Semantic SPARQL Templates
Christina Davril
Structure

• What is KGQA?

• What are semantic SPARQL templates?
• Key characteristics of the templates
• Wikipedia Lists (benchmark)

• Results of frequency analyses
What is KGQA?
Knowledge Graphs

- Knowledge Bases

- **Examples**: Wikidata, PubChem, UniProt

- **RDF**: Resource Description Framework

- Triples:
  
  `<subject>`  
  
  “The Hitchhiker's Guide to the Galaxy”  
  
  Q3107329  

  `<predicate>`  
  
  “author”  
  
  P50  

  `<object>`  
  
  “Douglas Adams”  
  
  Q42

- **SPARQL**: SPARQL Protocol and RDF Query Language
Knowledge Graph Question Answering (KGQA)

QLever

```
SELECT ?person_name ?party_name ?max_join_date ?sitelinks WHERE {
  { SELECT ?person (MAX(?join_date) AS ?max_join_date) WHERE {
  } GROUP BY ?person }
  ?person @en@rdfs:label ?person_name .
  ?party @en@rdfs:label ?party_name .
  ORDER BY DESC(?sitelinks)
}
```

Source: https://ad-blog.cs.uni-freiburg.de/post/semantic-sparql-templates-for-question-answering-over-wikidata/
What are semantic SPARQL templates?
Semantic SPARQL templates?

1. Politicians
2. Politicians are German
3. The most recent party for each politician
### Semantic SPARQL templates?

<table>
<thead>
<tr>
<th>Semantic Characteristic</th>
<th>Semantic Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Politicians</td>
<td>→ CONSTRAINT</td>
</tr>
<tr>
<td>2. Politicians are German</td>
<td>→ CONSTRAINT</td>
</tr>
<tr>
<td>3. The most recent party for each politician</td>
<td>→ ATTRIBUTE</td>
</tr>
</tbody>
</table>
Semantic SPARQL templates?

1. Politicians $\rightarrow$ CONSTRAINT

$$\text{?politician} \quad \text{wd:}P106/\text{wd:}P279^* \quad \text{wd:}Q82955 .$$

$$\text{<variable1>} \quad \text{<occupation>/<subclass-of>} \quad \text{<politician>}$$

**QLever Output:**

<table>
<thead>
<tr>
<th>?politician</th>
<th>Q1000051</th>
</tr>
</thead>
</table>

**Semantic Template**

"path": [varIRI] [pred] [obj].
Semantic SPARQL templates?

2. Politicians are German → CONSTRAINT

<variable1> <country-of-citizenship> <Germany>

QLever Output: | ?politician |
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1000051</td>
</tr>
</tbody>
</table>

“add_path”: [cont] [varIRI] [pred] [obj] .
Semantic SPARQL templates?

3. The most recent party for each politician → ATTRIBUTE

```
?politician  ???  ?party.

<variable1>  <current-political-party?>  <variable2>
```
### Oskar Lafontaine (Q110719)

**German politician**

#### Statements

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Start Time</th>
<th>End Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance of</td>
<td>human</td>
<td></td>
<td></td>
</tr>
<tr>
<td>member of political party</td>
<td>The Left</td>
<td>29 December 2005</td>
<td>March 2022</td>
</tr>
<tr>
<td></td>
<td>Social Democratic Party of Germany</td>
<td>1966</td>
<td>May 2005</td>
</tr>
</tbody>
</table>
“the party with the maximum start time value (for each politician)”
3. The most recent party for each politician → ATTRIBUTE


P102 <member_of_political_party> P580 <start_time>

“add_path”: [cont] [varIRI] [pred] [obj] .
Template identifier: \texttt{arg\_agg}

Parameters:

- \texttt{[distinct]}: aggregation type modifier $\in \{ \text{DISTINCT}, \emptyset \}$
- \texttt{[agg]}: aggregation type $\in \{ \text{MAX, MIN, AVG} \}$
- \texttt{[var1]}: variable whose value(s) corresponding to the aggregated value should be output (\texttt{arg})
- \texttt{[var2]}: variable whose values are aggregated
- \texttt{[var3]}: variable to store the aggregated value
- (\texttt{[var4.1], \ldots}): variables to group by
- (\texttt{[var5.1], \ldots}): additional variables to project
- \texttt{[cont]}: query graph to add to

Template:

```sql
SELECT [var1] [var3] ([var4.1], \ldots) ([var5.1], \ldots)
WHERE {
  
  SELECT ( [agg] ( [distinct] [var2] ) AS [var3] ) ([var4.1], \ldots)
  WHERE { [cont] } GROUP BY ([var4.1], \ldots)
}

SELECT ( [var2] AS [var3] ) [var1] ([var4.1], \ldots) ([var5.1], \ldots)
WHERE { [cont] }
```
Arguments:

- no DISTINCT
- MAX
- ?party (the “arg“)
- ?start_time
- ?max_start_time
- ?politician
- no other variables to project
  ...

```sql
SELECT [var1] [var3] ([var4.1], ...) ([var5.1], ...)
WHERE {
  {
    SELECT ( [agg] ( [distinct] [var2] ) AS [var3] ) ([var4.1], ...)
    WHERE { [cont] } GROUP BY ([var4.1], ...)
  }
  {
    SELECT ( [var2] AS [var3] ) [var1] ([var4.1], ...) ([var5.1], ...)
    WHERE { [cont] }
  }
}
```
3. The most recent party for each politician → ATTRIBUTE

QLever Output:

<table>
<thead>
<tr>
<th>party</th>
<th>max_start_time</th>
<th>politician</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q166027</td>
<td>2004-01-01T00:00:00Z</td>
<td>Q1000800</td>
</tr>
<tr>
<td>Q49768</td>
<td>1930-01-01T00:00:00Z</td>
<td>Q1001323</td>
</tr>
</tbody>
</table>
Human-readable output, duplicate-free, showing well-known entities first

4. Names of politicians and political parties → ATTRIBUTE

```
“add_name”: [cont]
[var1.1] [pred] [var2.1] . FILTER ( LANG[var2.1] = [lang1] )
[var1.2] [pred] [var2.2] . FILTER ( LANG[var2.2] = [lang2] )

?politician rdfs:label ?politician_name . FILTER(LANG(?politician_name)="en")
```

5. Number of Wikipedia site links per politician → ATTRIBUTE

```
“add_path”: [cont]
[varIRI] [pred] [obj] .

```
6. Output politician names and party names *distinct pairs* → OUTPUT

```
“select”: SELECT [distinct] ([var1], ...) WHERE {
    [cont]
}
```

7. Order by descending number of Wikipedia site links → OUTPUT

```
“order”: [cont]
ORDER BY [order1] ([var1]) [order2] ([var2])
DESC(?sitelinks)
```
## QLever Output:

<table>
<thead>
<tr>
<th>politician_name</th>
<th>party_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angela Merkel</td>
<td>Christian Democratic Union</td>
</tr>
<tr>
<td>Rosa Luxemburg</td>
<td>Communist Party of Germany</td>
</tr>
<tr>
<td>Konrad Adenauer</td>
<td>Christian Democratic Union</td>
</tr>
</tbody>
</table>

[https://qlever.cs.uni-freiburg.de/wikidata/nuT7zG](https://qlever.cs.uni-freiburg.de/wikidata/nuT7zG)
Key characteristics of the templates
Why does “arg_agg” contain code duplication / two subqueries?

```sql
SELECT ?party ?max_start_time ?politician
WHERE {

{ SELECT ( MAX(?start_time) AS ?max_start_time ) ?politician
  WHERE {
  } GROUP BY ?politician
}

{ SELECT ( ?start_time AS ?max_start_time ) ?party ?politician
  WHERE {
  }
}
```
Why does “arg_agg” contain code duplication / two subqueries?

SELECT ?party ?max_start_time ?politician
WHERE {
{
    SELECT ( MAX(?start_time) AS ?max_start_time ) ?politician
    WHERE {
    } GROUP BY ?politician
}
{
    SELECT ( ?start_time AS ?max_start_time ) ?party ?politician
    WHERE {
    }
}
} → To account for ties.
“Who is the Formula 1 race driver with the most races?”

Gold query:

```
SELECT DISTINCT ?uri
WHERE {
}
ORDER BY DESC(?num) LIMIT 1
```

Q10841764  P1350
<Formula-One-driver>  <number of matches played/races/starts>

“Correct result”: Michael Schumacher (Q9671)
“Who is the Formula 1 race driver with the most races?”

Gold query:

SELECT DISTINCT ?uri
WHERE {
}
ORDER BY DESC(?num) LIMIT 1

Q10841764 <Formula-One-driver> P1350 <number of matches played/races/starts>

“Correct result”: Michael Schumacher (Q9671)

Problems:

- Does not account for ties → Uses world knowledge
- Does not provide a source / verifiable ground truth → Why is the result not Kimi Räikkönen?
- Generic variables
This list is accurate as of the 2022 Australian Grand Prix. Drivers who only participated in Friday practice and who were not actually entered for the race are not included.

<table>
<thead>
<tr>
<th>Driver Name</th>
<th>Nationality</th>
<th>Seasons Competed</th>
<th>Drivers' Championships</th>
<th>Race Entries</th>
<th>Race Starts</th>
<th>Pole Positions</th>
<th>Race Wins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rubens Barrichello</td>
<td>Brazil</td>
<td>1993–2011</td>
<td>0</td>
<td>326</td>
<td>322</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>Jenson Button</td>
<td>United Kingdom</td>
<td>2000–2017</td>
<td>1 2009</td>
<td>309</td>
<td>306</td>
<td>8</td>
<td>15</td>
</tr>
</tbody>
</table>

Inclusion of zero counts using the “agg” template

Examples: “Film series and how many films they contain”, “Actors and their number of won awards”, ...

QALD-9-plus: 8 out of 412 examples
QALD-10: 3 out of 394 examples

Problems:
→ Indicates lack of variety regarding query structures
→ Counts of zero are only included in one example where an average count is required

QALD-10, ID 23: “How many spouses do heads of state have on average?”

1) “agg”: Count spouse statements for each head of state, incl. 0
2) “agg_all”: Take the mean
Inclusion of zero counts using the “agg” template

“agg” sub-type for COUNT:

```
SELECT (COUNT ( [distinct1] [var1.1] ) AS [var2.1] ) ([var3.1], ...)  
WHERE {  
    [cont2]  
    OPTIONAL { [cont1] }  
}  
GROUP BY ([var3.1], ...)
```

**Pro:**
- Includes (real) counts of zero
- One version of “agg” for COUNT aggregation is enough.

**Con:**
- Includes missing values.
- More complicated to use.
Summary

• 18 templates
• Knowledge graph-independent
• For lower-level semantic purposes: e.g., “path”
• For higher-level semantic purposes: e.g., “arg_agg”
• Made to be generally applicable
• Use a basic set of SPARQL 1.1 constructs, e.g., no HAVING in addition to FILTER
• Based on Wikidata-based benchmark/dataset: Wikipedia Lists
Wikipedia Lists (benchmark)
Wikipedia Lists

- 60 examples
- examples are based on (information in) Wikipedia lists
  - re-creations of full Wikipedia lists → often tabular or table-like
  - questions about aspects of Wikipedia lists
- Handwritten query version (sometimes created using templates as aid!)
- Generated, template-based query version
  - The Wikidata output (QLever) was compared to the info in the Wikipedia list and discrepancies resolved or documented
  - The results of the handwritten and generated queries match
## Zone 5: South and Central Americas

- **500** – 🇦🇺 Falkland Islands  
  - **500** – 🇬🇸 South Georgia and the South Sandwich Islands  
- **501** – 🇧🇿 Belize  
- **502** – 🇬🇹 Guatemala  
- **503** – 🇸🇻 El Salvador

<table>
<thead>
<tr>
<th>Country Calling Code</th>
<th>Sample Flag Image</th>
<th>Country Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>+500</td>
<td><img src="image" alt="Falkland Islands Flag" /></td>
<td>Falkland Islands</td>
</tr>
<tr>
<td>+500</td>
<td><img src="image" alt="South Georgia and the South Sandwich Islands Flag" /></td>
<td>South Georgia and the South Sandwich Islands</td>
</tr>
<tr>
<td>+501</td>
<td><img src="image" alt="Belize Flag" /></td>
<td>Belize</td>
</tr>
<tr>
<td>+502</td>
<td><img src="image" alt="Guatemala Flag" /></td>
<td>Guatemala</td>
</tr>
<tr>
<td>+503</td>
<td><img src="image" alt="El Salvador Flag" /></td>
<td>El Salvador</td>
</tr>
</tbody>
</table>

[en.wikipedia.org/wiki/List_of_country_calling_codes](en.wikipedia.org/wiki/List_of_country_calling_codes)  
[qlever.cs.uni-freiburg.de/wikidata/fKDg2G](qlever.cs.uni-freiburg.de/wikidata/fKDg2G)
Results of frequency analyses
The 60 examples had 1,130 instances of templates.
Which (other) templates were inserted into the templates?

“add_path”:

```plaintext
[cont] [varIRI] [pred] [obj] .
```
Example: “arg_agg_all” containing “agg”

“Which US president was played by the most actors in a movie? Also show the actors”

“Which Formula One driver won the most championships and in which years?”

“Which movie has won the most Oscars?”

“Who composed the music for the most Pixar films (excluding short films)?”

“Which country borders the most other countries?”

→ Clear pattern that may help recognize that this combination of templates is required

→ Same aggregators used every time:

    COUNT in “agg”

    MAX in “arg_agg_all”
Syntactic structure created with the help of the Stanford Lexicalized Parser v2.0.4. Model: English PCFG; Tags: Penn Treebank syntactic and POS tagsets
Thank you for your attention and participation!

Questions?
Extra slides
Overview: Semantic Categories & Templates

CONSTRAINT: path, add_path, connect, filter, minus, arg_ranks_all, val_ranks_all

ATTRIBUTE: add_path, add_name, add_desc

AGGREGATE: agg, agg_all, arg_agg, arg_agg_all

COMBINE: union, bind

OUTPUT: select, order
How many nuclear reactors operate in each country? Sort by descending number and only include countries that have operational reactors.

Wikipedia list: [https://en.wikipedia.org/wiki/Nuclear_power_by_country#Overview]

Wikidata query:
```
```

Results comparison:
Even though the Wikidata output contains much fewer entities, four out of the top 5 of the Wikipedia list are within the top 5 of the Wikidata output, suggesting that the query is adequate. Countries were retrieved using a simple triple as the filtering by the number of operational reactors already eliminated many undesirable result entities.

Table query:
```
```
"44": {
  "[1]": {
    "desc": "countries",
    "template": "path",
    "arguments": [
      "?country",
      "wdt:P31/wdt:P279*",
      "wd:Q6256"
    ]
  },
  "[2]": {
    "desc": "nuclear reactors per country",
    "template": "add_path",
    "arguments": [
      "?nuclear_reactor",
      "wdt:P17",
      "?country",
      "[1]"
    ]
  },
  "[3]": {
    "desc": "nuclear reactors are nuclear reactors",
    "template": "add_path",
    ...
  }
}
Which (other) templates were inserted into the templates?

**“add_path”:**

```plaintext
[cont]
[varIRI] [pred] [obj]
```
Missing functionality in SPARQL

Among the templates, there are “arg_ranks_all” and “val_ranks_all” ...

\begin{itemize}
  \item \texttt{arg\_ranks\_all} e.g., needed for “Which are the top three electronegative chemical elements?”
  \begin{itemize}
    \item \texttt{ranks 1-3 overall}
  \end{itemize}
  \item \texttt{val\_ranks\_all} e.g., needed for “How fast is the world’s second fastest man (100 m sprint)?”
  \begin{itemize}
    \item \texttt{rank 2 overall}
  \end{itemize}
\end{itemize}

... but there are no good templates for “arg\_ranks” and “val\_ranks” (with grouping)!

\textbf{Example:} “What is the second largest country on each continent?”

Attempts either require computationally expensive operations or pre-generation!