Can Documents be Linked Data?

Kate Byrne, Language Technology Group, University of Edinburgh

This paper expands upon a presentation given at the CIGS workshop on Linked Data, held in Edinburgh on 18th November 2013. It starts by reminding the reader about the vision behind the semantic web and Linked Data, by looking back at the roots of these developments. I then describe some of my own research work on extracting Linked Data from free text documents, as a way of indexing their content and – ideally – summarising their meaning. Finally the paper examines some of the issues and pitfalls around Linked Data projects, looking at how we can actually make connections between our data and remote but related data. I suggest that the basic principle is analogous to using authority files, but one of the problems is that when "Anyone can say anything about anything" it may be hard to tell which of the many authorities is the one to trust.

I should probably mention at this point that, although I used to work in a National Library, I am a geek not a librarian, so I ask the reader to forgive any blunders made in talking of library matters.

The Semantic Web Vision

Things can move very fast in computer technology. The web – that is to say the "document web", which we can here contrast with the Data Web (the name Tim Berners-Lee says he should probably have coined instead of "semantic web") – grew explosively in its first few years. It's hard to believe that before the 1990s there was no World Wide Web. Of course the Internet had been in use for decades, but it was HTML that turned it into a vehicle for the information revolution we are experiencing. By the end of the 1990s we had all realised that creating a web presence for our organisations was not "nice to have" but essential.

In comparison, the semantic web has been a very slow burner. The first version of RDF became a W3C Recommendation in the 1990s and Tim Berners-Lee started talking publicly about the semantic web in 1999. It was back in 2001 that the now famous Scientific American article appeared (Berners Lee et al. 2001)[1] in which the authors proposed a web of data intended to be read by software agents, not humans. Alongside the raw data (in RDF) the agents would use structured vocabularies and inference rules that would enable them to examine large quantities of information and draw conclusions that could be justified to the human user if required. As Sir Tim put it: "The day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machine, leaving humans to provide the inspiration and intuition".[2]

Six years after the first *Scientific American* article a follow-up appeared (Feigenbaum et al., 2007) which argued that truly significant progress had been made, though the goals described seemed less ambitious than those in the original vision. Another seven years have passed and the semantic web is still hardly a household term. The somewhat ambiguous "Linked Data" label[3] has become fashionable, and one can now say, cautiously, that the movement towards a web of data is at last gaining momentum. In the library and archive worlds, more and more bodies are feeling that it is time to start experimenting with publishing their holdings as RDF triples, with "Cool URIs" that will hook them into the growing web of Linked Data.

---

[1] Tim Berners-Lee's famous remark about how the semantic web is supposed to work (incorporated in W3C *RDF Concepts*, 2002 draft).
[2] Quotation from Berners-Lee and Fischetti, 1999.
[3] Whether Linked Data **has** to involve RDF or not is a question guaranteed to produce lengthy arguments.

Bringing Free Text Content into the Data Web

There are plenty of exciting projects going on that aim to convert structured databases to RDF for Linked Data purposes. I want to consider unstructured information, *viz* text documents. After all, whilst indexes and field lists are obviously vital for finding information, we know that the really interesting facts are often contained in free text notes or documents that are associated with the database records. If machines are going to be able to make inferences about the knowledge in our datasets they will have to be able to "read" text in some way analogous to how humans do it. My research has involved trying to extract the factual statements embodied in texts (taken from an archaeological site archive), and turn them into RDF triples that could easily be added to the RDF graph derived from a structured database.

To illustrate the process, let us consider the following text:

> *"In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort."* [4]

RDF triples, consisting of "Subject – Predicate – Object", can be thought of as simple declarative sentences, where the nodes (the subject and object of a triple) are nouns and the predicate edges (or "arcs") joining them are verbs. Thus one could represent the first sentence above with the following simple RDF graph of two connected triples:

Figure 1 shows a graph for the whole of the quotation, involving a graph of five triples. The nodes are nouns and the edges are verbs. The representation is just an example and there are plenty of other ways of expressing the content as triples – just as there are other ways of conveying the same meaning in text sentences.

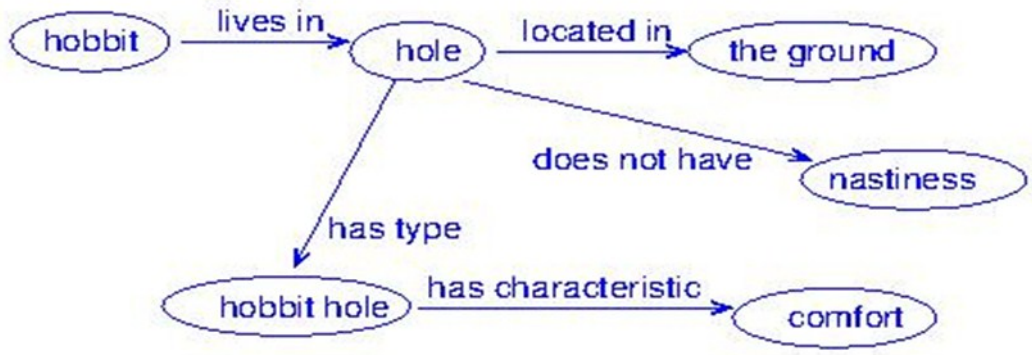

Figure 1. The opening of "The Hobbit" as RDF.

[4] This is of course the opening of *The Hobbit*, by J R R Tolkien.

In this example the RDF graph was of course produced a human editor (me) reading the text and deciding what were the key facts to be expressed as triples. I am interested in how well this task can be achieved automatically by software. My *txt2rdf* pipeline reads in text documents and passes them through a sequence of language processing steps (hence the term "pipeline") including tokenisation, part of speech tagging, named entity recognition and relation extraction. The end result is a set of binary relations between entities mentioned in the text; for example in the *Hobbit* extract, two of the entities detected might be "hole" and "the ground" and, if all goes to plan, the software should assert a relationship between them, namely that the first is located within the second. Once one has the binary relations between the entities – which are essentially the content-bearing nouns in the text – it is relatively straightforward to assign suitable URIs and turn them into RDF triples. I don't propose to explain the details of the pipeline here, but the interested reader can find them in Byrne and Klein, 2009, which is a paper written for a non-technical audience (that is, for archaeologists interested in processing archive data, rather than for language processing specialists).

The *txt2rdf* pipeline uses statistical techniques and was trained on archaeological data. It performs reasonably well (extracting around 60% of the available "facts" correctly) on data from this domain but very poorly on text from other domains. At present this is still typical of all such tools. The key steps are:

1. Identify and classify the entities mentioned in the text, eg decide whether a given term is a date, a place, a person's name, an archaeological site description, or completely irrelevant. This step is known as Named Entity Recognition or NER.
2. Having found a collection of entities, decide whether there are relationships between them, and if so how these relations should be labelled, eg "site A is located in place B", "event C occurred on date D", and so forth. This step is Relation Extraction or RE.

Each of these steps is a standard language processing task and a rough rule of thumb is to expect around 80% success, where "success" is usually measured as the harmonic mean of recall (how many of the available entities the NER step found, say) and precision (how many of those it identified were actually correct). If a pipeline combines NER and RE it multiplies the errors, so the best overall scores to expect are in the region of 64%, which is roughly what my research project produced. With careful training and tweaking these scores could certainly be improved but that means a lot more human effort in writing software and running tests.

The holy grail is to create automatic tools that perform well without having to be carefully and expertly tailored to a particular domain – we are still some way from that. Several research prototypes and a few commercial products have appeared in the past couple of years, for information extraction tasks of this kind. They are generally trained on "news" text – rich in dates, well-known people, etc – and will work best on this kind of material. Table 1 lists some of the currently available tools.[5]

[5] These are not recommendations, just a list of systems I am aware of. I haven't evaluated them.

Name	URL	Notes
Open Calais	http://www.opencalais.com/	One of the first on the web. From one of the top research labs. Language engineering tied in to semantic web. Free, but commercial services also available.
RelFinder	http://www.visualdataweb.org/relfinder.php	Also from a top research team. Free and open source. Probably requiring a de-
AlchemyAPI	http://www.alchemyapi.com/	One of the first commercial products, with some free services. Has a sophisticated online demo on the website to ena-
PoolParty	http://www.poolparty.biz/	Commercial products with some free demo software. Founded by respected research team. Range of different tools available.
TextWise	http://textwise.com/	A commercial product, with a simple online demo (paste in your own text) available on the website.
OpenUp DES	http://openup.tso.co.uk/des	Data Enrichment Service. A nice site put up by TSO, based on the well-known GATE tools from Sheffield University. Has a simple demo online that you can paste text into.

Table 1. Online language engineering tools.

We've looked at how language engineering can help to liberate the contents of text documents and turn the facts they contain into RDF triples. This process could also be used for related tasks, such as populating structured database fields, or checking the existing contents of these fields. The relation “event C occurred on date D” can equally well be expressed as a triple, or by putting event C and date D into fields of a database table.

If current tools are still so inaccurate are they of any use? I would say “Yes”, even at current levels of performance. After all, if you are faced with a data cleaning task over millions of records, having a machine do the “easy” 60-80% for you is a very great deal better than nothing. I would expect that, over the next few years, an increasing number of data managers with text-rich archives will turn to language processing tools to assist in their curation tasks.[6]

[6] It's another topic altogether, but “assisted curation” is an interesting problem – where a human and a software agent collaborate on editing data.

Authorities and ontologies

Assuming that we have somehow produced a graph of RDF triples – whether by extracting them from text or by converting a structured database, or even by hand-coding them – how do we make them part of the Linked Data enterprise? I would argue that the fundamental requirement is to hook our RDF data into established public ontologies, much as library catalogue entries are tied to authority records.

The beauty of RDF is that if two graphs share a node they are automatically connected. For example, if my graph happens to include a resource node for J R R Tolkien and so does yours, then our datasets are linked – no protocols, no special software; we're linked, and a SPARQL query can cross our datasets. But there are a couple of assumptions in that statement. One of the assumptions is that both our datasets are exposed on the web. The other, more germane to this discussion, is that we both use precisely the same data web node for J R R Tolkien. This is in fact very unlikely, unless we take trouble to make it true.

In general, when local data is converted to RDF the first thing that's done is to decide the "base URI". I work in the Language Technology Group at Edinburgh University and we have a domain URL of "www.ltg.ed.ac.uk". When I mint RDF URIs I will prefix them with something based on that. Therefore my J R R Tolkien node might be "http://rdf.ltg.ed.ac.uk/mygraph#jrrtolkien", say. It will clearly not match yours. Therefore, if I am planning ahead I should replace my "default" URI with a canonical one; "http://live.dbpedia.org/resource/J_R_R_Tolkien" would be a good candidate. Assuming you do the same thing in your data, we have Linked Data.

A moment's thought will show that this process is probably intractable. Every time a triple is generated (and it usually will be generated, by some automatic process) a search has to be made for the best available canonical URI. A simpler procedure is to mint local URIs but to connect them to published standards whenever possible. If there is more than one authority there's nothing to stop us making multiple connections. In my example, I might add the following triple to my graph:[7]

```
mygraph:jrrtolkein          owl:sameAs          dbpedia:J_R_R_Tolkein
```

If I am the first to publish Linked Data about Tolkien then at this stage I am not connected to anything except the DBpedia node – which will certainly enrich my data, as it gives me access to all the metadata in DBpedia. But as soon as another data publisher comes along and connects to the same resource, I gain a connection to a new remote dataset. As more and more publishers connect to the same shared authority nodes, the web of Linked Data grows automatically. Clearly if we are connected by multiple nodes, not just this single one, then the richness of the network grows and possible SPARQL queries across it can be more sophisticated.

This is an exciting notion, and it's already happening all around the data world. An example from the cultural heritage field that I work in is the AHRC-funded *Seneschal* project.[8] that has turned a collection of domain thesauri (for monument and artefact types in particular) into RDF ontologies. Projects are underway to link archive datasets that use this standard terminology to the new canonical URIs with a "heritagedata.org" prefix.[9]

Personally, I think this is great. But it's not problem free. One of the issues is that there are just too many competing ontologies out there. This doesn't necessarily mean that the network won't get joined up eventually, but it would obviously be more efficient if a query didn't have to step across multiple ontologies to reach a remote data graph. I doubt if anything can be done: the usual result of setting up a committee of the great and good to

[7] I am following standard conventions for compact URIs, see http://www.w3.org/TR/curie/.

[8] http://www.heritagedata.org/blog/about-heritage-data/seneschal/

[9] Note that I am talking here of ontologies in the sense of hierarchical terminology lists. There is a separate issue about embedding standard framework schemas in local graphs – much as I used the OWL "sameAs" predicate in the example above.

resolve the problem of having *x* competing standards is that one ends up with *x*+1 competing standards.

A more insidious problem is that, if we decide not to worry about finding canonical URIs as we go along, because we can always sprinkle in some “owl:sameAs” or “skos:exactMatch” links later, we may become lazy about our local RDF design. One comes across some extremely opaque RDF – a long way from the ideal of clean, simple data that generic software agents will read and “understand”. It may become easy enough to link across to a remote dataset – via one of the established stepping stones like DBpedia or geonames – but how will our poor agent find its way about when it gets there, if the local schema is as complex and messy as the relational database it probably grew out of. I think there's a clear case for RDF schema design as an art in its own right.

The subject of schema design is too large a one to start on at this point. But let me finish with what I think is the most pressing requirement. If one accepts my premise that the nodes of the graph are the content-bearing nouns then there is a good case for these being genuinely local, with connections to established authorities whenever possible. But the relationships between them could be made much more generic: one thinks of the “Who?, What?, Where? When?” mantra, that might result in a predicate set like “has agent”, “has classification”, “has location”, “has date”. Readers may see similarities to the Dublin Core set – which at least started out as a laudably simple set of generic properties. If we can make our RDF use as simple and generic a set of predicates as possible, the vision of software agents that can interpret distributed data conveniently for us might just materialise. If we all use locally minted predicates it won't; we'll end up with the same kind of distributed query problems we see now, just based on a different technology.

References

Berners-Lee and Fischetti, 1999. Berners-Lee, Tim; Fischetti, Mark (1999).*Weaving the Web*. Harper San Francisco. Chapter 12. ISBN 978-0-06-251587-2.

Berners-Lee et al., 2001. Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284:34–43.

Byrne and Klein, 2009. Byrne, K. and Klein, E. (2009). Automatic Extraction of Archaeological Events from Text. In *Proceedings of Computer Applications and Quantitative Methods in Archaeology (CAA2009) – Making History Interactive*, Williamsburg, Virginia, USA. Selected for printed proceedings, 2010. ISBN: 978-1-905739-32-5.

Feigenbaum et al., 2007. Feigenbaum, L., Herman, I., Hongsermeier, T., Neumann, E., and Stephens, S. (2007). The Semantic Web in Action. *Scientific American*, 297:90–97.