Which of the following SPARQL Queries are Similar? Why?

Renata Dividino and Gerd Gröner

WeST, University of Koblenz-Landau, Germany

Abstract. Linked data on the Web can be accessed by SPARQL queries. Previously executed queries to SPARQL endpoints are valuable information sources about the underlying data structure and schema of data sources. These queries reveal how resources are related to each other and they reflect the user interests on the data. Therefore, methods for query logs analysis provides a basis for extracting relevant and qualitative data from the LOD cloud. These query logs analysis rely on similarity measures for SPARQL queries in order to determine whether a query is related to another query. However, there is no commonly agreed notion of similarity for SPARQL queries. Instead, there are several measures that not only vary computational complexity and output values, but also on their purpose. In this paper, we present a comparison and a discussion of the existing similarity measures for SPARQL queries in the literature. Our results are guidelines designed to provide guidance on the application of similarity measures for comparing SPARQL queries.

1 Introduction

Information extraction (IE) methods that rely on linked data information have to deal with various interconnected domains, highly heterogeneous vocabularies, and missing information about the used data schema. Thus, IE applications face difficulties when exploring linked data since they may not have the necessary knowledge about the underlying data source, its content and the used vocabulary, which make it difficult to select and optimize training seeds.

Apart from direct requests, linked data sources can be accessed by SPARQL requests, either directly by users or indirectly by applications. When querying the LOD cloud, agents need to be familiar with the data, their links and their vocabularies. We argue that previously executed queries to SPARQL endpoints are valuable information sources about the underlying data structure and schema of data sources. These queries reveal how resources are related to other resources and they reflect the user interests on resources and their relationships. Therefore, methods for query logs analysis provide a basis for the extraction of relevant and qualitative data (from the user's point of view) from the LOD cloud to be used, for instance, in learning methods or even to improve crawling.

Several techniques are based on the analysis of SPARQL requests to the LOD cloud such as methods for query completion [1], approximation [2,3] and relaxation [4,5], which aims to help users in formulating SPARQL queries. Furthermore, techniques for analyzing and mining of query logs of SPARQL endpoint

```
PREFIX foaf: <http://xmlns.com/foaf/0.1>
SELECT ?person ?email
FROM <http://dig.csail.mit.edu/timbl/foaf.rdf>
WHERE{
{?persopn foaf:name "John Doen".>
UNION
<?persion foaf:name "Peter Doen".>
?person foaf:mbox ?email
}LIMIT 50
```

(a) It returns the email of the persons named John Doen and Peter Doen

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?email
FROM <http://dig.csail.mit.edu/timbl/foaf.rdf>
WHERE {
{?person foaf:name "John Doen".}
UNION
{?person foaf:name "Sarah Carey"}
?person foaf:mbox ?email.
}LIMIT 50
```

(b) It returns the email of the persons named John Doen and Sarah Carey

Fig. 1. Similar SPARQL Queries

also provide a valuable knowledge to enhance a range of semantic applications, e.g., to investigate user behavior [6], or even to extract frequent query pattern to be used as benchmarking [7] or as templates for data pre-fetching [8]. All these techniques rely on similarity measures for SPARQL queries in order to determine whether two SPARQL query are similar and to which degree. However, it is not always obvious to determine the similarity degree of SPARQL queries. Queries can be similar and dissimilar with respect to different dimensions. For example, queries can be similar encoded such as the query presented in Fig. 1(a) and the query presented in Fig. 1(b), but lead to very different results. Further, queries can be written in different ways using various alternative constructs, such as the query presented in Fig. 1(a) and the query presented in Fig. 2 but still leading to the same or to a similar result.

In the literature, there are several measures for SPARQL query similarity. For instance, the Levenshtein edit distance is often used [9,8,4]. Either the entire query is considered as a string or just parts of the query (e.g. the triple patterns). These measures are simple and have low computation complexity, but they do not explore the properties of SPARQL query. Also popular are graph-based similarity. Basically, these metrics exploit the structure of a SPARQL query (represented as a graph) and it is basically defined on graph isomorphism [10]. The definition of query containment and equivalence [11] is often adopted to measure the degree of similarity between queries. Graph-based metrics are computationally expensive, and thus highly avoided in real-time applications.

Often it is, a priori, hard to say which measure is best to be used when comparing SPARQL queries. This directly leads to the question "What is a good similarity measure for SPARQL queries?". We show that this answer depends on the application context in which the similarity measure is used.

In order to assess the applicability of a similarity metric, a thorough investigation of SPARQL query foundations and the different kinds of similarity measures is needed. The analysis of query similarity measures and metrics has several facets like the investigation of similarity assessment dimensions, as well as the computation complexity of a measure. A thorough analysis requires a common formalization of query similarity and an abstraction of implementation-oriented aspects of existing similarity measures in order to allow for a proper comparison under a uniform umbrella.

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?person ?email
FROM <http://dig.csail.mit.edu/timbl/foaf.rdf>
WHERE {
    {?person foaf:mbox ?email.
    ?person foaf:name "John Doen"}
    UNION
    {?person foaf:mbox ?email.
    ?person foaf:name "Peter Doen".}
}LIMIT 50
```

Fig. 2. Equivalent to the query in Fig. 1(a)

In this paper, we present a foundational study of similarity measures for SPARQL queries. We compare different metrics regarding their assessment, the computational complexity, the intended purpose and usage of a certain measure and its practical applicability. An analysis shows which similarity measures are intuitive with the user's expectations and how measures are related to other ones. Our results serve as guidelines to provide support when applying similarity measures for SPARQL queries.

2 Foundations

This section describes foundations of SPARQL, introduced in [12]. We will use these notions later to describe a similarity measures for SPARQL queries.

We consider pairwise disjoint alphabets U, a set of URIs, L, a set of terminal literals, and B, a set of blank nodes. An RDF statement is a triple $(s, p, o) \in (U \cup B) \times U \times (U \cup L \cup B)$. An RDF graph is a finite set of triples. SPARQL is based on matching graph patterns against RDF graphs.

A SPARQL query is formally represented by a tuple defined as $Q = (A, V, G, P, M, R)$, where A is the set of prefix declarations (Line 1 in Fig 1(a)), V is the output form (Line 2 in Fig 1(a)). The output of a SPARQL query can be of different types: yes/no queries, selections of values of the variables, which match the patterns, construction of new RDF graphs from these values, and descriptions of resources. G is the RDF graph(s) being queried (Line 3 in Fig 1(a)), P is a graph pattern (Lines 5-8 in Fig 1(a)), which includes features of pattern matching of graphs, like optional parts, union of patterns, nesting, filtering values of possible matchings, and the possibility of choosing the data source to be matched by a pattern, M are query modifiers, which allow to modify the results by applying projection, order, limit, and offset options (Line 9 in Fig 1(a)) and R is the set of variable bindings given by partial substitutions (mappings) of the query variables by RDF terms $(B \cup L \cup U)$.

Two graph pattern expressions $P1$ and $P2$ are equivalent, denoted by $P1 \equiv P2$, if $[\![P1]\!]_G^D = [\![P2]\!]_G^D$ for every graph G and RDF dataset D.

3 State of the Art

Similarity measures serve for different purposes. Accordingly, the intention and the way they are computed vary significantly. This section provides an overview of existing similarity measures for SPARQL queries.

The first group of similarity measures are based on string metrics. Basically, the main goal is to find strings in a query that (partially) match to the strings in another query. The most popular string similarity is the Levenshtein distance. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertion, deletion, substitution) needed to transform one string into the other.

In [9], the authors use the Levenshtein distance in a performance benchmark of different triple stores. Frequently executed queries extract from the query log are clustered based on the Levenshtein distance. These clusters are subsequently used to devise the query generation patterns used in the benchmark. In a follow-up work [7], a combination of vector similarity and the Levenshtein distance is used. A query is described by an additional 17-dimensional vector of SPARQL features. This principle can be extended to SPARQL queries to incorporate the comparison of SPARQL operators such as the UNION, OPTIONAL and FILTER constructors, the DISTINCT solution modifier, operators like REG and STR.

Lorey at al. [8] group similar SPARQL queries in order to detect recurring patterns in queries. They introduce the notion of query templates, which represent clusters of similar queries exhibiting these recurrences. A query is not treated as one string, but as a set of strings, i.e., the set of individual elements (subject, predicate and object) of triple patterns. Similarity between each pair of query elements is defined based on the Levenshtein distance. Their aggregated score gives the similarity score of the triple. This notion is extended to the comparison of graph patterns.

Reddy et al. [2] make use of similarity of triples for the purpose of SPARQL query relaxation. Subject, predicate and object of triples can be replaced by super-classes and super-properties with respect to an RDFS ontology. A similar principle is applied in [5]. SPARQL queries are extended by a relax clause to describe triple patterns that can be relaxed regarding to an ontology. Instead of an ontology, a pre-specified mapping that assigns pairs (URI or literal pairs) to a value between 0 and 1 is applied in the so-called fuzzy relaxation [4]. The value to which URI or literal pairs are mapped represent the similarity value between these URIs or literals. Similarity of triples relies on matchers that exploit the element structure. Similarity between numerical values is based on normalized Euclidean distance, for string values on normalized Levenshtein edit distance, and for categorical values based on a similarity table computed by a background matching process.

Similarity measures based on graph matching are commonly used in query optimization/rewriting applications. Basically, queries are represented as a graph and the main goal is to find the isomorphic sub-graphs among the corresponding query graphs, i.e., to find the maximum common sub-graph of two graphs. In [10],

the authors propose an efficient algorithm to address these challenges. They exploit the fact that nodes and edges in query graphs have types.

Other approaches define similarity measures between two queries as a function that associates the similarity of their result sets to a similarity score in the range $[0; 1]$. Result sets are similar if the values bound to their query variables are similar. The RDF data query language iRDQL [13] and the SPARQL extension iSPARQL [3] offer constructs to write RDQL or SPARQL like queries with additional constructs like the IMPRECISE clause in iRDQL to explicitly state for which variables of a query an "imprecise" result binding. Both language extensions implement TF-IDF and Levenshtein distance, as well as Jensen-Shannon information divergence (in iRDQL) or Jaccard distance (in iSPARQL).

Query similarity is also defined over query equivalence and containment. Checking query containment is, however, undecidable. There are some approaches that improve the complexity of query containment computation by certain language restrictions [12,11].

In order to allow for a more efficient containment checking, a minimization technique for RDFS query patterns is presented in [14]. Efficiency is achieved by additional restrictions like a strict distinction between data and schema layer and that domain and range restrictions are always defined and unique.

4 SPARQL Similarity Measures

For the purpose of this work, we need a generic definition of a similarity function for queries that supports the comparison of the state-of-the-art approaches.

Basically, SPARQL query similarity is a function θ that maps two SPARQL queries to a value in the closed interval between 0 and 1. (The higher value indicates the higher degree of similarity). $\mathcal{Q}$ denotes the set of all queries (universe).

Definition 1 (Signature of a Query Similarity Function θ). *A SPARQL query similarity θ is a function $\theta : \mathcal{Q} \times \mathcal{Q} \mapsto [0, 1]$.*

Let s be a similarity measure. The similarity of queries is obtained by aggregating the similarity of their components C, depending on the particular measure. The components for the similarity is denoted by $\mathcal{C}$ ($\mathcal{C} \subseteq A \times V \times G \times P \times M \times R$) (see Sec. 2 for the definition of a SPARQL query). We refer to the set of possible components of queries by $\mathcal{C}^s$. The similarity of query components is a function γ that maps two SPARQL query components to a value in $[0, 1]$.

Definition 2 (Query Components and Component Similarity). *Let $q \in \mathcal{Q}$ be a SPARQL query and s a similarity measure.*

1. *Π^s is a projection that maps a query q to its components $c \in \mathcal{C}^s$, Π^s_i is the projection of the i^{th} component*
2. *$|\Pi^s(q)|$ denotes the number of components that constitute q wrt. measure s*
3. *A SPARQL query component similarity γ^s is a function $\gamma^s : \mathcal{C}^s \times \mathcal{C}^s \mapsto [0, 1]$*

Based on the signatures of θ and γ, we define the query similarity function.

Definition 3 (Query Similarity Function). *Let $q_1, q_2 \in \mathcal{Q}$ be SPARQL queries, s a similarity measure and C^s the domain of query components for measure s. Let $n = min(|\Pi^s(q_1)|, |\Pi^s(q_2)|)$ be the minimum number of components in q_1 and q_2 respectively. The coefficient α_i is a value in $[0, 1]$ that is used to weight the similarity values of the components / elements, such that $\sum_{i=1}^{n} \alpha_i = 1$. σ is a permutation.*

$$\theta^s(q_1, q_2) = \sum_{i=1}^{n} \alpha_i \cdot \gamma^s \cdot (\Pi^s_{\sigma^s(i)}(q_1), \Pi^s_{\sigma^s(j)}(q_2))$$

Table 1. Overview of the existing similarity measures for SPARQL queries

Approaches	$\Pi(q)$ maps to ...	Similarity γ	Param.
Morsey et al. [9]	the string of q $\Pi : \mathcal{Q} \mapsto String$	Levenshtein	$n = 1$,
Lorey et al. [8]	set of string triples $\Pi : \mathcal{Q} \mapsto$ $(String \times String \times String)^k$	Levenshtein	$n = 3k$
Hogan et al. [4]	the set of (1) literals (L), (2) numerical values (Num) (3) categorical attributes (U) $\Pi : \mathcal{Q} \mapsto$ $(String \times String \times (L\|Num\|U))^k$ (direct match for subj. and pred.)	(1) Levenshtein distance (for literals (L)) (2)Euclidian distance (for num. values (Num)) (3) attribute table (for cat. attr. (U))	$n = k$
Reddy et al. [2] and Hurtado et al. [5]	the set of terms/nodes (T) in an ontology $\Pi : \mathcal{Q} \mapsto (T \times T \times T)^k$ (subj., pred., obj. can be relaxed)	distance of terms (T) to super-classes/ super-properties in an ontology	$n = 3k$
Morsey et al. [7]	binary feature vector ($0 \widehat{=}$ unused, $1 \widehat{=}$ used in the query) $\Pi : \mathcal{Q} \mapsto \{0, 1\}^c$ c is number of features (dimension of vector (fix))	(1) squared Euclidean distance	$n = c$
Le et al. [10]	the set of vertices V and edges E of the $\Pi : \mathcal{Q} \mapsto (V, E)$ $V \subseteq (B \cup U \cup L)^{2k}$ (subjects and objects), $E \subseteq U^k$ (predicates)	Jaccard sim. based on the maximum common sub-graph	$n = \|V\| +$ $\|E\|$
Kiefer et al. [3]	set of variable bindings as strings $\Pi : \mathcal{Q} \mapsto B$ $(B \subseteq String^{3k})$	(1) Levenshtein or (2) Jaccard or (3) cosine sim. over TF-IDF (on strings)	$n = \|B\|$

Table 2. Similarity metrics vs. Dimension Assessment

Dimensions	Morsey[9]	Lorey[8]	Hogan[4]	Reddy[2]	Hurtado[5]	Morsey[7]	Le[10]	Kiefer[3]
Structure	✓	—	—	—	✓	—	✓	—
Language	—	—	—	—	—	✓	—	—
Content	—	✓	✓	✓	✓	✓	✓	—
Result	—	—	—	—	—	—	—	✓

The definition of similarity of the query components (γ^s) depends on the particular similarity measure s. $\Pi^s(\sigma^s(i))$ projects the $(\sigma^s(i))^{\text{th}}$ component, while the permutation σ^s depends on s. α_i is used to ensure that the results are normalized. Table 1, shows the variations of the function γ^s proposed by the existing approaches, as discusses in Section 3. k in Table 1 denotes the number of triple patterns in a query-

Def 3 imposes restrictions on the number of comparisons of components of both queries. For instance, if a measure s compares only triples, and assume query q_1 consists of three triples, and q_2 consists of four triples, then the query similarity θ^s aggregates the components similarity, which is the similarity of triples, of three pairs that have the highest similarity value. The selection of three triples out of four from the second query depends on the particular measure. This is covered by the permutation σ^s in the definition.

5 Similarity Dimensions

In Section 4, we compared the computation of a variety of similarity measures. When mapping or decomposing queries to their components (Π in Table 1), we can observe different ways how queries (or their components) are "used" for a certain similarity computation. While this mapping is obviously determined by the similarity measure, an interesting additional issue is the correlation between the way the query is constituted and the actual application of its constituents in the similarity computation.

We can observe different types of elements of a query that are considered in the similarity computation: (1) only the query *structure*, expressed by a string (as the sequence of symbols) or a graph structure, is considered; (2) only the query *content*, i.e. its triple patterns are considered, and they are expressed as strings, ontological terms or numerical values; (3) only the use of *language features*, such as query modifiers, are considered; (4) the query *result sets* are considered.

Based on these observations, we extract four dimensions of query similarity assessment: *structure*, *content*, *language* and *result set*. The most common criteria used in numerous methods is the correspondence of the query structure, followed by content, results and language. Nevertheless, queries may be assessed with respect to not only one, but also two or even all dimensions. A grouping of similarity measures into these dimensions is shown in Table 2. For each assessment dimension, we discuss application and complexity issues.

5.1 Structure-based Similarity

A structure-based metric considers the query elements A, V, P, $M \in \mathcal{C}^s$ where their representation is expressed by a sequence of symbols (string), or a set of vertices and edges (graphs). For every pair of queries, the metrics at this level, checks the similarity of their structure, i.e, if their sequence of symbols or graph representation agree.

More formally: (1) two queries q_1 and q_2 have similar structure if there is correspondences between symbols of $string(q_1)$ and those of $string(q_2)$ with the property that the ordering of the symbols is the same. (2) Two queries q_1 and q_2 have similar structure if there is correspondences on their graph representations, denoted as $graph(q_1)$ and $graph(q_2)$, i.e. between vertices of $graph(q_1)$ and those of $graph(q_2)$, and that two vertices of $graph(q_1)$ are adjacent if and only if their images in $graph(q_2)$ are adjacent. Correspondences may also includes other properties such as size. In particular, for graphs, it may include the properties depth (graph property related to the cardinality of paths in a graph) and breadth (graph property related to the cardinality levels in a graph).

In general, structure-based similarities are used in pattern matching techniques. The goal in these applications is to recognizes common structural patterns in queries, to cluster queries with a certain structure or to merge queries with common shapes. Existing approaches, which make use of structural measures, like Morsey et al. [9,7] use the string representation of queries to devise the generation of query patterns.

The computational complexity varies with the actual measure. String similarity measures has low computational complexity, graph comparison has higher complexity instead. Nevertheless, the most used string similarity, Levenshtein Distance, has complexity $O(m^2)$ (m is the string length), which is still too slow for large datasets.

5.2 Content-based Similarity

Content-based similarities assess the content (the agent information need) covered by the query, i.e. a content-based metric considers the element $P \in \mathcal{C}^s$. Thus, the term *content* refers to the set of triple patterns in a query.

Two queries q_1 and q_2 are equal if they share the same content. This means, if for each triple $t_1 \in q_1$, and $t1 = (s_1, p_1, o_1)$ where s_1, p_1, and o_1 are the subject, predicate, and object of t_1 respectively, and $t_2 \in q_2$, and $t2 = (s_2, p_2, o_2)$, there is a one-to-one correspondences between s_1 and s_2, p_1 and p_2, o_1 and o_2.

Similar to structure-based metrics, triple patterns can be expressed by sequence of symbols or graph representation, and thus all structure-based metrics can also be applied here. However, content-metrics are more flexible since they do not rely on the ordering of the triples. For every two queries that have two equal triple patterns t_1 and t_2, even if they are encoded in different ordering, for instance, in the first query t_1 follows t_2, and in the second t_2 follows t_1, they are still equal at this dimension.

Additionally, it is usual practice that each constituent of a triple pattern $(s, p, o) \in t$ is expressed using different representation (string representation for literals, numerical representation for numbers, ontology categorization for resources). Reddy et al. [2] and Hurtado at al. [5] express subject, predicate and object of triples regarding to ontology terms, i.e. replaced by super-classes and super-properties with respect to an RDFS ontology. Hogan at al. [4] make used of normalized Euclidean distance to compute similarity of two numerical values, for string values on normalized Levenshtein edit distance, and for categorical values based on a similarity table which is pre-computed. The similarity measure in [10] just checks for the correspondences of the predicates.

The complexity depends on the used structure-based metrics for the triple patterns. Besides this, if additional characteristics like the relaxation based on ontologies are used, the complexity also depends on the ontology size and structure, e.g., the hight of concept hierarchies.

5.3 Language-based Similarity

Language-based similarities assess only the language terms covered by the query language, i.e. a language-based metric considers only the element $V, M \in \mathcal{C}^s$. If two queries q_1 and q_2 are described by same set of language terms (i.e. SPARQL features) then they are equal.

Morsey et al. [7] use language-feature metrics to incorporate the comparison of SPARQL operators such as the UNION, OPTIONAL and FILTER constructors, the DISTINCT solution modifier, operators like REG and STR. An advantage of this assessment is that the complexity is linear in $n = c$ (c refers to the number of features). Additionally, this metrics is can be extended to measure the similarity of the corresponding present features, e.g., by using string similarity. In this case, the total complexity is the complexity of the feature vector comparison and the complexity of the feature comparison of present features (e.g., the complexity of string similarity).

5.4 Result-Based Similarity

The computation on the result-based level measures the similarity of SPARQL queries based on the relations between set of their result sets. In this level, the similarity is defined based on the set of data retrieve when evaluating the query against to one or all possible RDF graph(s). The assessment at the structure, content, and language level aim at measuring the relations between two queries based on their syntax. For instance, if two queries are encoded using the same symbols (and ordering), for either all the elements in A, V, P, and/or M, for $A, V, P, MinC$, then they are equal. The assessment at the result-based level aims at measuring the relations of the elements in $R \in C$. Obviously, they are highly related, for instance, if for all elements in A, V, P, and M, are equal, the all the elements in C are equal when evaluated over a given RDF graph G.

Query equivalence is defined for queries q_1 and q_2 that have a one-to-one correspondence between the result sets over all possible RDF graphs. Query

equivalence is, however, an undecidable problem. Therefore, existing approaches restrict to the one-to-one correspondence between the result sets of r_1 and those of r_2, when q_1 and q_1 are evaluated against a given RDF graph. The works of [13] and [3] relies on similarity at this dimension.

The complexity depends on the query size and on implementation aspects. In the worst case, the complexity is NP-complete.

6 Analysis

The definition of a good similarity measure relies on the answer of what elements of the SPARQL queries are intended to be essential and how this can be actually measured. Therefore, a similarity metric is determined to be good or bad from a user's or application's point of view. In the literature, there is no consensus on the notion of SPARQL queries correspondence, there is no unique mechanisms to measure the "goodness" of a similarity measure. Accordingly, an evaluation of the above described metrics would not be possible, since they vary on their assessments and their results could not be compared. In the following, we discuss two aspects as a guideline for using and further developing query similarity measures.

What a metric is good for? — The Application Matters. The application context heavily determines the choice for a similarity measure, i.e. the application content reveals how a measure should deal with the similarity computation.

Structure-based measures are important in application where the representation of the query itself indicates similarity. Most of such applications aim to identify frequent (sub-) structures (e.g., sequences or graphs) in order to classify or cluster queries. A typical application for such metrics is data mining. However, structural similarity is not always enough. Even if structural similarity includes the assessment of the query content, it is still less flexible (i.e. it relies on the ordering of the symbols) than the content-based metrics and thus it is often not appropriate for content analysis.

Content-based similarity is preferred when concrete triples, particular resources and relationships among resources are relevant to determine relatedness of queries. Two queries that share the same content, but are expressed in different sequence ordering or even using different query modifiers or variables, are obviously the same regarding their content. Typical applications that use content-based metrics are query completion, query suggestion and query recommendation. In these cases, a part of a graph pattern is given and either potential extensions of this partial query or a list of related queries that have some content overlap are suggested as possible query continuations.

Result-based similarity is essential if the query representation does not interest at all (which is for sure not the case for applications such as pattern mining). Typical examples are relevance feedback-based applications. For instance, the user picks a query and the result of the query is the only criteria that matters.

Only few applications are based in language-based similarity. Still these metrics are the only ones which includes the pecularities of language the queries are encoded. Usually, these similarity are used in combination with metrics assessing other dimensions, in order to align the computation to the specific encoded language, for instance, the assessement of graphs which are, in turn, encoded with a specific language.

Is there a complete measure which is good for everything? No. Unfortunately, there is no complete similarity metric so far for all different application purposes. A complete SPARQL query similarity metric should suit the analysis of not only the query structure, content and language (its syntax), but also its semantic (result set). It seems to be rather challenging to come up with a similarity measure that covers all these different dimensions, maybe with some weighting mechanisms to adjust the measure for particular needs. At the same time, such a similarity measure should also be computational tractable. As discussed earlier, result-based measures are NP-complete. Thus, coming up with a comprehensive measure seems to be a computational very hard problem.

In order to build such a comprehensive but also computational tractable similarity measure, the best choice is probably to use the most promising building blocks from existing similarity measures. A first direction is to consider a mixed dimension assessment, depending on the relevance for a specific application. For example, considering IE applications, a mix of content and language assessments is the most promising. The use graph patterns instead of only triple pattern should enhance the information need about the combination of SPARQL features. Graph patterns can be combined using different features like optional parts, union of patterns, nesting, filtering (or restricting) values of possible matchings, and the possibility of choosing the data source to be matched by a pattern. Additionally, aspects of feature-based solution modifiers are a further issue for a similarity measure. For instance, the use of distinct solution modifier, which counts as the presence of a feature in the feature-based similarity measure, can influence a lot the result set, and thus, the presence or absences of this solution modifier might be a good indicator for query similarity.

7 Conclusion

Several applications for information extraction from linked data are based on query log analysis. These techniques rely on similarity measures for SPARQL queries. While there is a bunch of different measures, inspired from other research areas, a thorough comparison of these measures and their intended usage is still missing in order to assess the usefulness of a linked data source for a particular IE application. This gap has been closed by the research efforts that has been presented in this paper.

We have shown a comprehensive comparison of existing SPARQL similarity measures. In the first step, we have introduced a similarity specification as a common denominator for all presented similarity measures. In the comparison,

each similarity measure is related to this specification to account for a better comparability. In the second step, we have looked for the basic "building blocks" of SPARQL queries that are assumed by the different similarity measures. We ended up with categories of query similarity assessment, namely *structure*, *content*, *language* and *result set*.

We plan to proceed this research into two directions. First, we will conduct a user evaluation to get feedback which measure is intuitive with respect to human feeling of similarity. Second, we will extend the graph similarity measure by incorporating all operators of SPARQL.

References

1. Campinas, S., Perry, T.E., Ceccarelli, D., Delbru, R., Tummarello., G.: Introducing RDF Graph Summary with application to Assisted SPARQL Formulation. In: DEXA. (2012)
2. Reddy, B.R.K., Kumar, P.S.: Efficient Approximate SPARQL Querying of Web of Linked Data. In: URSW. (2010) 37–48
3. Kiefer, C., Bernstein, A., Stocker, M.: The Fundamentals of iSPARQL: A Virtual Triple Approach for Similarity-Based Semantic Web Tasks. In: ISWC/ASWC. (2007) 295–309
4. Hogan, A., Mellotte, M., Powell, G., Stampouli, D.: Towards Fuzzy Query-Relaxation for RDF. In: ESWC. (2012) 687–702
5. Hurtado, C.A., Poulovassilis, A., Wood, P.T.: Query Relaxation in RDF. J. Data Semantics **10** (2008) 31–61
6. Raghuveer, A.: Characterizing machine agent behavior through sparql query mining. In: USEWOD, Lyon, France, Arxiv (April 2012)
7. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.C.: Usage-Centric Benchmarking of RDF Triple Stores. In: AAAI 2012. (2012)
8. Lorey, J., Naumann, F.: Detecting SPARQL Query Templates for Data Prefetching. In: ESWC. LNCS, Springer (2013)
9. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.C.: DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In: ISWC 2011. (2011)
10. Le, W., Kementsietsidis, A., Duan, S., Li, F.: Scalable Multi-query Optimization for SPARQL. In: ICDE '12, IEEE Computer Society (2012) 666–677
11. Letelier, A., Pérez, J., Pichler, R., Skritek, S.: Static Analysis and Optimization of Semantic Web Queries. In: PODS. (2012) 89–100
12. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. **34**(3) (September 2009) 16:1–16:45
13. Bernstein, A., Kiefer, C.: Imprecise RDQL: Towards Generic Retrieval in Ontologies Using Similarity Joins. In: ACM Symposium on Applied Computing (SAC). (2006) 1684–1689
14. Serfiotis, G., Koffina, I., Christophides, V., Tannen, V.: Containment and Minimization of RDF/S Query Patterns. In: ISWC. Volume 3729 of LNCS., Springer (2005) 607–623