Three ways to sprinkle POWDER

Stasinos Konstantopoulos

Institute of Informatics and Telecommunications, NCSR ‘Demokritos’, Greece
konstant@iit.demokritos.gr

Abstract. In this paper, three alternative implementations of the POWDER W3C Recommendation are presented, compared, and discussed. The main issue with implementing POWDER is that POWDER inference accesses resources’ URIs in order to mass-annotate regular expression-delineated groupings of URIs.

1 Introduction

The *Protocol for Web Description Resources* (POWDER) is a W3C Recommendation primarily meant as a format for publishing metadata taking advantage of natural groupings of URIs.[1] As an example, consider the following data:

```
<http://nasa.dataincubator.org/spacecraft/1969-059A>
  rdf:type <http://purl.org/net/schemas/space/Spacecraft> .
<http://nasa.dataincubator.org/spacecraft/1969-099B>
  rdf:type <http://purl.org/net/schemas/space/Spacecraft> .
```

POWDER can be used to formally state what is already obvious to a human: that URIs from the `nasa.dataincubator.org` domain that have a path beginning with `spacecraft` denote instances of the `Spacecraft` class. POWDER documents are XML documents made up of *Description Resources* (DR), where each DR assigns one or more properties to all resources denoted by the URIs within a regular expression-delineated sub-space of the URI space:

```
<dr>
 <iriset>
  <includeregex>.*//nasa.dataincubator.org/spacecraft/.*</includeregex>
 </iriset>
 <descriptorset>
  <typeof src="http://purl.org/net/schemas/space/Spacecraft"/>
 </descriptorset>
</dr>
```

For the sake of brevity and clarity, only aspects relevant to the current discourse are presented, but it is worth noting that POWDER documents carry provenance information; that individual DRs can be combined in more complex structures and refer to DRs in other POWDER documents; and that the basic regular expression-based schema can be extended by defining more human-friendly `iriset` elements, such as `includehosts` and `includepathstartswith`, that can be gleaned into regular expression elements.

[1] See `http://www.w3.org/2007/powder` for more details.

2 Implementing POWDER

POWDER documents can be processed either directly as XML of by transforming into equivalent OWL/RDF documents. Either way, POWDER semantics provides the safeguard that having a matching URI is the only way to be assigned a property via POWDER, thus allowing for *layered inference* approaches where a POWDER layer is deployed under the semantic inference layer without the need for pushing semantic inference results back to the POWDER layer.

Here, three alternatives are explored:[2] via OWL inference, at the level of the triple store, and as combined POWDER/RDFS inference.

2.1 Implementing POWDER model theory

The formal semantics of POWDER uses an extension of RDF semantics that defines the `matchesregex` property that relates resources with regular expression strings such that the normalized string encoding of resource's URI matches the regular expression. Our spacecraft example is, then, equivalent to:

```
[ rdf:type owl:Restriction ;
  owl:hasValue ".*//nasa.dataincubator.org/spacecraft/.*"^^xsd:string ;
  owl:onProperty <http://www.w3.org/2007/05/powder-s#wdrs:matchesregex>
] rdfs:subClassOf <http://purl.org/net/schemas/space/Spacecraft> .
```

The most straightforward approach is to transform POWDER documents into their OWL/RDF equivalent, make all applicable `matchesregex` assertions, and then use OWL inference. This is the approach taken by SemPP, one of the reference implementations produced by the POWDER Working Group. SemPP is implemented within the Jena framework by extending the Jena `OntModel` with one that intercepts ensures that all appropriate `matchesregex` assertions are in the model. The extended `OntModel` is used as the explicit triple back-end for the `InfModel` implementation provided by the Pellet OWL reasoner.

Alternatively to implementing the `matchesregex` extension, one can implement an XML-based processor at the triple store, making POWDER-inferred statements appear as explicit triples to any inference stacked above it. In our implementation, we took advantage of the OpenRDF Sesame framework's architecture of 'stackable' layers that infer implicit RDF triples from the (explicit or implicit) data they receive from the layer immediately below. In this framework, a POWDER processor has been implemented as a layer stacked between the RDF store and semantic inference. This allows any inference engine to be deployed without modifications.

Querying is handled by manipulating the queries coming from above before being submitted to the RDF store below: triple patterns involving POWDER-assigned properties are first ignored (to accept *any* resource as having these properties) and the results are subsequently filtered so that only those bindings that are either explicit in the RDF store or satisfy the URI restrictions specified in the POWDER document are retained.

[2] All implementations are available from `http://powder.sourceforge.net`

2.2 Implementing POWDER proof theory

Looking at POWDER from another perspective, it is, effectively, a form of inference, not radically unlike semantic inference, except that the premises are the URI restrictions expressed in `<iriset>`s and the consequents are the properties expressed in `<descriptorset>`s.

Pursuing this angle, POWDER extends the proof-theoretic RDF and RDFS semantics rather than the model-theoretic semantics used in the two implementations above. The extension amounts, then, to augmenting the rules of RDFS inference by a rule that infers triples based on URIs matching regular expressions. This proof-theoretic extension was implemented by modifying the forward-chaining RDFS inference engine bundled in the Sesame distribution and using that instead of standard RDFS inference.

3 Results and Conclusions

3.1 Large-scale repositories

Many applications, especially large-scale RDF stores, regard efficiency as more important than expressivity of reasoning, use RDFS to define a relatively simple data schema, and operate over volumes of triples that make OWL reasoning unrealistic. In such applications, POWDER can be used as a means of compression and optimization by exploiting URI regularities to compress the explicit data.

The POWDER-S implementation cannot be used in such applications, because (a) asserting one statement for each URI $\times$ regular expression will make the repository *bigger* than if POWDER was not used; and (b) it introduces a dependency on OWL reasoning that could be avoided. We evaluated the other two implementations on the repository of the SYNC3 system that analyses news crawled from the web, extracts named entities, events, and the sentiment expressed in the news item with respect to events.[3]

Naturally, the amount of triples saved by using POWDER to mass-annotate resources depends on the application and the density of the POWDER-inferred statements. In SYNC3, there are several `rdf:type` assertions that cannot be semantically inferred but can be very naturally expressed in POWDER, given good URI design. For example, some of the types of named entities recognized in the content as participants in events (persons, organizations, etc.) cannot be semantically differentiated, but the distinction between these sub-types of event participants is necessary for the application.

In this domain, compression initially fluctuates but converges to roughly 20% over longer periods of time. Furthermore, queries involving a mixture of POWDER and non-POWDER triple patterns execute faster, since fewer triples are retrieved from the physical data store, especially when retrieving larger volumes

[3] Earlier results regarding the implementation of Sect. 2.1 have been presented by the author at the Intl WS on Semantic Web Information Management, ACM SIGMOD/PODS 2011, Athens, 12 June 2011.

Table 1. Querying time for repositories holding 38Mtriples of SYNC3 data. The 'vanilla' repository stacks the Sesame RDFS reasoner over the Sesame NativeStore.

Tuples	Time (in seconds)		
retrieved	Vanilla	RDFS/P	POWDER
538	32	13	32
706	22	14	19
2233	109	106	86
3418	117	114	96

of data compensating for the overhead of the POWDER layer. Comparing query rewriting with RDFS/P reasoning, the former approach is more efficient for larger queries, although is introduces more overhead, as it avoids asserting (either in the persistent store or in any in-memory structures used at query/inference time) the POWDER-inferred triples (Table 1).

3.2 The case for POWDER-S

POWDER is also used in the WAC Open Collaboration Platform to serve RDF annotations on applications' suitability for mobile devices and their security and privacy specifications.[4] Although the WAC specification does not dictate the use of POWDER-S, using the POWDER-S representation with OWL inference would have been a viable solution: in such contexts, data volumes are not such that scalability is an issue, but the qualitative information that TBox reasoning can offer is essential.

Applications such as WAC, access authorization for pay sites, trust-marking of websites by third-party authorities (e.g., as child-safe or not), etc. can use POWDER-S to support the annotation process by identifying inconsistencies between complex DRs overlapping in scope and possibly authored by different organizations.

3.3 Conclusion

We presented and compared three alternatives for implementing the POWDER Recommendation, outlining different applications where each is appropriate.

A promising future direction is optimizing RDF store indexes for using POWDER by, e.g., indexing URI fragments so that URIs matching regular expressions can be retrieved. Another interesting direction would be to investigate a combining POWDER with more expressive forms of reasoning, such as backward-chaining reasoning for OWL 2 RL.

Acknowledgement: I gratefully acknowledge the support of the European FP7-ICT project SYNC3[5] and the generous help and contribution by Phil Archer, `http://philarcher.org`. Naturally all errors and omissions are only my own.

[4] See `http://www.wacapps.net/web/portal` for more details.
[5] See `http://www.sync3.eu` for more details.