Analytics and Visualization over Semantic RDF Graphs

Maria-Evangelia Papadaki, Yannis Tzitzikas, Nicolas Spyrouto
ISIP, 9-10 May, 2019, Heraklion
OUTLINE

- Motivation
- HIFUN: A High Level Functional Query Language for Big Data Analytics
- Application of HIFUN to RDF Graphs
- Evaluating HIFUN Queries in SPARQL
- Concluding Remarks
- Future Work
MOTIVATION
Motivation (1/2)

We live in the era of data explosion. At the same time, more and more data sources are being produced as linked data, using the Resource Description Format (RDF).

However, the exploitation and the application of analytics over RDF data, is not so straightforward, since its structure is not so simple:

- different resources may have different sets of properties,
- resources may or may not have types,
- multi-value properties may exist
Even though, many analytical tools have been developed, they:

- focus on relational data
- work with a single homogeneous data set
- may demand a deep knowledge of specific query languages
- do not offer flexible choices of dimension, measurement, and aggregation
- neither support multiple central origins, nor RDF semantics
What do we need?

- An analytical **tool** that would:
  - be applied over one or multiple linked data sets,
  - **not** demand any **programming skill**,
  - **visualize** data **intuitively** and will support **collaborative** data exploration.

What are we doing?

- Currently, we are investigating an approach based on:
  - a **high level query language**, for **analytics** over RDF graphs [N. Spyratos et al. 2018], and
  - an interactive 3D **system** [M.E. Papadaki et al. 2018] for **visualizing linked data**.
HIFUN: A HIGH LEVEL FUNCTIONAL QUERY LANGUAGE FOR BIG DATA ANALYTICS
**Definition of HiFun (1/3)**

**HIFUN** [N. Spyratos et al. 2018], is a high level functional query language for defining analytic queries over big data sets, independently of how these queries are evaluated.

- It can be applied over a data set that is structured or unstructured, homogeneous or heterogeneous, centrally stored or distributed.
Definition of HiFun (2/3)

- **Data set assumptions**
  - consists of *uniquely identified items*

- has a set of *attributes*,
  - an attribute is *viewed as a function* from the data set to some domain of values: `nameOfAttribute: D → value`
    - e.g. date: `D → String`
Definition of HiFun (3/3)

- The set of attributes (direct or derived) that analysts are interested in is called context and D the origin of it.

The attributes that their values appear on the invoices are called direct, while those that can be computed from them are called derived (e.g. the attributes m and y can be extracted from d).
Definition of Hifun Query

A query \( Q \) in HiFun is viewed as an ordered triple, such that \( g \) and \( m \) are attributes of the data set \( D \), and \( \text{op} \) is an aggregate operation applicable on \( m \)-values.

\[ Q = (g, m, \text{op}) \]
a) **group the items** of the data set \(D\) using the values of \(g\) (i.e. items with the same \(g\)-value \(g_i\) are grouped together)
HIFUN: Evaluation of Analytic Query (2/4)

b) in each group of the items created, **extract** from D the **m-value of each item** in the group

\[
Q = (g, m, op)
\]

**Ordered triple**

measuring function
HIFUN: Evaluation of Analytic Query (3/4)

c) aggregate the m-values obtained in each group to obtain a single value $v_i$

$$Q = (g, m, \text{op})$$
HIFUN: Evaluation of Analytic Query (4/4)

c) aggregate the m-values obtained in each group to obtain a single value $v_i$

Ordered triple

$Q = (g, m, \text{op})$

aggregate operation

Such a query can be evaluated easily using either SQL or Map-Reduce.
Application of HiFun to Relational Data (1/5)

Example: suppose that, we would like to know the total quantities of products, delivered to each branch, for the following relational data, using HiFun.

<table>
<thead>
<tr>
<th>D</th>
<th>Date</th>
<th>Month</th>
<th>Year</th>
<th>Branch</th>
<th>Product</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoiceID1</td>
<td>date1</td>
<td>month1</td>
<td>year1</td>
<td>branch1</td>
<td>product1</td>
<td>100</td>
</tr>
<tr>
<td>invoiceID2</td>
<td>date2</td>
<td>month2</td>
<td>year2</td>
<td>branch1</td>
<td>product2</td>
<td>200</td>
</tr>
<tr>
<td>invoiceID3</td>
<td>date3</td>
<td>month3</td>
<td>year3</td>
<td>branch2</td>
<td>product3</td>
<td>300</td>
</tr>
<tr>
<td>invoiceID4</td>
<td>date4</td>
<td>month4</td>
<td>year4</td>
<td>branch3</td>
<td>product4</td>
<td>400</td>
</tr>
</tbody>
</table>
Application of HiFun to Relational Data (2/5)

Each invoice would be a uniquely identified item, that has a set of attributes, each of which could be seen as a Hifun function from dataset D to some domain of values.

<table>
<thead>
<tr>
<th>invoiceID1</th>
<th>date1</th>
<th>month1</th>
<th>year1</th>
<th>branch1</th>
<th>product1</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoiceID2</td>
<td>date2</td>
<td>month2</td>
<td>year2</td>
<td>branch1</td>
<td>product2</td>
<td>200</td>
</tr>
<tr>
<td>invoiceID3</td>
<td>date3</td>
<td>month3</td>
<td>year3</td>
<td>branch2</td>
<td>product3</td>
<td>300</td>
</tr>
<tr>
<td>invoiceID4</td>
<td>date4</td>
<td>month4</td>
<td>year4</td>
<td>branch3</td>
<td>product4</td>
<td>400</td>
</tr>
</tbody>
</table>
Application of HiFun to Relational Data (3/5)

a) **group** together all the **invoices** referring to the same **branch**

1. **Grouping (b)**

branch1: ID1, ID2
branch2: ID3
branch3: ID4

\[
Q = (b, m, op)
\]
Application of HiFun to Relational Data (4/5)

b) find the quantity corresponding to each invoice in the group

1. Grouping (b)  2. Measuring (q)

branch1: ID1, ID2  branch1: 100, 200
branch2: ID3  branch2: 300
branch3: ID4  branch3: 400

\[ Q = (b, q, op) \]
Application of HiFun to Relational Data (5/5)

- **c)** in each group of the previous step, we sum up the quantities found

<table>
<thead>
<tr>
<th>1. Grouping (b)</th>
<th>2. Measuring (q)</th>
<th>3. Reduction (sum)</th>
</tr>
</thead>
<tbody>
<tr>
<td>branch1: ID1, ID2</td>
<td>branch1: 100, 200</td>
<td>branch1: 300</td>
</tr>
<tr>
<td>branch2: ID3</td>
<td>branch2: 300</td>
<td>branch2: 300</td>
</tr>
<tr>
<td>branch3: ID4</td>
<td>branch3: 400</td>
<td>branch3: 400</td>
</tr>
</tbody>
</table>

\[ Q = (b, q, \text{sum}) \]
APPLICATION OF HIFUN TO SEMANTIC DATA
Application of HIFUN to semantic data (1/8)

What if the data was represented as an RDF graph?

Each property would correspond to a Hifun attribute, having as source the domain of that property and target the range of it.
Application of HIFUN to semantic data (2/8)

In fact, one could also derive easily attributes from literals (e.g. we could extract from date the attributes of “month” and “year”).

```xml
<ex:Invoice
  ex:hasDate="2019-05-09"
  ex:ID4="400"

  ex:Branch ex:Product
  ex:delivers

  ex:branch3 ex:product4
  ex:takePlaceAt

  xsd:date ex:inQuantity
  xsd:Integer

  2019-05-09 ex:hasDate
```
Application of HIFUN to semantic data (3/8)

Now, imagine that “date” was represented as a blank node. Could HiFun be applied over such data?
Application of HIFUN to semantic data (4/8)

Yes, since the attributes of it would correspond to 1-1 functions.
Application of HIFUN to semantic data (5/8)

For example, date’s functions would be:

hasDate(ID4) = _:b1
hasMonth(_:b1) = month4
hasYear(_:b1) = year4
Application of HIFUN to semantic data (6/8)

But, what if an attribute was not functional (i.e. nationality)?
Application of HIFUN to semantic data (7/8)

Then, such an attribute could still be expressed in Hifun, if the number of its values was finite.
So far we,
- have examined the basics for applying HIFUN over RDF
- have designed and implemented a HIFUN2SPARQL converter and have applied it over RDF data expressed using the RDF Data Cube vocabulary

Issues that worth further research:
- multi-valued attributes
- the interplay with inference
- complex dimension hierarchies
- heterogeneous graphs
EVALUATING HIFUN QUERIES IN SPARQL
Evaluating HIFUN Queries in SPARQL (1/7)

We could encode each Hifun query $Q$ as a SPARQL group-by query over a triple store, as follows:

HIFUN Query $Q = (e, e', op)$

SPARQL Query

Select ?target(e), op(?target(e')) As ?Result
WHERE {..}
GROUP BY ?target(e)
Query: find the total quantities of products, group by branch.

HIFUN Query

$$Q = (b, q, \text{SUM})$$

Results:

<table>
<thead>
<tr>
<th>branch</th>
<th>TOTALS</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;branch1&quot;</td>
<td>300</td>
</tr>
<tr>
<td>&quot;branch3&quot;</td>
<td>400</td>
</tr>
<tr>
<td>&quot;branch2&quot;</td>
<td>300</td>
</tr>
</tbody>
</table>
Evaluating HIFUN Queries in SPARQL (3/7)

Result restricted-query

Suppose now that, we would like to set restrictions to the final results. The query would be formulated, as follows:

HIFUN Query

\[ Q = (e, e', op)/F \]

SPARQL Query

\[
\begin{align*}
\text{Select} & \quad ?\text{target}(e), \quad \text{op}(?\text{target}(e')) \quad \text{As} \\
& \quad ?\text{Result} \\
& \quad \text{WHERE} \{..\} \\
& \quad \text{Group by} \quad ?\text{target}(e) \\
& \quad \text{HAVING}(\text{op}(?\text{target}(e'))\text{op}')
\end{align*}
\]
Evaluating HIFUN Queries in SPARQL (4/7)

Query: find all the branches that received more than 300 products, group by branch.

HIFUN Query
Q = \( \frac{b \cdot q \cdot \text{SUM}}{300} \)

Results:

<table>
<thead>
<tr>
<th>branch</th>
<th>TOTALS</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;branch3&quot;</td>
<td>400</td>
</tr>
</tbody>
</table>

SELECT ?branch SUM(?quantity) AS ?TOTALS
WHERE{
}
GROUP BY ?branch
HAVING(SUM(?quantity) > 300)
Evaluating HIFUN Queries in SPARQL (5/7)

Attribute restricted-query

Suppose now that, we would like to apply restrictions at the level of the attributes and filter the results, internally. The query would be formulated, as follows:

**HIFUN Query**

\[ Q = (e/E, e', \text{op}) \]

**SPARQL Query**

```
Select ?target(e), \text{op(?target(e'))} As ?Result
WHERE {
  FILTER(op(?target(e')) \text{op'})
}
Group by ?target(e)
```
Query: find the total quantities of products that received by branch “branch1”, group by branch.

HIFUN Query
$Q = (b/ \text{"branch1"}, q, \text{SUM})$

SELECT ?branch SUM(?quantity) AS ?TOTALS
WHERE{

FILTER regex((?branch), \text{"branch1"}, \text{"i"})}
GROUP BY ?branch

Results:

<table>
<thead>
<tr>
<th>branch</th>
<th>TOTALS</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;branch1&quot;</td>
<td>300</td>
</tr>
</tbody>
</table>
Evaluating HIFUN Queries in SPARQL (7/7)

Query: find the total quantities of products that received per branch, group by month.

HIFUN Query
Q = (bom, q, SUM)

Results:

<table>
<thead>
<tr>
<th>month</th>
<th>branch</th>
<th>TOTALS</th>
</tr>
</thead>
<tbody>
<tr>
<td>month4</td>
<td>branch3</td>
<td>400</td>
</tr>
<tr>
<td>month2</td>
<td>branch1</td>
<td>200</td>
</tr>
<tr>
<td>month3</td>
<td>branch2</td>
<td>300</td>
</tr>
<tr>
<td>month1</td>
<td>branch1</td>
<td>100</td>
</tr>
</tbody>
</table>

SELECT ?month ?branch (SUM(?quantity) AS ?TOTALS )

WHERE {
  ?ex:ID ex:hasDate ?date .
  ?date ex:month ?month .
  ?ex:ID ex:quantity ?quantity .}

GROUP BY ?month ?branch
VISUALIZATION FOR ANALYTICS
Generally, the results of OLAP usually are visualized using:

- 2D plots (in normal and/or log scale)
- Pie charts
- Histograms and bar charts
- Scatter plots etc.
In our work, we plan to investigate visualizations appropriate also for power law distributions.

Total quantities of products, which were sold per branch, group by product (for 50 branches).

When the data that is visualized is not too many, then the existing visualizations are adequate.
When the number of data is big, the result of is not so informative.

Total quantities of products, which were sold per branch, group by product (for 1000 branches).
VISUALIZATION FOR ANALYTICS (4/4)

Traditional plots vs. the proposed method:

Total quantities of products, which were sold per branch, group by product (for 1000 branches).
MORE ON THE SPIRAL LAYOUT

- The relative sizes are more clear
- The number of values is more evident
- It is like “coiling” the long tail of the normal plot
- Its complexity is linear

Moreover
- The 3rd dimension can be exploited for visualizing an additional function
- An interactive environment allows the user to zoom in any area and explore the space

Ref: [Papadaki et. Al. 2018]

An application that visualizes Only the datasets and their connections Is accessible at www.ics.forth.gr/isl/3DLod/
CONCLUDING REMARKS
Several models, languages and tools have been developed for data analysis.

- However, these tools are not adequate for applying analytics over RDF data.

Also, the existing visualization tools are not so informative, when the number of the results is too big.

So, we are investigating an approach based on:

- a high level query language, for analytics over RDF graphs and
- a 3D interactive system for visualizing linked data.
FUTURE WORK
Future Work

- We could **extend** our application to support:
  - more complex analytical queries over RDF Graphs
  - incremental algorithms
- Also, we could design **more layout algorithms** appropriate for analytics
  - and perhaps provide **visualizations with immersion**, for the **intuitive** and **collaborative** interpretation of the results.
References (1/2)


References (2/2)


Links to tools

3DLod: www.ics.forth.gr/isl/3DLod/
THANKS!

QUESTIONS?