Query Translator for Factorized RDF Data

Chandan Acharya, Devendra Hupri, Rakesh Lagare

RWTH Aachen
{chandan.acharya,devendra.hupri,rakesh.lagare}@rwth-aachen.de

Abstract. The Semantic Sensor Network ontology is an ontology used to describe sensors and their observations. Sensors are the major source of data available on web today. Semantic representation of this sensor data contains repeated values which is also a reason for high volume of the data. We use SR Benchmark queries to query this data. By factorizing the data we can reduce the volume of the semantic sensor data. The goal of the lab project is to develop a Query Translator for factorized semantic sensor data. The translator would translate the SR Benchmark queries to translated queries which will be used against the factorized semantic sensor data

1 Introduction

Sensors are now used in applications ranging from meteorology to medial care to environmental monitoring to security and surveillance. The growth in number of applications and sensors is accompanied by growth in the volume of data and the heterogeneity of devices and data formats. Semantics can help users to manage and query sensors and data. Semantic sensor data is represented in the form of Resource Description Framework. The Semantic sensor data contained large number of repeated values which resulted in large volume of the data. In order to overcome this Factorization of the Semantic sensor data was proposed. SR Benchmark queries are used to query the original semantic sensor data. These queries cannot be used against the factorized semantic sensor data due to change in the structure of the data. Hence there is a need for a translator which would translate the SR Benchmark queries so that we can use them to query the factorized semantic sensor data.
Since the factorized semantic sensor data does not contain repeated values the querying of factorized semantic sensor data is believed to consume less time compared to the time taken to query an original semantic sensor data. Considering the query translator as a unit, the translator receives a SR Benchmark SPARQL Query as input. The translator applies transformation rules to the query and outputs a translated SPARQL query, This SPARQL query we get can be used to evaluate the factorized semantic sensor R data. The translator is later embedded within the SPARQL Engine.

2 Background

Semantic Sensor Data The encoding of sensor descriptions and sensor observation data with Semantic Web languages enables more expressive representation, advanced access, and formal analysis of sensor resources. A semantic sensor network will allow the network, its sensors and the resulting data to be organized, installed and managed, queried, understood and controlled through high-level specifications.The World Wide Web Consortium (W3C) initiated the Semantic Sensor Networks Incubator Group (SSN-XG) to develop the Semantic Sensor Network (SSN) ontology, intended to model sensor devices, systems, processes, and observations. The Semantic Sensor Network (SSN) ontology enables expressive representation of sensors, sensor observations, and knowledge of the environment[1].
SR Benchmark SR Bench is a Streaming RDF/SPARQL Benchmark which aims at assessing the abilities of streaming RDF/SPARQL engines in dealing with important features from Semantic Web research areas combined in one real-world application scenario.SR Bench currently defines 17 queries. The queries below are implemented according to the syntax specified by the systems. These queries are used to evaluate the original semantic sensor data[2].

3 Related Work

For the optimization of the semantic sensor data factorization has been utilized. The representation of the semantic sensor data by applying logical axioms of relational algebra has been proposed[5,6]. There has been research on different query engines that can be used to query the semantic sensor data. Integration of the translator into the query engine is a new approach where still not enough finding has been made.

4 Proposed Approach

We propose a query translator which would translate the standard SR Bench SPARQL queries to queries which can be used against the factorized semantic sensor data. The translator will take a SPARQL Query as input and then perform the transformation over it and produce a translated query as output. This translated query that we get can be used to query the factorized semantic sensor data.

4.1 Problem Statement

SR Benchmark queries are used to evaluate the original Semantic Sensor RDF data. We need a translator which would translate the SR Benchmark SPARQL queries. The translated queries should take equal or less time to evaluate the factorized semantic sensor RDF data when compared to the time taken by SR Benchmark SPARQL queries for querying the original semantic sensor RDF data.

4.2 Proposed Solution

For the implementation of the translator which would translate the SR Benchmark Queries to the queries that can be used with factorized semantic sensor RDF we make use of a Query Reformulation Algorithm from [3].This algorithm gives a step by step approach on how to translate an original SPARQL query to a translated query. The steps are performed on each set of a triple pattern of input query. The input query may have multiple triple patterns.

Algorithm 1 Query Reformulation Algorithm

Input: A BGP SPARQL query Q
Output: A query plan for Q', the reformulation of query Q

```
1  Reformulate Measurements if (?m :value ?v) 2 BGP then
2  Add f (?obs :result ?m), (?oM:hasObservation ?obs), (?oM:result ?mM), (?mM
   :value ?v)g to BGP'
3  if (?m :uom ?u) 2 BGP then
4  Add f (?obs :result ?m), (?oM :hasObservation ?obs), (?oM:result ?mM), (?mM
   :uom ?u)g to BGP'
5  Reformulate Observations if (?o:observationProperty ?p) 2 BGP then
6  Add f (?oM :hasObservation ?o), (?oM:observationProperty ?p))g to BGP'
7  if (?o :procedure ?u) 2 BGP then
8  Add f (?oM :hasObservation ?o), (oM:procedure ?u))g to BGP'
9  if (?o rdf:type ?t) 2 BGP then
10 Add f (?oM :hasObservation ?o), (?oM rdf:type ?t))g to BGP'
11 return plan(Q')
```

4.3 Implementation

We make use of the query reformation algorithm proposed in [3].Translator was implemented on a Linux machine with a CPU Intel I7 2.3GHz and 8GB RAM DDR3. Eclipse was used as development environment.
To implement the above algorithm, Apache Jena libraries were imported. Queryfactory[7] was used to parse the original SPARQL query. ElementWalker was used to walk through the triples. NodeFactory[8] was used to create dynamic nodes as per the transformation rules. Then a triple path will added to the transformed query using the dynamically created nodes. The reference SPARQL Queries were taken from SRBench and the datasets from Knoesis[4].

5 Use Case

Name: Query Translation
Primary Actor: A SPARQL Query Engine

Precondition: The SPARQL query from SR Benchmark must be given as input to the engine.
Description: The translator applies the algorithm and translates the given SR Benchmark SPARQL Query so that it can be run against the factorized RDF data. We learn and evaluate the query execution time of the translated query against the factorized semantic sensor data and the result comparison of the original query against the original semantic sensor data with the translated query against the factorized semantic sensor data.We try to answer below two questions

1 Translated queries against factorized semantic sensor data can speed up the execution time?

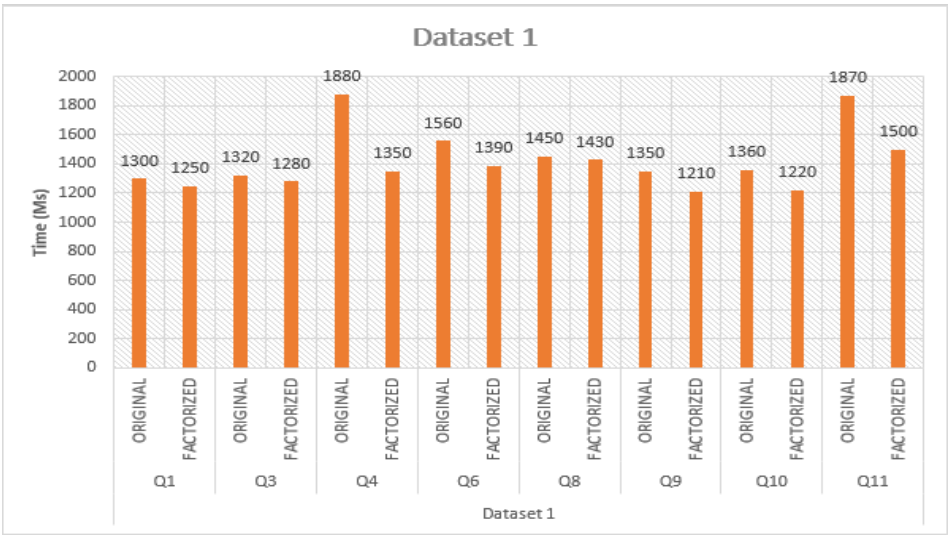
2 Is the result of queries against factorized semantic sensor data different from the result of original query against the original semantic sensor data?

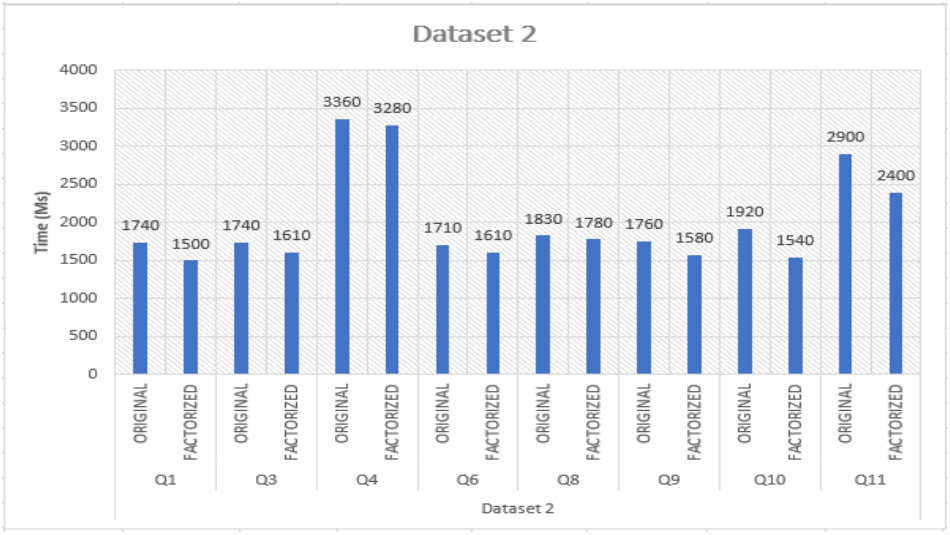
Datasets: Evaluation is conducted on three sensor datasets[4]. These datasets comprise observations of different climate phenomena e.g., temperature, visibility, precipitation, wind speed, and humidity, during the hurricane and blizzard seasons in the United States in the years 2003, 2004, and 2005.

Queries: We make use of SR Benchmark queries for querying the datasets. These queries are given as input to the translator. These queries contain the STREAM clause, ASK queries, aggregate modifiers like AVG, GROUP BY, and HAVING clauses. Queries range from simple queries with one triple pattern to complex queries with multiple triple pattern.

5.1 Efficiency of Translated Queries

To answer questions 1 and 2, we analyze the time efficiency of queries generated by algorithm 1, and the result count obtained after evaluating the queries against the semantic sensor data. Figure 1 reports on the query execution time (milliseconds. log-scale) , while Figure 2 depicts the observed execution query result count for both original and factorized datasets.


Dataset 1
Time (Ms)
2000
1800
1600
1400
1200
1000
800
600
400
200
0
1300
1250
1320
1280
1880
1350
1560
1390
1450
1430
1350
1210
1360
1220
1870
1500
ORIGINAL
FACTORIZED
Q1
Q3
Q4
Q6
Q8
Q9
Q10
Q11
Dataset 1


Dataset 2
Time (Ms)
4000
3500
3000
2500
2000
1500
1000
500
0
1740
1500
1740
1610
3360
3280
1710
1610
1830
1780
1760
1580
1920
1540
2900
2400
ORIGINAL
FACTORIZED
Q1
Q3
Q4
Q6
Q8
Q9
Q10
Q11
Dataset 2

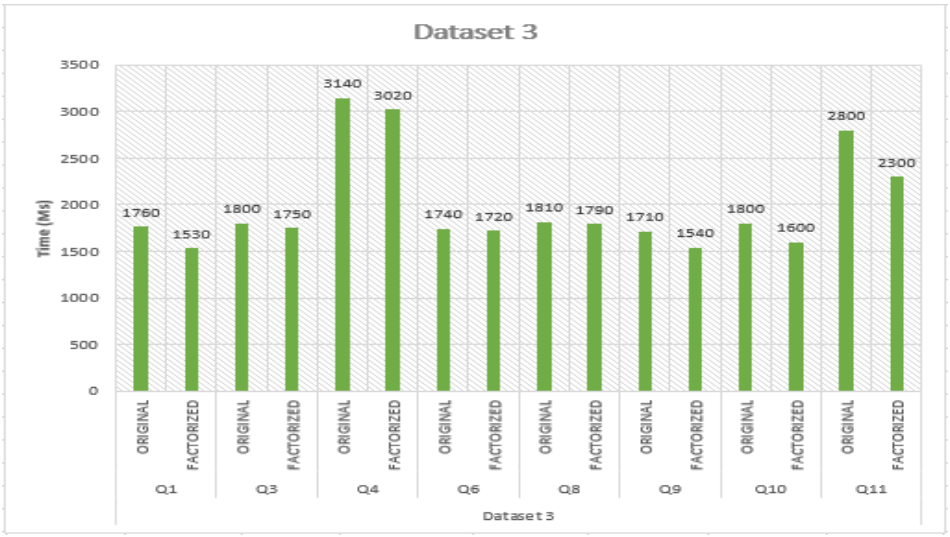


Figure 3: Graphical comparison of Original and Factorized Dataset (In terms of query execution times)

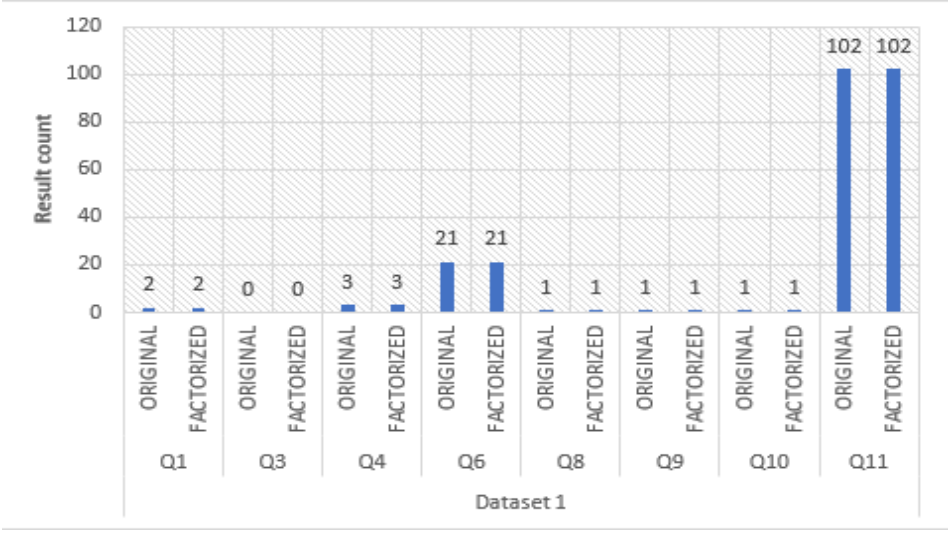


Figure 4: Graphical comparison of Original and Factorized dataset(In terms of query result counts)

6 Conclusions and Future Work

In this paper we present an approach to build a query translator for translating the SR Benchmark SPARQL queries to queries that can be used to evaluate the factorized semantic sensor RDF data. For this we make use of query reformulation algorithm. We also try to find out the time and the result efficiency of the translator by comparing it with the results that we get by querying the

original semantic sensor data. The results that we get from the validation proves that running the translated queries against the factorized semantic sensor data takes less time and also the results obtained from the execution matches with the original one. The present implementation does not support the translation of the query which contains nested body.

7 References

1 https://www.w3.org/2005/Incubator/ssn/
2 https://www.w3.org/wiki/SRBench
3 Farah Karim, Mohammed Najib Mami, Maria-Esther Vidal and Sren Auer: Large-scale Storage and Query Processing for Semantic Sensor Data
4 http://wiki.knoesis.org/index.php/LinkedSensorData
5 N. Bakibayev, T. Kocisky, D. Olteanu, and J. Zavodny. Aggregation and ordering in factorised databases. PVLDB, 6(14):1990-2001, 2013.
6 N. Bakibayev, D. Olteanu, and J. Zavodny. FDB: A query engine for factorised relational databases. PVLDB, 5(11):1232-1243, 2012.
7 https://jena.apache.org/documentation/javadoc/arq/org/apache/jena/query/QueryFactory.html
8 https://jena.apache.org/documentation/javadoc/jena/org/apache/jena/graph/NodeFactory.html