Frank: The LOD Cloud at Your Fingertips*

Wouter Beek and Laurens Rietveld

Dept. of Computer Science, VU University Amsterdam, NL
{w.g.j.beek,laurens.rietveld}@vu.nl

Abstract. Large-scale, algorithmic access to LOD Cloud data has been hampered by the absence of queryable endpoints for many datasets, a plethora of serialization formats, and an abundance of idiosyncrasies such as syntax errors. As of late, very large-scale – hundreds of thousands of document, tens of billions of triples – access to RDF data has become possible thanks to the LOD Laundromat Web Service. In this paper we showcase *Frank*, a command-line interface to a very large collection of standards-compliant, real-world RDF data that can be used to run Semantic Web experiments and stress-test Linked Data applications.

1 Introduction

Let's be frank: The Semantic Web is a big and dangerous place. Many researchers and application programmers spend a fair amount of their researching and application programming time by handling various serialization formats and juggling with syntax errors and other dataset-specific idiosyncrasies.

For instance, one of the authors of this paper has tried to run an evaluation on Freebase, one of the most valuable Linked Datasets out there. A human agent can assess that the Freebase dereference of the 'monkey' resource, `http://rdf.freebase.com/ns/m.08pbxl`, consists of approximately 600 statement. However, a state-of-the art RDF parser such as Rapper[1] is able to retrieve 32 triples, slightly more than 5% of the actual monkey info. Such results are not uncommon, even among often-used, high-impact datasets such as Freebase.

The problem of idiosyncrasies within one dataset is worsened by the fact that different datasets exhibit different deviations from RDF and Web standards. This means that a custom script that is able to run a Semantic Web evaluation on one dataset may fail to perform the same job for another, requiring ad hoc and thus human-supervised operations to be performed. We believe that this is one of the reasons why most evaluations in Semantic Web research publications are run on only a handful of, often the same, datasets (DBpedia, Freebase, Semantic Web Dog Food, SP²Bench, etc.).[2] It is simply impractical to run a Semantic Web algorithm against tens of thousands of datasets. Notice that the

* This work was supported by the Dutch national program COMMIT.

[1] Version 2.0.14, retrieved from `http://librdf.org/raptor/rapper.html`.

[2] Based on the observations from the previous paragraph, we can only guess as to what it actually means to run an evaluation against a dataset like Freebase. Does it mean that the evaluation was run against its $< 5\%$ syntactically correct triples?

challenge here is not scalability per se, as most datasets on the Semantic Web are actually quite small (much smaller than DBpedia, for instance). The problem seems to be with the *heterogeneity* of data formats and idiosyncrasies.

In [1] the LOD Laundromat was presented, an attempt to clean as many Linked Datasets as possible into a single, uniform and standards-compliant format. The LOD Laundromat has now (re)published a wealth of Linked Data in a format that can be processed by machines without having to pass through a dataset-specific and cumbersome data cleaning stage.

At the time of writing the LOD Laundromat disseminates over 650,000 data documents containing over 38,000,000,000 triples. In [4] the LOD Laundromat, which had been serving clean data files until that point, was combined with the Triple Pattern Fragments [5], thereby offering live query access to its entire collection of cleaned datasets.

By (re)publishing very many datasets in exactly the same, standards-compliant way, the LOD Laundromat infrastructure supports the evaluation of Semantic Web algorithms on large-scale, heterogeneous and real-world data. However, until now the LOD Laundromat, together with its Triple Pattern Fragments extension, has been disseminated as a collection of Web Services (`http://lodlaundromat.org`) where clean datasets can be downloaded and queried. In addition, metadata about the cleaning process and the structural properties of the data can be queried via a SPARQL endpoint. While this is a good interface for some use cases, e.g. downloading a specific data document, it is not suitable for others. For instance, using the Web Services it is relatively difficult to evaluate a Semantic Web algorithm against thousands of datasets, the main use case we are aiming for in this paper.

Moreover, it is precisely these large-scale use cases in which the LOD Laundromat excels. This is why we have created *Frank*, the computational companion to the LOD Laundromat Website and Web Services. *Frank* allows the same operations to be more easily performed on a larger scale and with added flexibility.

2 How to be *Frank*

In this section we show some of the key features of *Frank*.[3] For brevity, we sometimes abbreviate MD5 hashes and use common RDF prefix shortening in the results.

2.1 Data retrieval

Frank makes it easy to pose queries against the LOD Cloud, such as "Give me an arbitrary triple, Frank." This query is executable via the `frank statements` command, which returns a stream of RDF statements serialized as plain N-Triples or N-Quads. Access to a single statements is possible by using the power of Bash streams and pipes:

```
$ frank statements | head -n 1
<http://csarven.ca/#i> foaf:givenName "Sarven" .
```

[3] See `https://github.com/LODLaundry/Frank` for the source code.

In the above example, *Frank* is asked for any instantiation for the subject, predicate and object. The results are returned in a stream of arbitrary length, containing an arbitrary number of solutions. Since *Frank* uses the standard conventions for output handling, other processes can utilize the resultant triples by simply reading from standard input. Since *Frank* returns answers with anytime behavior, i.e., one-by-one, processes that utilize its output are able to run flexibly. Specifically, no cumbersome writing to file and/or waiting for complete result sets is needed.

We can ask for more than arbitrary triples though, as any Simple Graph Pattern [2] can be used to query LOD Laundromat data. For example, in order to retrieve only persons:

```
$ frank statements \
    --predicate rdf:type \
    --object foaf:Person \
    --showGraph \
  | head -n 2
<http://csarven.ca/#i> rdf:type foaf:Person ll:85d...33c.
dbp:Computerchemist rdf:type foaf:Person ll:0fb...813.
```

Notice that we have instantiated the predicate and object terms and have requested the graph from which a triple originates. Or, in this case, the graph from which a person was retrieved. These graphs are the LOD Laundromat identifiers that stand for the cleaned documents containing the respective FOAF persons.

To query a specific graph (or a specific collection of graphs), these LOD Laundromat document identifiers can be added as arguments to `frank statements`:

```
$ frank statements
    --predicate rdf:type \
    --object foaf:Person \
    http://lodlaundromat.org/resource/85d...33c
<http://csarven.ca/#i> rdf:type foaf:Person .
...
```

2.2 Data documents

Besides querying for individual triples, *Frank* can also load entire data documents. The advantage of loading documents, besides being a bit quicker, is that a document is a collection of triples this is published with a certain intent. Even though data documents can — in theory — be assembled randomly, in pratice it is often assumed that there is some cohesion present in a document that cannot be found in a random collection of triples. (This may be called a social aspect of RDF data.)

The following command prints every LOD Laundromat download URI.

```
$ frank documents --downloadUri
http://download.lodlaundromat.org/fcf...b92
http://download.lodlaundromat.org/134...344
http://download.lodlaundromat.org/d4a...b85
http://download.lodlaundromat.org/0b8...ade
http://download.lodlaundromat.org/f08...66f
...
```

The results of `frank documents` can be filtered with some basic options. For instance, in the following data documents are filtered by the number of (unique) triples that appear in them:

```
$ frank documents --downloadUri \
  --minTriples 100000
  --maxTriples 1000000
http://download.lodlaundromat.org/bd0...2a5
...
```

To fetch the triples for these filtered datasets, simply pipe the results of verb|frankDocuments| to `frank statements`:

```
$ frank documents --resourceUri \
    --minTriples 100000
    --maxTriples 1000000
  | frankStatements
dbp:1921Novels rdfs:label "1921 novels".
dbp:1921Operas rdfs:label "1921 operas".
...
```

2.3 Metadata

`frank meta` allows metadata descriptions of data documents to be retrieved and returned in N-Triples format. For example, the following returns metadata for one particular document.

```
$ frank documents --resourceUri | frank meta
ll:85d...33c ll:triples "54"^^xsd:int .
ll:85d...33c llo:added "2014-10-10T00:23:56"^^xsd:dateTime .
...
```

3 Implementation

Frank is implemented as a single Bash script, which allows piping of results to other processes. Figure 1 shows the relationships between *Frank* and the LOD Laundromat Web Services. We now give implementation details of the basic interface commands that were illustrated in Section 2.

Streamed triple retrieval `frank statements` allows individual statements to be retrieved. Its command-line flags `--subject`, `--predicate`, and `--object` mimics the expressivity of the Triple Pattern Fragments Web API. LDF provide a self-descriptive API which uses pagination in order to serve large results in smaller chunks, making streamed processing possible. `frank statements` interfaces with the Triple Pattern Fragments API for a given data document, or it enumerates all available LDF endpoints (using `frank documents`). For performance reasons, a `frank statements` call without subject, predicate or object flag retrieves the triples directly from the published LOD Laundromat Gzip files. For each LDF endpoint it handles the pagination settings in order to ensure a a constant stream of triples. The LDF API is able to answer triple pattern requests efficiently by using the Header Dictionary Triples[4] (HDT) technology.

[4] See `http://www.rdfhdt.org/`.

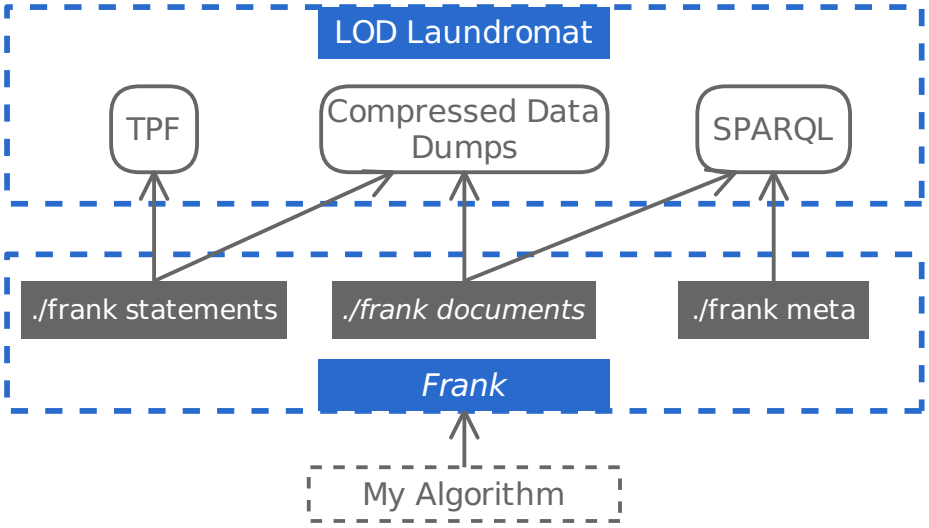

Fig. 1. The implementation architecture for *Frank* and its dependence on the LOD Laundromat Web Services.

HDT is a binary, compressed, and indexed serialization format that facilitates efficient browsing and querying of RDF data at the level of Simple Graph Patterns. HDT files are automatically generated for all data documents that are disseminated by the LOD Laundromat backend.

Streamed document retrieval `frank documents` allows individual documents to be retrieved. It interfaces with the SPARQL backend in order to find data documents that e.g. adhere to the given size restrictions, i.e., at least `--minTriples` and at most `--maxTriples` triples. It identifies a data document in the following two ways:

1. The URI from which the data document, cleaned by the LOD Laundromat, can be downloaded (`--downloadUri`)
2. The Semantic Web resource identifier assigned by LOD Laundromat for this particular document (`--resourceUri`)

When neither `--downloadUri` nor `--resourceUri` are passed as arguments *Frank* returns both separated by white-space.

The clean data documents are disseminated by the LOD Laundromat as Gzipped N-Triples or N-Quads. The statements are unique within a document so no bookkeeping with respect to duplicate occurrences needs to be applied. Statements are returned according to their lexicographic order. These statements can be processed on a one-by-one basis which allows for streamed processing by *Frank*.

Metadata `frank meta` retrieves the metadata description of a given data document. It interfaces with the SPARQL endpoint of LOD Laundromat and returns N-Triples that contain provenance for that particular resource, and a set of VoID statistics generated by the LOD Laundromat. [3]

4 Conclusion & Future work

Algorithmic access to the LOD *Cloud* used to be cumbersome to implement, required the use of crawling or incomplete catalogs, and often needed ad-hoc intermediate human-supervised operations to deal with deviations from RDF and Web Standards. Now – thanks to *Frank* and the LOD Laundromat – such algorithmic access is reduced to a single Bash line.

The current version of *Frank* focuses on performing *data consumption* tasks. It allows triples and documents to be retrieved from the LOD Laundromat. It does not, at the moment, allow data to be added for cleaning. This can be done though the LOD Basket Web Interface (`http://lodlaundromat.org/sparql/`). *Frank* does not yet allow metadata to be queried in non-trivial ways. This can be done though the SPARQL endpoint (`http://lodlaundromat.org/basket/`) which stores the scraping metadata as well as the structural metadata [3]. Better support in these areas may be added in future versions, depending on whether such support is needed in practice.

Frank currently allows Simple Graph Patterns to be queried. While it is technically possible to collate Simple Graph Patterns together into Basic Graph Patterns [2], complex queries cannot not yet be efficiently evaluated. The reason for this is that Linked Data Fragment requests are performed in the sequence in which they are supplied by the user and this sequence may not be optimal. In future research we want to allow complex queries to be optimized before being sent to the LOD Laundromat backend, thereby making it possible to perform queries of arbitrary complexity in a more efficient manner.

References

1. Beek, W., Rietveld, L., Bazoobandi, H.R., Wielemaker, J., Schlobach, S.: LOD laundromat: A uniform way of publishing other people's dirty data. In: The Semantic Web–ISWC 2014, pp. 213–228. Springer (2014)
2. Harris, S., Seaborne, A.: SPARQL 1.1 query language (March 2013)
3. Rietveld, L., Beek, W., Schlobach, S.: LOD in a box: The C-LOD meta-dataset (Under submission), `http://www.semantic-web-journal.net/system/files/swj868.pdf`
4. Rietveld, L., Verborgh, R., Beek, W., Sande, M.V., Schlobach, S.: Linked data as a service: The Semantic Web redeployed. In: The Extended Semantic Web Conference – ESWC. Springer (2015)
5. Verborgh, R., Hartig, O., De Meester, B., Haesendonck, G., De Vocht, L., Vander Sande, M., Cyganiak, R., Colpaert, P., Mannens, E., Van de Walle, R.: Querying datasets on the web with high availability. In: The Semantic Web–ISWC 2014, pp. 180–196. Springer (2014)