ABSTRACT
This paper studies the compressibility of RDF data sets. We show that big RDF data sets are highly compressible due to the structure of RDF graphs (power law), organization of URIs and RDF syntax verbosity. We present basic approaches to compress RDF data and test them with three well-known, real-world RDF data sets.

Categories and Subject Descriptors
E.4 [Coding and Information Theory]: Data compaction and compression

General Terms
Experimentation, Measurements

1. INTRODUCTION
RDF data management has become a major track in Web development. Real-world RDF data (as shown in Linked Open Data datasets[3]) reveal an increasing number of huge data collections, as well as great diversity in terms of sources and use. It is well known that they form labeled graphs and that their nodes and edges follow power law distributions[2]. Hence the data include redundancy from the graph itself (repeated nodes and edges), the hierarchical organization of URIs, and the verbosity of the given syntax (especially significant in RDF/XML).

Compression appears as a natural choice for exchanging this type of data in order to achieve a better time/space tradeoff, or for storing it modularly, as data dictionary plus the graph itself. The graph, in turn, can be represented by generalized adjacency lists, which can take advantage of the heavy-tailed graph structure of big RDF data sets. With some add-ons, this splitting can support basic searching/retrieving operations such as the common Resource → Identifier assignation in triples stores.

We present different approaches for compressing RDF data using its particularities with standard compression techniques, testing these methods in three well-known data sets: Billion Triples is a large data set given within the SemanticWeb Challenge from a mashup of sources, whereas Uniprot RDF and U.S. Census are real-world RDF data sets of protein sequences and U.S census information respectively. RDF data is normalized from its original format to plain N3, sampling a chunk of 3 Million triples (hereafter “Original”).

2. APPROACHES TO RDF COMPRESSION
Based on natural RDF features, we study four different approaches to compress RDF as shown in Fig. 1.

a) Direct Compression. First, we tested direct compression of the original file (Figure 1a). We consider three well-known techniques which cover the main compressor families: a dictionary-based gzip built on an LZ77 adaptation, bzip2 based on the Burrows-Wheeler Transform and ppmdi which implements a high-order predictive model on PPM. As we expected, a high repetition of data given by power-law distribution results in high levels of compression, shown in Table 1. PPM, as a high order compressor, gets the best results (up to 3.12% in Uniprot). The diversity of sources of Bil-
for a commonly used delta coding, in which URIs are also redundancy of their long shared prefixes. Figure 1c stands aged independently, as we focused on URIs, exploiting theals in the dictionary were sorted lexicographically and man-
turn, can be represented as one type of adjacency lists. Liter-
responding number assignation in the dictionary. Triples, in
tments and the triples substituting for each element, the cor-
compressibility: we split the data into the dictionary of ele-
compression levels previously presented, show a regularity in
rarely present. Nevertheless, power law assumption and high
facilitating compression too. In RDF, these properties are
some successors are shared by pages in the same domain,
cal order benefits the compression. By means of similarity,
is hardly applicable to RDF because they ex-
result in weaker compression, while the powers of delta and
coding size is bigger than delta before compression, as it only
Most of these URIs are named sequentially, so that tree
reached original compressing levels.
the improvement of both delta and tree coding do
reduce original size. In contrast, identifiers re-assignation
can slightly increase the size of triples representation. B.T and
U.S datasets are composed of a great variety of literals,
so that the improvement of both delta and tree coding do not
compensate for the uncompressed size of numeric literals
in U.S or the variability in B.T. In these cases, they do not
reach original compressing levels.
Uniprot is highly compressible due to the massive presence of
URIs, which benefits the compression of the dictionary.
Most of these URIs are named sequentially, so that tree
coding size is bigger than delta before compression, as it only
stores the difference and tree repeats the whole identifier.

### 3. CONCLUSIONS

Table 1 summarizes the results of the different approaches
tested. From this study we can conclude:

1. RDF data at big scale is highly compressible.
2. Dedicated data structures, e.g. adjacency lists, code
triples efficiently and facilitate compression (both string 
with ppmdi and integer with Huffman).
3. RDF URIs are prone to efficient compression with 
standard techniques, but compression of literals desire 
finer approaches.
4. The structure of RDF graphs differs from XML or Web 
data, hence, classical approaches such as [1] are not 
directly applicable.

### 4. REFERENCES


---

**Table 1: Original and compressed size (in MB) of Dictionary+Triples decomposition.**

<table>
<thead>
<tr>
<th></th>
<th>Dictionary</th>
<th>Triples</th>
<th>Uniprot</th>
<th>Triples</th>
<th>US Census</th>
<th>Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig.</td>
<td>Orig.</td>
<td>Orig.</td>
<td>Orig.</td>
<td>Orig.</td>
<td>Orig.</td>
<td>Orig.</td>
</tr>
<tr>
<td>gzip</td>
<td>114.0</td>
<td>29.9</td>
<td>153.2</td>
<td>28.4</td>
<td>10.7</td>
<td>21.6</td>
</tr>
<tr>
<td>bzip2</td>
<td>541.5</td>
<td>239.4</td>
<td>148.2</td>
<td>541.5</td>
<td>239.4</td>
<td>148.2</td>
</tr>
<tr>
<td>pppdi-6</td>
<td>37.9</td>
<td>7.5</td>
<td>6.4</td>
<td>37.9</td>
<td>7.5</td>
<td>6.4</td>
</tr>
</tbody>
</table>

**Table 2: Original and compressed size (in MB) of Dictionary+Triples decomposition.**

<table>
<thead>
<tr>
<th></th>
<th>B. Triples</th>
<th>Uniprot</th>
<th>US Census</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig. Size</td>
<td>541.5</td>
<td>239.4</td>
<td>148.2</td>
</tr>
<tr>
<td>gzip</td>
<td>114.0</td>
<td>29.9</td>
<td>153.2</td>
</tr>
<tr>
<td>bzip2</td>
<td>541.5</td>
<td>239.4</td>
<td>148.2</td>
</tr>
<tr>
<td>pppdi-6</td>
<td>37.9</td>
<td>7.5</td>
<td>6.4</td>
</tr>
</tbody>
</table>

**Table 3: Direct Compression (in MB).**

<table>
<thead>
<tr>
<th></th>
<th>B. Triples</th>
<th>Uniprot</th>
<th>US Census</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig. Size</td>
<td>541.5</td>
<td>239.4</td>
<td>148.2</td>
</tr>
<tr>
<td>gzip</td>
<td>114.0</td>
<td>29.9</td>
<td>153.2</td>
</tr>
<tr>
<td>bzip2</td>
<td>541.5</td>
<td>239.4</td>
<td>148.2</td>
</tr>
<tr>
<td>pppdi-6</td>
<td>37.9</td>
<td>7.5</td>
<td>6.4</td>
</tr>
</tbody>
</table>

**Table 4: Adjacency Lists Representations (in MB).**

<table>
<thead>
<tr>
<th></th>
<th>B. Triples</th>
<th>Uniprot</th>
<th>US Census</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig.</td>
<td>Orig.</td>
<td>Orig.</td>
<td>Orig.</td>
</tr>
<tr>
<td>gzip</td>
<td>114.0</td>
<td>29.9</td>
<td>153.2</td>
</tr>
<tr>
<td>bzip2</td>
<td>541.5</td>
<td>239.4</td>
<td>148.2</td>
</tr>
<tr>
<td>pppdi-6</td>
<td>37.9</td>
<td>7.5</td>
<td>6.4</td>
</tr>
</tbody>
</table>