Working with Knowledge Graphs

Eight exercises and a challenge

Markus Krötzsch, TU Dresden
EDBT Summer School 2019, Lyon, France

Knowledge graphs are an important asset for many AI applications, such as personal assistants and semantic search, and a valuable resource for related fields of research. They are also, of course, a conceptual umbrella that spans rather different methods and approaches in the intersection of databases, knowledge representation, information extraction, knowledge management, and web technologies. This course focusses on the effective handling and usage of knowledge graphs, in particular data access, query answering, and quality analysis.

This paper compiles eight tasks to exercise and deepen what you learned in the course, and it concludes with an open challenge that asks you to creatively combine this with other techniques. We will focus on *Wikidata* (the community-built, multi-lingual knowledge graph of Wikipedia), but also work with DBpedia (a family of knowledge graphs built by information extraction from Wikipedia articles in a particular language).

1 Resources

The following resources and services will be helpful to solve the below tasks:

- Wikidata SPARQL Query Service: https://query.wikidata.org/ (user interface;[1] use the *Examples* menu to view many queries and to filter them by topic and SPARQL feature)
- DBpedia SPARQL service: https://dbpedia.org/sparql
- SQID Wikidata browser: https://tools.wmflabs.org/sqid/ (alternative view on Wikidata)
- SPARQL help: most questions are best answered with a simple web search; the official reference is the W3C specification: http://www.w3.org/TR/sparql11-query/
- SPARQL & RDF for Wikidata: https://www.wikidata.org/wiki/Help:SPARQL (comprehensive documentation)

2 Exercises

Part A: Warm up

Task 1. Familiarise yourself with the data in DBpedia and Wikidata, and with its encoding in RDF, by looking at the following resources:

- Take a look at the Wikidata page for *Saint-Germain-au-Mont-d'Or* (https://www.wikidata.org/wiki/Q1617721). Note that all properties and items have numeric identifiers, such as Q142 for *France*, which you can see from their URLs.
- View this data in the SQID browser: https://tools.wmflabs.org/sqid/#/view?id=Q1617721. Find out who was born in this town using the tab *From related entities*.

[1] The web service for programmatic SPARQL queries against Wikidata is at https://query.wikidata.org/sparql

- View the data in RDF: http://www.wikidata.org/wiki/Special:EntityData/Q1617721.nt. How is the current population (including its time of validity and reference) encoded here? How many triples are there if you ignore the multi-lingual labels and descriptions?
- View related data in DBpedia: http://dbpedia.org/page/Saint-Germain-au-Mont-d'Or. Compared to Wikidata, which additional data can be found? And what is missing?
- In the Wikidata SPARQL query service, use the query

```
SELECT * WHERE { wd:Q1617721 ?p ?o . }
```

  to view all RDF triples where the town is the subject.
- In the DBpedia SPARQL query service, use the query

```
SELECT * WHERE { dbr:Saint-Germain-au-Mont-d\'Or ?p ?o . }
```

  to view all RDF triples where the town is the subject.

Task 2. Find your home town or home institution on Wikidata. Use the *add* and *edit* links to add some missing data to it. Try to include a reference (compare with other entities that have this information to see what is appropriate). Note: no login is needed, but IP addresses will be on public record for edits made without login.

Part B: Working with SPARQL

Task 3. Using the Wikidata SPARQL endpoint, find queries for each of the following tasks, or argue why the task cannot be solved. Some of these queries have millions of results – use LIMIT for testing.

(a) All child–parent pairs.

(b) All things that have an ancestor who is not human.
Hint: Negation is expressed with not exists. Humans are instances of (P31) entity Q5.

(c) All mothers who have an article on English Wikipedia.
Hint: English Wikipedia pages are identified by http://en.wikipedia.org/
Wikipedia pages are linked to their entities via schema:about.

(d) Indirect ancestors who are related via a chain of one or more intermediate ancestors, all of whom have articles on English Wikipedia.
Hint: For any SPARQL query, how many nodes of degree >2 are needed for a match?

(e) All present-day countries and, where available, their current head of government. (Note: Germany should be among the results, but the historical state of Burgundy should not.)

(f) All countries that existed in 2009 with their head of government at that time.
Hint: Look at the Wikidata example query „Largest cities with female mayor".

(g) All administrative regions that TU Dresden is currently contained in.

(h) All administrative regions that TU Dresden was contained in in 1980.

(i) For any human x, the number of pairs of ancestors a and b that it connects, i.e., where a has ancestor x and x has ancestor b.

(j) The length of the longest simple path along the parent (mother or father) relation between any two entities.

Task 4. For all tasks from the previous problem, find SPARQL queries that solve them on the DBpedia SPARQL endpoint, or argue why the task cannot be solved.

Part C: Querying and data analysis with rules

The following tasks can be solved using the VLog4j rule engine. An easy starting point is the example project https://github.com/knowsys/vlog4j-example, which can be cloned and modified as needed.

Task 5. Use VLog4j to integrate and compare *parent* relations from DBpedia and Wikidata.

In detail, use SPARQL queries to fetch *parent* relationships, restricting to items with articles on English Wikipedia for Wikidata. Augment the queries to contain a common feature that can be used for information integration, e.g., the URL of the related Wikipedia article.[2] Then use rules to compute the total number of (unique) relations found in both graphs, in Wikidata only, and in DBpedia only.

Task 6. English DBpedia is based on entities that have an article on English Wikipedia. Even if we are only interested in such entities, one can imagine cases where *indirect* relationships are missed due to the absence of other entities that make a connection.

Investigate if this is the case for the "ancestor" relationship: are there any pairs of ancestors in English Wikipedia that are not related by any chain of intermediate ancestors that are also in English Wikipedia? Use negation in VLog4j find missing relationships.

Task 7. A common type of error in knowledge graphs are cycles in relationships that should be hierarchical, such as *subclass of*, *part of*, or *located in*. Such errors are hard to spot since they cannot be seen from single entities.

Analyse the *mother* relationship of Wikidata for cycles, i.e., find all items that have a chain of *mother* relations to themselves. SPARQL can answer this query, but it cannot compute the cyclic paths causing the problems. Design a rule-based query that computes these paths, using existential quantification to introduce elements that represent paths. Try your query in VLog4j. Using a timeout might be helpful for testing and debugging.

Hint: To avoid non-termination, model set operations as shown in the course.

See also: Carral et al.: Chasing Sets: How to Use Existential Rules for Expressive Reasoning. IJCAI 2019.

Task 8. Instead of querying SPARQL, use the partial RDF export file found at https://doi.org/10.5281/zenodo.3379077, which contains (among other data) triples that encode full *mother* statements of Wikidata. Using the qualifier information in the RDF file, filter cases of adoptive parents, foster parents, etc. that suggest that there is no biological relationship. Compare your results with those of Task 7.

Hint: Use SPARQL (or grep) to find relevant values for qualifier type of kinship (P1039).

Bonus: Depending on the size of your working memory, you can also try to solve Tasks 7 and Tasks 8 for all parents (fathers or mothers). An RDF file with all father statements is at https://doi.org/10.5281/zenodo.3379079. You may have to experiment with various approaches of writing the rules to optimise performance. Another memory-saving technique is to compute some intermediate results and write them to disk (CSV export) first, and to draw further conclusions from this intermediate result in a second run.

[2] DBpedia uses the outdated "http" protocol for Wikidata and Wikipedia URLs. SPARQL string operations such as `SUBSTR` can be used with `BIND` to align this with "https" URLs stored in Wikidata.

3 Challenge

We present an open challenge that may require a wide range of diverse techniques to be solved, such as information extraction, data mining, machine learning, NLP, or crowd-sourcing.

Problem description Wikidata is becoming a huge repository of data, and many entities are described in great detail. However, not all of the data is equally important. One can ask for at least two different types of "importance" among the data associated with an entity:

1. **Claim to fame:** Which data is the essential reason for the notability of this entity?
2. **Trivia:** Which data are people likely to look up when they want to learn about this entity?

For example, for entity Q567, the answer to the first question is the statement "position held: Federal Chancellor of Germany" rather than "Erdös number: 5" or "award received: Time Person of the Year". The answer to the second question, in contrast, might include statements such as "spouse: Joachim Sauer" or "number of children: 0".

Clearly, there is no mathematical definition of what exactly the answer should look like, and characterising this further would already be a contribution. As an intuition, the data selected for the first question should be informative enough for a knowledgeable person to guess the entity that it came from (roughly: how to generate quiz questions from Wikidata?). In contrast, the data selected for the second question is more similar to the results shown in an information box by a Web search engine when searching for the entity.[3] Arguably, the first task is easier to evaluate (can humans find out which entity was meant?), while the second might be more subjective (and yet, statements like "Mathematics Genealogy Project ID: 223396" or "occupation: politician" are probably hardly of interest to anybody looking up the entity).

Motivation Both notions of importance have wide-reaching applications. The first can be used, for example, to create concise verbal descriptions that highlight the main features that set an entity apart. It could also be useful in search and ranking (if we look for German chemists, Q567 might not be the most relevant hit, no matter how large its absolute importance). Or one could indeed create quiz questions from data. The second notion of importance is valuable for informational displays, such as the aforementioned information box in web search engines. It can also help to organise the presentation of data when showing all of it to users. Finally, both types of importance can help Wikidata contributors to decide which statements should receive most attention, and which important pieces of information might be incomplete.

Starting points: One might ask some questions to approach the task:

- Can "importance" of a statement (in either sense) be gauged from the data alone?
- Can linguistic sources, such as Wikipedia pages or Wikidata descriptions, be used?
- Does the editing history hold relevant clues?
- Could importance be learned from training data?
- Are there classification-based patterns of what is important (e.g., are we always interested in the same facts when querying about politicians)?
- How could a reasonable baseline solution be created? How could we obtain gold-standard datasets for evaluation?

[3] How do you think Google finds out which properties should be displayed in the Knowledge Panel for a particular entity?

For computing more complicated statistics about all of Wikidata, it is advisable to use *Wikidata Toolkit* (https://github.com/Wikidata/Wikidata-Toolkit). There are already example programs there on how to count the occurrence of certain features across the whole knowledge base.

The popularity of an entity in absolute terms can also be estimated from the number of Wikimedia sites that have a page about it. This number is also stored in the RDF property `wikibase:sitelinks` (useful for ordering SPARQL results).

Towards a solution: To explore if either of these tasks can be solved, one could build small prototype tools.

1. Given an entity, extract up to five statements that are deemed to be most important to the entity's notability. Can you guess the entity from the statements?

2. Given an entity, extract the ten most interesting statements, or order all of its statements by importance.