ICT Seventh Framework Programme (ICT FP7)

Grant Agreement No: 318497

Data Intensive Techniques to Boost the Real – Time Performance of Global Agricultural Data Infrastructures

D4.2: Experimental Report on Current Datasets

Deliverable Form	
Project Reference No.	ICT FP7 318497
Deliverable No.	D4.2
Relevant Workpackage:	WP4: Rigorous Experimental Testing
Nature:	R
Dissemination Level:	PU
Document version:	1.0
Date:	21/11/2014
Authors:	UAH, NCSR-D
Document description:	This report documents experiments conducted on datasets contributed and prepared in WP2 using the research components developed in WP3.

Document History

Version	Date	Author (Partner)	Remarks
Draft v0.1	8/10/2014	NCSR-D, UAH	First draft circulated to consortium
Draft v0.2	7/11/2014	NCSR-D, UAH	Pre-final draft
Draft v0.9	14/11/2014	SWC	Internal review
Final v1.0	21/11/2014	NCSR-D, UAH	Submitted as D4.2

EXECUTIVE SUMMARY

This report documents rigorous testing experiments using the *SemaGrow* research prototypes and based on the *SemaGrow* use cases and datasets.

These experiments go beyond the technical testing of the project's prototype components carried out in the WP3 tasks: the experiments reported here test the overall methodology of the system, its underlying assumptions, and well as the individual components in the context of the overall system.

Besides evaluating *SemaGrow* methods, these experiments also contribute to the development of the *automatic rigorous testing* components that periodically and automatically test deployments of the *SemaGrow Stack*.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose and Scope

This report documents rigorous testing experiments using the *SemaGrow* research prototypes and based on the *SemaGrow* use cases and datasets.

These experiments go beyond the technical testing of the project's prototype components carried out in the WP3 tasks: the experiments reported here test the overall methodology of the system, its underlying assumptions, and well as the individual components in the context of the overall system.

Besides evaluating *SemaGrow* methods, these experiments also contribute to the development of the *automatic rigorous testing* components that periodically and automatically test deployments of the *SemaGrow Stack*.

1.2 Relation to other Work Packages and Deliverables

Work in this task receives:

- The experimental methodology (D4.1)
- Prototypes and infrastructure (WP3, WP5)
- Test data (WP2)

in order to provide:

- Evaluation of SemaGrow technologies
- Input to WP5 about the development of Automatic Rigorous Testing components.

That is, besides their value as evaluation, the experimental setups developed in this task are also the basis for the automatic components that measure system health on live deployments.

1.3 Big Data Aspects

We see two distinct challenges in querying a federation of big data sources:

- Small results from big data: in this "needle in a haystack" situation we are joining properties from different big datasets in order to retrieve a result set that does not constitute big data by itself. The challenge here is to have accurate *instance-level metadata* (Task 3.1) as well as an intelligent *query execution planner* (Task 3.4) that guides the executor engine along a query execution plan that never retrieves large quantities of results that will fail to join with subsequent query patterns. This is discussed in this deliverable.
- Big results from big data: in this situation the result set constitutes big data, and no amount of query plan optimization can avoid this since this is what has been requested by the client. In this case the *execution engine* must efficiently use its computational, memory, and disk resources in order to compute and dispatch results as quickly as they are coming in from the distributed data sources and, thus, avoid retaining large partial results. This will be discussed in the follow-up *D4.4 Experiment Report on Projected Datasets* where Semagrow technologies will be tested on even larger datasets.

2. EXPERIMENTS ON MAINTAINING HISTOGRAMS

2.1 Objectives and Requirements

In this experiment, we measure the behaviour of the *SemaGrow Stack* when querying evolving data sources. That is not to say that new data is added to the repository, but that the repository is updated so that typical and useful queries will return different results. This does not necessarily imply that previous statements are retracted, but that, for example, new information is added about previously described entities; or that new entities are retroactively added to past events.

The *SemaGrow Stack* features primarily tested are:

- The ability to take advantage of query feedback to automatically tune source descriptions to evolving datasets. This pertains to the *STRHist* method developed in Task 3.1 (cf. Deliverable 3.1.2i).
- The query execution overheads introduced by collecting query feedback. This pertains to the *SemaGrow Stack* integrated components developed in Task 5.4 (cf. Deliverable 5.4.3).
- The ability to take advantage of source descriptions in optimizing query execution. This pertains to the *Query Decomposition* component developed in Task 3.4 (cf. Deliverable 3.4.2).

In order to achieve this, we need a *data source* and a *query workload* that satisfy the following requirements:

1. The data source is regularly updated and, in particular, updated in a manner that affects the way a query is planned.
2. The URIs of the resources in the database are constructed in a way that it is meaningful to group them by prefix.
3. The query workload is realistic and representative of the *SemaGrow* use cases.
4. The query workload joins data so that the AGRIS database is queried on specific resource URIs, so that the query optimizer can exploit the self-tuning histograms.

In the remainder of this chapter, we will document the experimental setup developed for this test, present and analyse results that inform further work in Tasks 3.1 and Task 3.4, and also specify the *automatic rigorous testing* components developed in Task 5.3 based on these experiments.

2.2 Experimental Setup

This experiment is based on FAO's *AGRIS* bibliographic database on agricultural research and technology (cf. Section 3, Deliverable 2.2). AGRIS provides an annual record of its content since 1975, so that the evolution of the dataset can be reconstructed regardless of the year of publication of the referenced work. Since articles are often added retroactively and not during or immediately after the year of publication, querying for articles published within a given (past) time period yields different results depending on when the query is posed. Most crucially for our purposes, after each annual update the cardinality of query patterns that involve publication year are going to change, and the relevant histograms in the data source descriptions will need to converge to new values. This covers Requirement 1.

Furthermore, AGRIS constructs a unique identifier for each bibliographical entity following the pattern

<two-letter country code><year submitted to AGRIS><increment>

where <year> is represented by two digits up to 1997 and four digits from then onwards. This identifier is used as the value of the dct:identifier property and also to construct the entity's URI under the http://agris.fao.org/aos/records/ namespace. These URIs cover Requirement 2 as different prefixes correspond to national scientific output and have sharply un-uniform cardinality distributions.

The training query workload is based on logs collected by SemaGrow partners AK, recording queries executed as part of their daily workflow over several days. Although not specifically targeted at AGRIS, they are for the most part bibliography queries searching for content that matches thematic criteria, expressed as keywords. In the AGRIS search engine, thematic query terms are searched by joining AGRIS data with AgroVoc labels that link the abstract terms (such as those shown in **Error! Reference source not found.**) to linguistic terms (such as those shown in **Error! Reference source not found.**). Following standard practice in evaluating histogram performance, the evaluation queries are *point queries*, retrieving all properties of a given URI resource.

This workload satisfies Requirement 3 and Requirement 4.

2.3 Experimental Results

This experiment is based on FAO's *AGRIS* bibliographic database on agricultural research and technology (cf. Section 3, Deliverable 2.2). AGRIS provides an annual record of its content since 1975, so that the evolution of the dataset can be reconstructed regardless of the year of publication of the referenced work.

We define a 3-dimensional histogram over subject, predicate and object variables. Subject URIs are represented as strings while predicate URIs are treated as categorical values, since there is always a small number of distinct predicates. Each bucket is composed of a 3-dimensional subject/predicate/object bounding box, a size indicating the number of triples contained in the bucket, and the number of distinct subjects, predicates and objects.

We automatically generated a workload of query feedback as follows: we randomly select a URI in the dataset and construct a prefix by trimming the URI string to a random length. The query feedback record is constructed by selecting the fragment of the dataset that satisfies the aforementioned prefix. We estimate the size of the results of hypothetical queries that select rows that have this particular value. We then measure the average absolute estimation error.

For this experiment we use different versions of AGRIS from 1990 to 1999. The AGRIS size increases linearly over the year as depicted in Figure 1. We start the experiment using an initial histogram that contains the root bucket with statistics a gross average of the size of the repositories and iterate over the years using as initial histogram the trained histogram of the previous. For each AGRIS repository we generate a workload of 100 training queries using the aforementioned method. We measure the average absolute error of the 1% of the publications occurring in the specific repository. We use two histogram configurations (a) with unlimited number of buckets and (b) with limit of 100 bucket and allowing parent-child merges to occur.

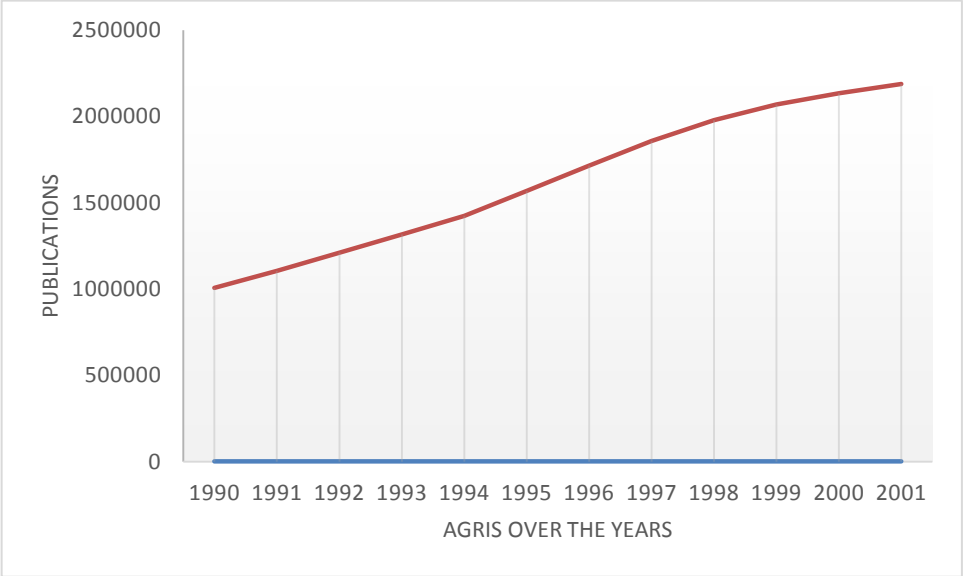

Figure 1 Number of AGRIS publications over the years

2.4 Automatic Rigorous Testing

This experimental setup will be the basis for testing components that periodically check histogram estimations against actual query results in order to produce reports about how well STRHist applies to a given deployment. This will be reported in D5.3.2 *Automatic Rigorous Testing* (M30).

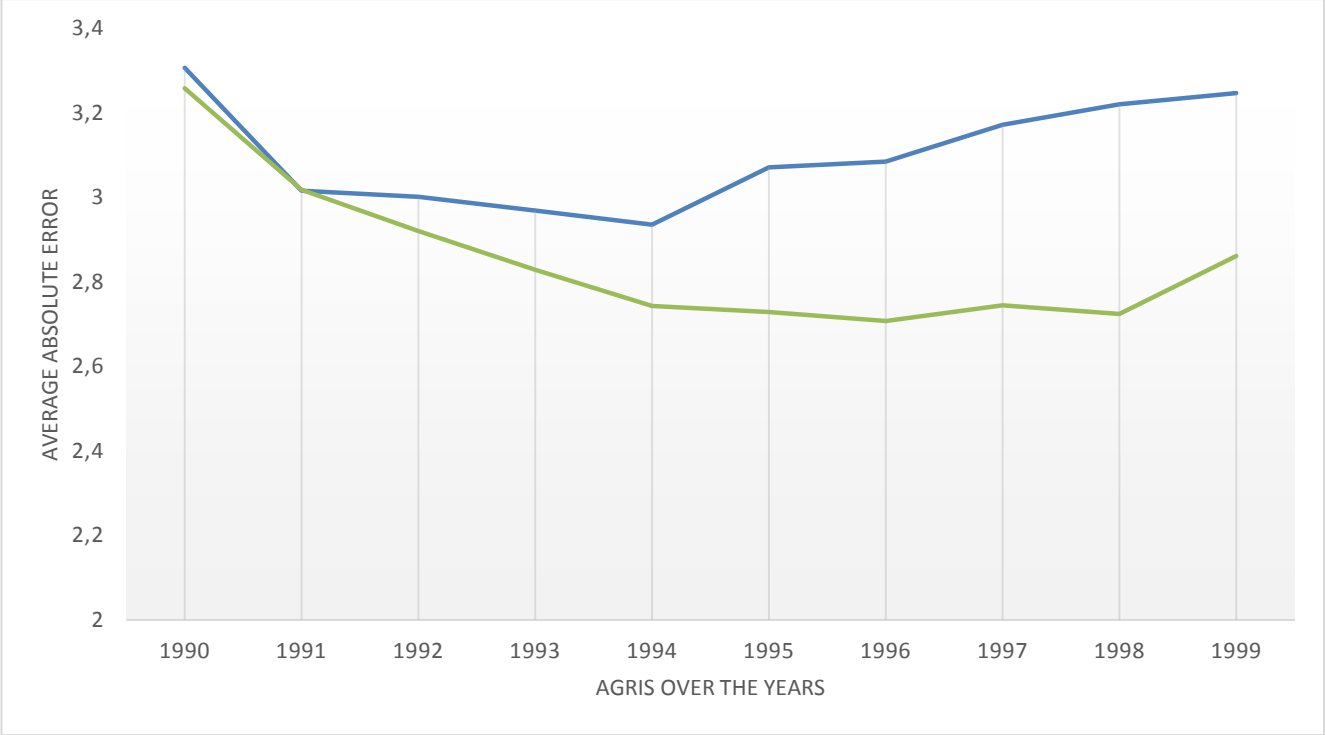

Figure 2 Comparison of the average absolute error using unlimited memory (green) and using limit of 100 buckets.

3. EXPERIMENTS ON QUERY EXECUTION PLANNING

3.1 Objectives and Requirements

In this experiment, we measure the effect on querying performance of having access to and taking advantage of, instance-level metadata.

In order to achieve this, we need a setup that:

1. Compares the SemaGrow Stack against two different query execution engines: (a) one that is algorithmically sophisticated in using histograms to optimize query execution; and (b) one that is a mature, well-engineered system, even if it implements a simpler query execution planner.
2. Uses a query workload that tests different aspects of federated querying.

In the remainder of this chapter, we will document the experimental setup developed for this test, present and analyse results that inform further work in Task 3.4.

3.2 Experimental Setup

FedBench[1] is commonly used to evaluate the performance of the SPARQL query federation systems. The benchmark is explicitly designed to represent SPARQL query federation on a real-world datasets. The FedBench suite contains two domains: the Cross-domain (CD) and Life Science (LS). The benchmark queries resemble typical requests on these datasets and their structure ranges from simple star and chain queries to more complex graph patterns. **Error! Reference source not found.** presents the various query characteristics.

We will evaluate the performance of SemaGrow against FedBench since it has been studied thoroughly and many SPARQL query federation system has been evaluated extensively. We compare SemaGrow Stack with FedX[2] and SPLENDID[3]. FedX and SPLENDID are the ones that outperform most of the existing state of the art systems.[4] These two systems employ different techniques for query processing and execution: FedX uses ASK queries to select source that potentially contain triples of the given query. The query decomposition algorithm is greedy and does not use any kind of statistics. Triple patterns are ordered using heuristics based on the number and type of variables occurring in the query. SPLENDID, on the other hand, is closer to SemaGrow ideas. In order to select the appropriate data sources, consults dataset metadata described in VoID and then uses ASK. SPLENDID employs dynamic programming for decomposing the query and make use of the cardinality statistics maintained in the VoID metadata.

In the evaluation setting of this experimental setup we assume the SPARQL endpoints are reliable and fast. Thus, the endpoints are deployed locally in different Virtuoso servers. All the federation engines can access the data sources via the SPARQL protocol. The statistics of each dataset used in the evaluation is presented in **Error! Reference source not found.**. SPLENDID and SemaGrow use VoID metadata that are generated by extracting statistics directly from the actual data. The VoID descriptions referred only to property and class partitions of the corresponding datasets. The federation in each case contain only the data sources that can potentially contribute to the queries of the collection.

This experimental setup satisfies both requirements.

3.3 Experimental Results

Error! Reference source not found. shows the query execution time (first run and average) for both query collections. FedX outperforms SemaGrow and SPLENDID in most of the queries. The main reasons for FedX's small execution time is the efficient execution engine and the source selection. In the latter, FedX uses ASK queries to probe the data sources for triple existence, which is quite efficient since data sources are responsive and close to the federator. Moreover, since only

[1] P. Haase, T. Mathäß, and M. Ziller, "An evaluation of approaches to federated query processing over linked data," Proc. 6th Int. Conf. Semant. Syst. - I-SEMANTICS '10, p. 1, 2010.

[2] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt, "FedX: a federation layer for distributed query processing on linked open data," Semanic Web Res. Appl., pp. 481–486, 2011.

[3] O. Görlitz and S. Staab, "SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions.," COLD, vol. 782, 2011.

[4] M. Saleem, Y. Khan, A. Hasnain, I. Ermilov, and A. N. Ngomo, "A Fine-Grained Evaluation of SPARQL Endpoint Federation Systems," vol. 1, pp. 1–5, 2009.

the data source that participate in the federation is limited, the technique of ASK queries can be viable. In several queries SPLENDID and FedX yield similar execution plans. Thus, we conclude that the difference of the query execution time is mainly due to the efficiency of FedX's execution engine. A characteristic example is query LS7 in which all systems yield the same execution plan.

Both SemaGrow and SPLENDID exhibit similar behaviour. In some cases SPLENDID perform better that SemaGrow. This is mainly due to the fact that SPLENDID uses heuristics to produce more efficient plan. For example, SPLENDID yield a different plan in queries CD2, CD3. However, these heuristics are not safe in terms of completeness of the query. For example, SPLENDID misses certain answers due to that heuristic in query CD7 while SemaGrow returns all the available answers.

Table 1 Query characteristics

Collection	Query	Type	# Triple Patterns	# Results
Cross Domain	CD1	Complex	3	90
	CD2	Star	3	1
	CD3	Chain-Star	5	2
	CD4	Complex	5	1
	CD5	Chain-Star	4	2
	CD6	Chain-Star	4	11
	CD7	Chain-Star	4	1
Life Sciences	LS1	Complex	2	1159
	LS2	Chain	3	333
	LS3	Chain-Star	5	9054
	LS4	Complex	7	3
	LS5	Complex	6	393
	LS6	Complex	5	28
	LS7	Complex	5	144

Table 2 Dataset statistics used in FedBench collection of queries.

Collection	Dataset	#triples	#subjects	#predicates	#objects	#types	#links
Cross Domain	Dbpedia subset	43.6M	9.50M	1063	13.6M	248	61.5k
	GeoNames	108M	7.48M	26	35.8M	1	118k
	LinkedMDB	6.15M	694k	222	2.05M	53	63.1k
	Jamendo	1.05M	336k	26	441k	11	1.7k
	New York Times	335k	21.7k	36	192k	2	31.7k
	SW Dog Food	104k	12K	118	37.5k	103	1.6k
Life Sciences	KEGG	1.09M	34.3k	21	939k	4	30k
	CheBI	7.33M	50.5k	28	772k	1	
	Drugbank	767k	19.7k	119	276k	8	9.5k
	Dbpedia subset	43.6M	9.50M	1063	13.6M	248	61.5k

There are queries that use extensively the predicates rdf:type and owl:sameAs. These predicates are encountered frequently in every data sources, obliging SemaGrow to ask every data sources. On the other hand, FedX and SPLENDID use ASK queries to validate the existence of matching triples before decomposing the query. Moreover, the generated VoID metadata contains only information about the properties of the dataset. It is expected though that SemaGrow, in the presence of URI prefixes, will perform better.

An interesting case is query LS6, in which SemaGrow outperforms both SPLENDID and FedX by an order of magnitude. The main reason is that SemaGrow yield an execution plan of better quality. The reason for that is twofold: first, SemaGrow use the dataset statistics and second the query decomposer guarantees the optimality of the plan with respect to the SemaGrow's cost function. On the other hand, FedX selects a suboptimal plan due to its greedy query decomposer, resulting in a vast difference of cost. This result is a concrete example that the optimality of the execution plan has a crucial role in the overall performance of the systems.

3.4 Automatic Rigorous Testing

This experimental setup does not contribute to D5.3.2 *Automatic Rigorous Testing* and is only used to evaluate the performance of the Semagrow Stack.

Table 3 Comparison of First Run and Average Overall Execution Time for SemaGrow, FedX and SPLENDID

		First Run Execution Time			Average Execution Time		
Collection	**Query**	**SemaGrow**	**FedX**	**SPLENDID**	**SemaGrow**	**FedX**	**SPLENDID**
Cross Domain	CD1	N/A	158	446	N/A	55	207
	CD2	552	47	126	243	26	70
	CD3	21419	111	190	15334	61	123
	CD4	1950	87	176	1579	56	94
	CD5	630	61	299	509	45	147
	CD6	9027	627	10480	8033	538	9258
	CD7	9287	629	2254	8649	567	1879
Life Sciences	LS1	N/A	219	351	N/A	77	144
	LS2	N/A	107	415	N/A	50	273
	LS3	N/A	8954	725	N/A	9974	560
	LS4	7400	64	182	6338	256	127
	LS5	35545	2603	N/A	33851	4053	N/A
	LS6	3226	139765	11504	3125	123528	7473
	LS7	15781	2307	20382	15696	3231	19458