The REPLACE clause presents a *template* that should be matched to a part of the graph pattern in the query being rewritten. This matching is the trigger of the rule. A *template* is a graph pattern including three kinds of *tokens*: *IRI tokens*, *variable tokens*, and *wild tokens*. A *IRI token* only binds to IRIs in the graph pattern of the query, a *variable token* only binds to variables, and a *wild token* binds to both. IRI tokens are prefixed by *s:* or *t:* meaning that they only bind to IRIs in the source or target dataset, respectively. Variable tokens are prefixed with a question mark. Wild tokens are prefixed with a hash, for instance *#u*. The matched part in the graph pattern of the query will be replaced by the binded template in the BY clause if the graph pattern in the WHEN clause find matches with the data graph of the datasets in question (the BY clause resembles the CONSTRUCT clause and the WHEN clause resembles the WHERE clause in SPARQL queries, but for replacement of triple patterns in a graph pattern).

For instance, the following rule:

```
PREFIX s:<source dataset>
PREFIX t:<target dataset>
REPLACE  #s s:p  #o .
BY       #s t:p  #o .
WHEN {
   s:p owl:sameAs t:p .
}
```

applied to the query in listing 1, produces the query

```
PREFIX mdb:<http://data.linkedmdb.org/
     resource/movie/>
PREFIX dbo:<http://dbpedia.org/ontology/>
SELECT DISTINCT ?movie ?name
WHERE {
  ?woody  mdb:director_name "Woody Allen".
  ?movie  dbo:director ?woody;
          mdb:actor ?actor;
          mdb:film_art_director ?art.
  ?actor  mdb:actor_name "Sean Penn".
  ?art  mdb:film_art_director_name ?name. }
```

Listing 3: mdb:director replaced with dbo:director.

by rewriting the triple pattern (?movie mdb:director ?woody) by (?movie dbo:director ?woody) due to the appearance of (mdb:director owl:sameAs dbo:director) in the consulted data graph, in this particular case in the local Virtuoso RDF store.

This rule language is sufficiently expressive since WHEN clauses can use the full expressivity of graph patterns in SPARQL 1.1. The core of the implementation of those rules can be supported by an almost direct generation of SPARQL queries from the rule expression. For instance, the query supporting the previous sample rule could be constructed in the following way: Assume that the template *(#s s:p #o)* (appearing in the REPLACE clause) matches a part of the graph pattern in the query being rewritten and yields a binding of the IRI token *s:p* to the IRI *s:IRIp* in the source dataset. Then, the triple pattern (*s:IRIp owl:sameAs ?t0p*) (originated from the WHEN clause) will be the WHERE clause and *?t0p* will be the projected variable in the SELECT clause due to the BY clause in the rule (see Listing 4).

```
SELECT  ?t0p
WHERE {
  s:IRIp  owl:sameAs  ?t0p  .
}
```

Listing 4: Query supporting rule implementation.

Then, the results of that query can be used to form the corresponding replacements specified in the BY clause.

Rules in this paper only consider the basic RDF entailment regime. Nevertheless, if the mediating SPARQL endpoint managing the query processing implements another entailment regime over the dataset of interest, the rewriting process leverages on that enriched regime without any harm.

A rewriting rule set requires an algorithm to manage the rewriting process, this is the role of element $\mathcal{A}$ in the abstract framework. Different algorithms managing the same set of rules may produce different outcomes.

The rewriting system of the proposed framework takes a given query Q_s (named *source query*), expressed with a vocabulary adequate[3] for the source dataset, and transforms it into another query Q_t (named *target query*), expressed with a vocabulary adequate for the selected target dataset. In this paper, the primary problem of a single target dataset is considered, although the process may be iterated with a different target dataset each time. Decomposition of the source query into parts and distribution of each part to a different target dataset is devoted to future work. However, it should be noted that it could be solved by combination of solutions of the primary problem. The rewriting process produces Q_t as a semantically equivalent query to Q_s as long as enough equivalence mappings between the vocabulary of the source dataset and the vocabulary of the target dataset are found. But the distinguishing point is that the process produces a mimetic query Q_t even in the case when no equivalent translation for Q_s is found. That is to say, semantic preservation cannot be guaranteed due to vocabularies heterogeneity and missing links with terms appearing in the source query. It is at this point where the definition of a quality estimation of the rewriting outcome becomes crucial to our approach, because the user needs to be aware of the quality of the produced target query. This is the goal of the $\mathcal{Q}$ element in the abstract framework.

Every application of a rule r is considered as a step in the progress to the target query, and such steps are valuated with a factor computed by the associated similarity measure $\mathcal{V}(r)$ from $\mathcal{M}$.

The function $\mathcal{SF}$ calculates a similarity factor for a target query in terms of the sequence of rules $\bar{r}$ that were applied to construct it and properly combining the measures $\mathcal{V}(r)$ (for each $r \in \bar{r}$). Similarity measures in $\mathcal{M}$ can be defined by simple functions or very complex ones. Usually they can be defined by combining similarity measures taken from a state-of-the-art repository [8].

As previously said in the introduction section, the intention of issuing a query to the selected target dataset is to look for more or different results than those obtained in the source dataset. Therefore, although the target query should try to maintain the spirit of the source query, the intended result set of the target query cannot be compared with the source query retrieved set but with that of an *ideal* expression of such source query in terms of the vocabulary acceptable by the target dataset. Notice that, due to the previously mentioned heterogeneity reasons, such ideal expression cannot be trivially constructed. In fact, we consider that the finding of such ideal expression, in the considered scenario, should be realized by a human expert who knows vocabularies of source and target datasets. And, therefore, the reference query against which the target query should be compared is a human designed one, that tries to express the most similar intention to the source query but in the context of the target dataset. We consider such a query our *gold standard* query against which the target query should be compared.

In the presence of a gold standard query, its results can be compared with those obtained by the target query. Statistical measures such as precision, recall and F1 score can be used to measure the quality of a target query. Of course, gold standards can only exist in an experimental scenario but not in the real setting, and that is the reason to incorporate machine learning techniques in the framework. The predictive model $\mathcal{P}$ is generated by a supervised machine learning method applied to a suitable experimental scenario consisting

[3] We say that a term is *adequate* for a dataset if its IRI prefix follows the proprietary format of the dataset or it appears in the dataset vocabulary.

of a selected benchmark of source queries with their respective gold standard queries for the target datasets, and the set of corresponding target queries generated by the rewriting system with their respective $\mathcal{SF}$ value and with their respective F1 score that will be the goal for prediction.

Note that this framework establishes, principally, a scenario for experimentation, where different materializations of each element of the framework can be assessed and compared.

In particular, it should be taken into account that the provision of gold standard queries involves a delicate work: knowledge of different vocabularies is needed and sometimes different choices can be considered as appropriate gold standard of a query. Furthermore, the production of desired quantities of training queries is a time consuming task. The editors of the gold standard queries considered in the experiment reported in this paper were just the authors of the paper. Queries were grouped by domain and each domain group was assigned to a different author. Then, the work was reviewed jointly. Therefore, the results of our experiments could be considered to be improved if we had a larger and much more supervised collection of queries.

4. Framework embodiment

This section presents a brief explanation of a specific embodiment of the abstract framework ($\mathcal{R}$, $\mathcal{A}$, $\mathcal{Q}$, $\mathcal{P}$) that was partially presented in [24] and which serves as a proof of concept for our proposal.

The set of rules $\mathcal{R}$ was devised from a pragmatic point of view. The rules set up common sense heuristics to obtain acceptable rewritings even when no semantically equivalent translations are at hand. Preconditions for the application of the rules take into account a carefully restricted context of the terms occurring in the graph pattern. Although restricted, the rule set has shown to be quite effective achieving acceptable rewritings (see section 5).

Five kinds of rules have been considered, each kind based on a different motif: Equivalence (E), Hierarchy (H), Answer-based (A), Profile-based (P), and Feature-based (F).

Furthermore, a pragmatic scenario has been considered in which a bridge dataset can be taken into account in the process of rewriting a query adequate for a source dataset into another query adequate for a target dataset. In order to favour the possibilities of finding alignments between resources, mappings between both the source (D_s) and target (D_t) datasets and a bridge (D_b) dataset are considered. Such a choice is justified because that scenario is quite frequent, since in almost any domain there is a popular dataset that may play such a reference role. For instance: BabelNet in the linguistic domain, DBLP in the Computer Science Bibliographic domain, NCI Thesaurus in the clinical domain, New York Times-Linked Open Data in the media domain, reference.data.gov.uk in government domain, or Dbpedia in cross domain. There is not a fixed bridge dataset for each domain, any dataset may play the role of bridge dataset for each occasion instead. More ambitious scenarios may consider bridge concatenations, but a balance between computational cost and completeness decided us for restricting to only one bridge dataset per rule application.

Equivalence rules basically consist in replacing a query fragment by an equivalent one. They are the most frequent kind of query rewriting rules in the technical literature. Of course, their use is the most reasonable decision when such equivalence mappings are at hand; and can be confident that such rewriting preserves the semantics of the query. In section 3 a simple equivalence rule regarding the predicate of a triple pattern was applied to the source query in listing 1. Next, the expression of another of our equivalence rules is presented, namely one that replaces the subject of a triple pattern. Notice that a bridge dataset is used and diverse equivalence mappings are considered (see the FILTER clauses):

```
PREFIX s:<source dataset>
PREFIX t:<target dataset>
PREFIX b:<bridge dataset>
REPLACE   s:u  #p  #o .
BY        UNION(t:u #p  #o)
WHEN {
   s:u  ?eq1  b:u .
   b:u ?eq2  t:u  .
   FILTER (?eq1 = owl:sameAs ||
           ?eq1 = owl:equivalentClass ||
           ?eq1 = owl:equivalentProperty ||
           ?eq1 = skos:exactMatch)
   FILTER (?eq2 = owl:sameAs ||
           ?eq2 = owl:equivalentClass ||
           ?eq2 = owl:equivalentProperty ||
           ?eq2 = skos:exactMatch)
}
```

where UNION(t:u #p #o) represents the UNION pattern of all the triple patterns constructed with the IRIs binded with t:u.

The similarity measure associated to equivalence rules (E) is simply the constant function $\phi(u) = 1$, representing the semantics preservation after the replacement of the non adequate term *u*.

Hierarchy rules consist in replacing a term by a semantic generalization or restriction of that term. Such kind of rules are considered in works that account for relaxing or narrowing queries. In cases where equivalence is not guaranteed, replacing a term by its most specific subsumer or its most general subsumee expression changes the semantics in a ontological biased way. Next, the expression of one of our hierarchy rules is presented:

```
PREFIX s:<source dataset>
PREFIX t:<target dataset>
REPLACE  #s  s:p  #o .
BY          AND(#s t:p #o)
WHEN {
    s:p  ?sub  t:p .
    FILTER (?sub = rdfs:subPropertyOf ||
              ?sub = skos:narrower)
}
```

where AND(#s t:p #o) represents the conjunction pattern of all the triple patterns constructed with the IRIs binded with t:p. Three successive applications of this hierarchy rule to the query in listing 3 rewrites it to the following query:

```
PREFIX mdb:<http://data.linkedmdb.org/
      resource/movie/>
PREFIX dbo:<http://dbpedia.org/ontology/>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?movie ?name
WHERE {
   ?woody   foaf:name "Woody Allen".
   ?movie   dbo:director ?woody;
            mdb:actor ?actor;
            mdb:film_art_director ?art.
   ?actor   foaf:name "Sean Penn".
   ?art   foaf:name ?name.  }
```

Listing 5: mdb:director_name, mdb:actor_name and mdb:film_art_director_name replaced with foaf:name.

by rewriting the properties mdb:director_name, mdb:actor_name and mdb:film_art_director_name by the property foaf:name due to the appearance of (mdb:director_name rdfs:subPropertyOf foaf:name), (mdb:actor_name rdfs:-subPropertyOf foaf:name) and (mdb:film_art_director_name rdfs:subPropertyOf foaf:name) as mappings provided by the own LinkedMDB dataset[4].

Similarity estimation of hierarchy related terms is usually based on a distance measure. It is generally considered that the depth associated to the compared terms in the hierarchy influences the conceptual distance between the terms. Low depth correspond to more general terms and high depth correspond to more specific terms. Nearby high depth terms tend to be more semantic similar than low depth ones. Consequently, the similarity function selected for hierarchy rules (H) was an adaptation of a distance proposed in [25] and elsewhere. Each term *u* in the hierarchy is associated with a *milestone* value *m(u)* depending on its depth in the hierarchy. Then, the distance between two terms *u* and *v* in the hierarchy is $d(u, ccp(u, v)) + d(v, ccp(u, v))$ where *ccp(u,v)* is the *closest common parent* of *u* and *v* in the hierarchy and $d(x, ccp(x, y)) = m(ccp(x, y)) - m(x)$. The milestone of a term *u* is defined as:

$$m(u) = \frac{1}{2 \times k^{depth(u)}}$$

where *k* is a predefined factor bigger than 1 that indicates the rate at which the value decreases along the hierarchy, and $depth(u)$ is the length of the longest path from the node *u* to the *root* ($depth(root) = 0$). In this case we used $k = 2$.

However, in the context considered in this paper, where *u* and *v* belong to different vocabularies and, therefore, different hierarchies, the notion of *ccp(u,v)* is not directly applicable. Then, we have adapted such distance. The intuition behind is that the deepest term (i.e. that with the least milestone) carries more information than the higher term:

$$\begin{aligned}
distance(u, v) &= m(u) \times (1 - \frac{1}{k}) \\
&\quad \text{if } m(u) = m(v) \wedge u \sqsubseteq v \\
distance(u, v) &= m(v) - m(u) \\
&\quad \text{if } m(u) < m(v) \wedge u \sqsubseteq v \\
distance(u, v) &= m(u) \times (1 - \frac{1}{k}) \\
&\quad \text{if } m(u) < m(v) \wedge v \sqsubseteq u
\end{aligned}$$

Once the distance is defined, the similarity function S_o between the terms *u* and *v* is

$$S_o(u, v) = 1 - distance(u, v)$$

[4] http://wiki.linkedmdb.org/Main/Interlinking

The number of terms involved in the replacement of a term u and its triple pattern can be more than one, depending on the particular bindings of the applied rewriting rule. In such a case, if there are n bindings $(u_1, \ldots, u_n)$, the similarity measure is the average of the n pairwise similarity values:

$$\phi(u) = \frac{\sum_{i=1}^{n} S_o(u, u_i)}{n}$$

In the particular case of the current query example, it resulted $\phi(mdb : director\_name) = 0.6$, $\phi(mdb : actor\_name) = 0.6$ and $\phi(mdb : film\_art\_director\_name) = 0.6$.

Equivalence and hierarchy rules can be applied when direct mappings for the term to be replaced are found. But, if the involved datasets are not completely aligned and those direct mappings are missing, new kinds of rules trying to leverage mappings of terms surrounding the focussed term should be considered. And that is precisely what our proposed Profile-based, Answer-based, and Feature-based kinds of rules do.

Let us call the *profile* of a resource x in a dataset D to the set of resources that are related to x, as subjects or objects, through triples in D. More specifically:

$$\begin{aligned}\mathcal{P}_D(x) = \{v \in Terms(D) \mid \\ (\exists p.(x, p, v) \in D \lor (v, p, x) \in D) \lor \\ (\exists a.(a, x, v) \in D \lor (v, x, a) \in D)\}\end{aligned}$$

The heuristic considered in **Profile-based** rules is the following: if a resource v, in the profile of the focused resource u, is equivalent to a resource $t : v$ in the target dataset, and there is a resource $t : u$ in the profile of $t : v$, sufficiently similar to u, then u could be replaced by $t : u$.

For instance, the following triples:

```
mdb:film/38778   mdb:film_art_director
     mdb:film_art_director/238 .
mdb:film/96785   mdb:film_art_director
     mdb:film_art_director/1 .
mdb:film/38180   mdb:film_art_director
     mdb:film_art_director/2 .
mdb:film_art_director/84    rdf:type
     mdb:film_art_director .
mdb:film_art_director/363  rdf:type
     mdb:film_art_director .
```

are only some of the triples in LinkedMDB that would determine the profile of mdb:film_art_director. Considering only such small set, the profile would be the set:

```
{mdb:film/38778,   mdb:film_art_director/238,
 mdb:film/96785,   mdb:film_art_director/1,
 mdb:film/38180,   mdb:film_art_director/2,
 mdb:film_art_director/84,
 mdb:film_art_director/363}
```

For the sake of the example, let us consider only one of those resources in the profile. For instance, mdb:film/38778. A mapping of equivalence was found betweeen mdb:film/38778 and the resource dbr:Sweet_and_Lowdown in DBpedia. And some triples were found in DBpedia involving dbr:Sweet_and_Lowdown. For instance:

```
dbr:Sweet_and_Lowdown   dbo:cinematography
     dbr:Zhao_Fei .
dbr:Sweet_and_Lowdown   dbo:director
     dbr:Woody_Allen .
dbr:Sweet_and_Lowdown   dbo:distributor
     dbr:Sony_Pictures_Classics .
dbr:Sweet_and_Lowdown   dbo:editing
     dbr:Alisa_Lepselter .
dbr:Sweet_and_Lowdown   dbo:gross
     4197015.0 .
dbr:Sweet_and_Lowdown   dbo:producer
     dbr:Jean_Doumanian .
dbr:Sweet_and_Lowdown   dbo:runtime
     5700.000000^(xsd:double) .
```

The same process should be done with all the resources of the profile. Next, calculating the similarity between mdb:film_art_ director and any of the predicates appearing in the preceding set of triples we found the following values:

```
S(mdb:film_art_director, dbo:cinematography)
     =0.72
S(mdb:film_art_director, dbo:director)=0.61
S(mdb:film_art_director, dbo:distributor)
     =0.56
S(mdb:film_art_director, dbo:editing)=0.54
S(mdb:film_art_director, dbo:gross)=0.31
S(mdb:film_art_director, dbo:producer)=0.20
S(mdb:film_art_director, dbo:runtime)=0.18
```

Then, the resource with the maximum similarity value was selected to replace mdb:film_art_director, namely dbo:cinematography. And the value of the similarity measure was $\phi(mdb : film\_art\_director) = 0.72$.

Next, the expression of the profile rule responsible of the previous replacement is presented:

```
REPLACE   #s  s:p  #o .
BY        #s  t:p  #o .
WHEN {
    ?s s:p ?o .
    ?s ?eq ?ts .
    ?o ?eq ?to .
    {?ts t:p ?to .}
    UNION
    {?to t:p ?ts .}
    FILTER (?eq = owl:sameAs ||
            ?eq = skos:exactMatch)
    FILTER (t:p = maxSim(s:p, h, profile(s:p))
}
```

Regarding the notion of sufficient similarity, we decided to establish a threshold h to be exceeded by the value of a similarity function between resources $S(u, v)$, in order to consider v sufficiently similar to u. When several resources exceed the threshold, the resource with maximum similarity value is selected for the replacement. Let us denote *maxSim*(u, h, R) to a resource w in the set of resources R which is the most similar to u and whose similarity value is greater than the threshold value h:

$$\begin{aligned} &maxSim(u, h, R) = w \\ &\qquad \text{such that} \quad w \in R, \\ &\qquad\qquad\qquad S(u, w) \geqslant h, \text{ and} \\ &\qquad\qquad\qquad \forall v \in R.\, S(u, w) \geqslant S(u, v) \end{aligned}$$

In the current framework, the similarity function of two resources is defined as a linear combination of some other three similarity measures, which are selected to compare a context of the terms.

$$S(u, v) = \alpha_n \cdot S_n(u, v) + \alpha_d \cdot S_d(u, v) + \alpha_o \cdot S_o(u, v)$$
$$\alpha_n, \alpha_d, \alpha_o \geqslant 0 \ \wedge \ \alpha_n + \alpha_d + \alpha_o = 1$$

S_n and S_d are string based methods, and S_o is the similarity measure previously defined. S_n is a similarity measure computed as the average of Levenshtein and Jaccard distances, corresponding to the rdfs:label property value of the two compared terms. And S_d takes into account the definition contexts of the terms: for each compared term, u and v, a bag of words is constructed containing words from their rdfs:comment and rdfs:label string valued properties. S_d is defined as a cosine similarity of two vectors $V(u)$ and $V(v)$ constructed by the frequency of word appearance (i.e. Vector Space Model technique):

$$S_d(u, v) = \frac{V(u) \cdot V(v)}{\|V(u)\| \, \|V(v)\|}$$

Definition of $S(u, v)$ can be considered simple if compared to functions that involve more sophisticated linguistic techniques, or use some other Information Content techniques, or take into account much more information about the resources. But we think that the computational cost of those alternatives must be carefully considered, given the use case scenario presented in the introduction of this paper. In spite of its simplicity, results of our experiments are encouraging for researching along that line (see section 5).

Then, the similarity measure associated to u after the application of a Profile-based rule is

$$\phi(u) = S(u, maxSim(u, h, profile(u)))$$

This kind of rule was used to replace mdb:film_art_director by dbo:cinematography, and also mdb:actor by dbo:starring, in the working example presented in the introduction, with a $\phi(mdb : film\_art\_director) = 0.72$ and $\phi(mdb : actor) = 0.89$. The result is the query presented in listing 2 which is completely expressed with the target dataset vocabulary.

However, after the application of some query rewriting rules, some non adequate terms of the query could still be unreplaced. The proposal in this paper considers two more kinds of rules in order to cover some more interesting circumstances. **Answer-based** rules are supported by bindings obtained, during the source query processing over the source dataset, for the piece of graph pattern to be replaced. Those binded resources are considered as examples of what the query is looking for in the target dataset. Triples involving those resources in the target dataset are used to mimic the triple pattern to be replaced. The intuition behind is that triples stated about the answer samples in the target dataset probably resemble expected answers of the original query.

For instance, consider the query

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?a ?p ?q
WHERE {
  dbr:The_Other_Side_of_the_Wind ?p ?a .
  ?a ?q dbo:Agent.
}
```

Listing 6: Information about The Other Side of the Wind.

for DBpedia as source dataset and consider LinkedMDB as the target dataset. Then, dbr:The_Other_Side_of_the_Wind and dbo:Agent are not adequate for LinkedMDB.

Consider the following set of triples in the source dataset (DBpedia):

```
dbr:The_Other_Side_of_the_Wind dbo:starring
     dbr:John_Houston.
dbr:The_Other_Side_of_the_Wind dbo:starring
     dbr:Peter_Bogdanovich.
dbr:The_Other_Side_of_the_Wind dbo:director
     dbr:Orson_Wells.
dbr:The_Other_Side_of_the_Wind
     dbo:cinematography dbr:Gary_Graver.
dbr:The_Other_Side_of_the_Wind dbo:starring
     dbr:Susan_Strasberg.
```

Consider the five bindings of variable $?a$, in the first triple pattern of the query, as answer samples to that triple pattern. And consider also that those answer samples are mapped as equivalent to corresponding resources in the target dataset (LinkedMDB):

```
dbr:John_Houston owl:sameAs mdb:actor/29769.
dbr:Peter_Bogdanovich owl:sameAs
                      mdb:actor/29762.
dbr:Orson_Wells owl:sameAs mdb:producer/9736.
dbr:Gary_Graver owl:sameAs mdb:actor/9677.
dbr:Susan_Strasberg owl:sameAs
                        mdb:actor/37472.
```

Then, triples about those answer samples in the target dataset could probably resemble expected answers of the original query. For each one of those answer samples in the target dataset (LinkedMDB), triples as the following are found in the dataset:

```
mdb:actor/29769   mdb:actor mdb:film/133;
                  mdb:actor mdb:film/1025;
                  mdb:actor mdb:film/46921;
                  mdb:actor mdb:film/23486.
mdb:actor/29762   mdb:actor mdb:film/38395;
                  mdb:actor mdb:film/46921.
mdb:producer/9736   mdb:actor mdb:film/38078;
                    mdb:actor mdb:film/46921;
                    mdb:actor mdb:film/66530.
mdb:actor/9677   mdb:actor mdb:film/89003;
                 mdb:actor mdb:film/46921.
mdb:actor/37472   mdb:actor mdb:film/274;
                  mdb:actor mdb:film/46921.
```

As a crude approximation, the most frequent resource appearing in such a context could be considered to replace the non adequate term dbr:The_Other_Side_of_the _Wind in the query. In this case, mdb:film/46921 appeared 11 times in the running experiment and was selected for the replacement. Then, its similarity value was calculated, yielding $\phi(dbr : The\_Other\_Side\_of\_the\_Wind) = 0.71$.

Then, the rewritten query obtained after applying that answer-based rule would be the following :

```
PREFIX mdb: <http://data.linkedmdb.org/
    resource/movie/>
SELECT DISTINCT ?a ?p ?q
WHERE{
   mdb:film/46921 ?p ?a .
   ?a ?q dbo:Agent .
}
```

Listing 7: Information about movie:46921.

The rule expression capturing the process described above could be as follows:

```
PREFIX s:<source dataset>
PREFIX t:<target dataset>
REPLACE s:u ?p ?x
BY      t:u ?p ?x
WHEN {
   ?s as t:u (COUNT(?s) as ?OCCURNUM)
   WHERE {
      s:u ?p ?x .
      ?x ?eq t:?o .
      ?s t:?pt t:?o .
      FILTER (?eq = owl:sameAs ||
              ?eq = skos:exactMatch)
   }
   GROUP BY ?s ORDER BY DESC ?OCCURNUM
   LIMIT 1
}
```

Only a restricted set of templates is selected for applying the Answer-based rewriting rules. In particular, triple patterns where only one term remains non adequate for the target dataset. The other two terms of the triple pattern are an adequate term for the target dataset and a variable or else two variables. Namely, the selected templates are: (?x t:p s:u), (s:u t:p ?x), (?x s:p t:o), (t:o s:p ?x), (?x ?p s:o), (s:u ?p ?x). The adequate term may be there from the beginning (i.e. some terms can be adequate for both source and target dataset) or else be the result of a previously applied rewriting rule.

As has been previously noted, when the number of terms involved in the replacement of the term u is more than one (let us say $(t\colon b_1, \ldots, t\colon b_k)$), every single measure is replaced by the corresponding average $(S_x(u, t\colon b_1) + \ldots + S_x(u, t\colon b_k))/k$. Then, a linear combination of the same aforementioned similarity measures is associated to the replaced term u:

$$\phi(u) = \alpha_n \cdot \frac{\sum_{i=1}^{k} S_n(u, t\colon b_1)}{k} +$$

$$\alpha_d \cdot \frac{\sum_{i=1}^{k} S_d(u,t\colon b_1)}{k} +$$
$$\alpha_o \cdot \frac{\sum_{i=1}^{k} S_o(u,t\colon b_1)}{k}$$

Values for parameter α_n, α_d, and α_o could be determined by an expert taking into account the desired weighting of the three facets. But it could be preferable to obtain those parameter values as the output of a specifically designed optimization algorithm. The explanation of the configuration of the optimization algorithm will be presented in section 5, within the experimental scenario.

The last kind of rules we are considering in the present embodiment is **Feature-based** rules. This kind of rules is the last option if non adequate terms remain in the query graph pattern after the aforementioned kind of rules have already been considered. Notice that running a query with a non adequate term for the considered dataset would yield the empty answer. In this case, the heuristic behind is to replace the non adequate term by a new variable (therefore, generalizing the query) but restricting that variable with features of the replaced term (that is to say, triples in the source dataset which the replaced term is the subject). The expression for such a rule is the following:

```
PREFIX  s:<source dataset>
PREFIX  t:<target dataset>
REPLACE #s #p s:u
BY         #s #p ?v .
           AND(?v #f #o)
WHEN {
        s:u #f #o   }
```

A summary of the rules considered for this framework embodiment is presented in tables 11, 12, 13, 14, 15, in the appendix A .

The algorithm $\mathcal{A}$ devised for applying the rewriting rules consists in applying first those rules that seem to maintain as much as possible the semantics of the current query. The rewriting algorithm applies the rules on a kind by kind basis. Within a kind of rules the algorithm repeats the application of each rule until no more application is possible. The rules of a kind are sequentially numbered and they are applied in that numbered sequence. In particular, the rules are numbered in the same sequence order that they appear in tables of appendix A. A rule is applied as long as its preconditions (described by columns *REPLACE* and *WHEN*) are satisfied. When a rule is no longer applicable, the algorithm drives to the following rule. Something specific takes place when application of a feature-based rule has finished. If any non adequate IRI remains in the query, Equivalence and Hierarchy rules are tried again and after that, any triple pattern presenting a non adequate IRI is deleted from the query. At any moment the current query being object of rewriting becomes adequate for the target dataset, the algorithm stops and return such target query. Next, a more explicit description of the algorithm is presented, where Q_s and Q_t represent the source and target query, respectively, and $\mathcal{C} = D_s \cup D_t \cup D_b$ represent the data graph context, composed of the source, target and bridge datasets, where the rewriting process takes place. $voc(Q_t)$ and $voc(D_t)$ respectively mean the vocabulary (i.e. set of terms) of Q_t and D_t.

```
REWRITE($Q_s, \mathcal{C}$)
// $\mathcal{C} = D_s \cup D_t \cup D_b$
$Q_t \leftarrow Q_s$
if $voc(Q_t) \nsubseteq voc(D_t)$ then
  $Q_t \leftarrow$APPLY(EquivalenceRules, $Q_t, \mathcal{C}$)
if $voc(Q_t) \nsubseteq voc(D_t)$ then
  $Q_t \leftarrow$APPLY(HierarchyRules, $Q_t, \mathcal{C}$)
if $voc(Q_t) \nsubseteq voc(D_t)$ then
  $Q_t \leftarrow$APPLY(AnswerBasedRules, $Q_t, \mathcal{C}$)
if $voc(Q_t) \nsubseteq voc(D_t)$ then
  $Q_t \leftarrow$APPLY(ProfileBasedRules, $Q_t, \mathcal{C}$)
if $voc(Q_t) \nsubseteq voc(D_t)$ then
  $Q_t \leftarrow$APPLY(FeatureBasedRules, $Q_t, \mathcal{C}$)
if $voc(Q_t) \nsubseteq voc(D_t)$ then
  $Q_t \leftarrow$APPLY(EquivalenceRules, $Q_t, \mathcal{C}$)
if $voc(Q_t) \nsubseteq voc(D_t)$ then
  $Q_t \leftarrow$APPLY(HierarchyRules, $Q_t, \mathcal{C}$)
$Q_t \leftarrow$ deleteNonAdequateTriplePatterns($Q_t, voc(D_t)$)
return $Q_t$

APPLY($RuleSet$, $Q$, $\mathcal{C}$)
for each $r \in RuleSet$
  while $applicable(r, Q, \mathcal{C})$
    $Q \leftarrow rewriteWith(r, Q, \mathcal{C})$
return $Q$
```

The similarity factor $\mathcal{SF}$ associated to a target query is an aggregation of the similarity values associated to each rule applied to reach such a target query. Among different possibilities, a measure based on the Euclidean distance on a n-dimensional space was selected. Given a sequence of rule applications $(r_i)_{i=1}^{N}$ for the rewriting of a source query into a target query, involving the corresponding non adequate terms $(u_i)_{i=1}^{N}$, the values $(\phi(u_i))_{i=1}^{N}$ can be considered the coordinates of a point in a N-dimensional space, where the point $(1,\ldots,1)$ represents the best and the point $(0,\ldots,0)$ the worst. Then, the Euclidean distance between the points $(\phi(u_i))_{i=1}^{N}$ and $(1,\ldots,1)$ provides a foundation for a similarity measure. In order to normalize the similarity value within the real interval $[0,1]$, with the value 1 representing the best similarity, the Euclidean distance between $(\phi(u_i))_{i=1}^{N}$ and $(1,\ldots,1)$ is divided by $\sqrt{N}$, and substracted from the best similarity 1. Let us use $(u_i)_{i=1}^{N}$ in representation of the sequence of rules $(r_i)_{i=1}^{N}$, then

$$\mathcal{SF}((u_i)_{i=1}^{N}) = 1 - \frac{1}{\sqrt{N}}\sqrt{\sum_{i=1}^{N}(1-\phi(u_i))^2}$$

Finally, the score selected to inform about the quality of the obtained target query was the F1 score calculated by comparing the answers retrieved by the target query with those retrieved by the corresponding Gold standard query. We call *Relevant answers* (Rel) to the set of answers obtained by running the Gold standard query, and *Retrieved answers* (Ret) to the set of answers obtained by running the target query. Then, the values Precision (P), Recall (R), and F1 score (F1) are calculated with the following formulae.

$$\mathrm{P} = \frac{|\mathrm{Rel} \cap \mathrm{Ret}|}{|\mathrm{Ret}|} \qquad \mathrm{R} = \frac{|\mathrm{Rel} \cap \mathrm{Ret}|}{|\mathrm{Rel}|} \qquad \mathrm{F1} = 2 \times \frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}}$$

The supervised learning model $\mathcal{P}$ devised to predict the F1 score of a target query was generated from the application of a Random Forest algorithm. Some other regression algorithms were considered and after an experimentation process, discussed in section 5, the Random Forest was selected because it offered the best results.

5. Framework validation

This section presents the main results of the process carried out to validate the proposed framework. The following resources are presented: (a) the LOD datasets selected for querying data, (b) the collection of training queries, (c) the optimization algorithm used to determine the proper parameter values for computing similarity measures and a discussion of its results, (d) the collection of features gathered from data to construct the learning datasets used by the machine learning algorithms and the results obtained by them, (e) the processing times needed by the framework implementation to get the answers in the corresponding SPARQL endpoints.

5.1. Datasets and queries

To validate the framework we trusted on well known datasets of the Linked Open Data environment, with accessible endpoints that facilitate the assessment of our experiments. Three domain areas were considered for the datasets: media-domain, bibliographic, and life science. For each one, a set of recognized datasets were selected. With respect to media-domain, the selected ones were: DBpedia, MusicBrainz, LinkedMDB, Jamendo, New York Times and BBC. With respect to bibliographic domain, we considered BNE (Biblioteca Nacional de España), BNF (Bibliothèque National du France), BNB (British National Bibliography), LIBRIS, and Cambridge. And finally for the life science area: Drugbank, SIDER, CHEBI, DISEASOME, and KEGG were the selected ones. Moreover, to achieve greater plurality in the tests, we used the SP2Bench, which is based on a synthetic dataset. In addition, our framework implementation also considered DBpedia, VIAF, Freebase, and GeoNames as bridge datasets. The available SPARQL endpoint for each mentioned dataset was used to answer the queries. In summary, we considered 17 different datasets (16 real + 1 synthetic) along with 4 bridge datasets that assisted us on the framework implementation. This is a evidence of the broad coverage of the experiments performed.

The set of experimental queries were selected after analyzing heterogeneous benchmarks such as QALD[5], FedBench [26], and real SPARQL endpoint logs (like BNE or DBpedia). A set of 100 queries was created for experimenting with the framework and providing data for the learning process. Those queries along with their corresponding gold standards and the names of source and target datasets are listed in the appendix published in [27, 28]. The idea underlying the selection process was to select queries that could be representative of the different SPARQL query types and that could cover heterogeneous domains in the Linked Open Data framework. Concerning provenance we selected 25 queries from well known cited benchmarks that were defined for some of the datasets listed in the previous paragraphs, and that presented a variety of

[5] https://qald.sebastianwalter.org/

graph pattern structures. Furthermore, 25 more queries were selected from the LOD SPARQL endpoints logs (year 2014 period). To select them, a clustering of the queries was carried out according to their graph pattern structure, and a random sample of each group was chosen, previously eliminating those queries that were malformed or those that exceed a maximum of 15 triple patterns, since they are usually triple pattern repetitions that do not provide structural diversity to the query set. In this way we got an initial set of 50 queries, which we doubled by converting their gold standards into source queries that were the origin of a new rewriting. In total we obtained a set of 100 queries expressed in terms of vocabularies of 17 different datasets, accessible via SPARQL endpoints.

Regarding the syntactic structure of the queries, a variety of the SPARQL operators (UNION, OPTIONAL, FILTER) and different patterns for joins of variables appeared in the queries. The number of triple patterns of each query ranges from 1 to 7.

A very important source of knowledge that supports this framework are repositories containing mappings between terms from different vocabularies and interlinkings between different IRIs for the same resource. For instance, we can find files with mapping triples in the DBpedia project[6] or data dumps of Freebase/Wikidata mappings. Some datasets, for instance Jamendo[7], are accompanied by sets of mappings that interlink their resources with resources in another domain sharing dataset, like Geonames and Musicbrainz. Moreover, some datasets can act as a central point of interlinking between some different datasets. It is the case of VIAF[8] on the bibliographic domain. A very useful web service, helping with this problem is http://sameas.org/, a service that helps to solve the existence of co-references between different data sets. But unfortunately there are no many well organized repositories and services of this kind, For this reason, to improve our approach we have created our own mapping repository in a local instance using Virtuoso Open Source triple store. So we have crawled the web finding possible mapping files and have incorporated them into the Virtuoso repository.

5.2. Suitability of the similarity factor

The similarity factor $\mathcal{SF}$ in the framework is intended to inform the user about a similarity estimation of a target query with respect to a source query. Our approach takes into account the query content and the query results as dimensions for query similarity, and its computation is based on a kind of graph-edit distance associated to each rewriting rule application. Assuming that the ideal for the target query would be to behave as similarly as possible to the corresponding gold standard query, along the query results dimension, it is natural to design $\mathcal{SF}$ in such a way that the similarity factor associated to a target query be correlated with the F1 score of that target query. Remember that such F1 score is computed for the target query with respect to the predefined gold standard query. Therefore, tuning of the similarity measures used to compute $\mathcal{SF}$ is desirable.

The similarity measure, presented in section 4, is based on similarity functions ϕ (associated to each rule application $\mathcal{V}(r)$) and many of them are defined as a linear combination of three similarity measures (S_n, S_d, S_o), involving three parameters α_n, α_d, and α_o. Instead of trying to determine their appropriate values by chance it seems preferable to devise a method to optimize their values towards the goal of moving $\mathcal{SF}$ closer to F1 score. This is the tuning process to which we refer in the previous paragraph.

We selected a method based on a genetic algorithm, specifically the Harmony Search (HS) algorithm [29]. Harmonies represent sets of variables to optimize, whereas the quality of the harmony is given by the fitness function of the optimization problem at hand.

In our case the variables to optimize are the parameters $(\alpha_n, \alpha_d, \alpha_o)$ appearing in the definition of the similarity measure. And the established fitness function was the maximization of the proportion of m queries whose absolute difference between the value of the similarity factor $\mathcal{SF}(q_i)$ and the F1 score for query number $i = 1 \ldots m$ was smaller than a given threshold β. The fitness function is as follows:

[6] http://wiki.dbpedia.org/services-resources/interlinking

[7] http://dbtune.org/jamendo/

[8] http://viaf.org/

$$\begin{aligned}&\text{maximize} \quad \sum_{i=1}^{m} \frac{1}{m} H(q_i))\\&\text{subject to} \;\; 0 \leqslant \alpha_n, \alpha_d, \alpha_o, \beta \leqslant 1\\&\text{where}\\&H(q_i) = \begin{cases} 1 & \text{if } |F_1(q_i) - \mathcal{SF}(q_i, \alpha_n, \alpha_d, \alpha_o)| < \beta \\ 0 & \text{otherwise} \end{cases}\end{aligned}$$

In order to carry out the optimization process the query set (see 5.1) was considered, and five-fold cross validation were performed. The sample data (initial query set) are divided into five subsets. One of the subsets is used as test data and the other four as training data. The cross-validation process is repeated five times (folds), with each of the five subsamples used exactly once as the test data. For each iteration (fold), the first step was to execute the HS algorithm on the set of training queries (composed by four subsets), in order to obtain the parameter values that achieve optimal fitness. For this, the algorithm was parametrized with the number of iterations and initial values for the parameters. The HS optimization process may obtain different solutions depending on the initial random values chosen for the parameters and the number of iterations allowed. With 100 iterations and an initialization defined by the HS algorithm itself, we obtained the parameter values shown in table 3, according to different values of β $(0.4, 0.2, 0.1)$. Each value shown in the cells of the table represents the different results obtained for training subsamples in each iteration (fold).

One example of convergence of the HS optimization process for the training dataset and $\beta = 0.2$ is shown in figure 1, where abscissas axis represents the number of algorithm iterations and the ordinate axis represents the fitness value. It can be observed how the fitness increases with the number of iterations.

To assess the validity of the parameter values obtained by the algorithm, the similarity factor was computed for each fold over the set of remaining test queries (in this case using the alphas obtained in the different scenarios with $\beta = 0.4, 0.2, 0.1$, respectively) and then, the absolute difference between these similarity factors and the F1 scores for the corresponding target queries were calculated. Training and Test Fitness in table 4 display the fitness mean values for the five different folds calculated over the training dataset and test dataset respectively, along with the corresponding Mean Absolute Error (MAE). The MAE is the difference between the training dataset fitness value and the one obtained with the test dataset; it measures the suitability of the optimization. It can be observed that the MAE never exceeds 0.15, which indicates that the optimization process is valid (values less than 0.3 are considered valid).

Taking into account that a tighter threshold as $\beta = 0.1$ produces worse test fitness values and higher absolute errors, and a looser threshold as $\beta = 0.4$ is too relaxed for similarity considerations, we decided to implement the computation of $\mathcal{SF}$ with the values $\alpha_n = 0.130$, $\alpha_d = 0.515$, and $\alpha_o = 0.352$, corresponding to the threshold $\beta = 0.2$, which offers a reasonable balance between test fitness values and closeness of $\mathcal{SF}$ and F1.

5.3. Discussion

In the following we discuss the validity of the similarity factor $\mathcal{SF}$ obtained by our embodied framework using the optimized parameter values calculated by the HS aforementioned method over the set of 100 experimental queries. In order to trust in the quality of the information conveyed to the user by the similarity factor, it is relevant to compare such factor with a measure of the behaviour of the target query. It is evident that the F1 score is a measure of that kind and therefore we proceeded with such a comparison.

Figure 2 shows a scatter plot of the 100 points with coordinates (F1 score, $\mathcal{SF}$) and table 5 shows numbers for the same points. An analysis of the results revealed the following considerations: In 59 out of 100 source queries the target queries provided the same set of results as the corresponding gold standard queries. From this set, in 50 of them their F1 score was 1, and in 9 of them (Q11, Q17, Q25, Q26, Q39, Q41, Q52, Q76, and Q91) the F1 score could not be calculated because the sets of relevant and retrieved results (see section 4) were both empty (notice that eventual dataset updates could change those results). The cases in which the F1 score equals 1 (50% of the whole set) can be divided into two groups depending on the similarity factor: (1) Cases whose similarity factor equals F1 score, represent a 23% of the whole query set. (2) Cases whose similarity factor is less than F1 score, represent a 27% of the whole query set. In the other case, there were 41 queries in which the target queries did not provide the same set of results as the corresponding gold standard queries. Therefore these queries had a F1 score lower than 1. From this set, in 14 of them the similarity factor

	0.1					0.2					0.4				
	1-fold	**2-fold**	**3-fold**	**4-fold**	**5-fold**	**1-fold**	**2-fold**	**3-fold**	**4-fold**	**5-fold**	**1-fold**	**2-fold**	**3-fold**	**4-fold**	**5-fold**
α_n	0.119	0.120	0.118	0.112	0.114	0.130	0.130	0.129	0.129	0.130	0.134	0.130	0.131	0.137	0.134
α_d	0.501	0.504	0.500	0.501	0.504	0.517	0.514	0.513	0.519	0.515	0.537	0.529	0.534	0.535	0.538
α_o	0.379	0.369	0.379	0.379	0.375	0.351	0.351	0.352	0.354	0.353	0.328	0.327	0.329	0.319	0.321

Table 3

Optimal parameter values for similarity function calculated from training subsamples for each fold.

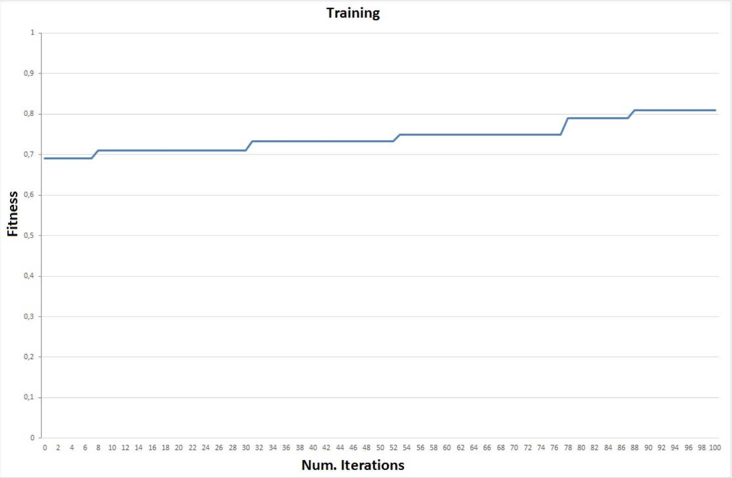

Fig. 1. Convergence of fitness with the training dataset and $\beta = 0.2$.

	Training Fitness	*Test Fitness*	*Mean Absolute Error (MAE)*
$\boldsymbol{\beta = 0.4}$	0.832	0.761	0.071
$\boldsymbol{\beta = 0.2}$	0.809	0.725	0.084
$\boldsymbol{\beta = 0.1}$	0.678	0.532	0.146

Table 4

Fitness values for different thresholds.

was lower than the F1 score. We want to highlight that in cases where $\mathcal{SF}$ < F1 score, the rewriting system is performing better than the offered similarity factor, since the higher F1 score shows that the target query performs more similarly to the gold standard than the offered information.

Finally, in 25 of them the similarity factor was higher than the F1 score. In those cases the similarity factor was too optimistic because the actual results provided by the target query were quite different from those provided by the gold standard query and there were cases where the F1-measure value was very low. This circumstance support the idea that would be interesting to complement the information to the user with a *quality score* reflecting the F1-score. As long as gold standard queries are not present in a real scenario, devising a prediction model for such a score is an option. Next, section 5.4 will explain an implementation of such a predictive model. There were also 2 queries where the retrieved results were empty (Q28, Q40).

Concluding this comparison, it can be said that $\mathcal{SF}$ is a cautious information to the user since in the majority of the cases $\mathcal{SF} \leqslant$ F1, and therefore frequently indicates a lower bound quality of the behaviour of the target query with respect to expected answers. It is interesting to remark that while $\mathcal{SF}$ reflects an intensional measure (semantic similarity of the replacement), the F1 score has an extensional character.

As discussed above, three domain areas were considered for the datasets: media-domain, bibliographic, and life science. Table 7 presents some statistics about the queries distribution and their $\mathcal{SF}$ and F1 values. In particular, the number of queries per domain, their similarity factor and F1 score averages ($\overline{\mathcal{SF}}$, $\overline{F1}$) and their corresponding standard deviations (σ).

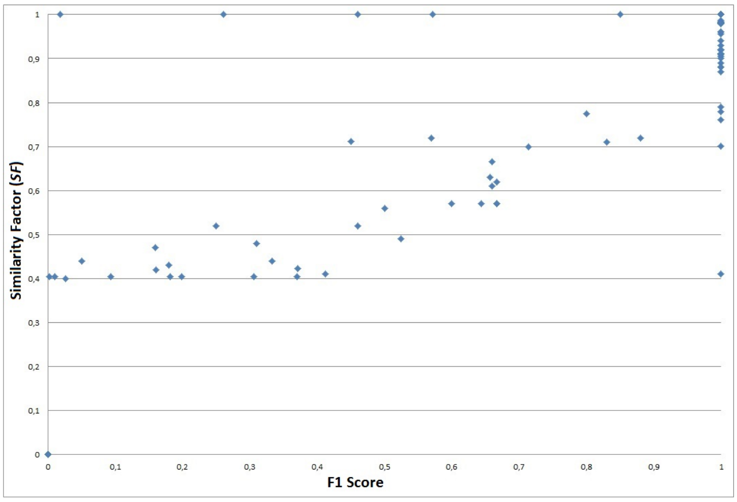

Fig. 2. Scatterplot for F1 score and Similarity factor (using similarity parameter values calculated for training dataset and $\beta = 0.2$).

	Q1	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**
F1	1	0.002	1	1	0.83	1	0.46	1	0.5	1
$\mathcal{SF}$	0.956	0.405	0.987	0.979	0.71	0.905	0.52	0.984	0.56	0.78
	Q11	**Q12**	**Q13**	**Q14**	**Q15**	**Q16**	**Q17**	**Q18**	**Q19**	**Q20**
F1	-	1	1	1	1	1	-	1	1	1
$\mathcal{SF}$	-	0.912	1	1	1	1	-	0.901	1	1
	Q21	**Q22**	**Q23**	**Q24**	**Q25**	**Q26**	**Q27**	**Q28**	**Q29**	**Q30**
F1	1	0.66	0.16	0.05	-	-	0.01	-	0.8	1
$\mathcal{SF}$	1	0.61	0.47	0.44	-	-	0.405	-	0.775	0.701
	Q31	**Q32**	**Q33**	**Q34**	**Q35**	**Q36**	**Q37**	**Q38**	**Q39**	**Q40**
F1	1	1	1	0.57	0.88	1	0.18	1	-	-
$\mathcal{SF}$	1	0.79	0.87	0.72	0.72	0.93	0.43	1	-	-
	Q41	**Q42**	**Q43**	**Q44**	**Q45**	**Q46**	**Q47**	**Q48**	**Q49**	**Q50**
F1	-	1	1	0.31	0.37	0.25	1	1	1	0.45
$\mathcal{SF}$	-	0.88	0.41	0.48	0,405	0.52	1	1	0.761	0.711
	Q51	**Q52**	**Q53**	**Q54**	**Q55**	**Q56**	**Q57**	**Q58**	**Q59**	**Q60**
F1	1	-	0.666	1	0.018	0.198	0.666	1	1	1
$\mathcal{SF}$	0.96	-	0.57	0.94	1	0.405	0.665	0.919	0.982	0,89
	Q61	**Q62**	**Q63**	**Q64**	**Q65**	**Q66**	**Q67**	**Q68**	**Q69**	**Q70**
F1	1	0.371	0.306	0.656	0.666	0.714	1	1	1	1
$\mathcal{SF}$	1	0.422	0.405	0.63	0.62	0.7	1	0.88	1	1
	Q71	**Q72**	**Q73**	**Q74**	**Q75**	**Q76**	**Q77**	**Q78**	**Q79**	**Q80**
F1	1	0.666	0.571	0.26	1	-	0.85	1	0.46	1
$\mathcal{SF}$	1	0.57	1	0.99	1	-	1	1	1	1
	Q81	**Q82**	**Q83**	**Q84**	**Q85**	**Q86**	**Q87**	**Q88**	**Q89**	**Q90**
F1	1	1	1	0.026	0.644	0.412	0.181	1	1	0.6
$\mathcal{SF}$	1	0.98	0.91	0.4	0.57	0.41	0.405	1	0.96	0.57
	Q91	**Q92**	**Q93**	**Q94**	**Q95**	**Q96**	**Q97**	**Q98**	**Q99**	**Q100**
F1	-	1	1	0.524	0.093	0.333	1	1	1	0.16
$\mathcal{SF}$	-	0.92	1	0.49	0.405	0.44	1	0.98	0.91	0.42

Table 5

$\mathcal{SF}$ (using similarity parameter values calculated for training dataset and $\beta = 0.2$) and F1 score for the experimental query set.

<table>
<tr><td colspan="6">Query set composed of
100 queries</td></tr>
<tr><td colspan="3">59 queries with
Retrieved answers = Relevant answers</td><td colspan="3">41 queries with
Ret$\neq$ Rel</td></tr>
<tr><td colspan="2">50 queries with
F1 = 1</td><td rowspan="2">9 queries
with |Ret|=|Rel|=0
F1 cannot
be calculated</td><td rowspan="2">25 queries with
$\mathcal{SF}$ &gt; F1</td><td rowspan="2">14 queries with
$\mathcal{SF}$ &lt; F1</td><td rowspan="2">2 queries with
|Ret|=0</td></tr>
<tr><td>23 queries with
$\mathcal{SF} =$ F1</td><td>27 queries with
$\mathcal{SF}$ &lt; F1</td></tr>
</table>

Table 6

Summary of the comparison between $\mathcal{SF}$ value and F1 score.

In that table 7 we can observe that $\overline{\mathcal{SF}}$ and $\overline{F1}$ values do not vary significantly depending on the domain. The greatest distances are, in the case of $\overline{\mathcal{SF}}$, between Media and Life Science domains (0.073) and, in the case of $\overline{F1}$, between Media and Bibliographic domain (0.07), never exceeding a difference greater than 0.08. Moreover, Life Science is the domain in which the values are more dispersed with relation to the average, which means that the quality of the rewriting, even within the same domain, has varied significantly. The best results are obtained in the Media domain, due to a greater number of links among source, target and bridge datasets belonging to that domain. Nevertheless, we are aware that the limited number of queries considered in the testbed may have an impact in the compared behaviour of the respective domains. However, the obtained results are quite promising and therefore they could be considered as a baseline.

Finally, we found interesting to know what was the correlation between the computed similarity factor and the F1 score. The Pearson correlation coefficient (usually named Pearson's *r*) is a popular measure of the strength and direction of the linear relationship between two variables. Pearson's correlation coefficient for continuous data ranges from -1 to $+1$. A value equals to 0 indicates no linear relationship between the variables. Positive correlation indicates that both variables increase or decrease together, whereas negative correlation indicates that as one variable increases, so the other decreases, and vice versa. In our case, the value was $r = 0.724$, which can be considered a high positive correlation, indicating that variables (F1 score, $\mathcal{SF}$) increase or decrease together providing a coherent metric for informing the user about the outcome of the rewriting process.

We think that the results in table 5 allow us to say that the similarity factor defined in section 4 is quite informative about the quality of the target query from an intensional point of view. Nevertheless, one goal of the presented framework is to serve as a tool for establishing benchmarks which promote improvement of query rewriting systems, and the embodiment presented in this paper could be considered a baseline.

5.4. Predictive model for the F1 score

As we have already mentioned, gold standard queries are not available in a real scenario, that is the reason why a predictive model $\mathcal{P}$ was considered in the framework. In the scenario of this paper, $\mathcal{P}$ is in charge of predicting the F1 score. Such predicted value is the *quality score*, referred in the example shown in section 1, that adds information to the user.

The construction of that predictive model was based on learning datasets that contain features of data that represent underlying structure and characteristics of the data subject of the prediction. In our scenario, features related to the structure of the source query such as number of triple patterns or number of operators, along with features related to rules that take part during the rewriting process, and finally features concerning the involved LOD datasets, were considered to build the feature datasets.

Following we present the 21 considered features:

1. Similarity and rules features, numbered from 1 to 11: (1) Number of times the equivalence rules are applied, (2) Similarity measure value associated to the equivalence rules application, (3) Number of times the hierarchy rules are applied, (4) Similarity measure value associated to the hierarchy rules application, (5) Number of times the answer-based rules are applied, (6) Similarity measure value associated to the answer-based rules application, (7) Number of times the profile-based rules are applied, (8) Similarity measure value associated to the profile-based rules application, (9) Number of times the

	Nº Queries	$\overline{\mathcal{SF}}$ **-** σ	$\overline{F1}$ **-** σ
Media domain	34	0.707 - 0.37	0.729 - 0.31
Bibliographic domain	40	0.649 - 0.4	0.659 - 0.33
Life Science domain	26	0.634 - 0.47	0.72 - 0.38

Table 7

$\mathcal{SF}$ and F1 averages with standard deviation

		F.1	F.2	F.3	F.4	F.5	F.6	F.7	F.8
1	Equivalence rule application times	X	X						
2	Equivalence rule similarity measure value	X	X	X					
3	Hierarchy rule application times	X	X						
4	Hierarchy rule similarity measure value	X	X	X					
5	Answer rule application times	X	X						
6	Answer rule similarity measure value	X	X	X					
7	Profile rule application times	X	X						
8	Profile rule similarity measure value	X	X	X					
9	Feature rule application times	X	X						
10	Feature rule similarity measure value	X	X	X					
11	Similarity factor	X	X	X	X	X	X	X	X
12	Number of source triple patterns	X	X						
13	Number of terms in source query	X	X	X	X	X	X	X	
14	Number of non-adequate terms	X	X	X	X	X	X		X
15	Number of union operators	X							
16	Number of projected variables	X							
17	Number of optional operators	X							
18	Number of filter operators	X							
19	Source Dataset	X	X	X	X				
20	Target Dataset	X	X	X	X				
21	Number of mappings between source and target	X	X			X			

Table 8

Selected features for the 8 different datasets.

Datasets	**LR**	**SVM**	**RF**
Features1	0.6751	-1.5072	**0.8219**
Features2	0.5902	-1.0313	0.7132
Features3	0.6045	-0.0437	0.7152
Features4	0.4572	-0.0689	0.7026
Features5	0.7146	**0.7731**	0.6835
Features6	**0.7529**	0.6815	0.7797
Features7	0.7511	0.7611	0.7221
Features8	0.7523	0.7146	0.7923

Table 9

R2 metric of the predictive models.

Query	**TT**	**AT**	**TAT**	**Query**	**TT**	**AT**	**TAT**
Q1	8926	214	9140	Q51	7699	307	8006
Q2	8525	652	9177	Q52	7836	672	8508
Q3	2991	108	3099	Q53	2643	121	2764
Q4	2005	364	2369	Q54	2893	408	3301
Q5	3500	405	3905	Q55	2560	396	2956
Q6	2688	210	2898	Q56	2971	221	3192
Q7	4261	573	4834	Q57	4562	475	5037
Q8	1142	847	1989	Q58	797	623	1420
Q9	981	1021	2002	Q59	3400	637	4037
Q10	1802	938	2740	Q60	3111	1002	4113
Q11	0	734	734	Q61	656	429	1085
Q12	2825	208	3033	Q62	2676	198	2874
Q13	167	290	457	Q63	581	335	916
Q14	268	312	580	Q64	671	281	952
Q15	137	205	342	Q65	592	112	704
Q16	330	277	607	Q66	407	392	799
Q17	1144	1482	2626	Q67	2273	752	3025
Q18	11264	523	11787	Q68	9380	503	9883
Q19	1953	109	2062	Q69	1774	122	1896
Q20	536	108	644	Q70	921	173	1094
Q21	1994	574	2568	Q71	2327	696	3023
Q22	5191	613	5804	Q72	6317	384	6701
Q23	2753	409	3162	Q73	3612	493	4105
Q24	3503	102	3605	Q74	3037	251	3288
Q25	5658	689	6347	Q75	7780	782	8562
Q26	3258	314	3572	Q76	3649	373	4022
Q27	1796	125	1921	Q77	2030	109	2139
Q28	2225	852	3077	Q78	2653	974	3627
Q29	1153	901	2054	Q79	1264	670	1934
Q30	1765	782	2547	Q80	1983	593	2576
Q31	1181	213	1394	Q81	1727	201	1928
Q32	5899	1297	7196	Q82	9052	904	9956
Q33	6749	708	7457	Q83	7264	775	8039
Q34	5096	514	5610	Q84	6613	562	7175
Q35	4658	160	4818	Q85	5883	118	6001
Q36	4005	898	4903	Q86	5216	905	6121
Q37	3013	1650	4663	Q87	3840	1284	5124
Q38	571	294	865	Q88	685	394	1079
Q39	4512	1834	6346	Q89	7523	928	8451
Q40	2899	775	3674	Q90	3137	551	3688
Q41	1103	1028	2131	Q91	1022	1005	2027
Q42	4020	383	4403	Q92	4705	462	5167
Q43	1029	469	1498	Q93	1280	392	1672
Q44	3598	493	4091	Q94	4659	347	5006
Q45	0	197	197	Q95	389	144	533
Q46	1779	1328	3107	Q96	1711	1206	2917
Q47	0	122	122	Q97	775	215	990
Q48	461	203	664	Q98	621	193	814
Q49	3613	2751	6364	Q99	5652	1431	7083
Q50	653	272	925	Q100	739	285	1024

Table 10

Processing times in *ms*

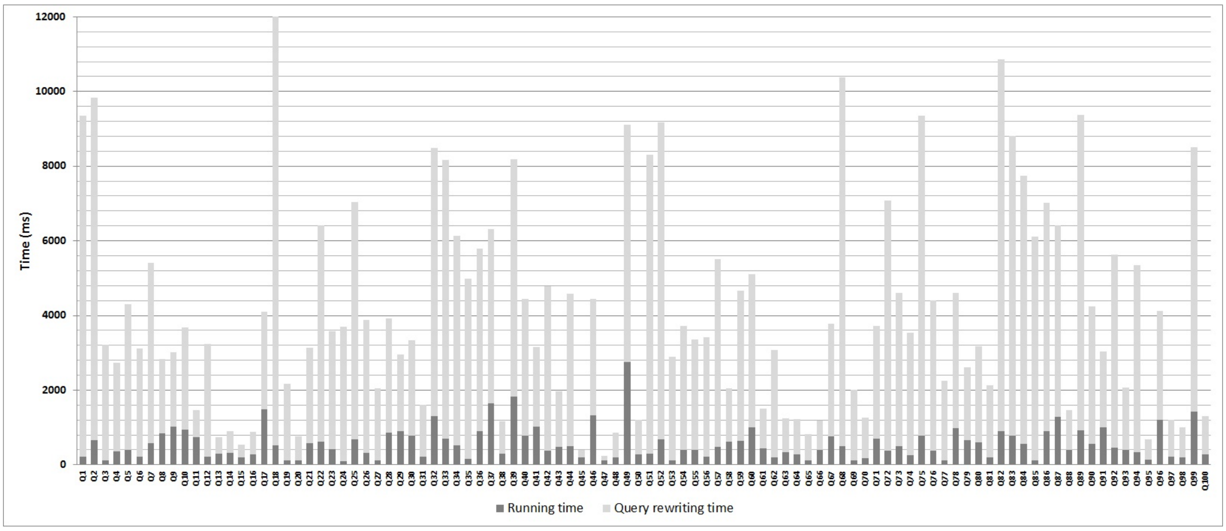

Fig. 3. Answering times plus rewriting times.

feature-based rules are applied, (10) Similarity measure value associated to the feature-based rules application, and (11) the similarity factor calculated for the target query.

2. Query structure features, numbered from 12 to 18: (12) Number of triple patterns of the source query, (13) number of terms of the source query, (14) number of non adequate terms for the target dataset, (15) number of union operators, (16) number of projected variables, (17) number of optional operators, and (18) number of filter operators.
3. LOD Datasets features, numbered from 19 to 21: (19) categorical data associated to the source dataset depending on its size. Three values are possible: 1, for small datasets with less than 10^5 triples; 2, for medium size datasets with a number of triples between 10^5 and 10^6; 3 for larger datasets with more than 10^6 triples, (20) categorical data associated to the target dataset depending on its size (the same possible values as in the case of feature number 20), and (21) number of mappings between source and target datasets.

In order to select a best fit model, we experimented with the following off-the-shelf algorithms [30, 31]: Linear regression (LR), Support Vector Machines (SVM), and Random Forest; and with 8 different datasets (F.1 to F.8) corresponding to distinct feature selection (see table 8).

The values used in the experiment were obtained from the rewriting of the 100 aforementioned queries. This set of queries were divided in three fragments: 80% for the training process, 15% for the validation process, and 5% for the test process, respectively. The score of each of those models was measured based on a 20-fold cross-validated average mean squared error-R2 metric. The results are presented in the table 9, where each cell of the table represents the coefficient of determination for each model trained with the feature dataset indicated by the row.

As can be seen, the model that best fit is that obtained using the Random Forest-RF algorithm with F.1 dataset, with a R2 equals to 0.8219 for validation set. Moreover, notice that the features datasets F.6, F.7, and F.8 are the ones that, in general, show a better behaviour with all the models. Therefore, the similarity factor (11), number of terms (13), and number of non-adequate terms (14) features can be considered the most significant ones. And it points out again the validity of the computed similarity factor. To asses the generalization error of the final chosen model, the value of R2 over the test set was computed, yielding a value of 0.8014.

5.5. Processing time

The framework is placed into the Linked Open Data environment, leveraging datasets SPARQL endpoints. In order to asses the performance of the framework we want to show runtimes and the evaluation conditions in which it was implemented.

The rewriting process (rules and algorithm) has been implemented with Attributed Graph Grammar System

(AGG)[32]. For similarity computation, we relied on the libraries Wordnet Similarity for Java (WS4J)[9] and SimMetrics[10]. The queries run by means of Jena semantic framework[11]. For performance testing, the system consisted of an Intel Core 2 Duo 2.67 GHz processor, 8 GB RAM, Windows 7 Professional, and Java Runtime Environment 1.8. All measurements were executed six times consecutively using the average of the last five measurements.

Table 10 displays the processing times for each query in benchmark executed over the corresponding dataset. The TT column indicates the time needed by the framework to obtain the rewritten query. In column AT the time to get the answers of the query over the SPARQL endpoint is indicated, and finally the column TAT is the sum of the previous times (TT+AT). The same information is graphically showed in Figure 3. The TT times, that really represents the performance of our system, are between a minimum value of $137ms$ (Q15) and a maximum value of $11264ms$ (Q18), regardless of the values of $0ms$ (Q45, Q47). And 90% of queries are executed in a maximum time of $6sec$. Taking into account this significant performance information about the framework implementation, we consider that the processing times are acceptable but amenable to improvement.

6. Conclusions

The current state of the Web of Data with so many different datasets of a heterogeneous nature makes it difficult for users to query those datasets in order to exploit the vast amount of data they contain. Different proposals are appearing to overcome that limitation. In this paper we have detailed the features of a framework that allows end users to obtain results from different datasets expressing the query using only the vocabulary which the users are more familiar with, and informs them about the quality of the answer. Moreover, this framework serves technical users as a tool for establishing query rewriting benchmarks.

The framework has been embodied with a selected set of rules, rule scheduling algorithm, similarity measures, and quality estimation model composed of similarity factor function and F1 score predictive model. Moreover, the framework has been validated in a real scenario and the results obtained are promising, and they could be considered a baseline to be improved considering smarter rewriting rules and better shaped similarity measures.

Acknowledgements

This work is supported by FEDER/TIN2013-46238-C4-1-R and FEDER/TIN2016-78011-C4-2-R projects, and also by the ELKARTEK program (BID3ABI project).

[9] https://code.google.com/p/ws4j/

[10] http://sourceforge.net/projects/simmetrics/

[11] https://jena.apache.org/

Appendix A. Summary of rewriting rules

Rule	REPLACE	WHEN	BY	$\mathcal{V}(r)$
E1	ELHS	$EDOAL : ELHS \rightarrow t : ERHS$	$t : ERHS$	$\phi(s : u) = 1$
E2	$(s : u, \#p, \#o)$	$(s : u, eq, t : u_i)$ $(i = 1, \ldots, k)$	$\text{UNION}_{i=1,\ldots,k}$ $(t : u_i, \#p, \#o)$	
E3	$(\#s, s : u, \#o)$	$(s : u, eq, t : u_i)$ $(i = 1, \ldots, k)$	$\text{UNION}_{i=1,\ldots,k}$ $(\#s, t : u_i, \#o)$	
E4	$(\#s, \#p, s : u)$	$(s : u, eq, t : u_i)$ $(i = 1, \ldots, k)$	$\text{UNION}_{i=1,\ldots,k}$ $(\#s, \#p, t : u_i)$	
E5	$(s : u, \#p, \#o)$	$(s : u, eq, b : u_i)(b : u_i, eq, t : u_i)$ $(i = 1, \ldots, k)$	$\text{UNION}_{i=1,\ldots,k}$ $(t : u_i, \#p, \#o)$	
E6	$(\#s, s : u, \#o)$	$(s : u, eq, b : u_i)(b : u_i, eq, t : u_i)$ $(i = 1, \ldots, k)$	$\text{UNION}_{i=1,\ldots,k}$ $(\#s, t : u_i, \#o)$	
E7	$(\#s, \#p, s : u)$	$(s : u, eq, b : u_i)(b : u_i, eq, t : u_i)$ $(i = 1, \ldots, k)$	$\text{UNION}_{i=1,\ldots,k}$ $(\#s, \#p, t : u_i)$	

Table 11

Summary of Equivalence rewriting rules.

Rule	REPLACE	WHEN	BY	$\mathcal{V}(r)$
H8	$(s : u, \#p, \#o)$	$(s : u, sub, v)$ $(v, sub, t : u_i)$ $(i = 1, \ldots, k)$	$\text{AND}_{i=1,\ldots,k}$ $(t : u_i, \#p, \#o)$	$\phi(s : u) = \frac{\sum_{i=1}^{k} S_o(s:u,t:u_i)}{k}$
H9	$(\#s, s : u, \#o)$	$(s : u, sub, v)$ $(v, sub, t : u_i)$ $(i = 1, \ldots, k)$	$\text{AND}_{i=1,\ldots,k}$ $(\#s, t : u_i, \#o)$	
H10	$(\#s, \#p, s : u)$	$(s : u, sub, v)$ $(v, sub, t : u_i)$ $(i = 1, \ldots, k)$	$\text{AND}_{i=1,\ldots,k}$ $(\#s, \#p, t : u_i)$	
H11	$(s : u, \#p, \#o)$	$(t : u_i, sub, v)$ $(v, sub, s : u)$ $(i = 1, \ldots, k)$	$\text{UNION}_{i=1,\ldots,k}$ $(t : u_i, \#p, \#o)$	
H12	$(\#s, s : u, \#o)$	$(t : u_i, sub, v)$ $(v, sub, s : u)$ $(i = 1, \ldots, k)$	$\text{UNION}_{i=1,\ldots,k}$ $(\#s, t : u_i, \#o)$	
H13	$(\#s, \#p, s : u)$	$(t : u_i, sub, v)$ $(v, sub, s : u)$ $(i = 1, \ldots, k)$	$\text{UNION}_{i=1,\ldots,k}$ $(\#s, \#p, t : u_i)$	

Table 12

Summary of Hierarchy rewriting rules.

Rule	REPLACE	WHEN	BY	$\mathcal{V}(r)$
A14	$(?x, t:p, s:u)$	Answers($?x, t:p, s:u$)= $(x_1,\ldots,x_n)$ $(x_k, eq, t:x_k)$ $(k=1,\ldots,n)$ $(t:x_k, t:p, t:o_{kj})$ $(j=1,\ldots,m_k)$	UNION$_{k=1,\ldots,n}$(AND$_{j=1,\ldots,m_k}$ $(?x, t:p, t:o_{kj})$)	$\phi(s:u) =$ $\alpha_n \cdot \frac{\sum_{i=1}^{k} S_n(s:u,t:o_i)}{k} +$ $\alpha_d \cdot \frac{\sum_{i=1}^{k} S_d(s:u,t:o_i)}{k} +$ $\alpha_o \cdot \frac{\sum_{i=1}^{k} S_o(s:u,t:o_i)}{k}$
A15	$(s:u, t:p, ?x)$	Answers($s:u, t:p, ?x$)= $(x_1,\ldots,x_n)$ $(x_k, eq, t:x_k)$ $(k=1,\ldots,n)$ $(t:x_k, t:p, t:o_{kj})$ $(j=1,\ldots,m_k)$	UNION$_{k=1,\ldots,n}$(AND$_{j=1,\ldots,m_k}$ $(t:o_{kj}, t:p, ?x)$)	
A16	$(?x, s:u, t:o)$	Anwers($?x, s:u, t:o$)= $(x_1,\ldots,x_n\ )$ $(x_i, eq, t:x_i)$ $(i=1,\ldots,n)$ $\forall j \in \{1\ldots k\}$ $(t:x_i, t:u_j, t:o)$	UNION$_{j=1,\ldots,k}$ $(?x, t:u_j, t:o)$	
A17	$(t:s, s:u, ?x)$	Anwers($t:s, s:u, ?x$)= $(x_1,\ldots,x_n\ )$ $(x_i, eq, t:x_i)$ $(i=1,\ldots,n)$ $\forall j \in \{1\ldots k\}$ $(t:x_i, t:u_j, t:o)$	UNION$_{j=1,\ldots,k}$ $(?x, t:u_j, t:o)$	
A18	$(?x, ?p, s:u)$	Answers($?x, ?p, s:u$)= $(x_1,\ldots,x_n)$ $(x_k, eq, t:x_k)$ $(k=1,\ldots,n)$ $(t:x_k, t:p_{ki}, t:o_{ki})$ $(i=1,\ldots,m)$ $t:o_z =$ mostFrequent($t:o_{ki}$: $k=1,\ldots,n.i=1,\ldots,m$)	$(?x, ?p, t:o_z)$	$\phi(s:u) =$ $\alpha_n \cdot S_n(s:u, t:o_z)$ $+\alpha_d \cdot S_d(s:u, t:o_z)$ $+\alpha_o \cdot S_o(s:u, t:o_z)$
A19	$(s:u, ?p, ?x)$	Answers($s:u, ?p, ?x$)= $(x_1,\ldots,x_n)$ $(x_k, eq, t:x_k)$ $(k=1,\ldots,n)$ $(t:x_k, t:p_{ki}, t:o_{ki})$ $(i=1,\ldots,m)$ $t:o_z =$ mostFrequent($t:o_{ki}$: $k=1,\ldots,n.i=1,\ldots,m$)	$(t:o_z, ?p, ?x)$	

Table 13

Summary of Answer-based rewriting rules.

Rule	REPLACE	WHEN	BY	$\mathcal{V}(r)$
P20	$(s:u, \#p, \#o)$	$(s:u, s:p_i, a_i)$ $(a_i, eq, t:a_i)$ $(t:a_i, t:p_i, t:o_i)$ $(i = 1, \dots, m)$ $(b_j, s:q_j, s:u)$ $(b_j, eq, t:b_j)$ $(t:b_j, t:q_j, t:o_j)$ $(j = m+1, \dots, n)$ $t:o_z$= maxSim($s:u, h, t:o_1, \dots, t:o_n$)	$(t:o_z, \#p, \#o)$	$\phi(s:u) =$ $S\,(s:u, \mathit{maxSim}(s:u, h, \mathit{profile}(s:u)))$
P21	$(\#s, \#p, s:u)$	$(s:u, s:p_i, a_i)$ $(a_i, eq, t:a_i)$ $(t:a_i, t:p_i, t:o_i)$ $(i = 1, \dots, m)$ $(b_j, s:q_j, s:u)$ $(b_j, eq, t:b_j)$ $(t:b_j, t:q_j, t:o_j)$ $(j = m+1, \dots, n)$ $t:o_z$= maxSim($s:u, h, t:o_1, \dots, t:o_n$)	$(\#s, \#p, t:o_z)$	
P22	$(\#s, s:u, \#o)$	$(s:u, s:p_i, a_i)$ $(a_i, eq, t:a_i)$ $(t:a_i, t:p_i, t:o_i)$ $(i = 1, \dots, m)$ $(b_j, s:q_j, s:u)$ $(b_j, eq, t:b_j)$ $(t:b_j, t:q_j, t:o_j)$ $(j = m+1, \dots, n)$ $t:o_z$= maxSim($s:u, h, t:o_1, \dots, t:o_n$)	$(\#s, t:o_z, \#o)$	

Table 14

Summary of Profile rewriting rules.

Rule	REPLACE	WHEN	BY	$\mathcal{V}(r)$
F23	$(s:u, \#p, \#o)$	$(s:u, s:p_k, s:o_k)$ $(k = 1, \dots, n)$	$(?v, \#p, \#o)$ AND$_{k=1,\dots,n}$ $(?v, s:p_k, s:o_k)$ $?v$ a new variable	$\phi(s:u) = 0$
F24	$(\#s, s:u, \#o)$	$(s:u, s:p_k, s:o_k)$ $(k = 1, \dots, n)$	$(\#s, ?v, \#o)$ AND$_{k=1,\dots,n}$ $(?v, s:p_k, s:o_k)$ $?v$ a new variable	
F25	$(\#s, \#p, s:u)$	$(s:u, s:p_k, s:o_k)$ $(k = 1, \dots, n)$	$(\#s, \#p, ?v)$ AND$_{k=1,\dots,n}$ $(?v, s:p_k, s:o_k)$ $?v$ a new variable	

Table 15

Summary of Feature-based rewriting rules.

References

[1] D.J. Odgers and M. Dumontier, Mining electronic health records using linked data, *AMIA Summits on Translational Science Proceedings* **2015** (2015), 217.

[2] P. Haase, T. Mathäß and M. 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, ACM, New York, NY, USA, 2010, pp. 5–159. ISBN 978-1-4503-0014-8. doi:10.1145/1839707.1839713. http://doi.acm.org/10.1145/1839707.1839713.

[3] O. Hartig, C. Bizer and J.-C. Freytag, Executing SPARQL queries over the web of linked data, in: *International Semantic Web Conference*, Springer, 2009, pp. 293–309.

[4] R. Dividino and G. Gröner, Which of the Following SPARQL Queries Are Similar? Why?, in: *Proceedings of the First International Conference on Linked Data for Information Extraction - Volume 1057*, LD4IE'13, CEUR-WS.org, Aachen, Germany, Germany, 2013, pp. 2–13. http://dl.acm.org/citation.cfm?id=2874472.2874474.

[5] J. Euzenat, F. Scharffe and A. Zimmermann, D2. 2.10: Expressive alignment language and implementation, *Knowledge Web project report, KWEB/2004/D2.2.10/1.0.* (2007).

[6] M. Arias, J.D. Fernández, M.A. Martínez-Prieto and P. de la Fuente, An empirical study of real-world SPARQL queries, *Proc. 1st International Workshop on Usage Analysis and the Web of Data, USEWOD* (2011).

[7] F. Scharffe, A. Ferrara and A. Nikolov, Data linking for the semantic web, *International Journal on Semantic Web and Information Systems* **7**(3) (2011), 46–76.

[8] J. Euzenat and P. Shvaiko, *Ontology matching*, 2nd edn, Springer-Verlag, Heidelberg (DE), 2013.

[9] M. Cheatham and P. Hitzler, The properties of property alignment, in: *Proceedings of the 9th International Conference on Ontology Matching-Volume 1317*, CEUR-WS. org, 2014, pp. 13–24.

[10] J. Li, J. Tang, Y. Li and Q. Luo, RiMOM: A dynamic multistrategy ontology alignment framework, *IEEE Transactions on Knowledge and Data Engineering* **21**(8) (2009), 1218–1232.

[11] M. Kejriwal and D.P. Miranker, An unsupervised instance matcher for schema-free {RDF} data, *Web Semantics: Science, Services and Agents on the World Wide Web* **35, Part 2** (2015), 102–123, Machine Learning and Data Mining for the Semantic Web (MLDMSW), ISSN 1570-8268. doi:http://dx.doi.org/10.1016/j.websem.2015.07.002. http://www.sciencedirect.com/science/article/pii/S1570826815000621.

[12] G. Correndo, M. Salvadores, I. Millard, H. Glaser and N. Shadbolt, SPARQL Query Rewriting for Implementing Data Integration over Linked Data, in: *Proceedings of the 2010 EDBT/ICDT Workshops*, EDBT '10, ACM, New York, NY, USA, 2010, pp. 4–1411. ISBN 978-1-60558-990-9. doi:10.1145/1754239.1754244. http://doi.acm.org/10.1145/1754239.1754244.

[13] K. Makris, N. Bikakis, N. Gioldasis and S. Christodoulakis, SPARQL-RW: transparent query access over mapped RDF data sources, in: *Proceedings of the 15th International Conference on Extending Database Technology*, ACM, 2012, pp. 610–613.

[14] C. Hurtado, A. Poulovassilis and P. Wood, Query Relaxation in RDF, *Journal on Data Semantics X* (2008), 31–61. doi:10.1007/978-3-540-77688-8_2. http://dx.doi.org/10.1007/978-3-540-77688-8_2.

[15] H. Huang, C. Liu and X. Zhou, Computing Relaxed Answers on RDF Databases, in: *Web Information Systems Engineering - WISE 2008*, Lecture Notes in Computer Science, Vol. 5175, Springer Berlin Heidelberg, 2008, pp. 163–175. ISBN 978-3-540-85480-7.

[16] H. Huang, C. Liu and X. Zhou, Approximating query answering on RDF databases, *World Wide Web* **15**(1) (2012), 89–114, ISSN 1386-145X. doi:10.1007/s11280-011-0131-7. http://dx.doi.org/10.1007/s11280-011-0131-7.

[17] S. Elbassuoni, M. Ramanath and G. Weikum, Query Relaxation for Entity-relationship Search, in: *Proceedings of the 8th Extended Semantic Web Conference on The Semanic Web: Research and Applications*, Springer Berlin Heidelberg, 2011, pp. 62–76. ISBN 978-3-642-21063-1. http://dl.acm.org/citation.cfm?id=2017936.2017942.

[18] R. De Virgilio, A. Maccioni and R. Torlone, A similarity measure for approximate querying over RDF data, in: *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, ACM, 2013, pp. 205–213.

[19] A. Hogan, M. Mellotte, G. Powell and D. Stampouli, Towards Fuzzy Query-Relaxation for RDF., in: *ESWC*, Lecture Notes in Computer Science, Vol. 7295, Springer, 2012, pp. 687–702. ISBN 978-3-642-30283-1. http://dblp.uni-trier.de/db/conf/esws/eswc2012.html#HoganMPS12.

[20] A. Poulovassilis, P. Selmer and P.T. Wood, Approximation and relaxation of semantic web path queries, *Web Semantics: Science, Services and Agents on the World Wide Web* **40** (2016), 1–21.

[21] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini and R. Rosati, Linking data to ontologies, in: *Journal on data semantics X*, Springer, 2008, pp. 133–173.

[22] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro and G. Xiao, Ontop: Answering SPARQL queries over relational databases, *Semantic Web* **8**(3) (2017), 471–487. doi:10.3233/SW-160217. https://doi.org/10.3233/SW-160217.

[23] W.W.W. Consortium et al., R2RML: RDB to RDF mapping language (2012).

[24] A.I. Torre-Bastida, J. Bermúdez and A. Illarramendi, Query approximation in the case of incompletely aligned datasets, *Actas de las XX Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2015)* (2015).

[25] J. Zhong, H. Zhu, J. Li and Y. Yu, Conceptual graph matching for semantic search, in: *International Conference on Conceptual Structures*, Springer, 2002, pp. 92–106.

[26] M. Schmidt, O. Görlitz, P. Haase, G. Ladwig, A. Schwarte and T. Tran, Fedbench: A benchmark suite for federated semantic data query processing, in: *The Semantic Web–ISWC 2011*, Springer, 2011, pp. 585–600.

[27] A.I. Torre Bastida, SPARQL Query Set (txt format) (2018). doi:10.6084/m9.figshare.5896627.v1. https://figshare.com/articles/100SPARQLgold_adequate_source_target_txt/5896627/1.

[28] A.I. Torre Bastida, SPARQL Query Set (pdf format), figshare, 2018. doi:10.6084/m9.figshare.5900236.v1. https://figshare.com/articles/SPARQL_Query_Set/5900236/1.

[29] Z.W. Geem, J.H. Kim and G. Loganathan, A new heuristic optimization algorithm: harmony search, *Simulation* **76**(2) (2001), 60–68.

[30] M. Mohri, A. Rostamizadeh and A. Talwalkar, *Foundations of machine learning*, MIT press, 2012.

[31] S. Wold, K. Esbensen and P. Geladi, Principal component analysis, *Chemometrics and intelligent laboratory systems* **2**(1–3) (1987), 37–52.

[32] G. Taentzer, AGG: A graph transformation environment for modeling and validation of software, in: *Applications of Graph Transformations with Industrial Relevance*, Springer, 2004, pp. 446–453.

[33] C. Bizer and A. Schultz, The R2R framework: Publishing and discovering mappings on the web, in: *Proceedings of the First International Conference on Consuming Linked Data-Volume 665*, CEUR-WS. org, 2010, pp. 97–108.

[34] C.A. Hurtado, A. Poulovassilis and P.T. Wood, Query relaxation in RDF, in: *Journal on data semantics X*, Springer, 2008, pp. 31–61.

[35] H. Huang, C. Liu and X. Zhou, Approximating query answering on RDF databases, *World Wide Web* **15**(1) (2012), 89–114.

[36] S. Scheider, A. Degbelo, R. Lemmens, C. van Elzakker, P. Zimmerhof, N. Kostic, J. Jones and G. Banhatti, Exploratory querying of SPARQL endpoints in space and time, *Semantic web* **8**(1) (2017), 65–86.

[37] F.M. Suchanek, S. Abiteboul and P. Senellart, Paris: Probabilistic alignment of relations, instances, and schema, *Proceedings of the VLDB Endowment* **5**(3) (2011), 157–168.

[38] W.E. Zhang, Q.Z. Sheng, K. Taylor, Y. Qin and L. Yao, Learning-Based SPARQL Query Performance Prediction, in: *International Conference on Web Information Systems Engineering*, Springer, 2016, pp. 313–327.

[39] R. Hasan and F. Gandon, A machine learning approach to SPARQL query performance prediction, in: *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, Vol. 1, IEEE, 2014, pp. 266–273.

[40] L. Breiman, Random forests, *Machine learning* **45**(1) (2001), 5–32.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12**(Oct) (2011), 2825–2830.