Simple Question Answering over Wikidata

Master’s thesis

Thomas Goette
October 28, 2021
A Knowledge Base contains many facts

Example

“The mother of Albert Einstein is Pauline Koch.”
Facts are stored using RDF

Example
“The mother of Albert Einstein is Pauline Koch.”

In RDF
"Albert Einstein" "has mother" "Pauline Koch"
We can use **SPARQL** to extract information

```sparql
SELECT ?target WHERE {
    "Albert Einstein" "has mother" ?target .
}
```
We can use SPARQL to extract information

Query

```sparql
SELECT ?target WHERE {
  "Albert Einstein" "has mother" ?target .
}
```

Result

```
?target

"Pauline Koch"
```
Names are ambiguous

Albert Einstein (famous scientist)

- [http://www.wikidata.org/entity/Q937](http://www.wikidata.org/entity/Q937)
- [wd:Q937](http://www.wikidata.org/entity/Q937)
Albert Einstein (famous scientist)

- [http://www.wikidata.org/entity/Q937]
- `wd:Q937`

“has mother” relation

- [http://www.wikidata.org/prop/direct/P25]
- `wdt:P25`
The question and the associated query are very different

Query

```sql
SELECT ?target WHERE {
}
```
The question and the associated query are very different

**Question**

“Who is the mother of Albert Einstein?”

**Query**

```
SELECT ?target WHERE {
}
```
The variable can also be in the subject position

Question

“Which books did J. R. R. Tolkien write?”

Query

```
SELECT ?book WHERE {
}
```
The result can contain more than one item

Question
“Which books did J. R. R. Tolkien write?”

Query
```
SELECT ?book WHERE {
}
```

Result
```
?book
  wd:Q1101425
  wd:Q15228
  wd:Q17029228
...
```
We use a shorter form for queries

This query ...

```sparql
SELECT ?t WHERE {
}
```

... becomes

```sparql
wd:Q937 wdt:P25 ?t
```

This query ...

```sparql
SELECT ?b WHERE {
  ?b wdt:P50 wd:Q892 .
}
```

... becomes

```sparql
?b wdt:P50 wd:Q892
```
Questions?
The input question goes through multiple steps

question → entity linking → candidate generation → relation matching → ranking → SPARQL query
The input question goes through multiple steps:

1. **question**
2. **entity linking**
3. **candidate generation**
4. **relation matching**
5. **ranking**
6. **SPARQL query**
Entity linking matches entities to words

<table>
<thead>
<tr>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Who is the mother of Albert Einstein?”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Albert Einstein”</td>
</tr>
<tr>
<td>“Albert Einstein”</td>
</tr>
<tr>
<td>“Einstein”</td>
</tr>
<tr>
<td>“the mother”</td>
</tr>
<tr>
<td>“the mother”</td>
</tr>
</tbody>
</table>

- Sort by number of matched words, then by entity popularity
- Keep the first $N_e$ of these matches
Every matched entity leads to several candidates

<table>
<thead>
<tr>
<th>Candidates for <strong>wd</strong>: Q76346</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>wd</strong>: Q76346  <strong>wdt</strong>: P25 ?0</td>
</tr>
<tr>
<td><strong>wd</strong>: Q76346  <strong>wdt</strong>: P26 ?0</td>
</tr>
<tr>
<td><strong>wd</strong>: Q76346  <strong>wdt</strong>: P569 ?0</td>
</tr>
<tr>
<td>?0  <strong>wdt</strong>: P25  <strong>wd</strong>: Q76346</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Candidates for <strong>wd</strong>: Q937</th>
</tr>
</thead>
<tbody>
<tr>
<td>?0  <strong>wdt</strong>: P1038  <strong>wd</strong>: Q937</td>
</tr>
<tr>
<td><strong>wd</strong>: Q937  <strong>wdt</strong>: P103 ?0</td>
</tr>
<tr>
<td><strong>wd</strong>: Q937  <strong>wdt</strong>: P1196 ?0</td>
</tr>
<tr>
<td><strong>wd</strong>: Q937  <strong>wdt</strong>: P25 ?0</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>wdt</strong>: P25 (mother)</td>
</tr>
<tr>
<td>• <strong>wdt</strong>: P26 (spouse)</td>
</tr>
<tr>
<td>• <strong>wdt</strong>: P103 (native language)</td>
</tr>
<tr>
<td>• <strong>wdt</strong>: P569 (date of birth)</td>
</tr>
<tr>
<td>• <strong>wdt</strong>: P1038 (relative)</td>
</tr>
<tr>
<td>• <strong>wdt</strong>: P1196 (manner of death)</td>
</tr>
</tbody>
</table>
We find relation matches for each candidate

Question

“Who is the mother of Albert Einstein?”

**wd**: Q937  **wdt**: P103 ?0

“mother” partially matches the alias “mother tongue” of **wdt**: P103 (native language)

**wd**: Q937  **wdt**: P1196 ?0

No match for **wdt**: P1196 (manner of death)

Keep only the candidates with some kind of relation match
question

entity linking

candidate generation

relation matching

ranking

SPARQL query
We map each candidate to a ten-dimensional vector of features

- number of entity words
- number of relation words
- word coverage
- entity popularity score
- ...

Feature vector of \texttt{wd}: Q937 \texttt{wdt}: P25 \texttt{?0}

(1, 1.0, 283, 1, 2, 2, 1, 1, 1, 1)
We rank the generated candidates

**Rule-based ranker**
Rank candidates with a hard-coded scoring function

**Learned ranker**
- Pairwise ranking as binary classification
- Random forest
Questions?
We use a subset of SimpleQuestionsWikidata as a benchmark

- Originally created from/for Freebase
- Subset classified as “answerable”
- 19481/5622 (train/test)
- Question together with gold SPARQL query
We use a subset of *SimpleQuestionsWikidata* as a benchmark

- Originally created from/for *Freebase*
- Subset classified as “answerable”
- 19481/5622 (train/test)
- Question together with gold *SPARQL* query

**Example**

Q2662597 P19 Q2868 where was paolo de la haza born?

**Corresponding query**

wd:Q2662597 wdt:P19 ?0
We compare the result sets of two SPARQL queries.
<table>
<thead>
<tr>
<th>QA system</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our system (rules, $N_e = 10$)</td>
<td>0.537</td>
</tr>
<tr>
<td>Our system (learned, $N_e = 10$)</td>
<td>0.564</td>
</tr>
<tr>
<td>Our system (rules, $N_e = 500$)</td>
<td>0.586</td>
</tr>
<tr>
<td>Oliya et al. (2021)</td>
<td>0.682</td>
</tr>
</tbody>
</table>
Questions?
QA over Wikidata

Given a natural language question $q$, find a SPARQL query $c$ such that the intended answer for question $q$ is the result of executing the SPARQL query $c$ over Wikidata.

Simple QA over Wikidata

The query $c$ is of the form

```
SELECT ?t WHERE {
  <body>
}
```

where $<body>$ is one triple pattern with the variable $?t$ being either in the subject or in the object position.
All features

- pattern complexity
- token coverage
- entity score
- entity label matches
- number of entity tokens
- number of entity tokens no stop
- number of exact relation matches
- number of no-stop relation matches
- number of contained relation matches
- number of relation tokens
Examples of feature vectors

- **wd**: Q937  **wdt**: P25  ?0
  
  (1, 1.0, 283, 1, 2, 2, 1, 1, 1, 1)

- **wd**: Q937  **wdt**: P103  ?0
  
  (1, 1.0, 283, 1, 2, 2, 0, 0, 1, 1)

- **wd**: Q76346  **wdt**: P25  ?0
  
  (1, 0.67, 57, 0, 1, 1, 1, 1, 1, 1)

Entities and relations

- **wd**: Q937 (Albert Einstein)
- **wd**: Q76346 (Mileva Marić)
- **wdt**: P25 (mother)
- **wdt**: P103 (native language)
Manual scoring function

\[ s(c) := 1000 \hat{f}_{10}(c) \]
\[ + 100 \left( \hat{f}_5(c) + \hat{f}_6(c) + \hat{f}_7(c) \right) \]
\[ + 10 \hat{f}_2(c) \]
\[ + \hat{f}_1(c) \]

- \( f_1(c) \): entity popularity score
- \( f_2(c) \): number of entity label matches
- \( f_5(c) \): number of exact relation matches
- \( f_6(c) \): number of contained relation matches
- \( f_7(c) \): number of no-stop relation matches
- \( f_{10}(c) \): word coverage
Training the ranker

• Run question through pipeline (except ranker)
• Find the correct candidates
• Build pairs of one correct and one incorrect candidate
  \((c_k, c_m)\)
• Create two training samples from every such pair:
  \((f(c_k) - f(c_m)), f(c_k), f(c_m)) \in \mathbb{R}^{30}\) with label 1
  \((f(c_m) - f(c_k)), f(c_m), f(c_k)) \in \mathbb{R}^{30}\) with label 0
<table>
<thead>
<tr>
<th>QA system</th>
<th>Accuracy (FB2M)</th>
<th>Accuracy (FB5M)</th>
<th>Accuracy (Wikidata)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bordes et al. (2015)</td>
<td>0.627</td>
<td>0.639</td>
<td>-</td>
</tr>
<tr>
<td>Yin et al. (2016)</td>
<td>0.683</td>
<td>0.672</td>
<td>-</td>
</tr>
<tr>
<td>Dai et al. (2016)</td>
<td>-</td>
<td>0.626</td>
<td>-</td>
</tr>
<tr>
<td>He et al. (2016)</td>
<td>0.709</td>
<td>0.703</td>
<td>-</td>
</tr>
<tr>
<td>Lukovnikov et al. (2017)</td>
<td>0.712</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Yu et al. (2017)</td>
<td><strong>0.787</strong></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mohammed et al. (2018)</td>
<td>0.749</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Huang et al. (2019)</td>
<td>0.754</td>
<td><strong>0.749</strong></td>
<td>-</td>
</tr>
<tr>
<td>Oliya et al. (2021)</td>
<td>-</td>
<td>-</td>
<td><strong>0.682</strong></td>
</tr>
<tr>
<td>Our system (rules)</td>
<td>-</td>
<td>-</td>
<td>0.586</td>
</tr>
<tr>
<td>Our system (learned)</td>
<td>-</td>
<td>-</td>
<td>0.564</td>
</tr>
</tbody>
</table>
Evaluation for different numbers of used entities

<table>
<thead>
<tr>
<th>$N_e$</th>
<th>R@1</th>
<th>R@2</th>
<th>R@3</th>
<th>R@5</th>
<th>R@10</th>
<th>R@100</th>
<th>AD (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>0.59</td>
<td>0.67</td>
<td>0.71</td>
<td>0.74</td>
<td>0.77</td>
<td>0.82</td>
<td>7.09</td>
</tr>
<tr>
<td>50</td>
<td>0.58</td>
<td>0.67</td>
<td>0.71</td>
<td>0.74</td>
<td>0.77</td>
<td>0.82</td>
<td>5.52</td>
</tr>
<tr>
<td>10</td>
<td>0.54</td>
<td>0.66</td>
<td>0.69</td>
<td>0.72</td>
<td>0.75</td>
<td>0.77</td>
<td>1.46</td>
</tr>
</tbody>
</table>

Rule-based ranker