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”).

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 distributions 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-
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 and delta coding dictionaries reach original compressing levels.

Table 2 details different Dictionary+Triples decompositions. In compression, triples and delta integers are coded by canonical bit-oriented Huffman, while the preorder sequence of the tree and two bits per node to represent 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 identifiers, an auxiliary structure is needed, with cost in-memory, an auxiliary structure is needed, with cost of literals, which benefits the compression of the dictionary. 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.

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 ppmid 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