RDF Compression: Basic Approaches*

Javier D. Fernández
Dept. of Computer Science
University of Valladolid (Spain)
jfergar@infor.uva.es

Claudio Gutierrez
Dept. of Computer Science
University of Chile (Chile)
cgutierr@dcc.uchile.cl

Miguel A. Martínez-Prieto
Dept. of Computer Science
University of Valladolid (Spain)
migumar2@infor.uva.es

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").

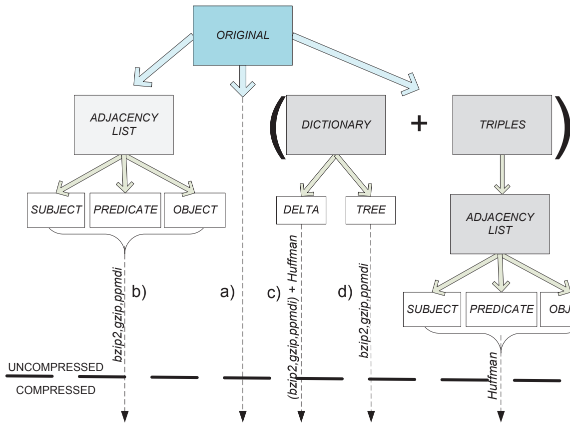

Figure 1: Approaches to RDF compression.

	B.Triples	Uniprot	US Census
Original Size	541.5	239.4	148.2
a) Direct Compr.	37.9 (7.0%)	7.5 (3.1%)	6.4 (4.3%)
b) Adjacency List	35.5 (6.6%)	5.3 (2.2%)	4.8 (3.3%)
c) Delta+Triples	37.9 (7.0%)	9.1 (3.8%)	9.0 (6.1%)
d) Trie+Triples	39.4 (7.3%)	9.7 (4.1%)	9.0 (6.1%)

Table 1: Results of different approaches (in MB).

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 3. PPM, as a high order compressor, gets the best results (up to 3.12% in Uniprot). The diversity of sources of Bil-

*Partly funded by Erasmus Mundus (first author), MICINN (TIN2009-14009-C02-02), a fellowship from the Regional Government of Castilla y Leon (Spain) and the European Social Fund (first and third authors) and Fondecyt 1090565 (C. Gutierrez, DCC).

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

<table>
<tr><th rowspan="3"></th><th colspan="4">B.Triples</th><th colspan="4">Uniprot</th><th colspan="4">US Census</th></tr>
<tr><th colspan="2">Dictionary</th><th colspan="2">Triples</th><th colspan="2">Dictionary</th><th colspan="2">Triples</th><th colspan="2">Dictionary</th><th colspan="2">Triples</th></tr>
<tr><th>Orig.</th><th>Compr.</th><th>Orig.</th><th>Compr.</th><th>Orig.</th><th>Compr.</th><th>Orig.</th><th>Compr.</th><th>Orig.</th><th>Compr.</th><th>Orig.</th><th>Compr.</th></tr>
<tr><td>Original</td><td>153.2</td><td>28.0 (18.3%)</td><td rowspan="3">65.5</td><td rowspan="3">9.5 (4.2%)</td><td>26.3</td><td>1.4 (5.3%)</td><td>58.6</td><td rowspan="3">8.1 (13.8%)</td><td>6.8</td><td>0.9 (13.2%)</td><td>49.6</td><td>10.7 (21.6%)</td></tr>
<tr><td>Delta</td><td>105.3</td><td>28.4 (18.5%)</td><td>4.4</td><td>1.0 (3.8%)</td><td rowspan="2">59.6</td><td rowspan="2">5.7</td><td rowspan="2">1.3 (19.1%)</td><td rowspan="2">52.8</td><td rowspan="2">7.7 (15.5%)</td></tr>
<tr><td>Tree</td><td>114.0</td><td>29.9 (19.5%)</td><td>15.9</td><td>1.6 (6.1%)</td></tr>
</table>

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

	B.Triples	Uniprot	US Census
Original Size	541.5	239.4	148.2
bzip2	46.9 (8.7%)	10.8 (4.5%)	9.7 (6.6%)
gzip	55.2 (10.2%)	18.3 (7.7%)	13.4 (9.0%)
ppmdi-6	37.9 (7.0%)	7.5 (3.1%)	6.4 (4.4%)

Table 3: Direct Compression (in MB).

	B.Triples	Uniprot	US Census
Original Size	541.5	239.4	148.2
Subject A.L	35.5 (6.6%)	5.3 (2.2%)	4.8 (3.3%)
Predicate A.L.	50.7 (9.4%)	6.60 (2.8%)	7.9 (5.3%)
Object A.L.	42.5 (7.9%)	6.6 (2.8%)	7.5 (5.0%)

Table 4: Adjacency Lists Representations (in MB).

lion Triples data set results in weaker compression, while the numeric data nature of U.S. Census is also punished.

b) Adjacency Lists. A second approximation focus data repeatability by using Adjacency Lists (Figure 1b). For example, the set of triples $\{(s, p_1, o_{11}), \cdots, (s, p_1, o_{1n_1}), \cdots (s, p_2, o_{21}), \cdots (s, p_2, o_{2n_2}), \cdots (s, p_k, o_{kn_k})\}$ would be written as the adjacency list $s \rightarrow [(p_1, \text{ObjList}_1), \cdots (p_k, \text{ObjList}_k)]$.

We considered three types of adjacency lists: subject (present in Turtle and N3), predicate and object headed. Predicates and subjects precede objects in sublists, and both lists and sublists are ordered lexicographically. Table 4 shows that subject adjacency lists compression (using ppmdi to compare best results) improves direct compression in every case, while predicate and object have better results only in Uniprot. Adjacency lists levels of compression depend on both their compact power and their ability to produce repeated elements in sublists.

c,d) Dictionary+Triples. The next two tests are focused on the graph nature of RDF. Note that standard Web graph compression is hardly applicable to RDF because they exploit locality and similiarity features of their link structure[1]. Locality implies that the source and the target of a link tend to share the same domain, therefore lexicographical order benefits the compression. By means of similarity, some successors are shared by pages in the same domain, facilitating compression too. In RDF, these properties are rarely present. Nevertheless, power law assumption and high compression levels previously presented, show a regularity in triples that might be exploited.

A simple graph representation is proposed to test RDF compressibility: we split the data into the dictionary of elements and the triples substituting for each element, the corresponding number assignation in the dictionary. Triples, in turn, can be represented as one type of adjacency lists. Literals in the dictionary were sorted lexicographically and managed independently, as we focused on URIs, exploiting the redundancy of their long shared prefixes. Figure 1c stands for a commonly used delta coding, in which URIs are also sorted lexicographically and each one is coded by two integers and a string. Integers delimit the number of characters shared with the previous symbol and the number of different characters, while the string represents the portion of the URI that differs from the previous one.

Delta encoding is a compression-oriented representation, so that operations with the dictionary (mainly to find the identifier of an element and vice versa, commonly in RDF triples stores) become more complex. In order to facilitate these operations, we considered a compact tree representation (referred to as DFUDS) built on a sequence of balanced parentheses, as shown in Figure 1d. We stored the preorder traversal sequence of the tree and two bits per node to represent the structure. When using this succinct representation in-memory, an auxiliary structure is needed, with cost $o(n)$, where n is the number of nodes. This structure facilitates common operations without uncompression.

Table 2 details different Dictionary+Triples decomposition. In compression, triples and delta integers are coded by canonical bit-oriented Huffman, while the preorder sequence and the literals are compressed with ppmdi. Compressed values of triples correspond to subject adjacency lists. As a general observation, both delta and tree coding dictionaries 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 uncompressibility 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 deserve 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

[1] P. Boldi and S. Vigna. The webgraph framework i: compression techniques. In *WWW*, pp. 595–602, 2004.

[2] L. Ding and T. Finin. Characterizing the semantic web on the web. In *ISWC*, pp. 242–257, 2006.

[3] LOD data sets. http://linkeddata.org/data-sets.