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

Query

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

We can use *SPARQL* to extract information

Query

```
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>`
- `wd:Q937`

Names are ambiguous

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

```
SELECT ?target WHERE {
  wd:Q937 wdt:P25 ?target .
}
```

The question and the associated query are very different

Question

"Who is the mother of Albert Einstein?"

Query

```
SELECT ?target WHERE {
  wd:Q937 wdt:P25 ?target .
}
```

The variable can also be in the subject position

Question

"Which books did J. R. R. Tolkien write?"

Query

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

The result can contain more than one item

Question

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

Query

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

Result

?book
wd:Q1101425
wd:Q15228
wd:Q17029228
...

We use a shorter form for queries

This query ...

```
SELECT ?t WHERE {
  wd:Q937 wdt:P25 ?t .
}
```

... becomes

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

This query ...

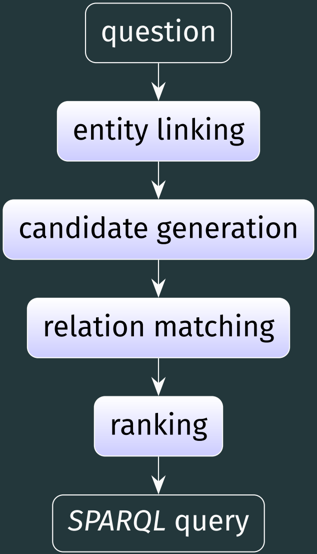
```
SELECT ?b WHERE {
  ?b wdt:P50 wd:Q892 .
}
```

... becomes

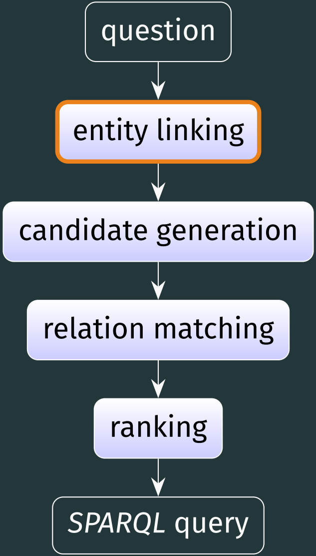
```
?b wdt:P50 wd:Q892
```

Questions?

The input question goes through multiple steps



The input question goes through multiple steps



Entity linking matches entities to words

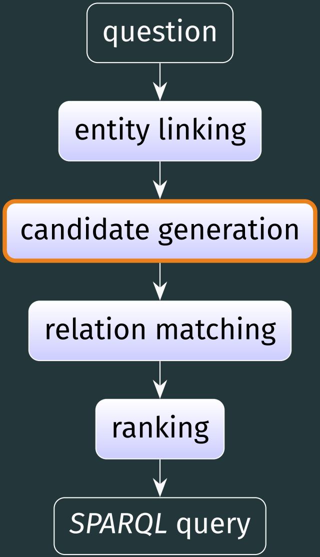
Question

"Who is the mother of Albert Einstein?"

Matches

"Albert Einstein"	wd:Q937 (famous scientist)
"Albert Einstein"	wd:Q13426745 (music album)
"Einstein"	wd:Q76346 (Mileva Marić)
"the mother"	wd:Q169632 (novel)
"the mother"	wd:Q464879 (spiritual guru)
	...

- Sort by number of matched words, then by entity popularity
- Keep the first N_e of these matches


question
entity linking
candidate generation
relation matching
ranking
SPARQL query

Every matched entity leads to several candidates

Candidates for wd:Q76346

```
wd:Q76346 wdt:P25 ?O
wd:Q76346 wdt:P26 ?O
wd:Q76346 wdt:P569 ?O
?O wdt:P25 wd:Q76346
...
```

Candidates for wd:Q937

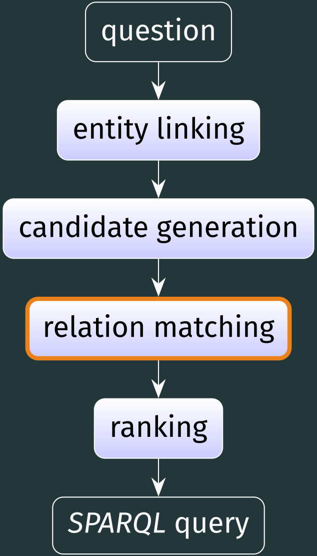
```
?O wdt:P1038 wd:Q937
wd:Q937 wdt:P103 ?O
wd:Q937 wdt:P1196 ?O
wd:Q937 wdt:P25 ?O
...
```

...

...

Relations

- wdt:P25 (mother)
- wdt:P26 (spouse)
- wdt:P103 (native language)
- wdt:P569 (date of birth)
- wdt:P1038 (relative)
- wdt:P1196 (manner of death)


question
entity linking
candidate generation
relation matching
ranking
SPARQL query

We find relation matches for each candidate

Question

“Who is the mother of ~~Albert Einstein~~?”

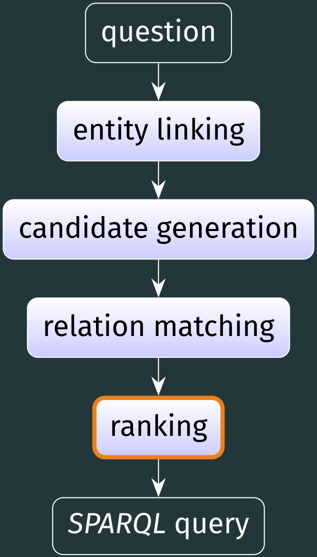
`wd:Q937 wdt:P103 ?0`

“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 `wd:Q937 wdt:P25 ?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

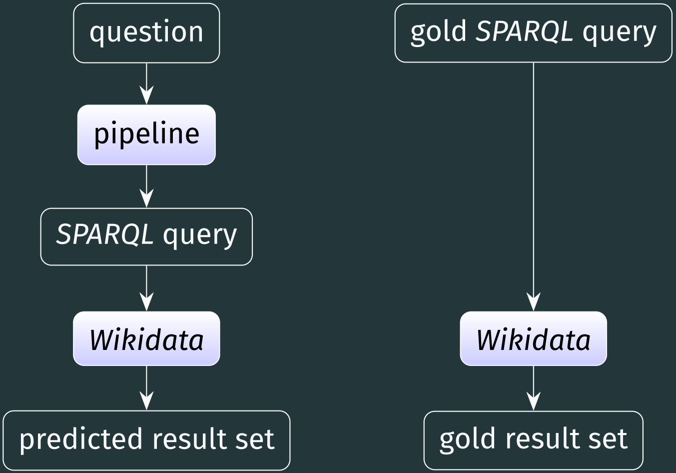
Example

```
Q2662597 P19 Q2868 where was paolo de la haza born?
```

Corresponding query

```
wd:Q2662597 wdt:P19 ?O
```

We compare the result sets of two *SPARQL* queries



QA system	Accuracy
Our system (rules, $N_e = 10$)	0.537
Our system (learned, $N_e = 10$)	0.564
Our system (rules, $N_e = 500$)	0.586
Oliya et al. (2021)	0.682

Questions?

Problem definition

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 ?O`

$(1, 1.0, 283, 1, 2, 2, 1, 1, 1, 1)$

`wd:Q937 wdt:P103 ?O`

$(1, 1.0, 283, 1, 2, 2, 0, 0, 1, 1)$

`wd:Q76346 wdt:P25 ?O`

$(1, 0.67, 57, 0, 1, 1, 1, 1, 1, 1)$

Entities and relations

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

Manual scoring function

$$
\begin{aligned}
s(c) := {} & 1000\hat{f}_{10}(c) \\
& + 100\left(\hat{f}_5(c) + \hat{f}_6(c) + \hat{f}_7(c)\right) \\
& + 10\hat{f}_2(c) \\
& + \hat{f}_1(c)
\end{aligned}
$$

$f_1(c)$	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

QA system	Accuracy (FB2M)	Accuracy (FB5M)	Accuracy (*Wikidata*)
Bordes et al. (2015)	0.627	0.639	-
Yin et al. (2016)	0.683	0.672	-
Dai et al. (2016)	-	0.626	-
He et al. (2016)	0.709	0.703	-
Lukovnikov et al. (2017)	0.712	-	-
Yu et al. (2017)	**0.787**	-	-
Mohammed et al. (2018)	0.749	-	-
Huang et al. (2019)	0.754	**0.749**	-
Oliya et al. (2021)	-	-	**0.682**
Our system (rules)	-	-	0.586
Our system (learned)	-	-	0.564

Evaluation for different numbers of used entities

Rule-based ranker

N_e	R@1	R@2	R@3	R@5	R@10	R@100	AD (s)
500	**0.59**	**0.67**	**0.71**	**0.74**	**0.77**	**0.82**	7.09
50	0.58	**0.67**	**0.71**	**0.74**	**0.77**	**0.82**	5.52
10	0.54	0.66	0.69	0.72	0.75	0.77	**1.46**