Linked Data, Logic Programming and Black Risotto

Alessandra Mileo[1] and Aidan Hogan[1]

Digital Enterprise Research Institute (DERI)
College of Engineering & Informatics
National University of Ireland, Galway
{alessandra.mileo,aidan.hogan}@deri.org

Many of you will be familiar with the hype sweeping a number of areas of Computer Science, which consists of adding catching adjectives (like "big", "open") to the word "data". One can trace this trend back to seminal works by the Web community, which began experimenting with adjectives like "Linked", "Open" and, more recently, the very imaginative "Linked Open". But the mother of all such adjectives burst into the scene in 2001 with Laney's discovery of the adjective "Big". To this day, the full impact of the resulting noun-phrase "Big Data" has yet to be realised (see Figure 1).

Now many of you, logicians, will resist the idea of sticking adjectives in front of the word *data* as being "merely the matter of syntax" and request a more formal treatment of the semantics of this new artefact. We will get to that.

But first let x be a free variable over English adjectives. At this juncture, a Logic Programming person may be asking why should one care about what's going on in the x Data community. After all, what relevance could data possibly have to Logic Programming?

Conversely, a person from the x Data community may ask what relevance does Logic Programming have to their work. "We're already using Programs, isn't that enough?" they might ask themselves. "Why do we have to use *Logic* Programs? Isn't logic the stuff Sherlock Holmes used to solve crimes? How is *that* relevant?"

Well we'd like to discuss a little bit about how the two communities are related, and what sort of inspiration can Linked Data provide to the Logic Programming crowd.

But first, let's get back to that semantics, where we now give interpretations for x Data under three common substitutions. In the following, we assume the reader possesses some kind of axiomatic conceptualization of "data", whatever this might mean.

First, we can formally define *Big Data* as data, but bigger. Given that this is a blog post and not a journal article, we do not wish to expose all of the technical details; we instead refer the interested reader to the Oxford English Dictionary for the entry pertaining to the adjective "big" as well as the four words starting with letter 'v' that are used to define what *Big Data* is.[1]

[1] http://www.ibmbigdatahub.com/infographic/four-vs-big-data

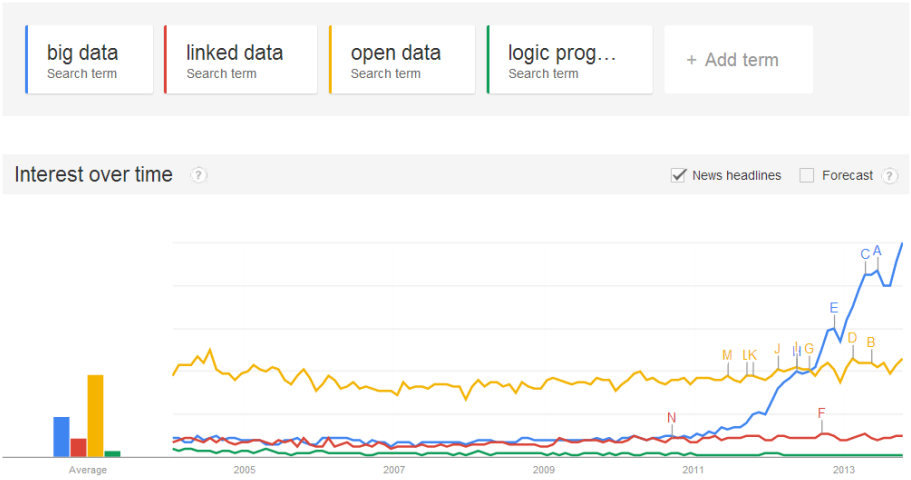

Fig. 1. Google Trends for "*Logic Programming*" and "x *Data*" where $x \in \{Big, Open, Linked\}$

Second, we can formally define *Open Data* as data you can get your hands on and, crucially, that you won't get sued for using.

Third, we can formally define *Linked Data* ... well actually, there's some non-trivial semantics associated with *Linked Data*, which some of you will be excited to hear: a well-defined notion of data and a model theory! So it is probably best to start a new section. But before we start a section called "Linked Data", we first need more motivation.

A Better Web?

Being a Computer Scientist, and therefore a keen conversationalist, you have likely been invited to varies dinner parties. Given that Computer Scientists are also renowned for their culinary skills, you will likely be asked, if not begged, to bring a dish. But for this particular hypothetical dinner party, one of the attendees is allergic to citrus and another is coeliac, which rules out your signature dishes "Duck a l'Orange" and "Fromage sur le Pain Grillé".

So you turn to Google for advice. Looking through the various recipe sites, you find lots of delicious looking dishes. Upon further examination, you find that many of these dishes contain wheat flour. So you head to a coeliac-friendly recipe site and now many of the delicious-looking dishes contain citrus fruits.

After some frustration, you eventually come across a recipe for Black Risotto, which was a personal highlight for you at the dinner banquet of a recent conference. The recipe will let you flex your considerable culinary muscle during the offline compilation phase and should trigger an amusing anecdote about the conference at runtime. And lo and behold, the recipe does not contain citrus or

gluten! Ideally the recipe should use fresh cuttlefish but if that's not available, squid can be used as a substitute. But where can you buy these ingredients nearby? You trawl through the websites of local supermarkets angling for cuttlefish but all you land with is some squid, and it's a twenty minute drive. Is there fresh cuttlefish available elsewhere? Is squid available somewhere closer? Maybe another recipe would be better? Time to go back to Google?

Well no. Eventually reality filters back in: you close down Google, settle on making a variant of "Duck a l'Orange" without orange, and get back to work.

Although the Web is unarguably pretty neat, it is not all what it could be. All of the pieces of information that you needed to successfully make Black Risotto were available on the Web. But the current Web cannot bring those pieces together for you. You have to go to different sites and piece together the puzzle yourself, which can often be like trying to find a needle in a haystack.

Instead of the current Web, close your eyes and imagine— actually that won't work. While reading this with your eyes open, imagine a future Web where the recipes from different sites were available as structured data under a common model. Imagine that the recipe sites agreed upon some basic vocabulary to use in these data: relations like CONTAINS from recipes to the things that make them, classes like DESSERT for meals that you eat when you're already full, and instances of those classes like APPLEPIE. Imagine that (as a result) the recipes from different sites could be aggregated, integrated and queried.

But when querying them, we may still struggle to express certain criteria that we are looking for. For example, we would love to be able to query for recipes that do not (or, more accurately, are not known to) contain CITRUS, but this might involve ruling out recipes with the ingredients ORANGE or LIME or LEMON or GRAPEFRUIT or TANGERINES or ... what else? And what about ingredients that contain the fruits like MARMALADE or LEMONJUICE or ...

If we could assign some lightweight semantics—like that the CONTAINS relation is transitive, or that DESSERT is a sub-class of DINNERCOURSE—this would allow for automated inference during query time. The engine could then know that because ORANGE is of TYPE AGRUME, it CONTAINS CITRUS, and thus any recipe, like DUCK A L'ORANGE, that CONTAINS ORANGE (or an ingredient containing ORANGE) also CONTAINS CITRUS, and so forth.

Now, instead of having to manually check through the ingredients of each recipe to see if it contains CITRUS, this can be done (at least partially) by automatic inference. Instead of every recipe site having to duplicate this process by defining nutritional information about ingredients, generic facts like that ORANGE is of TYPE AGRUME and that all instances of AGRUME contain CITRUS could be published by one reliable source (e.g., the FDA) and be reused (in a modular fashion) in multiple locations.

Last but not least, this future Web needs links. It needs links to connect the different pieces of information. We need links, for example, to bridge the ingredients in the recipe data and the products sold in local stores, or to link social networking sites of your friends to how they rated the recipes, or to link ... well, anything really. But these links are not necessarily hyperlinks: even

on a utopian Web, we would not be so naïve to assume that each recipe site links to each store in the world selling each ingredient. But if the store calls a CUTTLEFISH a CUTTLEFISH, (or a product that contains CUTTLEFISH a product that contains CUTTLEFISH), then linking the two sources of data becomes easy. Your recipe contains CUTTLEFISH? These stores are nearby? Well, here are the products related to CUTTLEFISH in nearby stores.

This vision of a utopian Web and the vision of "Linked Data" (aka. the "Web of Data" or, perhaps, even "Semantic Web") are pretty much isomorphic. And yes, this vision is clearly ambitious and a healthy critical review of the necessary concepts will naturally yield some pessimism. We might get to that later. But we just wanted to give you a taste of what Linked Data is about before going further.

Linked Data in brief

Linked Data is (based on) four principles:

1. Use URIs to name things (not just documents)
2. Use HTTP URIs (that can be looked up)
3. Return a description of the thing when its URI is looked up (using RDF)
4. Provide links to external data

To explain, let's just view the Web as a global map. Currently the keys are URLs and the values are documents: you look up a URL via HTTP and you get a document. The core idea of Linked Data is similar to viewing the Web as a map, where the keys are URIs that identify things like recipes, ingredients, stores, people, etc. (not just documents). Looking up a key via HTTP returns you a new kind of value: a structured description of the thing identified by the key. The structured description is modeled in *RDF* (we'll get to that in a minute) and embeds keys for related things. That's pretty much what Linked Data is: keys are URIs for things, lookups are done over HTTP, and the values returned are structured descriptions of those things containing keys of related things. Hence Linked Data is often seen as forming a "Web of Data" that co-exists with the traditional "Web of Documents".

The notion of "Data" in Linked Data is concrete, well-defined and standardised: it's RDF. RDF is a data model built with the Web in mind and is based on two foundational aspects: triples and RDF terms. Triples are facts with a fixed arity of three and a fixed predicate: $T(subject, predicate, object)$. So you can say things like $T(\text{ORANGE}, \text{CONTAINS}, \text{CITRUS})$. Sometimes these facts are thought of as binary relations $\text{CONTAINS}(\text{ORANGE}, \text{CITRUS})$; same thing. Triples are great because we can deal with sets of them. When we have two sets of triples from, say, two different Web documents, we can just union/merge them into one set (though in reality, it's often important to track where triples came from). And because triples have fixed arity, they are quite easy to process. For example, we can just store them as rows in a table with three columns.

But is ORANGE a fruit or a colour? URIs are used to identify things in RDF; these are globally unique identifiers on the Web. So we can create a URI `http://example.org/food/Orange` and say that it refers to the fruit. URIs are abbreviated with prefixes like `exf:` which refers to `http://example.org/food/`. So now RDF triples look more like $T($`exf:Orange`, `exf:contains`, `exf:Citrus`$)$. We know we're talking about orange the fruit. If it were in doubt, we could add two more triples to remove any doubt like $T($`exf:Orange`, `rdf:type`, `exf:Agrume`$)$ and $T($`exf:Agrume`, `rdfs:subClassOf`, `exf:Fruit`$)$. Now we know that the thing referred to by `exf:Orange` is an instance of `exf:Agrume` and all instances of `exf:Agrume` are also instances of `exf:Fruit`.

Now, per the principles of Linked Data, we have a URI like `exf:Orange` that identifies a thing, in this case the fruit orange; and we can look up that URI to get a structured description of that thing, in this case, a set of triples describing `exf:Orange`.

Along with the RDF data model, RDFS is a standard semantic extension of RDF which allows to express simple taxonomies and hierarchies among properties and resources, as well as domain and range restrictions for properties. According to this standard semantics of properties like $rdf : type$ and $rdfs : subClassOf$, we now know that $T($`exf:Orange`, `rdf:type`, `exf:Fruit`$)$ without having stated it explicitly. RDF and RDFS provide a number of terms with standard semantics like this and standard (Datalog-style) mechanisms for inference.

And aside from RDF and RDFS, there are further standards such as OWL: a complex but decidable ontology language representable in RDF based primarily on Description Logics, and RIF: a format for representing and exchanging rules. Such standards allow RDF to be "self-describing" by embedding the semantics of terms into the RDF descriptions themselves: the data and the semantics of the data can be packaged together and interpreted by any tool that understands the core semantics of RDF/RDFS/OWL. And lest we forget, there's also a standardised query language: SPARQL, which is a bit like SQL but tailored for querying RDF.

So in summary, we have Linked Data, based primarily on RDF and URIs. And on top of that, we have a collection of Semantic Web standards for getting things done with RDF; namely RDFS, OWL, RIF, and SPARQL.

In fact, we have all the necessary ingredients for the future Web we spoke about earlier: a structured data format, a way of publishing structured data on the Web, some well-defined languages that enable inference, methods for linking data, and query languages to make specific requests. But we're still very very far away from that vision.

Linked Data today

Returning to the vision of the future Web and reasons why one might be pessimistic, we can try to anticipate and address some of those concerns.

First, not only will publishers structure their data and agree on terms, they already do. A Google search for "black risotto" will show various rich snippets

with images, ratings, preparation time, etc., all based on structured data that publishers already provide. Publishers don't even need to structure their data: most web-sites are powered by databases, so publishers merely need to choose to expose the structure of their data. Structured data promotes clickthroughs and commerce.

Second, yes, Web data can be messy: errors, spam, etc. If you universally trust all data you find, even on the future Web, the only recipe you'll find is for disaster. No sane person would advocate trusting arbitrary Web data. But the very same lessons we've learned from the current Web apply to the future Web. Choose your sources carefully. Look at the network of links. Apply social rating and feedback schemes. Trace provenance.

Third, yes, the Web is big. Running inferencing and querying over structured data at "Web scale" is challenging. Transitive closure is expensive and not easily parallelised. But applications rarely need to run such algorithms over the entire Web. Partition your problem. Select only relevant, trustworthy data-sources. Then run inferencing and querying over combinations of partitions. Intuitively, this might give us a chance to dare and ask for 'a little more' reasoning on the Web (or rather on its relevant portion) such as "What is the *most preferred alternative* to cuttlefish?" or "How many ways are there to prepare black risotto?" and "Which one is the easiest? And the fastest? And the cheapest?".

In fact, Linked Data has already had moderate success in getting structured data published on the Web. There's a number of governmental organisations, media and retail companies, scientific communities and social sites publishing data according to the Linked Data paradigm: check out `http://lod-cloud.net/`. Some 40 billion facts have been published across hundreds of linked datasets in the past few years, and yet despite this wealth of openly available structured data in a common format, there are some key ingredients missing.

Among those, we want to focus on what we think might capture the attention of the Logic Programming community, and trigger more interest in Linked Data.

'A little more' Reasoning

If we look at the Semantic Web Stack, we can see layers named "rules", "logic" and "proof".[2]

In recent years, the theme of Reasoning for the Semantic Web had the Logic Programming communities lining up. We believe they are intrigued and provoked by the theoretical challenges of combining Classical logic (into which OWL and, in general Description Logic fall), and Logic Programming (into which Prolog and ASP [1, 2] fall). Such challenges are related to the crucial differences between Classical Logic and Logic Programming. These include the differences between default and strong negation vs. classical negation [3], as well as the way decidability is ensured.

[2] http://www.w3c.it/talks/2005/openCulture/slide7-0.html

Along these formal considerations, some concrete approaches have been investigated to combine rules (in Logic Programming sense) and ontologies [4] to provide 'a little more' Reasoning for the Semantic Web. But if we consider the Semantic Web vision and what Linked Data enabled in reality, the toy problems will not be relevant in the context of the ocean of (structured and linked!) data. In fact, leaving the theoretical challenges unsolved (or partially solved) would make reasoning (in the Logic Programming sense) on the Semantic Web totally unfeasible, unless... Unless a bottom up approach is combined with the top-down, theory driven investigation. Unless the problem - and the Web - is distilled into a relevant subset of its Data, relevant for the task at hand.

What if we refocus our attention from Semantic Web theoretical underpinnings (Description Logic) to its concrete instantiation, namely Linked Data? Can this provide a new inspiration for combining Linked Data and Logic Programming? What if we characterize and isolate some of the peculiarities of x in "x Data", maybe starting from the trendiest $x = Big$ and each of the salient 'v' words attributed to Big Data? Could we come up with better (scalable, more expressive) reasoning solutions, based on Logic Programming, for x Data?

At this point we think we made a connections between (Linked) Data and Logic Programs, and the reader might be interested in our intuitions expressed in a more explicit manner. Assuming we distill the Web (and thus Linked Data), maybe using data links or peculiarities of x Data itself, can Linked Data be used as a (remarkably rich) source of "facts" for problem solving? If we cannot make our partition consistent through RDFS/OWL, due to missing information, maybe Logic Programming can help? The reasoning capabilities that embed nonmonotonicity, decidability, incomplete knowledge (in the Logic Programming sense) are especially important here. To follow up on our initial culinary matter, we think this would allow us to gain deeper insights into the problem of the black risotto, save money and time, or become experts in alternative procedures to optimize one or more of these criteria when needed. But we are sure there are more questions, beyond cooking, where a deeper tie-in between Logic Programming and Linked Data can open new opportunities, mostly unexplored to-date, which could bring real Linked Data a few steps closer to the potential of the future Web pictured in this essay.

References

1. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
2. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proc. of the 5th Int'l Conference on Logic Programming. Volume 161. (1988)
3. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9** (1991) 365–385
4. Eiter, T., Ianni, G., Krennwallner, T., Polleres, A.: Rules and ontologies for the semantic web. In Baroglio, C., Bonatti, P.A., Maluszyński, J., Marchiori, M., Polleres, A., Schaffert, S., eds.: Reasoning Web. Springer-Verlag, Berlin, Heidelberg (2008) 1–53