Duplicate-Aware Federated Query Processing over the Data Web

Muhammad Saleem
Digital Enterprise Research Institute,
National University of Ireland, Galway
muhammad.saleem@deri.org
Third Author Name
Affiliation
Addrss
Email

Axel-Cyrille Ngonga Ngomo
Universität Leipzig, IFI/AKSW
PO 100920, D-04009 Leipzig
ngonga@informatik.uni-leipzig.de
Fourth Author Name
Affiliation
Addrss
Email

ABSTRACT

Over the last years, the Linked Data Web has developed into a large compendium of interlinked data sets from diverse domains. Due to this distributed character, several of these data sets contain either duplicate or complementary data. So far, several federated querying approaches have been developed to retrieve information from these data sources. Still, only little attention has been paid to the effect of duplicates and fragmented data on the ranking of data sources during the processing of federated queries. This work presents a novel approach federated querying based on min-wise independent permutation vectors applied to RDF predicates. Our approach allows evaluating the ranking of data sources accurately while taking into account duplicate data. By these means, our approach achieves significantly smaller mean squared errors that the state of the art with respect to the approximation of result set sizes as well as in the ranking of data sources.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Distributed databases*

General Terms

Algorithms, Experimentation, Theory

Keywords

Federated queries, SPARQL, deduplication

1. INTRODUCTION (AN+MS)

Over the last years, the Linked Data Web has developed into a large compendium of interlinked data sets from diverse domains. One of the central principles underlying the architecture of these data sets is the reuse of URIs and vocabularies as well as the linking of knowledge bases[20]. One of

the results of this architectural choice is that certain queries can only be answered by retrieving information from several knowledge bases. This type of querying, called *federated querying*, is of central importance for manifold applications such as question answering, knowledge retrieval and data integration. In addition to the information necessary to answering queries being distributed, certain pieces of information (i.e., triples) can be found in several knowledge bases. For example, the name of movie directors can be found both in DBpedia and LinkedMDB. Similarly, the authors of papers can be found in both the ACM and DBLP libraries. We call triples that can be found in several knowledge bases across the Web of Data *duplicates*.

While the importance of federated queries over the Web of Data has been stressed in previous work, the impact of duplicates has not received much attention. Yet (as we will show in the remainder of this paper), a duplicate-aware approach to query processing can lead to more time-efficient and effective algorithms for federated queries. In this paper, we address this drawback by presenting a duplicate-aware approach for query processing based on min-wise independent permutation (MIPs) vectors. Our approach is able to predict the amount of new information contained in a knowledge base with a higher accuracy than the state of the art, leading to a better performance with respect to source ranking and result set size optimization. In the rest of this paper, we aim to show experimentally that our approach outperforms the state-of-the-art by running it against two other approaches on a benchmark of 79 queries. We begin by giving a brief overview of the state of the art in federated query processing. In addition, we argue for the use of MIPS for the approximation of result size sets. Thereafter, we present our duplicate-aware federated query processing approach. After an overview of the datasets used in this paper, we present experimental results which corroborate the superior efficiency of our approach. We conclude the paper with a discussion of our findings and an overview of future work.

2. RELATED WORK (MS)

2.1 Federated SPARQL queries

The research work on federated query processing over the Data Web can be categorized into two main directions:

- Service description/index-assisted approaches, which make use of the dataset summaries that have been

collected in a pre-processing stage. These approaches lead to a more efficient query federation. However, the index needs to be constantly updated to ensure up-to-date results. Also, the index size should not be large to ensure that it does not increase the overall query processing time.

- Index-free approaches, in which the query federation is performed without using any stored data summaries. This approach is less efficient however; the query response time is faster than index assisted approach.

Both of the above approaches can further be divided into two categories (a) Query federation with complete result retrieval (b) Query federation with partial (sufficient) record retrieval. In our case, we are focusing on index-assisted which can be used both for complete and partial record retrieval .

Heiner et. al[1] describes how to extend the Sesame RDF repository to support distributed SeRQL queries over multiple Sesame RDF repositories. They use a special index structure to determine the relevant sources for a query. Quilitz and Leser[2] uses the theoretical knowledge of [1] to develop the first federated query engine (named DARQ) for remote RDF data sources. DARQ uses service descriptions for relevant data source selection. A service description describes the capabilities of a SPARQL endpoint. They used query rewriting mechanism based on[8] and a cost-based optimization algorithm to reduce the query processing time with the minimum bandwidth usage. DARQ is compatible with any SPARQL endpoint that implements the SPARQL standard. Their experimental results show that the optimization algorithm can greatly reduce the query processing time.

Andreas et al.[3] proposed a solution similar to DARQ using a mediator approach. All the SPARQL endpoints need to register first with the mediator using HTTP POST requests with an RDF document attached. The mediator continuously monitors the SPAQL endpoints for any dataset changes and updates the service descriptions automatically. Unlike DARQ, the service descriptions remain up-to-date all time.

Olaf et al.[4] present a new approach for federated query processing over the Web of Data. Their approach discovers data that might be relevant for answering a query during the query execution itself. The discovery of relevant data is accomplished by traversing RDF links. They used an iterator-based pipeline and a URI prefetching approach for efficient query execution. Both DARQ and [3] are not able to discover relevant data sources by the query itself.

The previous techniques cannot answer all types of queries due to index or service description limitations. Umbrich et al.[5], [6] proposed a Qtree-based index structure which summarizes the content of data source for query execution over the Web of Data. They concluded that this approach is able to handle more expressive queries and return more complete results than previous approaches. Schwarte et al.[7] proposed an index-free query federation for the Web of Data. Their approach gives a reasonably fast data retrieval as compared to all the previous techniques. However, since they do not keep any statistics, it works only for limited data sources.

Li and Heflin[15] build a tree structure which supports federated query processing over multiple heterogeneous sources. They make use of an OWL reasoner to answer queries over the selected sources and their corresponding ontologies. Kaoudi

et al.[16] propose a federated query technique on top of distributed hash tables(DHT). The overall goal of this technique is to minimize the query execution time and the bandwidth consumption by reducing the cardinality of intermediate results. The DHT-based optimizer make use of three greedy optimization algorithms for best plan selection.

Ludwig and Tran[18] proposed a mixed query engine; that assumes some incomplete knowledge about the sources to select and discover new sources at run time. A Symmetric Hash Join is implemented to incrementally produce answers. This approach was extended to query both remote and local linked data[19].

Maribel et al.[21] presented ANAPSID, an adaptive query engine that adapts query execution schedulers to SPARQL endpoints data availability and run-time conditions. ANAPSID provides physical SPARQL operators that detect when a source becomes blocked or data traffic is bursty. The operators produce results as quickly as data arrives from the sources. Amol et al.[17] presents a survey of the adaptive query processing techniques, common issues, and the settings in which each piece of work is most appropriate. Avalanche[24] gathers endpoints dataset statistics and bandwidth availability on-the-fly before the query federation. The advantage of this technique is the live query execution over the Web of Data. However, on-the-fly statistics calculation can increase the overall query execution time. Olaf[25] proposed a heuristics-based technique to minimize query intermediate results.

Other noteable contributions LIMES[26], SILK[27], EAGLE[28] provides time-efficient approaches for Link Discovery using various distance measure techniques. LIMES framework utilize the mathematical characteristics of metric spaces to compute similarity estimates between instances. It makes use of the triangle inequality to partition the metric space into different portions. Each of these portions of the space is then represented by an exemplar[33] that allows to compute an accurate distance approximation between different instances. SILK implements a multi-dimensional blocking approach for link discovery. The Levenshtein distance[34] is used for approximate for similarity estimation between different instances. EAGLE presents a novel active learning approach to learning link specifications based on genetic programming. EAGLE generates highly accurate link specifications while reducing the annotation burden for the user.

None (to the best of our knowledge) of the above techniques has taken into account the possibility of mutual overlap between different RDF datasets before query execution. Like DARQ engine, we use service descriptions for query federation. In addition, we store MIPs vector statistics for each of the indexed data source. Query federator makes use of this special statistic information to estimate the possible overlap between the results of different sub-queries.

2.2 Dataset Overlap Estimation

Avoiding the retrieval of duplicate records is a crucial issue in large scale distributed datasets. The overlap between two datasets can be estimated using min-wise independent permutations[9], bloom filters[10], or compressed bloom filters[11]. Bender et al.[12] also proposed a technique for predicting the query-specific mutual overlap between peers. However, MIPs have proved to be the more accurate overlap estimation with small bandwidth consumption[9].

3. NOTATION

In this section, we present the core of the notation that will be used throughout this paper. We denote data sources with S and the total number of data sources with n. The set of all data sets is dubbed $\mathfrak{S}$. The set of all possible result sets is denoted R while the set of all possible SPARQL queries is labeled with Q. A data source ranking function $rank : S \times Q \rightarrow \{1 \dots n\}$ is a function that assigns a ranking to each data source given a particular query $q \in Q$. Note that for any source ranking function $rank$, we assume that $\forall S, S' \forall q \in Q : S \neq S' \iff rank(S, q) \neq rank(S', q)$. A result set estimation function $est : S \times Q \rightarrow \mathbb{N}$ aims at approximating the size of the result set that will be returned by a given query. Note that this function plays a crucial role in the processing of federated queries as it is most commonly used to decide upon the ranking of data sources for a given query. The aim of a federated query system such as the one described in this work is thus to optimize its estimation function est so as to ensure a ranking of the source close to the optimal ranking.

4. APPROACH (MS)

4.1 Overview

Overall, our approach works as follows: The user begins by issuing a SPARQL query using any input form. The query is then forwarded to the query engine. Upon receiving, it is parsed to get all triples. The parser rewrites the original query in an execution-optimized form. The federator then makes use of the services descriptions directory/index to decompose single query into multiple sub-queries. The query optimizer takes all sub-queries to generate an optimize execution plan. At last, all the optimized sub-queries are forwarded to the relevant SPARQL endpoints. The results of each sub-query execution are integrated and finally sent back to the user. This architecture for distributed query processing is shown in figure 1.

While this architecture is the same for all federated query systems, the federator of our system differ significantly from those of other approaches as it make use of the MIPs statistical information to generate duplicate-free query execution plan. The generation of MIPs statistical information is explained in the next section.

4.2 Service Description Directory

To find the set of relevant servers and generate multiple sub-queries, the query engine needs information about the different data sets available at different servers. We call this catalog the *service description directory*. Each service description provides a declarative description of the data available from a server along with statistical information.

Our *service description directory* stores *server url, graph name*, and for each distinct predicate p we record *predicate name, MIPs vector, subject selectivity, object selectivity* and *total number of triples* having predicate p. The MIPs vectors are used to estimate the dataset overlap among the capable datasets before the sub-query distribution. The *subject* and *object selectivity* are used for sub-query result set estimation when *subject, object* are bound respectively. A sample service description is given in the listing 1.

It is very crucial to keep the index size enough to minimize pre-query federation time. However, it must contain
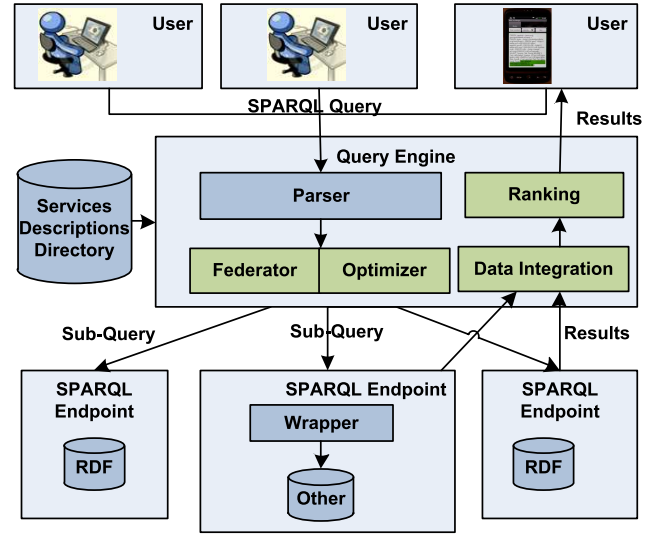

Figure 1: Federated SPARQL query processing architecture

Listing 1: A Service Description Example

```
[] a sd:Service ;
    sd:url    <http://localhost:8890/sparql> ;
    sd:graph ``diseasome/DS1'';
    sd:capability
    [
     sd:predicate   diseasome:name ;
     sd:MIPv  ``-6908232 -7090543 -6892373 -7064247 ...'';
     sd:subjectSelectivity   ``0.0068'';
     sd:objectSelectivity    ``0.0069'' ;
     sd:triples    147  ;
    ] ;
    sd:capability
    [
     sd:predicate   diseasome:chromosomalLocation ;
     sd:MIPv  ``-7056448 -7056410 -6845713 -6966021 ...'';
     sd:subjectSelectivity   ``0.0062'' ;
     sd:objectSelectivity    ``0.0072'' ;
     sd:triples    160  ;
    ] ;
```

sufficient information to enable sub-query generation, optimization and pre-processing dataset overlap estimation. Our index size is mostly depended upon the size of the MIPs vectors which can be adjusted to any length. The generation of the MIPs vectors is explained in the next section.

4.3 MIPs Operation

Min-Wise Independent Permutations (MIPs) assumes that the set elements can be ordered and computes N random permutations of the elements. Each permutation uses a linear hash function of the form $h_i(x) := a_i * x + b_i \mod U$ where U is a big prime number and a_i, b_i are fixed random numbers. By ordering the resulting hash values, we obtain a random permutation. For each of the N permutations, the MIPs technique determines the minimum hash value, and stores it in an N-dimensional vector, thus capturing the minimum set element under each of these random permutations. The technique is illustrated with an example in figure 2. Its fundamental rationale is that each element has the same probability of becoming the minimum element under a random permutation. By using sufficiently many different permutations, we can approximate the set cardinality.

An unbiased estimate of the pair-wise resemblance of sets using their N-dimensional MIPs vectors is obtained by counting the number of positions in which the two vectors have the same number and dividing this by the number of per-
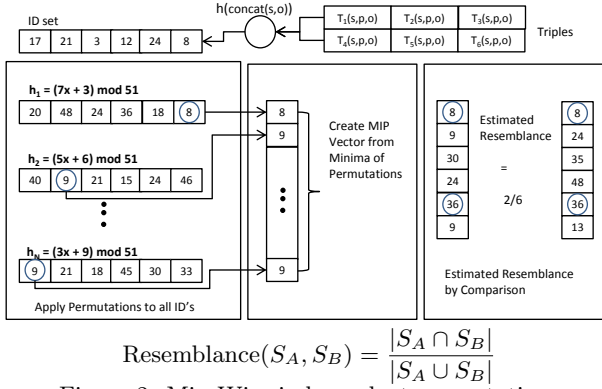
$$\text{Resemblance}(S_A, S_B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|}$$

Figure 2: Min-Wise independent permutations

Algorithm 1 Sub-query generation
Require: $T = \{t_1, .., t_n\}$, // set of triple patterns
$D = \{D_1, .., D_n\}$ // sets of data sources matching to the triple patterns
1: $queries = \emptyset$, $separateQueries = \emptyset$
2: **for** each $t_i \in T$ **do**
3:     **if** $D_i = \{d\}$ **then**
4:         $q = queries.getQuery(d)$
5:         **if** q not null **then**
6:             $q.T = q.T + t_i$
7:         **else**
8:             $queries = queries + (\{t_i\}, \{\}, d)$
9:         **end if**
10:    **else**
11:        **for** each $d_j \in D_i$ **do**
12:            $separateQueries = separateQueries + (\{t_i\}, \{\}, d_j)$
13:        **end for**
14:    **end if**
15: **end for**
16: **return** $queries \cup seperateQueries$ // Return all queries

Figure 3: Sub-query Generating Algorithm

mutations N. Essentially, this holds as the matched numbers are guaranteed to belong to the intersection of the sets.

For a predicate $p \in S$, the set elements are all triples $T \in S$ having predicate p. We concatenate the *subject* and *object* of each triple in T and apply a hash function to get the *ID set*. Finally, we computes N random permutations of the *ID set*. The resulting MIP vector is stored against each capability of the service description. Note that we are not applying hash function to the complete triple because the *predicate* is already stored with each service capability and source selection is bound by query *predicate*.

4.4 Source Selection

A SPARQL query contains one or more basic graph patterns. The query planning step is performed separately for each basic graph pattern. The source selection algorithm used in DARQ for a query simply matches all triple patterns against the capabilities of the data sources. The matching compares the predicate in a triple pattern with the predicate defined for a capability. Let BGP be a set of triple patterns in a filtered basic graph pattern. The result of the source selection is a set of data sources S_j for each triple pattern

$$t_j = (s_j, p_j, o_j) \in \text{BGP with}$$

$$S_j = \{s | (s,C) \in R \wedge \exists (p_j, r) \in C : r(s_j, o_j) = \text{true}\},$$

Where $r(s_j, o_j)$ represents any constraint on subject or object of a triple with predicate p_j.

We extend the DARQ source selection algorithm by comparing MIP Vectors for similarity. For a triple pattern, we compare the MIP vectors of all the relevant data sources for possible similarity. A specific source S is only added to the set of capable sources t_j, if S is able to produce new results above a specific threshold value defined for similarity. For a triple T in dataset S, if the new record add-up is less than the threshold, S is ignored from set S_j. For complete records retrieval the threshold value should be 1. This is explained in the next section.

4.5 Source Ranking

Result set estimation

4.6 Subquery generation

The sub-query generating algorithm is given in figure 2. We represent a sub-query as triple *(T,C, s)*, where T is a

set of triple patterns, C is a set of value constraints and s is the data source that can answer the sub-query. Algorithm 1 shows how the sub-queries are generated. If a triple pattern matches exactly one data source ($S_i = \{s\}$) the triple will be added to the set of a sub-query for this data source. All triples in this set can later be sent to the data source in one sub-query. If a triple matches multiple data sources the triple must be sent individually to all matching data sources in separate sub-queries.

5. EXPERIMENTS AND RESULTS

The goal of our experiments was to quantify how well our duplicate-aware solution approximates the result sets from each of the data source from which it can choose and how this approximation affects the way the data sources are ranked. For this purpose, we compared our approach with two other approaches: DARQ and the optimal duplicate-unaware solution to result set estimation and query ranking. In the following, we describe our experimental setup in detail and contrast the results achieved by each of the approaches. All data used is either publicly available or can be found at the project webpage.[1]

5.1 Experimental Setup (AN)

5.1.1 Datasets

Within our experiments, we used the four datasets described in Table 1. We chose data sets of different data sizes to gain some insights on the scalability of our approach. The size of the MIPS vectors was set to achieve a compression ratio of approx. 95% as used in previous work [?]. In addition, we emulated the difference between the number of relevant triples across datasets by introducing the discrepancy constant d. Given n data sources of sizes $|S_i|$

$$d = \max_{1 \le i \le n} |S_i| - \min_{1 \le j \le n} |S_j|. \tag{1}$$

Note that the discrepancy accounted for between ca, 1.6% and 2.8% of the triples and was to increasingly higher values for increasingly larger data sets. The number of slices was set to 10 in all experiments.

5.1.2 Queries

We used three main types of queries. Simple basic graph pattern (BGP) queries consisted of exactly one triple pat-

---
[1] http://???

Dataset	Size (MB)	Summary Size (MB)	CR (%)	Slices	Discrepancy	Duplicate Slices	Duplicate Datasets	Total Triples	Sum. Gen. Time (sec)
Diseasome	18.6	1.0	95	10	1,500	1	10	91,122	9
DBLP	39.0	2.8	93	10	2,500	1	10	234,405	16
Geo Coordinates	274.1	24.4	92	10	50,000	2		1,900,006	2705
LinkedMDB	448.9	41.3	91	10	100,000	1	2	3,579,616	3556

Table 1: Overview of datasets. CR stands for Compression Ratio.

Dataset	BGP	S-1	S-2	P-1	P-2	P-3	Total
Diseasome	5	5	5	4	5	2	26
Geo	5	5	5	0	0	0	15
LinkedMDB	5	0	0	0	0	0	5
Publication	5	5	5	7	7	4	33
Total	20	15	15	11	12	6	79

Table 2: Distribution of query types across datasets

Compression Ratio	$RSMSE$	$RankMSE$
91%	253.9×10^6	3.4
88%	33.3×10^6	0.6
85%	0	0

Table 3: Effect of compression ratio

tern in the query. Star-shaped and path-shaped queries were defined as in [**?**]: star-shaped queries (denoted S-k) have one variable as subject and contained k triple patterns. Path-shaped queries, denoted P-k on the other hand were generated by using a random walk approach to generate a path of length k. Note that previous work has shown that these query shapes are the most common shapes found in real-world RDF queries [**?**]. Overall, our benchmark data consisted of 79 queries as shown in Table 2. Note that some query shapes could not be used on certain datasets due to the topology of the ontology underlying the datasets not being permissive for such queries. For example, S-2 queries could not be sent to LinkedMDB because

5.1.3 Metrics

We used two measures to compare our approach with the state of the art. First, we measured the mean squared error of each approach with respect to their result set evaluation as follows: For each query q_i, each source S_i contained a number $R_i(q_i)$ of non-duplicate results. Each of the approaches generated an approach $\tilde{R}_i(q_i)$ of $R_i(q_i)$. The mean squared error with respect to the result set size (short $RSMSE$) was then computed as follows:

$$RSMSE = \frac{1}{|Q|} \sum_{q_i \in Q} |R_i(q_i) - \tilde{R}_i(q_i)|^2. \qquad (2)$$

In addition to the MSERS, we also computed how well the approaches were able to rank the data sources with respect to the unseen triples they would return for a given query. Given the ideal ranking $rank(S_i, q_j)$ of each data source S_i for each query q_j, we computed the mean squared ranking error (short $RankMSE$) of an approach characterized by the the ranking function $\tilde{r}(S_i, q_j)$ by using the following formula:

$$RankMSE(q_j) = \frac{1}{|\mathbb{S}|} \sum_{S_i} |rank(S_i, q_j) - \tilde{r}(S_i, q_j)|^2. \quad (3)$$

$$RankMSE = \frac{1}{|Q|} \sum_{q_j \in Q} RankMSE(q_j). \qquad (4)$$

For both measures, higher values are obviously worse.

5.2 Results

### 5.2.1 Result set estimation and ranking

The RSMSE results are shown in Figures 4 while the ranking results are shown in Figure 5. In each figure, we compared the results of the approaches w.r.t. both the whole input query and to each of the triples patterns contained in the input queries. Our results clearly show that we outperform the state of the art w.r.t to both result set estimation and ranking. Especially, our estimation of the result set sizes is approximately 4 orders of magnitude better than DARQ's or even DU's. This is due to our approach being able to capture the existence of duplicates in the data sets and thus to better approximate the number of new results that can be expected from an endpoint. As expected, the ranking error for datasets decreases when carrying the ranking of data sources at triple pattern level instead of query level (see Figures 4b and 4a). This is simply due to the overall error increasing when combining the several triple patterns to a query. The same overall behavior is also displayed when considering the result set estimation error (see Figures 5b and 5a). Interestingly, the approximation of our approach is slightly poorer on star-shaped queries (approx. 0.2 on S-2 queries). This is a result of our approach to generating the estimations for whole query, which is carried out by using the lower bound of the estimation of single query patterns in the query. Small errors in the estimation of the size of the result size can thus lead to minor absolute errors.
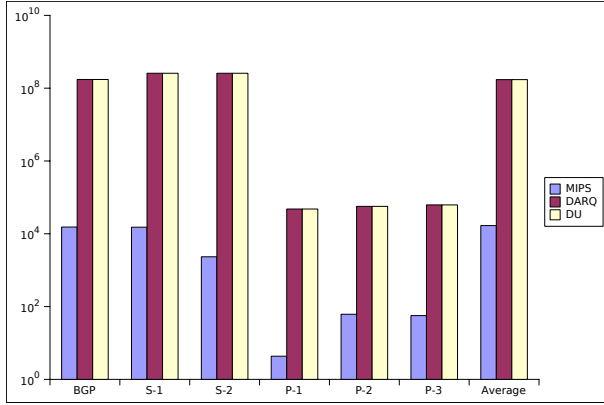
As stated before, we used a compression ratio of approx. 95% across all experiments. The main advantage of such a high compression ratio is the small size of the index. Yet, the higher the compression ratio, the larger the error margin of the ranking function. When given a query whose sub-queries lead to very similar result set estimations, this larger error margin can lead to substantial $RankMSE$ and $RSMSE$ rates. We thus studied the effect of decreasing the compression ratio on the $RSMSE$ and $RankMSE$ for such a query. Our results in Table 3 suggest that MIPS vectors can successfully detect even small differences in the sizes of the result sets when given a sufficiently small compression ratio.

5.2.2 Query Escaping

One of the main advantages of systems that process federated queries at sub-query-level is that refrain for sending sub-queries to all available data sources if these sources appear not to contain relevant information. The advantage of this strategy obviously lies in superior runtimes and better resource management (less network traffic, smaller space requirements, etc.). We thus measured the number of sub-
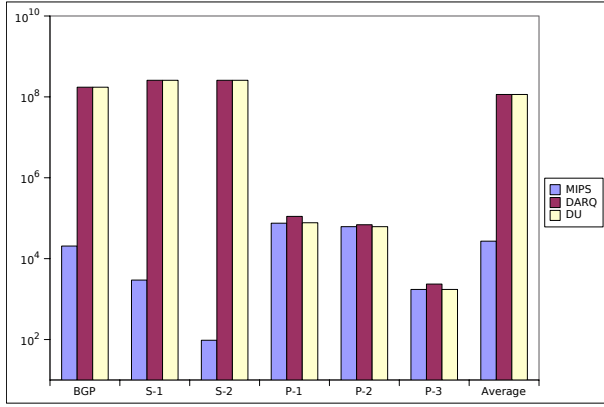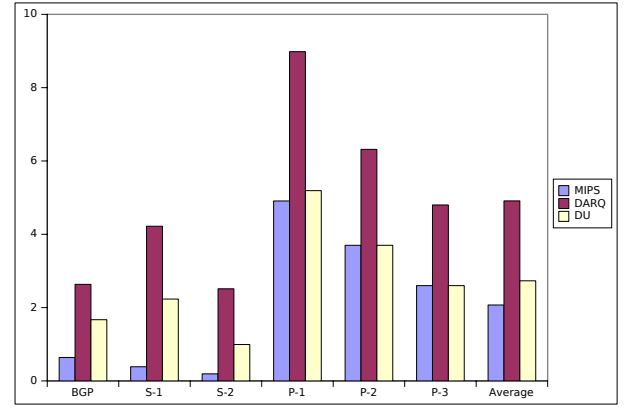
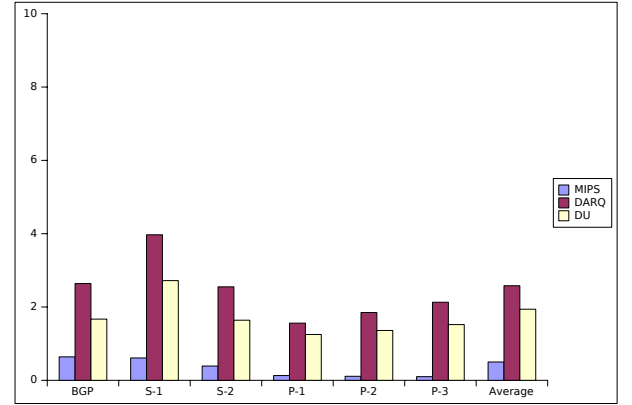(a) RSMSE for query-wise selection (vertical log scale)

(a) RankMSE for query-wise selection

(b) RSMSE for subquery-wise selection (vertical log scale)
Figure 4: Mean squared errors with respect to result set evaluation

(b) RankMSE for subquery-wise selection
Figure 5: Mean squared errors with respect to ranking

Dataset	BGP	S-1	S-2	P-1	P-2	P-3	Total
Diseasome	7	21	21	8	20	12	89
Geo	10	15	23	0	0	0	48
LinkedMDB	5	0	0	0	0	0	5
Publication	1	2	5	0	0	0	8
Total	23	38	49	8	20	12	150

Table 4: Distribution of query escaped by our approach

query executions that could be escaped by our approach without any loss in recall. The results shown in Table 4 clearly display that our approach can effectively reduce the total number of queries executed overall. Especially, on BGP queries, we were able to escape up to 10 subqueries (%?). Our approach was most successful on star-shaped queries, where up to 21 subqueries could be escaped.

6. DISCUSSION (AN)

7. REFERENCES

[1] Stuckenschmidt, H., Vdovjak, R., Houben, G.J., Broekstra, J.: Index structures and algorithms for querying distributed rdf repositories. In: WWW'04. (2004).

[2] Quilitz, B., Leser, U.: Querying distributed RDF data sources with SPARQL. In: European semantic web conference on the semantic web: research and applications (ESWC 2008). LNCS, vol. 5021, pp. 524-538. Springer, Heidelberg (2008).

[3] Langegger, A., Waoay, W., Blaochl, M.: A semantic web middleware for virtual data integration on the Web. In: European semantic web conference on the semantic web: research and applications (ESWC 2008). LNCS, vol. 5021, pp. 493-507. Springer, Heidelberg (2008).

[4] Hartig, C. O, Bizer, J.-C. Freytag. Executing SPARQL queries over the web of linked data. 8th International Semantic Web Conference (ISWC 2009).

[5] Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.U. Umbrich, J.: Data summaries for on-demand queries over linked data. In: (WWW 2010), USA.

[6] Umbrich, J., Hose, K., Karnstedt, M., Harth, A., Polleres.: Comparing data summaries for processing live queries over Linked Data. In:(WWWJ 2011), special issue querying the data web.

[7] Schwarte, .A, Haase, .P, Hose, .K, Schenkel, R, Schmidt, M.: FedX: Optimization Techniques for Federated Query Processing on Linked Data. In: 10th International semantic web conference (ISWC 2011).

[8] Perez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. In: 4th International SemanticWeb Conference (ISWC), Athens, GA, USA (November 2006).

[9] Broder, A. Z., Charikar, M., Frieze, A. M., & Mitzenmacher, M. (2000). Min-wise independent permutations. Journal of Computer and System Sciences, 60(3), 630-659.

[10] Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. Communications of ACM, 13(7), 422-426.

[11] Mitzenmacher, M. (2002). Compressed bloom filters. In IEEE/ACM Transactions on Networking, 10(5), 604-612.

[12] Bender, M., Michel, S., Weikum, G., & Zimmer, C. (2005). Minerva: Collaborative p2p search. In Proceedings of the (VLDB) conference (Demonstration).

[13] Parreira, J.X, Michel, S., Weikum, G.: p2pDating: Real life inspired semantic overlay networks for Web search. In: Journal of Information Processing and Management 43 (2007) 643-664.

[14] http://www4.wiwiss.fu-berlin.de/lodcloud/state/

[15] Y. Li and J. Heflin. Using reformulation trees to optimize queries over distributed heterogeneous sources. In ISWC, pages 502-517, 2010.

[16] Z. Kaoudi, K. Kyzirakos, and M. Koubarakis. Sparql query optimization on top of dhts. In ISWC, pages 418-435, 2010.

[17] A. Deshpande, Z. G. Ives, and V. Raman. Adaptive query processing. Foundations and Trends in Databases, 1(1):1-140, 2007.

[18] G. Ladwig and T. Tran. Linked data query processing strategies. In ISWC, pages 453-469, 2010.

[19] G. Ladwig and T. Tran. Sihjoin: Querying remote and local linked data. In ESWC (1), pages 139-153, 2011.

[20] Auer.S, Lehmann. J, Ngonga, A.C.: Introduction to Linked Data and Its Lifecycle on the Web. In: Reasoning Web. Semantic Technologies for the Web of Data, Pages 1-75, 2011.

[21] M. Acosta, M.-E. Vidal, T. Lampo, J. Castillo, and E. Ruckhaus. Anapsid: An adaptive query processing engine for sparql endpoints. In Proc. of International Semantic Web Conference (ISWC), 2011.

[22] C. Buil-Aranda, M. Arenas, and O. Corcho. Semantics and optimization of the sparql 1.1 federation extension. In ESWC (2), pages 1-15, 2011.

[23] . M. Stoker, A. Seaborne, A. Bernstein, C. Keifer, and D. Reynolds. SPARQL Basic Graph Pattern Optimizatin Using Selectivity Estimation. In WWW, 2008.

[24] C. Basca and A. Bernstein. Avalanche: Putting the Spirit of the Web back into Semantic Web Querying. In The 6th International Workshop on SSWS at ISWC, 2010.

[25] O. Hartig. Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In ESWC, pages 154-169, 2011.

[26] Axel-Cyrille Ngonga Ngomo and Soren Auer. LIMES - A Time-Ecient Approach for Large-Scale Link Discovery on the Web of Data. In Proceedings of IJCAI, 2011.

[27] R. Isele, A. Jentzsch, and C. Bizer. Ecient Multidimensional Blocking for Link Discovery without losing Recall. In WebDB, 2011.

[28] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. EAGLE: Ecient Active Learning of Link Specications using Genetic Programming. In: ESWC(2012).

[29] Unger, C.; Buhmann, L.; Lehmann, J.; Ngonga Ngomo, A.-C.; Gerber, D.; and Cimiano, P. Templatebased question answering over RDF data. In WWW (2012).

[30] G.Koutrika, and Y.Ioannidis, Personalized Queries under a Generalized Preference Model, In ICDE 2005.

[31] G.Koutrika, and Y.Ioannidis, Personalization of Queries in Database Systems, In ICDE 2004.

[32] S.Magliacane, A. Bozzon.E.D. Valle. Efficient Execution of Top-k SPARQL Queries. In: ISWC(2012), USA.

[33] B. J. Frey and D. Dueck. Clustering by passing messages between data points. Science, 315:972-976, 2007.

[34] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In Soviet Physics Doklady, volume 10, 1966.

[35] M. Morsey, J. Lehmann, S. Auer, and A.-C. Ngonga Ngomo. DBpedia SPARQL Bench- mark - Performance Assessment with Real Queries on Real Data. In 10th Interna- tional Semantic Web Conference (ISWC2011), 2011.