# Context Aware Source Selection for Linked Data

Barbara Catania, Giovanna Guerrini, and Beyza Yaman

University of Genoa, Italy
{firstname.name@dibris.unige.it}

**Abstract.** The traditional Web is evolving into the Web of Data, which gathers huge collections of structured data over distributed, heterogeneous data sources. *Live queries* are needed to get current information out of this global data space. In live query processing, *source selection* allows the identification of the sources that most likely contain relevant content. Due to the semantic heterogeneity of the Web of Data, however, it is not always easy to assess relevancy. *Context* information might help in interpreting the user's information needs. In this paper, we discuss how context information can be exploited to improve source selection.

## 1 Introduction

In the last decade, significant efforts have been devoted to the development of the Web of Data, i.e., a Web in which not only documents, but also data, can be first class citizens. A huge amount of Linked Data has been published to construct a global data space based on open standards [7]. However, it is still difficult to benefit from data in the Web of Data in an effective way. Typical systems operating on Linked Data collect (crawl) and pre-process (index) large amounts of data, and evaluate queries against a centralized repository. Given that crawling and indexing are time-consuming operations, data in the centralized index may be out of date at query execution time. On the other hand, an ideal query answering system for *live querying* [6] should return current answers in a reasonable amount of time, even on corpora as large as the Web.

Since in the Linked Data context there is a large number of small size sources, the main problems in processing queries in live query systems become (*i*) finding the sources containing triples that can contribute to the overall query answer, (*ii*) efficiently fetching triples from these sources, possibly in a parallel way. The first process is referred to as *source selection* [6] and requires knowledge of the data source content, with the aim of returning a ranked list of sources, ordered by relevance. Lightweight data summaries for determining relevant sources during query evaluation have been proposed and are exploited for this task [11]. Such data summaries are approximate multidimensional index structures that store descriptions of the content of data sources.

The diversity of publishers, however, results in data fulfilling an information need being stored in many different sources and described in many different

SEBD 2018, June 24-27, 2018, Castellaneta Marina, Italy. 

ways. The semantic enrichment of data, indeed, does not solve the problem of identifying relevant information with respect to a given domain, rather causes a shift of heterogeneity issues from the syntactic to the semantic level. By exploiting a more abstract notion of *context*, we can better interpret the request and devise the most relevant sources for the user's information needs.

Context may capture different properties of data, users or user-data interactions, and may come from data themselves, the querying device (e.g., ambient information) as well as from the execution environment (e.g., history of previous queries) [1, 5]. Despite the different additional information it can capture and rely on, matching data not only with respect to the user request but also with respect to a reference context associated with such request can help in determining the portion of the entire information that is most relevant to the user. Contexts have been used, e.g., in offering personalized services and recommendations and improving data selection [2, 8, 10], however, surprisingly, as far as we know, no approach has been defined yet exploiting context information for improving the linked data source selection process.

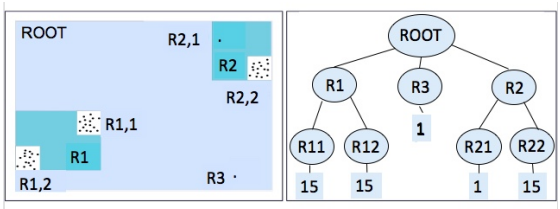
This paper goes in this direction and faces the problem of effective and efficient source selection for Linked Data, with the aim of proposing techniques that automatically discover sources providing most relevant answers, by exploiting a *domain-dependent context hierarchy* (as part of the more general framework envisioned in [3, 4]). In this paper, we consider RDF data sources and SPARQL queries and present: (*i*) two distinct extended data summaries enabling context-aware source selection (Section 2); (*ii*) an instantiation of the approach with a specific context taxonomy (Section 3); (*iii*) a preliminary experimental evaluation of the proposed structures on such instantiation (Section 4). Future work directions are discussed in Section 5.

## 2 Extended Data Summaries for Context-Aware Source Selection

In this section, we first briefly introduce the data summary our approach relies on, i.e., QTree [6], and the two extensions we propose to enable context-aware source selection. Both approaches rely on: (*i*) a context taxonomy $(\mathcal{C}, \preceq)$, where $\mathcal{C}$ is the set of contexts and $\preceq$ is a subconcept relationship among them; (*ii*) a distance function $d : \mathcal{C} \times \mathcal{C} \rightarrow [0, 1]$ measuring the distance between contexts in the taxonomy; (*iii*) a function $\gamma$ for associating contexts with triples and queries. $(\mathcal{C}, \preceq)$ and $\gamma$ can come with data or be (semi)automatically extracted from them while $d$ is data independent and is provided as an input parameter for the proposed techniques. Additional details on the proposed approaches can be found in [12].

### 2.1 Data Summaries: QTree

Data summaries [6, 11] have been proposed as an approach for efficiently determining which sources may contribute answers to a query in live distributed query




(a) Data points and regions (b) Region hierarchy and buckets

Fig. 1: 2-dimensional QTree [6]

systems. They describe the data provided by the sources in a summarized form. We specifically refer to the QTree indexing structure [6] which is a combination of R-trees and multidimensional histograms. The first step is to transform the RDF triples provided by the source into numerical points in a 3-dimensional space. Hash functions are applied to the individual components of RDF triples (i.e., *subject*, *predicate*, *object*) to obtain 3 numerical values for each triple. The employed hash functions, relying on a prefix-hashing approach, are such that triples with similar content are assigned close points in the multidimensional space, so that they are clustered close to each other. This 3-dimensional space is then indexed by a hierarchical data structure (the QTree, shown in Fig.1.b) in which nodes represent multidimensional regions, through minimal bounding boxes (MBBs). The region of a node always covers all the regions of its children. Fig.1.a) illustrates the 2-dimensional data points and regions indexed by the QTree. (A 2-dimensional space is used for the sake of simplicity.)

The leaves of the QTree (referred to as *buckets*) keep statistical information about the data items contained in the respective region. In Fig.1.b) leaves contain the number of points within the corresponding region. To summarize the sources, the number of triples contributed by the source is maintained inside the bucket as a set of pairs $\langle source, number\_of\_triples \rangle$. This set is kept ordered on the first component, so that sources contributing the highest number of triples appear first. Thus, the bucket corresponding to a region $R$ contains a list of pairs: $\langle s, card(\{t | hash(t) \in R, origin(t) = s\}) \rangle$ for each source $s$ containing at least one triple which is hashed to a point in region $R$. $origin(t)$ denotes the data source (a RDF set of triples available on the Web) containing $t$, obtained by dereferencing the subject of $t$.

When a SPARQL query is executed, each basic graph pattern (BGP) is converted into a set of coordinates by applying the hash functions to the pattern elements. BGPs containing variables are represented by regions spanning the whole range of hash values for the respective dimension. Then, the buckets containing sources contributing to the query are discovered by looking for overlapping MBBs in the QTree. Buckets returned by each BGP search are then intersected to identify joins. Sources contained in the resulting buckets are ranked based on an estimation of the *number of triples* contributing to the query result (computed

taking into account the bucket region and the size of the intersection between bucket and query regions).

### 2.2 Context-Aware Source Selection via QTree

The first approach we propose does not modify QTree internal nodes and exploits context information in the ranking phase only. Indeed, each source in a bucket is associated not only with the number of triples contributed by the source inside the bucket but also with the set of related contexts. Thus, buckets now contain sets of triples $\langle source, set\_of\_contexts, number\_of\_triples \rangle$. More specifically, the bucket corresponding to a region $R$ contains a set of triples $\langle s, \bigcup_{t|hash(t)\in R, origin(t)=s} \gamma(t), card(\{t|hash(t) \in R, origin(t) = s\}), \rangle$ for each source $s$ containing at least one triple which is hashed to a point in region $R$.

We assume now that a SPARQL query comes together with a reference context. The query representation and the index search do not change but, once the intersecting buckets are discovered and the contained sources retrieved, they are ranked according to: (*i*) *context distance*, computed as the aggregate (based on either AVG, MIN, or MAX function) of the distances $d$ in the context taxonomy between any context associated with the source and the query context; (*ii*) *number of triples*, defined as in Subsection 2.1. These two indicators are normalized between $[0, 1]$ and combined in a linear way.

### 2.3 Modelling the Context as a Fourth Dimension

The second, alternative, approach models the context as a fourth dimension of the data space (in addition to subject, predicate, and object), thus extending the QTree data structure. This leads to an increase of the dimensionality of the data summary but this increases the pruning potential as well and, thus, potentially, the effectiveness of the indexing structure.

Each context is mapped to a number, so that close numbers reflect similar contexts. This is achieved by an appropriate node numbering in the tree corresponding to the context taxonomy. This way, similar data will still be clustered into the same bucket. Thus, *hash* function, first applied to triples and returning a point in the 3D space, is now extended, leading to function $hash_4$, so that a quadruple consisting of a triple $t$ plus one of its contexts in $\gamma(t)$ is hashed to a point in the 4D space. The structure of internal nodes does not change with respect to the original QTree, while 4D QTree buckets keep list of pairs $\langle source, context \rangle$ with the number of triples they contribute to a bucket. Thus, assuming $R$ is the 4D region associated with a bucket, the bucket contains a set of pairs $\langle \langle s, c \rangle, card(\{t|hash_4(\langle t, c \rangle) \in R, c \in \gamma(t)\}) \rangle$.

Queries and the associated contexts are mapped to four dimensional regions, similarly to what happens for BGPs in the original QTree, but taking into account the context as fourth component. Index search now selects buckets that intersect a region representing *BGP+context* instead of only BGP. Ranking is performed similarly to what we discussed in Section 2.2 but ranking each $\langle source, context \rangle$ pair in a bucket separately.

## 3 Context-Modeling Using SKOS

Different context models [1, **?**] lead to different instantiations of the proposed approaches. In the following, as a concrete example, we discuss *context modeling* by means of a SKOS concept taxonomy.

*SKOS.* SKOS (Simple Knowledge Organization System) is a W3C recommendation for defining, publishing, and sharing concepts over the Web and it is used for knowledge organization. Concepts are entity categories identified by URIs. Though not providing semantic and reasoning capabilities as powerful as ontologies, concepts enhance the potential of search capabilities. Moreover, mappings between concepts from different concept schemes and in diverse data sources, and a hierarchical organization of concepts, are provided. The fundamental element of SKOS is the `(skos:Concept)` class, which instances, identified by URIs, represent domain concepts. Properties `(skos:broader)` and `(skos:narrower)` are one the inverse of the other and are used to assert a generalization/specialization relationship between two concepts [9]. For instance, the RDF triple `(:Palace skos:broader :TouristAttraction)` states that the touristic attraction concept is broader (i.e., more general) than the palace concept. SKOS is foreseen to be applied for associating concepts (i.e., resources of type `(skos:Concept)`) with resources through the `(dct:subject)` property.

*Context modeling through SKOS.* We model each context as a SKOS concept. In order to create a domain dependent context taxonomy, we started from a portion of DBpedia English SKOS concepts and related hierarchical relationships, as a *seed*. The portion consists of a subtree of the tree rooted at the concept `Tourist_attractions_in_Europe`. Starting from English DBPedia, resources related to contexts in the seed taxonomy through the `dct:subject` predicate, as well as triples describing them, are retrieved (via SPARQL queries posed to endpoints) and become part of our dataset (and they are associated with the corresponding concept as a context). Then, `(owl:sameAs)` links both at the instance and at the concept level are exploited for retrieving further concepts and instances.

The set of collected SKOS concepts is finally organized into a taxonomy $(\mathcal{C}, \preceq)$, containing 1632 nodes (SKOS concepts), related by the `(skos:broader)` relationship. Nodes are then numbered through a visit of the tree, and the generated numbers are used for hashing. Distance $d$ between two contexts is defined as the length of the shortest path connecting them in the taxonomy, normalized between 0 and 1. Contexts for triples (function $\gamma$) are defined as the contexts associated with their subjects or objects. The resulting dataset contains 1.086.874 triples, from 13.466 RDF sources.

---

https://www.w3.org/TR/skos-reference/

| Index | # generated buckets/maximum # buckets | Index Size | Hash interval |
|---|---|---|---|
| QTree | 8.092/25.000 | 31,3 MB | $[0-2000]$ |
| Ext QTree | 8.092/25.000 | 64,8 MB | $[0-2000]$ |
| 4D QTree | 50.000/50.000 | 188.4 MB | $[0-2000]$ |

Table 1: QTree details

## 4 Experimental Results

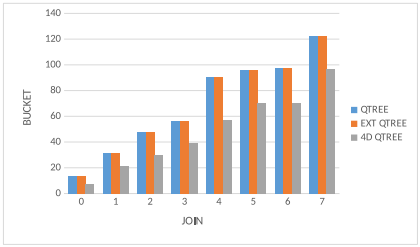
The experiments were performed on a machine with CPU Intel Core i3-2350M 2.30GHz, 6 GB of RAM size, running 64-bit Windows 7 Professional. Applications were developed in Java HotSpot(TM) 64-Bit Server VM under Java SE 1.8. Starting from the dataset described in Section 3, we created Qtree, Ext QTree, and 4D QTree, according to the values pointed out in Table 1. Notice that the maximum number of buckets (a parameter of the creation process) is higher for 4D QTree since the number of tuples to be indexed is higher in this case.

In this paper, we report experimental results related to the execution of *star-shaped queries*, sets of triple patterns sharing the same variable at the subject position. Star-shaped queries with a variable number of joins, ranging from 0 to 7, were randomly generated, guaranteeing at least one result on the considered dataset. For each number of join, 10 queries have been created, each associated with a random context. Each query was executed 10 times.

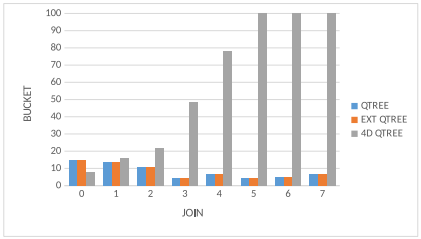
*Selectivity and Performance.* Fig. 2 compares QTree, Ext QTree, and 4D QTree with respect to the number of returned buckets, sources, and execution time. From Fig.2.a we note that, thanks to context information which is indexed together with triples, 4D QTree filters out more buckets during BGP search than 3D approaches. The same number of buckets is always returned by QTree and Ext QTree, since the two data structures differ only for information contained at the bucket level. On the other hand, 4D QTree returns a higher number of buckets as result for the join execution (Fig.2.b) since, as shown in Table 1, setting the maximum number of buckets to 50.000 does not guarantee a good data distribution among buckets (the number of created buckets coincide with the maximum value), many bucket regions intersect, and are returned as result. The number of sources returned by 4D QTree is however much lower compared with QTree and Ext QTree (Fig.2.c), thanks to context filtering during index search. For what concerns average execution time (in milliseconds, Fig.2.d), as expected, it coincides for QTree and Ext QTree and is often higher for 4D QTree, due to the higher number of intersections among buckets to be performed.

*Context Accuracy.* Accuracy is computed as the average distance between the contexts returned as results by Ext QTree and 4D QTree index searches and the context associated with the query. In the Ext QTree, since each source is associated with a set of contexts, we first compute an aggregate distance for

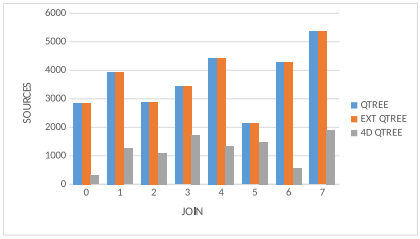
Experiments related to *path-shaped queries* are not reported due to space constraints.



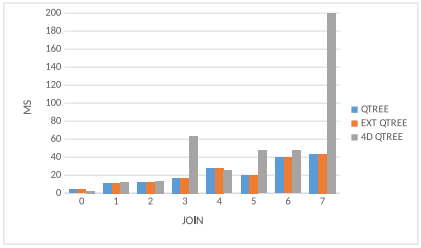

(a) Avg num. of returned buckets by BGP search




(b) Avg num. of join buckets




(c) Avg num. of returned sources




(d) Avg execution time

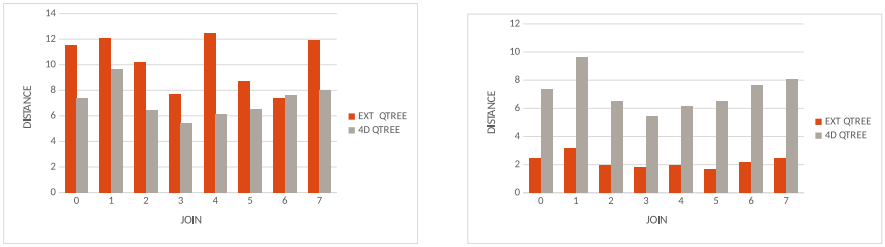
Fig. 2: Returned buckets, sources and execution time

each source (based on either AVG or MIN function) and then we average all them. Fig.3 shows that 4D QTree average context distance is always lower than the Ext QTree average context distance, when using AVG (Fig.3.a). Combined with Fig.2, this means that 4D QTree always retrieves less sources with a more relevant context, on average, than Ext QTree. When considering MIN (Fig.3.b), Ext QTree performs better than 4D QTree, since each data source returned by Ext QTree is associated with all its related contexts, including that leading to the minimum distance. This is not necessarily true with 4D QTree.

*Impact of Ranking.* We considered various ranking functions differing in the weights used in the linear combination. Aggregate context distances in Ext QTree are computed using the MIN aggregate function. The performed experiments, not reported here for space constraints, show that 4D QTree average ranking values computed for all the returned sources are lower than the corresponding Ext QTree ranking values. This is coherent with the previously presented results. When we consider the first $k$ ranked sources, 4D QTree ranking values are higher than corresponding Ext QTree ones. Thus, 4D QTree is able to select, among all the returned ones, "better" sources than the Ext QTree.

## 5 Conclusions and Future Work

In this paper, we proposed two approaches exploiting *context* information in increasing source selection effectiveness and efficiency. This way of exploiting




(a) Average Distance (b) Average Min. Distance

Fig. 3: Context result accuracy

context is, to the best of our knowledge, novel. Experimental results show that the proposed context-based data summaries succeed in selecting the most relevant data sources with respect to the query and the associated context, with a reasonable overhead. Future work includes the design of algorithms supporting top-k search in QTree and the extension of the proposed data structures with data source quality information, with the aim of selecting not only relevant but also accurate answers, by preferring trusted sources.

## References

1. C. Bettini et Al. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing* 6(2):161–180, 2010.f
2. C. Bolchini et Al. A data-oriented survey of context models. *ACM Sigmod Record*, 36(4):19–26, 2007.
3. B. Catania et Al. Wearable queries: Adapting common retrieval needs to data and users. In *DBRank*, 2013.
4. B. Catania, G. Guerrini, and B. Yaman. Context-dependent quality-aware source selection for live queries on linked data. In *EDBT*, 2016.
5. C. Ghidini and F. Giunchiglia. Local Models Semantics, or contextual reasoning=locality+compatibility. *Artif. Intell.* 127(2):221–259, 2001.
6. A. Harth et Al. Data summaries for on-demand Queries over Linked Data. In *WWW*, 2010.
7. T. Heath and C. Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.
8. R. Meusel et Al. Towards automatic topical classification of lod datasets. In *CEUR Workshop Proceedings. Vol. 1409*, 2015.
9. A. Miles et Al. Skos core: simple knowledge organisation for the web. In *International Conference on Dublin Core and Metadata Applications*, pages 3–10, 2005.
10. T. Rekatsinas et Al. Finding quality in quantity: The challenge of discovering valuable sources for integration. In *CIDR*, 2015.
11. J. Umbrich et Al. Comparing Data Summaries for Processing Live Queries over Linked Data. *World Wide Web*, 14(5-6):495–544, 2011.
12. B. Yaman. *"Exploiting Context-Dependent Quality Metadata for Linked Data Source Selection"*. PhD thesis, 2018.