Big (Linked) Semantic Data Compression

Motivation & Challenges

Antonio Fariña, Javier D. Fernández and Miguel A. Martinez-Prieto

3rd KEYSTONE Training School
Keyword search in Big Linked Data

23rd AUGUST 2017

Image: ROMAN AQUEDUCT (SEGOVIA, SPAIN)
Agenda

- **Linked Data & Semantic Technologies**
  - Foundations
  - RDF
  - SPARQL

- **(Some) Open Issues**
  - Linked Data Workflow
  - Big Linked Data Challenges

- **Semantic Data Compression**
  - Why is Semantic Data Redundant?
  - Compression Approaches
  - Achievements & Challenges
Big (Linked) Semantic Data Compression

Linked Data & Semantic Technologies

- Foundations
- RDF
- SPARQL
Linked Data is simply about using the Web to create typed links between data from different sources.
Linked Data refers to a set of best practices for publishing and connecting data on the Web.

These best practices have been adopted by an increasing number of data providers, leading to the creation of a global data space:

- Data are machine-readable.
- Data meaning is explicitly defined.
- Data are linked from/to external datasets.

The resulting data network connects data from different domains:

- Publications, movies, multimedia, government data, statistical data, etc.
Linked Data Principles

1. Use **URIs as names** for entities.

2. Use **HTTP URIs** so that people can look up those names.

3. When someone looks up a URI, provide useful information, using **standards** (e.g. RDF, SPARQL).

4. Include **links to other URIs**, so that they can discover more things.
#1 URIs as names

Names must ensure that any data entity has its own identity in the global Linked Data space.

- Human conventions are not effective to name data entities:
  - They are not universal → *ambiguity*.

- The use of **URIs** (*Universal Resource Identifier*) enables any real-world entity to be identified at universal scale:

  http://example.org/person/homer-simpson

  http://example.org/person/homer-simpson-guy

  What is his name? **“Homer Simpson”**
Dereferenceable URIs ensure the corresponding entity descriptions to be retrieved when an HTTP URI is accessed (via HTTP client).
#3 Standards

- Many and varied stakeholders coexist within the Linked Data ecosystem...
  - Data providers from diverse domains (economy, bioinformatics, multimedia...).
  - Application developers.
  - End-users...

- ... but all of them “must speak the same languages” for effective understanding.

- Standardized semantic technologies:
  - **URIs** for naming.
  - **Serialization formats** (*XML*, *N3*, *Turtle*, *HDT*...) for data storage.
  - **RDF** for data modelling and exchange.
  - **SPARQL** for RDF querying.
  - ...
#4 Links to Other URIs

- Data entities are individually described:
  - A particular **HTTP URI** is assigned as name.
  - Its **features** are stated.

- Linking two URIs establishes a particular type of connection between two existing entities:
  - This principle materializes the aim of data integration in Linked Data.

```
@prefix person   : <http://example.org/person/> .
@prefix property : <http://example.org/property/> .
```

```
person:homer-simpson

person:hommer-simpson

property:name

"Homer Simpson"

"Marge Simpson"

property:name

property:address

"742 Evergreen Terrace"

property:address

person:marge-simpson
```
#4 Links to Other URIs

@prefix person : <http://example.org/person/> .
@prefix property : <http://example.org/property/> .
The Web of Linked Data revisits WWW foundations to build a cloud of data-to-data labelled hyperlinks.

- The Web of Linked Data converts raw data into a first-class citizen of the Web:
  - **Data entities** are the **atoms** of the Web of Linked Data.
  - Each entity has its own **identity**.

- Relies on the WWW infrastructure:
  - It uses **HTTP** as communication protocol.
  - Entities are named using **URIs**.

**Knowledge from different fields can be easily integrated and universally shared/exploited using WWW infrastructure.**
The Web of Linked Data (2007 – 2011)

http://lod-cloud.net/
~10K datasets organized into 9 domains which include many and varied knowledge fields.

150B statements, including entity descriptions and (inter/intra-dataset) links between them.

>500 live endpoints serving this data.

http://lod-cloud.net/
http://stats.lod2.eu/
http://sparqles.ai.wu.ac.at/
RDF is a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ...
RDF Basics

- RDF is a standard model for **data publication, interchange, and consumption** on the Web of Linked Data.

- RDF allows any class of data to be described using a simple **triple** structure:
  - **Subject**: the resource being described.
  - **Predicate**: a property of that resource.
  - **Object**: the value for the corresponding property.

```
http://example.org/person/homer-simpson
http://example.org/property/name
"Homer Simpson"
```

```
http://example.org/person/homer-simpson
http://example.org/property/father
http://example.org/person/abe-simpson
```
RDF Triples

- An RDF triple can be seen as a **labelled directed subgraph** in which subject and object nodes are linked by a particular (predicate) edge:
  - The subject node contains the **URI** which names the resource.
  - The predicate edge labels the relationship using a **URI** whose semantics is described by any vocabulary/ontology.
  - The object node may contain a **URI** or a **Literal** value.

- **RDF links (between entities) also take the form of RDF triples.**
RDF Graphs

@prefix person : <http://example.org/person/> .
@prefix property : <http://example.org/property/> .

"Bart Simpson"

property:name

person:bart-simpson

property:age

10

property:name

location:springfield

"Springfield"

property:name

person:marge-simpson

property:mother

property:father

person:homer-simpson

property:name

"Homer Simpson"

property:father

person:abe-simpson

property:name

"Bart Simpson"

property:age

83

property:name

"Marge Simpson"

property:name

"742 Evergreen Terrace"

property:name

person:abe-simpson

property:father

person:homer-simpson

property:father

person:marge-simpson

property:mother

property:father

person:bart-simpson

property:name

"Homer Simpson"

property:father

person:abe-simpson

property:name

"Bart Simpson"

property:age

83

property:name

"Marge Simpson"

property:name

"742 Evergreen Terrace"
RDF Graphs

- An RDF graph is only a mental model which must be **serialized** for effective storage:
  - Choosing a particular serialization format is an important decision for the most relevant tasks in the Web of Linked Data.
RDF Serialization Formats

@prefix ns0: <http://example.org/property/> .
@prefix person : <http://example.org/person/> .
@prefix property : <http://example.org/property/> .

N3

<http://example.org/person/homer-simpson> ns0:name "Homer Simpson" ;
ns0:address "742 Evergreen Terrace" ;
ns0:father <http://example.org/person/abe-simpson> .

NTriples

<http://example.org/person/homer-simpson> ns0:name "Homer Simpson" .
<http://example.org/person/homer-simpson> ns0:address "742 Evergreen Terrace" .
<http://example.org/person/homer-simpson> ns0:father <http://example.org/person/abe-simpson> .

RDF/XML

<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ns0="http://example.org/property/"
>

<rdf:Description rdf:about="http://example.org/person/homer-simpson">
  <ns0:name>"Homer Simpson"</ns0:name>
  <ns0:address>"742 Evergreen Terrace"</ns0:address>
  <ns0:father rdf:resource="http://example.org/person/abe-simpson"/>
</rdf:Description>

JSON/LD

[{{"@id":"http://example.org/person/abe-simpson"},
{"@id":"http://example.org/person/homer-simpson","http://example.org/p
roperty/name":{{"@value":"Homer Simpson"}},
"http://example.org/property/address":{{"@value":"742 Evergreen Terrace"}},
"http://example.org/property/father":{{"@id":"http://example.org/person/abe-simpson"}}}]

http://www.easyrdf.org/converter
SPARQL is a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format.
### SPARQL Basics

- **SPARQL** is a ***semantic query language***, able to retrieve and manipulate RDF data.

- **SPARQL query resolution** performs ***graph pattern matching***:
  - The objective is to determine all the candidates matching to a pattern query in a large data graph.

- **Triple Patterns** are the most basic SPARQL query:
  - Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable.

- More complex **Graph Patterns** comprise *joins or unions* of multiple triple patterns.
The semantics of “lives in” is captured by the property http://example.org/property/location.

The entity describing “Springfield” is named by the URI http://example.org/location/springfield.

Who lives in Springfield?

```sparql
@prefix person : <http://example.org/person/> .
@prefix property : <http://example.org/property/> .
SELECT ?Who
WHERE { ?Who property:location location:Springfield }
```
SPARQL Querying

Person: Homer Simpson
- Name: Homer Simpson
- Address: 742 Evergreen Terrace
- Father: Abe Simpson
- Age: 83

Person: Abe Simpson
- Name: Abe Simpson
- Address: Marge Simpson
- Father: Homer Simpson
- Age: 40

Person: Marge Simpson
- Name: Marge Simpson
- Address: Springfield
- Husband: Homer Simpson
- Son: Bart Simpson

Person: Bart Simpson
- Name: Bart Simpson
- Address: Springfield
- Mother: Marge Simpson
- Father: Homer Simpson
- Age: 10
Big (Linked) Semantic Data Compression

(Some) Open Issues

- Linked Data Workflow
- Big Linked Data Challenges
Data moves from data providers to end users within the Linked Data ecosystem. It evolves along many stages to consolidate effective results which satisfy end-user requirements.


- **Generation**: data is ingested from real-world systems and transformed into an RDF dataset.

- **Publication**: the RDF dataset is exposed using one or more Linked Data compliant services.

- **Consumption**: data is retrieved from consumers using well-defined interfaces.
RDF Data Generation

- **Data is extracted** from real-world sources:
  - Real-time sources require streams of raw data to be continuously ingested.
  - Data from other sources are ingested in batch.

- **Data cleansing** deals with the transformation of raw data into high-quality data which is finally modeled using RDF:
  - Streaming or batch processing workflows are deployed for data transformation.

- **Data integration** enriches the current data with links from/to other relevant entities in the Linked Data Web:
  - *Where is the entity published? What is its URI?*

- **A new RDF dataset (or and RDF stream) is generated as a result.**
The new RDF dataset must be exposed in the Linked Data Web.

a. **RDF Dump:**
   I. Triples are serialized (and possible compressed) into a valid RDF format.
   II. The resulting dump file is hosted at a web server and registered into a central catalog (e.g. datahub.io) for discovering purposes.

b. **HTTP URIs:**
   I. Triples are stored and indexed (using possible a semantic database).
   II. An interface is exposed for dereferencing URIs.

c. **SPARQL endpoint:**
   I. Triples are stored and indexed using a semantic database.
   II. An SPARQL interface is exposed for querying.

d. **Linked Data fragment:**
   I. Triples are self-indexed using an in-memory RDF engine (HDT).
   II. An LD interface is exposed for (federated) querying.
RDF Data Consumption

Consumers retrieve RDF data to meet their information requirements.

- Consumers can download **RDF dumps** for local use:
  - A (possibly) voluminous file is retrieved and stored locally.
  - Data are usually indexed using a semantic database.

- The remaining mechanisms are used to retrieve pieces of RDF data matching particular needs:
  - Dereferencing **HTTP URIs** allows individual entity descriptions to be retrieved from the corresponding publisher.
  - **SPARQL endpoints** are used to retrieve RDF data which matches complex SPARQL queries in a particular dataset.
  - Finally, **Linked Data Fragments** are accessed to resolve federated queries along the Web of Linked Data.
This processing workflow suffers when Big Linked Data must be managed along it.
Big is not a matter of size... it is a matter of representativity & consumption capacity.
We refer as Big Linked Data (BLD) an RDF dataset that exceeds the capacity of conventional tools used to implement the processing workflow.

- BLD management is a challenge because of its **volume**:
  - Individual datasets are not typically considered BLD [+Gb, ++Gb].
  - Integrated RDF datasets (mashups) can be considered BLD [++Gb, +Tb].
  - The whole Web of Linked Data is obviously BLD [++Tb].

- BLD management is a challenge because of its **velocity**:
  - Data is generated continuously in many scenarios.
  - Data is queried continuously from many users.

- BLD management is a challenge because of other **Vs** (*variability, veracity, validity, vulnerability*...).
Big Linked Data Challenges

- **Generation Challenges:**
  - **Efficient streaming solutions** are required for processing continuous RDF sources (e.g. meteorology or traffic sensors, social networks...).
  - **Scalable batch-processing tools** able to store and manage large amounts of RDF data along the cleansing/integration stages.

- **Publication Challenges:**
  - **Compact serialization formats** for saving storage space (RDF dumps).
  - **Efficient indexes** for performing fast URI lookups and retrieving the corresponding triples (HTTP URIs).
  - **Lightweight indexes and efficient in-memory algorithms** for fast query resolution (SPARQL endpoints).
  - **Linked Data Fragments** is a good example of scalable solution.

- **Consumption Challenges:**
  - **Compact serialization formats** for reducing network latencies.
Big (Linked) Semantic Data Compression

Semantic Data Compression

- Why is Semantic Data Redundant?
- Compression Approaches
- Achievements & Challenges
A **semantic data compressor** provides an alternative method for serializing RDF triples:

- It detects **redundant information** within the dataset and removes it.
- In practice, the resulting encoding **uses (much) less bits** than that required by traditional formats.

**This lecture deals with the use of semantic data compression to address some of the Big Linked Data challenges:**

- Compressed RDF saves **storage space**.
- Compressed RDF requires less bandwidth for **exchanging purposes**.
- Compressed RDF is loaded faster than other traditional solutions, and (possibly) requires less amounts of memory for **triples parsing**.
- Some classes of compressed RDF allows **fast lookups** to be performed with no prior triples decoding.
Why Semantic Data is Redundant?

Data redundancy means that the same information can be encoded using less bits.
Data is redundant when the same information can be encoded using less bits:

- Compression is achieved when redundant data is removed from the original dataset.

Why is semantic data redundant?

- Some facts can be inferred from other ones → semantic redundancy.
- Sequences of symbols are repeated along the dataset → symbolic redundancy.
- The underlying RDF graph structure is redundant by itself → syntactic (structural) redundancy.
Semantic Redundancy

- **Semantic redundancy** occurs when the same meaning can be expressed using less triples:

  - http://example.org/property/age
  - http://www.w3.org/2000/01/rdf-schema#domain
  - http://example.org/classes/person
  - http://example.org/person/bart-simpson
  - http://example.org/property/age
  - 10
  - http://example.org/person/bart-simpson
  - http://www.w3.org/1999/02/22-rdf-syntax-ns#type
  - http://example.org/classes/person

- The third triple is redundant because the entity named as http://example.org/person/bart-simpson is the type http://example.org/classes/person because of the second triple (it provides the age of the person).
Symbolic Redundancy

- **Symbolic redundancy** is due to symbol repetitions in triples:
  - This is the “traditional” source of redundancy removed by universal compressors.

- Symbolic redundancy in RDF is mainly due to **URIs**:
  - URIs tend to be very large strings which share long prefixes, but also has other common infixes or suffixes.

```
http://example.org/class/person
http://example.org/property/address
http://example.org/property/age
http://example.org/property/location
http://example.org/person/abe-simpson
http://example.org/person/bart-simpson
http://example.org/person/homer-simpson
http://example.org/person/marge-simpson
```

- **Literals** also contribute to this redundancy.
Syntactic Redundancy

- **Syntactic redundancy** depends on **how the RDF graph structure is serialized**:
  - For instance, a serialized subset of n triples (which describes the same resource) writes n times the subject value. **It can be abbreviated.**

- ... and also on the underlying graph structure by itself:
  - For instance, resources of the same classes are described using (almost) the same sub-graph structure.

- Syntactic compression also has (many) room for optimization:
  - Structural RDF features must be better understood.
The current state of the art comprises a rich and varied set of compressors for RDF data. These are mainly lossless compressors (because they preserve the original knowledge in the dataset), yet lossy compressors are also emerging.
Compression Approaches

- Semantic compressors proposes many and varied techniques for **lossless compression**:
  - They preserve the original knowledge in the dataset.

- **Lossy compressors** are not in the scope of this lecture because losing knowledge is not acceptable for the Linked Data workflow.

- Three classes of lossless compressors:
  - **Physical compressors** detect and removes symbolic and/or syntactic redundancy from the original dataset.
  - **Logical compressors** detect and removes semantic redundancy from the original dataset.
  - **Hybrid compressors** perform at physical and logical levels.
Physical Compressors

- Physical compressors adapt traditional concepts from data compression to the particular case of RDF compression:
  - **Dictionary compression** removes *symbolic redundancy* and allows triples to be rewritten as 3-IDs tuples (ID graph).
  - **Graph compression** is applied to the ID graph representation and removes different kinds of *syntactic redundancy*.

- Many and varied physical compressors have been proposed:
  - **It reports good space savings** → compressed datasets take (much) less space than their uncompressed counterparts.

- A subset of such compressors (e.g. HDT, k2triples, or RDFCSA) are really **self-indexes**:  
  - Datasets are *compressed* and *indexed* in a single encoding.  
  - **Triples can be searched accessed with no prior decompression.**
Logical Compressors

- Logical compressors look for **redundant triples** (those than can be inferred), and remove them from the dataset.

- The resulting dataset encoding includes two components:
  - The **canonical subgraph** which organizes all canonical triples.
  - The **set of inference rules** which must be applied to recover “redundant triples”.

- Different rule-based algorithms have been proposed to obtain canonical subgraphs.

- Compression effectiveness is less competitive than for physical compressors.
Hybrid Compressors

- Hybrid compressors combine the best of both worlds:
  - Consider different strategies to compact the graph by deleting semantic redundancy at **logical level**.
  - Detect and remove syntactic/symbolic redundancy at **physical level** (dictionary + graph compression).

- These topic has not been much explored yet... but it is (maybe) the **most promissory**:
  - Canonical subgraphs are smaller than their original counterparts → **less space requirements**.
  - Implicit triples can be efficiently accessed by applying inference rules → **better overall search performance**.

- HDT++ or k2triples++ report competitive (preliminary) space/time tradeoffs.
RDF compression is a mature field of research, but the current state of the art has many room for optimization.
RDF compression is a mature field of research:
  - **We are working on this field from 2009.**

HDT is W3C Member Submission from 2011:
  - It is a *de facto* standard for publishing compressed RDF datasets (e.g. *LOD Laundromat*).
  - It is the RDF engine of *Linked Data Fragments* and has been adopted by many other tools in the Semantic Web community.
  - **HDT+Jena** allows in-memory SPARQL resolution to be efficiently performed.

K2triples and RDFCSA are the state of the art for RDF compression and SPARQL triples pattern resolution.

We have recently compressed the Linked Data Web in a single dataset: *LOD-a-lot.*
RDF Compression Challenges

- RDF compression is yet underexploited:
  - Managing compressed RDF allows the memory footprint to be reduced, improving **scalability**.
  - Self-indexes can be adopted by semantic databases to improve **query performance**.
  - A full-compressed triplestore has not been yet released, but we currently working on it.

- Although our techniques has room for optimization!

- Some other semantic applications can be improved using compression:
  - RDF archives, Linked Closed Data...
Bibliography


Let’s the lecture continues...

Basics of Data Compression