Dandelion: from raw data to dataGEMs for developers

Stefano Parmesan, Ugo Scaiella, Michele Barbera, and Tatiana Tarasova

SpazioDati S.r.l. HQ,
Via Adriano Olivetti, 13, "Le Albere", 38122, Trento (TN), Italy
{parmesan,scaiella,barbera,tarasova}@spaziodati.eu
http://spaziodati.eu/

Abstract. The amount of data generated on the Web is increasing day by day, but it is not always accessible and usable: developers have to go through numerous steps from obtaining the data to building applications on top of it. The mission of SpazioDati is to eliminate these steps from the development process. SpazioDati pulls the data from various disparate data sources together in one big knowledge graph, Dandelion, and provides a set of APIs over it. In this paper we provide an overview of the data curation and data access platform we develop at SpazioDati.

Keywords: data curation, knowledge graph, API

1 Problem Statement

An ever increasing amount of data is generated everyday and it is made available on the Web in structured and semi-structured forms. Three major trends are contributing to the original vision of the Semantic Web as a unique global database. First, W3C's Linked Data activity[1] promotes interoperability and facilitates machine access to data produced by different providers.

Second, a growing pressure on Governments and Public Administrations to publish their data on the Web and to promote its reuse has generated an increasing amount of valuable open information. Despite the intrinsic value of this information, its usability is hampered by the proliferation of several formats, which often lack the explicit semantics necessary to facilitate its programmatic reuse.

Third, private and corporate information producers have emerged as key data producers: they sell or freely offer data for third-party consumption via REST APIs. Yet, each API has its own interface, data semantics and terms of use, which complicates the re-use of information.

Beside these trends, virtually the majority of all information published on the Web is in the form of unstructured documents.

The current priority issue for Semantic Web and Linked Data experts is the shortage of enterprise-class applications able to creatively recombine and reuse Web data. We argue that the shortage of enterprise-grade applications is not caused by a lack of demand, but rather by the complexity of finding, meaningfully aggregating, repurposing, and finally reusing heterogeneous data. A first step to initiate a growth-enhancing

[1] http://www.w3.org/standards/semanticweb/data Last Accessed: July 2014

market process is the emergence of a cluster of data brokers performing a twofold role. On the one hand, they need to guarantee a simple and coherent access to information; on the other hand, they need to certify its quality for reuse within the enterprise.

Most of the major Web enterprises are gathering and merging Web data into their own private "Knowlege-Graphs", that they leverage to enhance search and other value-added services[2]. These valuable "Private Linked Data Clouds", often complemented with tools that facilitate linkage of unstructured documents to the graph, are considered as core assets and not made accessible to third-parties. We discuss existing related work in Section 2.

The mission of SpazioDati is to act as a data-broker by offering an enterprise-grade Knowledge-Graph-as-a-Service. We present our approach in Section 3 and conclude with the discussion on future directions in Section 4.

2 Related Work

There are many services and tools that make data accessible to the masses; some of them focus on facilitating accessibility, making data quickly usable by means of rich APIs, while others focus on the semantics of the data, allowing very powerful interactions and applications.

Products in the first category usually work on tabular data, e.g., Factual[3]. In some cases they allow one to upload custom data and manipulate it, e.g., DataMarket[4]. Such tools provide – at different degrees – both access to the data through APIs and useful visualisations built automatically on top of them. However, the data remains locked in tables, i.e., *verticals*. It requires additional efforts to integrate this data with other sources.

The second category of products, instead, follows a graph-approach. Sources as DBpedia[1], Freebase[2] and Yago[5] for example, provide cross-domain knowledge by means of a knowledge graph. In contrast to the works mentioned above, such services allow *horizontal* applications and give developers the ability to access data that broadly covers many different parts of the human knowledge. However, to access them programmatically, one must be knowledgeable in Semantic Web technologies such as SPARQL, RDF, which is not the case with an average programmer.

There are also technologies that try to merge the two worlds: the Linked Data Platform[5] is a W3C candidate recommendation for a Linked Data architecture; Apache Marmotta[6] is one of its open implementations, and allow developers to build their own knowledge graph. But the whole process is complicated, and often not affordable by small companies.

---

[2] For example, the Google's Hummingbird algorithm uses its Knowledge Graph to improve search results `http://www.icrossing.com/sites/default/files/Google-Hummingbird-Explained-iCrossing-POV.pdf` Last-accessed: July 2014

[3] `http://www.factual.com/` Last-accessed: July 2014

[4] `https://datamarket.com/` Last-accessed: July 2014

[5] `http://www.w3.org/TR/2012/WD-ldp-20121025/`

[6] `http://marmotta.apache.org/` Last-accessed: July 2014

3 The Dandelion Approach

Dandelion is a knowledge graph of places, events, organisations, people and other information that we are building at SpazioDati. We collect data from numerous proprietary and open data sources, harmonise the data, store it within our infrastructure and provide access to it from a single API. Figure 1 illustrates the process we follow and tools and technologies we leverage to build Dandelion.

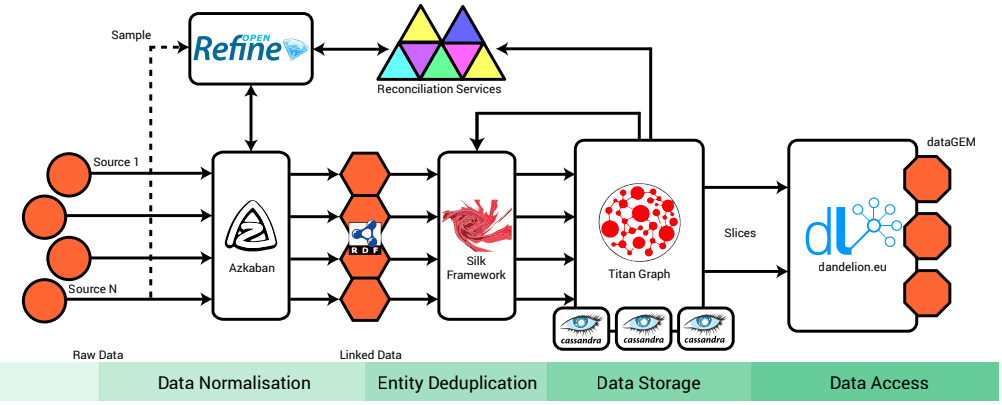

Fig. 1. A schematic view of the SpazioDati pipeline, with each step highlighted on the bottom line.

3.1 Data Normalisation

The process of data normalisation includes several steps, among which data cleaning and data harmonisation. We rely on OpenRefine[7] to implement these steps.

Data cleaning We apply various rules to transform data into standard representation formats. For example, to represent geographical information, we re-use standards developed by the Open Geospatial Consortium[8]. Our Refine extension, geoXtension[9], enables conversion between various projections. Another Refine extension that SpazioDati is actively developing is the Named-Entity Recognition (NER) extension[10].

Data Harmonisation Input data for our knowledge graph is highly heterogeneous, as it comes from different sources, organisations and domains. We rely on ontologies to resolve semantic heterogeneity of the data. To build ontology we use Neologism[11].

We implement a master ontology and local ontologies, that formalise original data sources. The purpose of the master ontology is to provide a common shared vocabulary for original data sources. To add new data into our knowledge graph, we, first,

[7] `http://openrefine.org/` Last-accessed: July 2014
[8] `http://www.opengeospatial.org/` Last-accessed: July 2014
[9] `https://github.com/giTorto/geoXtension`
[10] `https://github.com/RubenVerborgh/Refine-NER-Extension`
[11] `http://neologism.deri.ie/`

formalise it in a local ontology. Second, we define mappings from local to the master ontologies. Currently, the most developed domain of the Ontology is *organisations*. To build vocabulary of organisations, we re-used existing ontologies and vocabularies, such as the W3C Organization Ontology[12] and the Registered Organization Vocabulary[13].

To overcome syntactic and structural data heterogeneity, we transform all data into RDF using the Refine extension for exporting RDF[14]. RDF mappings implement a "table-to-class and column-to-predicate" approach:

- One row corresponds to one entity of a class defined by a table. Every entity in our knowledge graph is uniquely identified by a URI, called *acheneID*. Normally, we build acheneIDs out of unique identifiers that can be found in a data source.
- Columns in the table correspond to the properties of the entities. We map columns to properties of the master ontology.

3.2 Entity Deduplication

The next step in our data curation pipeline is entity deduplication. This is an essential step in obtaining a connected knowledge graph out of disperse independent data sources.

After entity mapping performed at the end of data normalisation, each source is transformed in a single, separated graph stored in Virtuoso[15]. Before inserting entities in the knowledge graph, we need to make sure that two entities in two different sources, representing semantically the same object[16], are merged in a single entity containing the data of both.

The Silk Framework [6] is used to deduplicate entities. With Silk it is possible to quickly calculate a similarity measure on any two entities, given a set of matching rules defined by the user; such rules are also used to automatically create a blocking algorithm to reduce the number of comparisons needed to match two large data sources[4].

Before importing new data from a source, the data itself is matched with Silk against the whole knowledge graph to find duplicated entities. As output Silk produces a list of `owl:sameAs` links that are then used to merge entities when importing the new data.

3.3 Data Storage

The knowledge graph is stored in a graph database, Titan[17], which runs on top of a Cassandra[18] cluster, and allows to store key-value maps on either vertices and edges and to submit queries using Gremlin[19] for fast traversals. Entities are not simply stored as a single vertex, because it is important to keep track of the provenance of each information stored on it. In the knowledge graph therefore four different kinds of vertices can be found, (see figure 2):

[12] `http://www.w3.org/TR/vocab-org/`
[13] `http://www.w3.org/TR/vocab-regorg/`
[14] `http://refine.deri.ie/` Last-accessed: July 2014
[15] `http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/` Last-accessed: July 2014
[16] i.e.: the Wikipedia page of the Leaning Tower of Pisa and its OpenStreetMap geometry should not generate two different entities in the graph.
[17] `http://thinkaurelius.github.io/titan/` Last-accessed: July 2014
[18] `http://cassandra.apache.org/`
[19] `https://github.com/tinkerpop/gremlin/wiki`

– *achene nodes* – the square nodes, they represent an entity; they do not store any kind of information with the only exception of the PURL[20] of the entity they represent;
– *bristle nodes* – the circular nodes, they are connected to achene nodes and store the actual data associated to an entity; an achene node may have multiple bristle nodes, one for each source from which it was imported;
– *provenance nodes* – the triangular nodes, they are connected to bristle nodes to represent the source that provided the information stored on each bristle;
– *type nodes* – the hexagonal nodes, they represent our entity taxonomy and are used to keep track of the type of each entity (e.g., Company, Person, POI, Geographical Location, etc.)

Entities in the knowledge graph are therefore represented by one achene node and multiple bristle nodes. Bristles and edges store semantic information using the master ontology as well as other public ontologies. The entity name will therefore be stored on the bristle as `name`, while link between type nodes will be labelled `rdfs:subClassOf`.

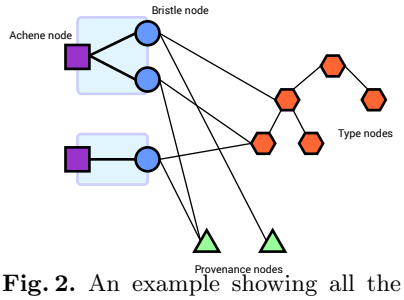

Fig. 2. An example showing all the different kinds of vertices that are used in the knowledge graph.

3.4 Data Access

Once data is available as a consistent and well-defined graph, we have to let developers access the data, so that they can get and browse the data programmatically.

As we mentioned at the beginning, this is not a simple task because not all developers are able to deal with graph data and build queries to browse and get what they really need for. Given this, the ultimate goal is to simplify as much as possible accessing the data: we argue that the most basic data structure that can be easily integrated and digested by an average programmer is the table.

Our idea is to provide a simplified access to the data, by means of *slices* of the graph: we identify a specific partition of graph, in terms of: types of nodes, set of properties for each node, and set of properties for each linked node, usually by traversing the graph with a limited number of steps.

Then data is formatted into a table where rows are nodes of the graph and columns are filled with values of selected properties. Possibly, properties are collections or collections of objects, that contain information coming from graph traversal. We can create any number of slices, changing the way to select nodes and the set of properties to be included.

These slices are called dataGEM: developers can access them using a REST-like web API and standard HTTP parameters to query the data in a simple manner. We argue that this model has several benefits with respect to a single and complex SPARQL endpoint:

[20] Permanent URL

– Data is centralised and can be better maintained thanks to the graph structure. At the same time, data is available through tables that are well-known and very easy to understand and deal with for all developers.
– Through dataGEM it is possible to realise different views on the graph using the same data sources. Having a unique data source enables interoperability between dataGEM and allows for easier updates and maintenance of the data.
– Open and proprietary data can coexist and be linked in the same graph because open data can be included in a free dataGEM, while proprietary data can be sliced into a private one for which paid subscription is required.

4 Conclusion and Future Work

We presented our infrastructure for content curation and the innovative approach for simplifying access to semantically structured data to let all developers that are not Semantic Web specialists add value to their applications.

In addition to this approach, we will also let developers access the graph using our text analytics API, called dataTXT. dataTXT is the evolution of a state-of-the-art algorithm [3], and it is able to identify on-the-fly and with high accuracy, meaningful sequences of terms in unstructured text and link them to a pertinent DBpedia resource. dataTXT solves ambiguity and synonymy by means of the knowledge graph extracted from DBpedia, but in the future we plan to extend this approach. DBpedia will be kept as a backbone graph that is used to disambiguate and provide context for common topics, but thanks to extended Dandelion's knowledge graph, dataTXT will be able to link also specific entities coming from external sources. Actually, this is another approach for accessing the graph: given an unstructured text, dataTXT can help developers to link it to the Dandelion's knowledge graph adding structure and semantics to plain texts.

References

1. Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia – a crystallization point for the web of data. *J. Web Sem.*, 7(3):154–165, 2009.
2. Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.
3. Paolo Ferragina and Ugo Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1625–1628, New York, NY, USA, 2010. ACM.
4. Robert Isele, Anja Jentzsch, and Christian Bizer. Efficient multidimensional blocking for link discovery without losing recall. In Amélie Marian and Vasilis Vassalos, editors, *WebDB*, 2011.
5. Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.
6. Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk – a link discovery framework for the web of data. `http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/`.