Efficient SPARQL Autocompletion on Large Knowledge Bases

Johannes Kalmbach

University of Freiburg

johannes.kalmbach@gmail.com

August 2021
A RDF knowledge base with three triples:

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;2004 Indian Ocean earthquake&gt;</td>
<td>&lt;is-a&gt;</td>
<td>&lt;Earthquake&gt;</td>
</tr>
<tr>
<td>&lt;2004 Indian Ocean earthquake&gt;</td>
<td>&lt;magnitude&gt;</td>
<td>9.3</td>
</tr>
<tr>
<td>&lt;1944 San Samuel earthquake&gt;</td>
<td>&lt;is-a&gt;</td>
<td>&lt;Earthquake&gt;</td>
</tr>
<tr>
<td>&lt;1944 San Samuel earthquake&gt;</td>
<td>&lt;magnitude&gt;</td>
<td>9.0</td>
</tr>
</tbody>
</table>

A simple SPARQL query:

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

The query result:

<table>
<thead>
<tr>
<th>?eq</th>
<th>?magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;2004 Indian Ocean earthquake&gt;</td>
<td>9.3</td>
</tr>
<tr>
<td>&lt;1944 San Samuel earthquake&gt;</td>
<td>9.0</td>
</tr>
</tbody>
</table>
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).
3. SELECT ?entity (COUNT(?entity) AS ?score_2) WHERE {
4.   \%context\% \%subject\% \%predicate\% ?object 
5. } GROUP BY ?object 

All possible ?objects that continue the partial query, together with the number of continuations (?score_2).
3. `SELECT ?entity (COUNT(?entity) AS ?score) WHERE {
    ?context %subject% %predicate% ?object 
} GROUP BY ?object 


7. `FILTER REGEX(?name, "^%prefix%")`

Add labels and filter by the typed prefix
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 }
7.   FILTER REGEX(?name, "~%prefix%")
8. } GROUP BY ?object ORDER BY DESC (?score)

If one ?object has multiple matching ?names, 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>`).
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
• 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...).
<table>
<thead>
<tr>
<th>Wikidata</th>
<th>≤ 0.2 s</th>
<th>≤ 1.0 s</th>
<th>Max / TO</th>
<th>MRR&lt;sub&gt;7&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agnostic Qlever</td>
<td>100%</td>
<td>100%</td>
<td>696ms</td>
<td>0: 6% 3: 64% 7: 92%</td>
</tr>
<tr>
<td>Sensitive Blazegraph</td>
<td>3%</td>
<td>27%</td>
<td>59%</td>
<td>0: 26% 3: 36% 7: 37%</td>
</tr>
<tr>
<td>Sensitive Virtuoso</td>
<td>35%</td>
<td>53%</td>
<td>24%</td>
<td>0: 38% 3: 67% 7: 67%</td>
</tr>
<tr>
<td>Sensitive Qlever</td>
<td>71%</td>
<td>90%</td>
<td>6%</td>
<td>0: 50% 3: 92% 7: 95%</td>
</tr>
</tbody>
</table>