LOD is all about evolution

Querying and Managing evolving Linked Open Data

Javier D. Fernández

Drift-a-LOD’17

Special thanks to Axel Polleres for his input
About me:

- since 2015 @WU, Inst. for Information Business
  - Research interest: Semantic Web, Open Data, Big (Semantic) Data Management, Databases, Data Compression, Privacy and Security

- https://www.wu.ac.at/en/infobiz/team/fernandez/
General agenda

- Monitoring Evolution and Archiving
- Archiving the Web of Data
- Representing and querying evolving semantic data
- Open Data evolution
Why evolution matters
(Creationists: please ignore this slide...)

- Monitoring evolution is relevant

![Global temperature change (1850–2016)](image)

![Google Trends graph](image)
Evolution matters

- Changes tell us “something”
- Uncertain information
- Validity of the information

From Wikipedia, the free encyclopedia

Donald Trump: Difference between revisions

In April 2011, Trump questioned President Barack Obama’s comments on US citizenship conspiracy theories-proof-of-citizenship-referraltomapost7717-today’snewsWatsonTrump speaking at the 2011[C]onservative Political Action Conference[c] (CPAC) in[Numbered Hanch, Maryland]].

In April 2011, Trump questioned President Barack Obama’s comments on US citizenship conspiracy theories-proof-of-citizenship-referraltomapost7717-today’snewsWatsonTrump speaking at the 2011[C]onservative Political Action Conference[c] (CPAC) in[Numbered Hanch, Maryland]].

In April 2011, Trump questioned President Barack Obama’s comments on US citizenship conspiracy theories-proof-of-citizenship-referraltomapost7717-today’snewsWatsonTrump speaking at the 2011[C]onservative Political Action Conference[c] (CPAC) in[Numbered Hanch, Maryland]].

In April 2011, Trump questioned President Barack Obama’s comments on US citizenship conspiracy theories-proof-of-citizenship-referraltomapost7717-today’snewsWatsonTrump speaking at the 2011[C]onservative Political Action Conference[c] (CPAC) in[Numbered Hanch, Maryland]].

In April 2011, Trump questioned President Barack Obama’s comments on US citizenship conspiracy theories-proof-of-citizenship-referraltomapost7717-today’snewsWatsonTrump speaking at the 2011[C]onservative Political Action Conference[c] (CPAC) in[Numbered Hanch, Maryland]].
Preservation matters

- Web archives: Common Crawl, Internet Memory, Internet Archive, ...
Time-based access matters

- The Memento protocol
  RFC 7089

Follow your nose
(HTTP content negotiation with datetime)

Batch discovery
(list of URIs of Mementos of the Original Resource)

But...
Challenges (Web archives)

- **Poor granularity** ("some" snapshots)
- **Aggregated data**, only, rather than raw data access
  - (e.g. in Google trends)
- What is the right query language?
  - basic retrieval features (get version at timestamp t)
  - when did a certain information disappear?
  - when was it changed?
  - structured queries?
- Scalability problems

Is it easier/better for RDF/Linked Data?
Arching the Web of Data
Linked Data is evolving

Update rate
- second
- minute
- hour
- day
- week
- month
- year

Virtual/Augmented Reality
- Internet of Things

Number of sources
- Dynamic LD Observatory
- LOD Laundromat
- BTC

Linked Data is evolving in various domains such as Virtual/Augmented Reality and the Internet of Things. The update rates range from seconds to years, with different sources available, including DBpedia, WIKIDATA, and LOD Laundromat.
One of the first (and last?) LOD archives: The Dynamic Linked Data Observatory (evolving Linked Data since 2012)

Weekly dumps of crawl snapshots...
Granularity?
Queries?
Crawl failures?
Most semantic Web/Linked Data tools are focused on this “static view” but do not consider versioning/evolution.

Sindice, SWSE, Swoogle, LOD Cache, LOD-Laundromat... so far, no versions!
RDF Archiving. Example

RDF Graph $V_1$

ex:C1 ex:hasProfessor ex:P1 .
ex:S1 ex:study ex:C1 .
ex:S2 ex:study ex:C1 .

RDF Graph $V_2$

ex:C1 ex:hasProfessor ex:P1 .
ex:S1 ex:study ex:C1 .
ex:S2 ex:study ex:C1 .
ex:S3 ex:study ex:C1 .

RDF Graph $V_3$

ex:C1 ex:hasProfessor ex:P1 .
ex:C1 ex:hasProfessor ex:P2 .
ex:S1 ex:study ex:C1 .
ex:S1 ex:study ex:C1 .
ex:S3 ex:study ex:C1 .
ex:S3 ex:study ex:C1 .
Research challenges on evolving structured interlinked data

- How can we **represent** archives of continuously evolving linked datasets?

- How can we **minimize** the redundant information of archives? (e.g. duplicates in snapshots)

- How can we improve **completeness** of archiving?

- How can emerging **retrieval** demands in archiving be satisfied?
  - e.g. time-traversing and traceability? Avoiding bottlenecks?

- How can certain **time-specific queries** over archives be answered?
  - Can we re-use existing technologies (e.g. SPARQL or temporal extensions)?
  - What is the right query language for such queries?
  - e.g. knowing if a dataset has changed, and how, in a certain time period?
...in the last few years:

Managing the Evolution and Preservation of the Data Web (FP7)

Preserving Linked Data (FP7)

The Dynamic Linked Data Observatory

Research projects

Archives

Tools

Benchmarking
Representing and querying evolving semantic data
The cold-start problem

How we can get archives of RDF data
Pull changes (crawl) vs. Push changes (notify)

- Some services that publish or are mapped to RDF change regularly, but we don’t know the frequency upfront!

- Some services mapped to RDF announce/archive their changes already, so they already keep an archive...

Towards capturing and preserving changes on the Web of Data 50-65 Jürgen Umbrich, Nina Mrzelj, Axel Polleres. DIACHRON WS 2015
DBpedia Wayback Machine

- Retrieve historical versions of a DBpedia resource
  - What was the version of “Donald Trump” on dd/mm/yyyy?
- Re-apply DBpedia mappings on the Wikipedia revision history

http://data.wu.ac.at/wayback
How can one represent revisions while respecting DBpedia?

- a) quads → `<dbpediaSubject> <pred> <obj> <Revision>`.
- b) proprietary triples → `<ownSubject/Revision> <pred> <obj>`.

Operations?

- Get revisions meta-data for one resource (by revisionID or timestamp)
- Get “materialised” versions of a resource (by revisionID or timestamp)
- Get difference between two revisions
More complex operations/queries? Open challenge
- a) On-demand? Query rewriting, similar to RDB2RDF
- b) Batch: Fetch the desired information, then store and query it
We are (obviously) not the only ones looking into this...

However: Only one version per “irregular” dbpedia dump
LOD Laundromat

- Lodlaundromat.org: a central repository of LD

- Problems?
  - Still you need to access/query 650K datasets
  - Of course the solution is not complete, but “a good approximation”
LOD-a-lot: Low cost archiving of LOD

Dataset 1

Dataset 650K

Linked Open Data

- crawl -

- clean -

- index & store -

SPARQL endpoint (metadata)

Kudos to Javier D. Fernandez, Wouter Beek, Miguel A. Martínez-Prieto, Mario Arias, Ruben Verbogh
**LOD-a-lot (some numbers)**

<table>
<thead>
<tr>
<th>#Triples</th>
<th>#Subjects</th>
<th>#Predicates</th>
<th>#Objects</th>
<th>#Common SO</th>
<th>#Literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>28,362,198,927</td>
<td>3,214,347,198</td>
<td>1,168,932</td>
<td>3,178,409,386</td>
<td>1,298,808,567</td>
<td>1,302,285,394</td>
</tr>
</tbody>
</table>

- **Disk size:**
  - HDT: 304 GB
  - HDT-FoQ (additional indexes): 133 GB

- **Memory footprint (to query):**
  - 15.7 GB of RAM (3% of the size)
  - 144 seconds loading time
    - 8 cores (2.6 GHz), RAM 32 GB, SATA HDD on Ubuntu 14.04.5 LTS

- **LDF page resolution in milliseconds.**

  (LOD-a-lot creation took 64 h & 170GB RAM. HDT-FoQ took 8 h & 250GB RAM)
LOD-a-lot

http://purl.org/HDT/lovd-a-lot

https://datahub.io/dataset/lod-a-lot
LOD-a-lot (some use cases)

- Archiving
  - We plan to have quarterly releases
- Query resolution at Web scale
- Evaluation and Benchmarking
  - No excuse 😊
- RDF metrics and analytics
ACKs LOD-a-lot
The archiving problem

Now, how can we efficiently archive and perform time-based retrieval queries of a dataset?
RDF Archiving. Archiving policies

a) Independent Copies/Snapshots (IC)

b) Change-based approach (CB)

c) Timestamp-based approach (TB)

d) Hybrid approaches
BEAR

https://aic.ai.wu.ac.at/qadlod/bear.html

1. Overview

There is an emerging demand on efficient, archiving and (temporal) querying different versions of evolving semantic Web data. As novel archiving systems are starting to address this challenge, foundations/standards for benchmarking RDF archives are needed to evaluate its storage space efficiency and the performance of different retrieval operations.

To this end, we have developed a BEerchmark of RDF Archives (BEAR), a test suite composed of three real-world datasets together with queries with varying complexity, covering a broad range of archiving use cases.

The DATA:

BEAR comprises three main datasets, namely BEAR-A, BEAR-B, and BEAR-C, each having different characteristics.
BEAR:
Benchmarking the Efficiency of RDF Archives

- Blueprint on benchmarking archives of semantic data
  - How can one define the corpus?
  - How can one design benchmark queries? Which queries?

- BEAR: concrete basic benchmark
  - Data: Crawl from Linked Data Observatory
  - Basic queries: Materialize, get Version...
  - Initial evaluation on archiving policies

https://aic.ai.wu.ac.at/qadlod/bear.html
BEAR: Benchmarking the Efficiency of RDF Archives

- Blueprint on benchmarking archives of semantic data
  - How can one define the corpus?
  - How can one design benchmark queries? Which queries?

- BEAR: concrete basic benchmark
  - Data: Crawl from Linked Data Observatory
  - Basic queries: Materialize, get Version...
  - Initial evaluation on archiving policies

https://aic.ai.wu.ac.at/qadlod/bear.html
Benchmarking: Define the corpus

- Number of versions / size
- Data dynamicity
  - Version change ratio
  - Version data growth
- Data static core
- Total triples (version-oblivious)
- Others
  - RDF vocabulary
  - Per version / evolution

Definition 1 (RDF Archive). A version-annotated triple is an RDF triple \((s, p, o)\) with a label \(i \in \mathcal{N}\) representing the version in which this triple holds, denoted by the notation \((s, p, o) : [i]\). An RDF archive graph \(\mathcal{A}\) is a set of version-annotated triples.

\[
\delta_{i,j} = \frac{|\Delta^+_{i,j} \cup \Delta^-_{i,j}|}{|V_i \cup V_j|}
\]

\[
growth(V_i, V_j) = \left| \frac{|V_j|}{|V_i|} \right|.
\]

\[
\mathcal{C}_\mathcal{A} = \{(s, p, o) | \forall i \in \mathcal{N}, (s, p, o) : [i] \in \mathcal{A}\}
\]

\[
\mathcal{O}_\mathcal{A} = \{(s, p, o) | \exists i \in \mathcal{N}, (s, p, o) : [i] \in \mathcal{A}\}
\]
Benchmarking: **Define the queries**

- Structured query languages managing time.
  - Temporal databases (T-Quel, TSQL2)
    - Overlapping, meeting, before, equal, during, finish
  - RDF/Linked Data
    - SPARQL extensions
      - T-SPARQL, SPARQL-ST
      - AnQL
      - DIACHRON Query Language
        - SPARQL with specific constructors such as DATASET (similar to a named graph), VERSION, or CHANGES
Benchmarking: Define the queries

- Design of benchmark queries
  - Archive-driven Cardinality + Selectivity (disregard versions)
  - Version-driven Cardinality + Selectivity + dynamicity

- Basic temporal retrieval features of queries
  - Mat \((Q, V_i)\): version materialization
  - Diff \((Q, V_i, V_j)\): delta materialization
  - Version\((Q)\): results of \(Q\) annotated with the version
  - Join\((Q1, V_i, Q2, V_j)\)
  - Change\((Q)\): Returns versions in which \(\text{Diff}(Q, V_i, V_{i-1}) \neq \emptyset\)

\[
\text{dyn}(Q, V_i, V_j) = \frac{|(\Omega_i \setminus \Omega_j) \cup (\Omega_j \setminus \Omega_i)|}{|\Omega_i \cup \Omega_j|}
\]
Benchmarking: Define the queries

- Instantiation of archive queries in AnQL [1]


- \texttt{Mat(Q,V)}
- \texttt{Diff(Q,V1,V2)}
- \texttt{Ver(Q)}
- \texttt{join(Q1,vi,Q2,vj)}
- \texttt{Change(Q)}

```sql
SELECT * WHERE { Q :[v] }
```
Benchmarked: Define the queries

- Instantiation of archive queries in AnQL [1]


- \( \text{Mat}(Q,V) \)
- \( \text{Diff}(Q,V1,V2) \)
- \( \text{Ver}(Q) \)
- \( \text{join}(Q1,vi,Q2,vj) \)
- \( \text{Change}(Q) \)

```sql
SELECT * WHERE {
  \{ \{Q:[v1]\} \text{ MINUS } \{Q:[v2]\} \} \text{ BIND } (v1 \text{ AS } ?V )
} \text{ UNION }
\{ \{Q:[v2]\} \text{ MINUS } \{Q:[v1]\}\} \text{ BIND } (v2 \text{ AS } ?V )
}```
Benchmarks: Define the queries

- Instantiation of archive queries in AnQL [1]


- Mat(Q,V)
- Diff(Q,V1,V2)
- Ver(Q)
- join(Q1,vi,Q2,vj)
- Change(Q)

```sql
SELECT * WHERE { Q : ?V }
```
Bench marking: Define the queries

- Instantiation of archive queries in AnQL [1]


- Mat(Q,V)
- Diff(Q,V1,V2)
- Ver(Q)
- join(Q1,v1,Q2,v2)
- Change(Q)

```sql
SELECT * WHERE { {Q : [v1]} {Q : [v2]} }
```
Benchamkering: Define the queries

- Instantiation of archive queries in AnQL [1]

- **Mat**($Q,V$)
- **Diff**($Q,V1,V2$)
- **Ver**($Q$)
- **join**($Q1,vi,Q2,vj$)
- **Change**($Q$)

Benchmarking: Define the queries

Open question remains: What is the right query syntax for archive queries?

```query
SELECT ?V1 ?V2 WHERE
{ {{Q :?V1} MINUS {Q :?V2}} UNION {{Q :?V2} MINUS {Q :?V1}} }
FILTER(abs(?V1-?V2) = 1)
```
BEAR: Benchmarking the Efficiency of RDF Archiving

- blueprint on benchmarking archives of semantic data
  - How can one define the corpus?
  - How can one design benchmark queries? Which queries?

- BEAR: concrete basic benchmark
  - Data: Crawl from Linked Data Observatory
  - Basic queries: Materialize, get Version...
  - Initial evaluation of archiving policies

https://aic.ai.wu.ac.at/qadlod/bear.html
BEAR: Benchmarking the Efficiency of RDF Archiving

- Queries and systems
  - We implemented and evaluate archiving systems on Jena-TDB and HDT, based on IC, CB and TB policies.
    - Serve as an initial baseline to compare archiving systems
    - More info: https://aic.ai.wu.ac.at/qadlod/bear.html
BEAR datasets

Dynamic Linked Data  BEAR-A
BEAR-A is composed of 58 weekly snapshots from the Dynamic Linked Data Observatory. BEAR-A provides triple pattern queries to test atomic queries such as Materialization, Diff, Version, etc.

The Dynamic Linked Data Observatory
The Dynamic Linked Data Observatory is a framework to monitor Linked Data over an extended period of time. The core goal of our work is to collect frequent, continuous snapshots of a subset of the Web of Data that is interesting for further study and experimentation, with an aim to capture new data about the dynamics of Linked Data. The resulting corpora will be made openly and continuously available to the Linked Data research community.

Datasets

Results

“Good news! We started with the weekly crawls.”

DBpedia Live  BEAR-B
The BEAR-B dataset has been compiled from DBpedia Live changesets over the course of three months and contains the 100 most volatile resources along with their updates and real-world triple pattern queries from user logs.

Open Data portals  BEAR-C
BEAR-C used the Open Data Portal Watch project to take the datasets descriptions of the European Open Data portal for 32 weeks. With the help of Open Data experts, we created 10 complex queries that retrieve different information from datasets and files.

Open Data Portal Watch
Quality Assessment and Monitoring of 250 Open Data Portals

Available Content

Quality Metrics
A comprehensive set of metrics for measuring the quality of open data portals.

Ports List
Official registered portals. Basic information such as the country, software, and the number of datasets.

Q3
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

{ dataset rdfs:type dcat:Dataset .
  dataset dcat:contactPoint /contact .
  contact vcard:fName ?name .
  OPTIONAL { contact vcard:hasEmail ?email .
  } }
RDF Archiving. Archiving policies

a) Independent Copies/Snapshots (IC)

```
ex:C1 ex:hasProfessor ex:P1 .
ex:S1 ex:study ex:C1 .
ex:S2 ex:study ex:C1 .
ex:C1 ex:hasProfessor ex:P2 .
ex:C1 ex:hasProfessor ex:S2 .
ex:S1 ex:study ex:C1 .
ex:S3 ex:study ex:C1 .
ex:C1 ex:hasProfessor ex:P1 .
ex:S1 ex:study ex:C1 .
ex:S2 ex:study ex:C1 .
ex:S3 ex:study ex:C1 .
ex:C1 ex:hasProfessor ex:P2 .
ex:C1 ex:hasProfessor ex:S2 .
ex:S1 ex:study ex:C1 .
ex:S3 ex:study ex:C1 .
ex:C1 ex:hasProfessor ex:P1 .
ex:S1 ex:study ex:C1 .
ex:S2 ex:study ex:C1 .
ex:S3 ex:study ex:C1 .
ex:C1 ex:hasProfessor ex:P1 [
V_1, V_2, V_3].
ex:C1 ex:hasProfessor ex:P2 [V_1].
ex:C1 ex:hasProfessor ex:S2 [V_1].
ex:S1 ex:study ex:C1 [V_1, V_2, V_3].
ex:S2 ex:study ex:C1 [V_1].
ex:S3 ex:study ex:C1 [V_2, V_3].
```

b) Change-based approach (CB)

```
ex:S2 ex:study ex:C1 .
ex:C1 ex:hasProfessor ex:P2 .
ex:C1 ex:hasProfessor ex:S2 .
ex:S1 ex:study ex:C1 [V_1, V_2, V_3].
ex:S2 ex:study ex:C1 [V_1].
ex:S3 ex:study ex:C1 [V_2, V_3].
ex:S3 ex:study ex:C1 .
ex:C1 ex:hasProfessor ex:P1 .
ex:S3 ex:study ex:C1 .
ex:C1 ex:hasProfessor ex:P1 .
ex:S3 ex:study ex:C1 .
ex:C1 ex:hasProfessor ex:P1[
V_1, V_2].
ex:C1 ex:hasProfessor ex:P2 [V_1].
ex:C1 ex:hasProfessor ex:S2 [V_1].
ex:S1 ex:study ex:C1 [V_1, V_2, V_3].
ex:S2 ex:study ex:C1 [V_1].
ex:S3 ex:study ex:C1 [V_2, V_3].
```

c) Timestamp-based approach (TB)
RDF Archiving. Archiving policies

a) Independent Copies/Snapshots (IC)

b) Change-based approach (CB)

c) Timestamp-based approach (TB)
Time-based access. Queries

- Triple Pattern queries
- Queries with “similar” number of results in all versions.
  - $\varepsilon$-stable query
    
    $\max_{\forall i \in N} \text{CARD}(Q, V_i) \leq (1 + \varepsilon) \frac{\sum_{\forall i \in N} \text{CARD}(Q, V_i)}{|N|}$
    
    $\min_{\forall i \in N} \text{CARD}(Q, V_i) \geq (1 + \varepsilon) \frac{\sum_{\forall i \in N} \text{CARD}(Q, V_i)}{|N|}$

\[ \varepsilon = 0.5 \]
Time-based access. Queries

Materialize (s,?,? ; version)

<table>
<thead>
<tr>
<th>IC</th>
<th>CB</th>
<th>HB₄</th>
<th>HB₈</th>
<th>HB₁₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>48 GB</td>
<td>28 GB</td>
<td>34 GB</td>
<td>31 GB</td>
<td>29 GB</td>
</tr>
</tbody>
</table>

Hybrid approach
Time-based access. Queries

\[ \text{diff(?, ?, o ; version0 ; version t)} \]

**Hybrid approach**

IC | CB | HB\(_4\) | HB\(_8\) | HB\(_{16}\)
---|----|-------|-------|-------
48 GB | 28 GB | 34 GB | 31 GB | 29 GB

**Query time in ms (logscale)**

![Query time graph](image)
Self-Indexing RDF Archives: v-RDFCSA

- RDFCSA: RDF index based on a Compressed Suffix Array
- v-RDFCSA[2] is designed as a lightweight TB approach
- Version information encoding
  - Any triple can be identified by the position of its subject within SA
  - Let be N the number of different versions and n the set of version-oblivious triples
- Two alternative encoding strategies
  - tpv: N bitsequences, each position i encodes if the triple i appears in the version
  - vpt: n bitsequences, each position i encodes if the version i includes the triple

Performs more than one order of magnitude faster than Jena-TDB

Open Data evolution

How is Open Data evolving?
OPEN DATA PORTAL WATCH
... a first step.

- Periodically monitoring a list of Open Data Portals
  - 90 CKAN powered Open Data Portals
- Quality assessment
- Evolution tracking
  - Meta data
  - Data

http://data.wu.ac.at/portalwatch/

Jürgen Umbrich
Sebastian Neumaier
ECDA: Evolving CSV Data Analyzer

DataMonitor

GET metadata
(JSON)

GET CSV (version)

{CSV}

Seed URLs

DATA EXTRACTOR

{CSV}

Analyzer

{CSV}

{Daff}

DATA CLEANSER

{CSV}

DATA GENERATOR AND VALIDATOR

{Daff}

GET data

ECDA: Evolving CSV Data Analyzer

Daff (diff for CSV)

Babelfy (Entity Linking)

Jaccard (Apache C.)
ECDA: Evolving CSV Data Analyzer

- Analysis of 726 datasets
  - Mean of 18 versions per file
  - Mean of 430 rows and 5.4 columns
- Increasing nature (x1.85 number of rows)
  - Small value modifications (0.85 jaccard)
  - Mostly string types (80% of 8-25 characters)

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Number of versions</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.landesdatenbank.nrw.de">www.landesdatenbank.nrw.de</a></td>
<td>493</td>
</tr>
<tr>
<td>ourairports.com</td>
<td>83</td>
</tr>
<tr>
<td>data.gov.au</td>
<td>23</td>
</tr>
<tr>
<td>Other domains</td>
<td>127</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of changes</th>
<th>Number of sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATIC: No changes in number of rows or columns</td>
<td>10 (1.4%)</td>
</tr>
<tr>
<td>INCREASING: No deletions of rows or columns, just additions.</td>
<td>71 (9.8%)</td>
</tr>
<tr>
<td>DECREASING: No deletions of rows or columns, just deletions.</td>
<td>2 (0.3%)</td>
</tr>
<tr>
<td>SAME SCHEMA: Only changes (additions or deletions) of rows</td>
<td>117 (16.12%)</td>
</tr>
<tr>
<td>SAME ROWS: Only changes (additions or deletions) of columns</td>
<td>9 (1.24%)</td>
</tr>
<tr>
<td>IRREGULAR: Changes (additions or deletions) of rows and columns</td>
<td>600 (82.64%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of versions</td>
<td>mean= 17.83, std=6.18</td>
</tr>
<tr>
<td>Original rows</td>
<td>mean= 431.23, std=1596.23</td>
</tr>
<tr>
<td>Original columns</td>
<td>mean= 5.40, std=8.48</td>
</tr>
<tr>
<td>Added rows</td>
<td>mean= 1.85, std=21.37</td>
</tr>
<tr>
<td>Deleted rows</td>
<td>mean= 0.79, std=2.02</td>
</tr>
<tr>
<td>Modified rows</td>
<td>mean= 0.09, std=0.80</td>
</tr>
<tr>
<td>Added columns</td>
<td>mean= 0.23, std=0.28</td>
</tr>
<tr>
<td>Deleted columns</td>
<td>mean= 0.22, std=0.29</td>
</tr>
<tr>
<td>Modified columns</td>
<td>mean= 0, std=0</td>
</tr>
<tr>
<td>Changed cells</td>
<td>mean= 1663.96, std=29564.60</td>
</tr>
</tbody>
</table>
ECDA: Evolving CSV Data Analyzer

- Analysis of 726 datasets
  - Entities (Babelfy)
    - On average there are around 0.07 entities per cell
    - Entities are static in the header (a mean of 3)
    - 1/3 of the entities change across time
    - Number of entities slightly decrease in time

<table>
<thead>
<tr>
<th>Variable</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of cells</td>
<td>mean= 2597.09, std=9890.53</td>
</tr>
<tr>
<td>Original entities</td>
<td>mean= 180.86, std=184.41</td>
</tr>
<tr>
<td>Actual entities</td>
<td>mean= 149.40, std=202.90</td>
</tr>
<tr>
<td>New entities</td>
<td>mean= 86.65, std=170.97</td>
</tr>
<tr>
<td>Entities maintained</td>
<td>mean= 62.75, std=131.23</td>
</tr>
<tr>
<td>Original header entities</td>
<td>mean= 2.68, std=11.09</td>
</tr>
<tr>
<td>Actual header entities</td>
<td>mean= 2.74, std=11.14</td>
</tr>
<tr>
<td>Header entities maintained</td>
<td>mean= 2.64, std=11.07</td>
</tr>
</tbody>
</table>
Finally, many open questions remain still!

- Archiving and querying evolving semantic Web data

<table>
<thead>
<tr>
<th>Objective</th>
<th>Research Question</th>
</tr>
</thead>
</table>
| Representation of archives| - minimize the redundant information  
- respect the original modeling and provenance information (e.g. LOD-a-lot) |
| Query language            | - design a query language satisfying these requirements for evolving interlinked data  
- our BEAR operations are meant to be an extensible starting point |
| Indexing                  | - index archives at large scale  
- keep up with evolution rate (streaming vs. archiving) to process the queries efficiently |
| Analysis/Optimization     | - use evolution patterns to optimize representations and queries  
- Querying archives of structures and non-structured sources? E.g. Open Data! |
| Application               | - LOD-a-lot is a good examples but modularity can be improved  
- Any low-cost but functional archiving at LOD scale can be a major milestone for the community |
Thank you!

“The measure of intelligence is the ability to change”

Albert Einstein