Towards Vertebrate Linked Datasets

Aidan Hogan

Digital Enterprise Research Institute, National University of Ireland, Galway

The Problem ...

Compared to the closed, cold, metallic, top-down and decidedly inorganic structure of a typical relational database, when I think of Linked Datasets, I think of evolving, seething, organic creatures with an emergent and complex structure. These organic creatures come in different shapes and sizes and can sometimes work together for various complementary purposes: a loosely interlinked ecology of sorts.

When I think of a Linked Dataset *like* DBpedia[1], I picture a gelatinous invertebrate entity—something like the creature from the 1958 movie "The Blob": organic, vaguely formed, turbid, expansive, expanding, wandering around in an Open World and occasionally assimilating some of the more interesting scenery. There is a certain unique beauty to this complex creature, no doubt, but it is a beauty that few can appreciate. If one is versed in SPARQL, one can strike up a conversation with the creature and ask it a variety of questions about the things it has assimilated. If you phrase your question right (use the correct predicates and so forth) and the creature is in the mood (the endpoint is available), and it has assimilated the things you're interested in (has the data), you can sometimes get a response.

But it is an enigmatic creature: oftentimes the response will not be what you expected. For example, if you ask DBpedia something simple like how many countries are there:

```
SELECT (COUNT(?c) as ?count) WHERE { ?c a dbpedia-owl:Country }
```

it tells you 2,710. Sounds a bit high, no? Should be just shy of 200? Why did DBpedia say 2,710?

The Position ...

Let's leave the admittedly belaboured and entirely unfair analogy of The Blob aside.[2] Linked Datasets are often analogous to black boxes. All we typically know about these black boxes are the the shape of content they house (triples), the size of that content (number of triples), the category of that content (domain of data), and some features of that content (classes and properties used).[3] Figuring out the rest is left as an exercise for the consumer. Oftentimes this is simply not enough and the consumer loses patience. So the question then is: how best to describe the contents of these black boxes.

The main aim of such a description should be to inform a consumer as to what expectations of freshness, accuracy and coverage of results they can expect for various types of queries over various chunks of the data (and let's assume that the endpoint performs perfectly now; an assumption that does not nearly hold in practice). The description should summarise the content of the black-box in such a way that it can be searched and browsed as a catalogue: it should allow a consumer to, hopefully at a few glances, *make sense* of the dataset they are querying, or at least a *core* of the dataset. My position in a few words: *we need intuitive mechanisms for humans to make sense of Linked Datasets (in whole and in part).*

How can this be done? Relying on the semantics of the used terminology, as given in a vocabulary/ontology, says nothing about the data itself. We can look at instances of classes and common relationships between instances of various classes, which is quite a useful exercise, but class-centric descriptions do not tell much of the story. Again, in DBpedia, we have 2,710 instances of `Country`. So what definition of `Country` permits

[1] ... here selecting a prominent example purely for argument's sake.
[2] I have nothing but cautious affection for DBpedia and Linked Data.
[3] If you're very lucky, you might find all of these details encapsulated in a VoID description.

2,710 instances? We can play around a bit and determine that a lot of listed countries no longer exist, or are not independent states, or are aliases or historical names, etc. But which are the current countries? And how many countries have flags or populations defined? How recent are those population measures?

It's great that DBpedia and other such datasets have lots of organic tidbits of information like historical countries and aliases and so forth. However, the result is a highly "non-normalised" soup of data that is difficult to describe and difficult to query over. We need a little *ordo ab chao*. One solution would be to trim the fat from datasets like DBpedia so they fit into a highly normalised database-like schema that best fits the most common needs and that gives consumers a smoother experience. But this is not only unnecessary, it is inorganic, inflexible and, dare I say it, not very Semantic-Webby.

One proposal ...

Instead of imposing a rigid inorganic structure on Linked Datasets like DBpedia so that they fit neatly into familiar rectangular frames of conceptualisation, perhaps we can just try find a natural shape to the dataset: a "spine". We can begin by clustering instances in an extensional sense: looking at clusters of instances defined using the same predicates and the same types. For example, a cluster might be a group of instances that all have type `Country`, at least one `capital`, at least one `gdp`, and at least one `city`. We can call these clusters "abstract classes" or "prototypes" or "templates" or "least common factors" or "instance cores" or ... well in fact, such clusters are not even an entirely new idea (and are similar to, e.g., "characteristic sets" [2]) but no matter.

More important is what we can use these clusters for. In this model, clusters naturally form a subsumption hierarchy where instances within a cluster are also contained within less specific sub-clusters. The number of clusters and the level of detail they capture can be parameterised for their computation [1]. A person can browse the hierarchy of clusters—the "spine"—of a dataset to see at a glance what it contains and to find the cluster he/she can target with a query. One might start exploring the super-cluster containing instances with type `Country` and see it branch into one sub-cluster with 910 instances with `dbprop:dateEnd` (dissolved countries) and a disjoint sub-cluster of 191 instances with the subject `category:Member_states_of_the_United_Nations` (current countries recognised by the UN; 2 are missing). Browsing the hierarchy thus helps with understanding the scope and breadth of data in increasing detail, helps with disambiguation, and helps with formulating a query that targets only the instances of interest.

Furthermore, the richer and more complete the clusters, the higher the degree of homogeneity in those instances. Child-clusters in the hierarchy with similar cardinalities may indicate incomplete data: e.g., taking the UN member cluster of 191 instances, we find a sub-cluster with of 188 instances with defined capitals, where we could conclude that capitals are missing in 3 instances. Filling in the blanks will merge one step of the hierarchy and increase the homogeneity of the country descriptions. Merging highly-overlapping sub-clusters in the hierarchy then becomes a quantifiable bottom-up goal for local normalisation.

Subsequently, clusters can be annotated for the instances they encapsulate; e.g., in this cluster, capitals rarely change, populations are all as recent as 2008, GDP values are 87% accurate, etc. A directed (subject-to-object), labelled (predicate), weighted (count) graph can be constructed between clusters as the aggregation of the most common links between their instances. Furthermore, the integration of two or more Linked Datasets can then be coordinated through these clusters, with the goal of identifying and consolidating conceptually overlapping clusters to a high (and easily quantifiable) degree.

The resulting spine of the dataset then gives a core and a shape to the dataset; an entry point to follow; a way of distinguishing normalised and complete data from non-normalised and incomplete data; a basis for coordinating integration; an emergent structure from which all the other organic matter can extend.

References

1. J. H. Friedman and J. J. Meulman. Clustering objects on subsets of attributes. *J. R. Stat. Soc*, 66:815–849, 2004.
2. T. Neumann and G. Moerkotte. Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In *ICDE*, pages 984–994, 2011.