http://webdam.inria.fr

Web Data Management

Putting into Practice: Ontologies in Practice (by Fabian M. Suchanek)

Serge Abiteboul
INRIA Saclay & ENS Cachan

Ioana Manolescu
INRIA Saclay & Paris-Sud University

Philippe Rigaux
CNAM Paris & INRIA Saclay

Marie-Christine Rousset
Grenoble University

Pierre Senellart
Télécom ParisTech

`http://webdam.inria.fr/Jorge/`

Contents

1 Exploring and installing YAGO **2**

2 Querying YAGO **2**

3 Web access to ontologies **4**
3.1 Cool URIs 4
3.2 Linked Data 5

This chapter proposes exercises to manipulate and query real-world RDFS ontologies, and especially YAGO. YAGO was developed at the Max Planck Institute in Saarbrücken in Germany. At the time of this writing, it is the largest ontology of human quality that is freely available. It contains millions of entities such as scientists, and millions of facts about these entities such as where a particular scientist was born. YAGO also includes knowledge about the classes and relationships compositing it (e.g., a hierarchy of classes and relationships).

1 Exploring and installing YAGO

Go to the YAGO Web site, `http://mpii.de/yago`, click on the "Demo" tab and start the textual browser. This browser allows navigating through the YAGO ontology.

1. Type "Elvis Presley" in the box. Then click on the Elvis Presley link. You will see all properties of Elvis Presley, including his biographic data and his discography.

2. You can follow other links to explore the ontology. Navigate to the wife of Elvis Presley, Priscilla.

3. The ontology is held together by a taxonomy of classes. Its top class is called "entity". Verify this by repeatedly following `type` and `subClassOf` links.

4. Go back to Elvis Presley. Navigate to one of his songs. You will see the date the song was recorded. Can you find all songs together with their record dates? Why would this be a tedious endeavor?

Then, to install YAGO on your machine, make sure that you have Java installed and around 5 GB free disk space. Proceed as follows:

1. Download the YAGO ontology. YAGO is available at the project homepage `http://mpii.de/yago`. We also provide a version of YAGO on the Web site of this book, `http://webdam.inria.fr/Jorge/`. There are multiple versions of the ontology; the easiest is to download the Jena TDB store, that can be directly queried. Select the smallest data set available. Save the file on your hard drive and unzip it. This may take some time.

2. Download the YAGO converters from the same site. Unzip the file to your hard drive.

You are all set, YAGO is installed on your machine and is ready for querying!

2 Querying YAGO

YAGO is expressed in RDFS. In RDFS, the facts and the ontological statements are written as triples. These triples can be queried using SPARQL , the standard querying language for RDFS facts. SPARQL was introduced in Chapter **??**. The query engine of YAGO uses the Jena framework `http://openjena.org/`. Jena is an open-source project that has grown out of work with the HP Labs Semantic Web Program. Jena ships with YAGO, so that we only need to download and install YAGO.

To query YAGO, open a terminal window, navigate to the folder where the converters live and run the SPARQL script (called `yago2sparql.bat` on Windows and `yago2sparql.sh` on Unix). You will be invited to type SPARQL queries for YAGO. For ease of notation, the following namespaces are already defined:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

In addition, the default namespace, referred to as simply ":", is already set to the namespace of YAGO, `http://www.mpii.de/yago/resource/`. We can ask for simple YAGO facts through SPARQL queries of the form

```
SELECT ?V WHERE { A R B }
```

Here, ?`V` is a variable name, as indicated by the question mark. The `SELECT` clause may contain several variables separated with whitespace. `A` and `B` are entities (with proper namespace prefix) and `R` is a relation (also with proper namespace prefix). Each of these components may also be a variable. The `WHERE` clause possibly contains multiple triples, separated by a dot. Try out the following:

1. Ask

   ```
   SELECT ?x WHERE { :Elvis_Presley rdf:type ?x }
   ```

   This query lists all classes that Elvis Presley is an instance of. (Be sure to type all characters in the query exactly as written here.) Note that the results show the full URI of the entities, not the equivalent short form with namespace prefixes.

2. Can you modify the query so that it lists all facts with Elvis Presley in the position of the subject, not just the ones with the `rdf:type` relation? You should find facts about his songs, his wife, his birth date and the movies he acted in.

3. List all the entities that Elvis created (with the relation `:created`). Now list only those of them that were created on 1974-03-20. Use the relation `:wasCreatedOnDate` and remember to put quotes around the literal `1974-03-20`. Your query should retrieve the entity `:Good_Times_(Elvis_Presley_album)`. Can you imagine why the entity is not simply called `:Good_Times`?

4. Knowing that `rdfs:label` is the relation between an entity and its name, retrieve all entities that are called "Good Times".

5. Write another query to list all entities called "Good Times" together with their creator. Compare the results of this query to the results of the previous query. Verify that not all of the entities have a creator. Can you imagine why this is the case, even though every entity was certainly created? What does this imply for the notion of negation in SPARQL?

6. Write another query to retrieve all classes that Elvis Presley is an instance of. Is Elvis an instance of the class `singer`? Can you imagine why?

   In YAGO, the ontological statements expressing the subclass and subproperty relations as well as the range and domain restrictions of properties are stored as RDFS triples. However, the semantics of these statements is not taken into account. In particular, the facts that follow from the RDFS entailment rules (see Chapter **??**) are not derived. To derive these facts, one can use the saturation algorithm given in Chapter **??**. It is possible, using the converters, to generate a Jena store of YAGO that includes (some of) these derived facts. This requires, however, downloading the YAGO2 ontology in its default format, and the conversion process can be a lenghty one.

Enter a blank line to quit the SPARQL interface.

3 Web access to ontologies

3.1 Cool URIs

An ontology refers to an entity through a URI. YAGO, for example, refers to the entity of Elvis Presley as `http://mpii.de/yago/resource/Elvis_Presley`. Another ontology may use a different identifier to refer to that entity. The DBpedia ontology, for instance, refers to Elvis as `http://dbpedia.org/resource/Elvis_Presley`. In general, these URIs (i.e., identifiers) do not have to be URL (i.e., locators). In other words, they do not have to refer to Web pages. In principle, when a URI is entered in a browser, one might simply get an error message. However, some ontologies implement the "Cool URI" protocol[1] of the W3C. This means that each URI in the ontology is actually understood by a Web server that is configured to respond to a request of this URI. (In other words, each such URI is also an URL.) This allows a machine to retrieve fragments of the ontology from the server. Let us try this out:

1. If you do not have a Unix-based operating system, search online for a version of the tool `wget` that works with your operating system. There should be a free version available. Download and install this tool. `wget` allows accessing a URL and downloading its content – much like a Web browser does it.

2. Open a terminal and type

   ```
   wget -O elvis.html http://dbpedia.org/resource/Elvis_Presley
   ```

   This accesses the URI as a URL, just like a Web browser. If you look into `elvis.html`, you will see the Wikipedia page of Elvis.

3. Now we tell `wget` to ask the server specifically for RDFS files:

[1] `http://www.w3.org/TR/cooluris/`

```
wget -O elvis.rdfs --header "Accept: application/rdf+xml"
     http://dbpedia.org/resource/Elvis_Presley
```

The file `elvis.rdfs` should now contain everything YAGO knows about Elvis Presley. The file format is RDF, encoded in XML.

4. These facts in the answer contain again other URIs. Find one of Elvis's albums in the answer. Then use another `wget` command to retrieve information about that album.

By following the URIs in the results, a machine can navigate the entire ontology.

3.2 Linked Data

As we have seen, different ontologies can use different URIs to refer to the same entity. The Linked Data Project, found at `http://linkeddata.org/`, tries to establish links between such synonymous URIs. Such a link takes the form of an RDFS statement. The predicate is `sameAs` of the OWL namespace:

```
              http://mpii.de/yago/resource/Elvis_Presley
owl:sameAs    http://dbpedia.org/resource/Elvis_Presley
```

These links allow jumping from one ontology to another. If both ontologies implement the Cool URI protocol, a machine can gather information about one entity from multiple servers. Let us try this out: Go to the Web site of the *Sig.ma* semantic Web search engine, `http://sig.ma/`. This engine gathers information from different ontologies about a given entity. It uses `sameAs` links, Cool URIs, and RDFa annotations hidden in HTML pages[2]. This leads to a lot of data, but potentially also very noisy data. Ask Sig.ma for

`http://mpii.de/yago/resource/Elvis_Presley`.

You can also try out keywords (such as "United States"). See how Sig.ma gathers data from multiple ontologies. The Linked Data project was pioneered by the DBpedia ontology, which is therefore a hub in this Web of data.

[2] `http://www.w3.org/TR/rdfa-syntax/`