Efficient SPARQL Autocompletion on Large Knowledge Bases

Johannes Kalmbach

University of Freiburg

johannes.kalmbach@gmail.com

August 2021

RDF and Sparql

- A Rdf knowledge base with three triples:

Subject	Predicate	Object
<2004 Indian Ocean earthquake>	<is-a>	<Earthquake> .
<2004 Indian Ocean earthquake>	<magnitude>	9.3 .
<1944 San Samuel earthquake>	<is-a>	<Earthquake> .
<1944 San Samuel earthquake>	<magnitude>	9.0 .

- A simple SPARQL query:

```
SELECT ?eq ?magnitude WHERE {
  ?eq <is-a> <Earthquake> .
  ?eq <magnitude> ?magnitude .
}
```

- The query result:

?eq	?magnitude
<2004 Indian Ocean earthquake>	9.3 .
<1944 San Samuel earthquake>	9.0 .

Simple in Principle, Complex in Practice

- The same query (earthquakes and their magnitude) for Wikidata:

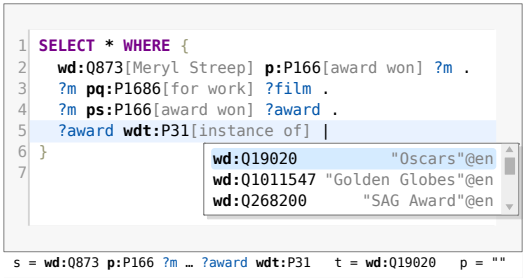
```
SELECT ?eqLabel ?magnitude WHERE {
  ?eq wdt:P31 wd:Q7944 .
  ?eq wdt:P2528 ?magnitude
}
```

Demo: Information on Earthquakes

Problem Definition (informal)

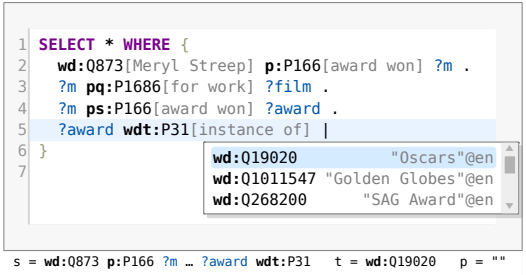
- User types a prefix of a Sparql query.
- User is presented with context-sensitive suggestions, on how to continue.
- User can restrict the suggestions by typing a prefix of a label of the next token.
- Suggestions are obtained via special SPARQL queries (Autocompletion/AC queries).

Sensitive AC Query Template for Wikidata Objects



3. *SELECT ?entity (COUNT(?entity) AS ?score_2) WHERE {*
4. **%context% %subject% %predicate%** *?object }*
5. *} GROUP BY ?object }*

All possible *?object*s that continue the partial query, together with the number of continuations (*?score_2*).



3. *SELECT ?entity (COUNT(?entity) AS ?score_2) WHERE {*
4. * **%context% %subject% %predicate%** ?object }*
5. *} GROUP BY ?object }*
6. *?object rdfs:label|skos:altLabel ?name .*
7. *FILTER REGEX(?name, "^**%prefix%**")*

Add labels and filter by the typed prefix

AC Query Templates for Wikidata Objects

```
1.    SELECT ?object (SAMPLE(?name) AS ?name)
2.                   (SAMPLE(?score_2) AS ?score) WHERE {
3.      { SELECT ?entity (COUNT(?entity) AS ?score_2) WHERE {
4.        %context% %subject% %predicate% ?object
5.      } GROUP BY ?object }
6.      ?object rdfs:label|skos:altLabel ?name .
7.      FILTER REGEX(?name, "^%prefix%")
8.    } GROUP BY ?object ORDER BY DESC (?score)
```

If one *?object* has multiple matching *?name*s, eliminate the duplicates

AC Query Templates

- Similar templates exist for *subjects* and *predicates*
- Can be turned into concrete AC queries by substituting placeholders **%context%**, **%prefix%** etc.
- Can easily be adapted to other knowledge bases.
- Alternative: *Agnostic* (non-sensitive) templates (ignore **%context%**). Cheap to compute, but worse results (*?x <is-a> <Angela Merkel>*).

QLever

- Open source SPARQL engine developed at AD chair in Freiburg.
- Very efficient on complicated, ranked queries (like the AC queries).
- Supports very efficient prefix filtering on strings.
- Allows caching of building blocks which appear in every AC query (e.g. the names of all entities).
- Immediately worked well with AC queries, however several improvements were still required for robustness.

Improvements for QLever

- Implementation of a Timeout and Memory Limit (for the few AC queries that don't work fast)
- Improvements to the query planner to efficiently utilize prefix filters and the cache
- Parallelization of several operations to speed up the query processing

Evaluation

- Query Engines: QLever (with improvements), Blazegraph, Virtuoso
- Knowledge Bases: Fbeasy(362M triples), Freebase (1.9B triples), Wikidata (6.9B triples)
- Queries: 301 examples from Wikidata, manually translated for Freebase (115) and Fbeasy (99)
- "Type" query from left to right, for each token issue AC queries for prefix length 0, 3, 7

Metrics

- Percentage of queries with time $< 0.2s$, $< 1.0s$, $> 5.0s$ (timeout)
- MRR_7 (Mean reciprocal rank): Suggestions are shown on pages of 7 results each, Score for individual token is $1/k$ where k is the page on which the target token is shown (100% for first page, 50% for second page...).

Results for Wikidata (1258 tokens)

Wikidata		$\leq$ **0.2**s	$\leq$ **1.0**s	**Max / TO**	**MRR$_7$**
Agnostic	Qlever	100%	100%	696ms	0: 6% 3: 64% 7: 92%
Sensitive	Blazegraph	3%	27%	59%	0: 26% 3: 36% 7: 37%
Sensitive	Virtuoso	35%	53%	24%	0: 38% 3: 67% 7: 67%
Sensitive	Qlever	71%	90%	6%	0: 50% 3: 92% 7: 95%