Improved Simple Question Answering over Wikidata
Bachelor’s thesis presentation

David Otte
University of Freiburg

June 28, 2023
Simplified subset of Wikidata in RDF format:

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Eiffel Tower&quot;</td>
<td>&quot;named after&quot;</td>
<td>&quot;Gustave Eiffel&quot;</td>
</tr>
<tr>
<td>&quot;Eiffel Tower&quot;</td>
<td>&quot;visitors per year&quot;</td>
<td>6,207,303</td>
</tr>
<tr>
<td>&quot;Gustave Eiffel&quot;</td>
<td>&quot;place of birth&quot;</td>
<td>&quot;Dijon&quot;</td>
</tr>
</tbody>
</table>
Introduction to Wikidata

- Simplified subset of Wikidata in RDF format:

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Eiffel Tower&quot;</td>
<td>&quot;named after&quot;</td>
<td>&quot;Gustave Eiffel&quot;</td>
</tr>
<tr>
<td>&quot;Eiffel Tower&quot;</td>
<td>&quot;visitors per year&quot;</td>
<td>6,207,303</td>
</tr>
<tr>
<td>&quot;Gustave Eiffel&quot;</td>
<td>&quot;place of birth&quot;</td>
<td>&quot;Dijon&quot;</td>
</tr>
</tbody>
</table>

- Simplified example query:

```sql
SELECT ?o WHERE {
    "Eiffel Tower" "named after" ?o .
}
```
Introduction to Wikidata

- Simplified subset of Wikidata in RDF format:

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Eiffel Tower&quot;</td>
<td>&quot;named after&quot;</td>
<td>&quot;Gustave Eiffel&quot;</td>
</tr>
<tr>
<td>&quot;Eiffel Tower&quot;</td>
<td>&quot;visitors per year&quot;</td>
<td>6,207,303</td>
</tr>
<tr>
<td>&quot;Gustave Eiffel&quot;</td>
<td>&quot;place of birth&quot;</td>
<td>&quot;Dijon&quot;</td>
</tr>
</tbody>
</table>

- Simplified example query:

```sql
SELECT ?o WHERE {
  "Eiffel Tower" "named after" ?o .
}
```

- Results:

```
?o
"Gustave Eiffel"
```
Introduction to Wikidata

Subset of Wikidata in RDF format (Prefixes omitted):

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q243</td>
<td>P138</td>
<td>Q20882</td>
</tr>
<tr>
<td>Q243</td>
<td>P1174</td>
<td>6,207,303</td>
</tr>
<tr>
<td>Q20882</td>
<td>P19</td>
<td>Q7003</td>
</tr>
</tbody>
</table>

Example query:

```
SELECT ?o WHERE {
}
```

Results:

```
?o
Q20882
```
Question: What is the height of Mount Everest?
Problem: Motivation

- Question: What is the height of Mount Everest?
- What are the required Wikidata IDs?
  - How is the required data organized in Wikidata?
  - How to formulate the correct query?
Question: What is the height of Mount Everest?

What are the required Wikidata IDs?
How is the required data organized in Wikidata?
How to formulate the correct query?

Query that answers question:
SELECT ?o WHERE {
    wd:Q513 wdt:P2044 ?o .
}
Problem: Definition

- Focus on Simple Questions
Focus on Simple Questions

Given: Natural language question \( q \)

Goal: Find query that answers \( q \) using one of the following two patterns:

**Target: Object**

```sql
SELECT ?o WHERE {
    <entity> <relation> ?o .
}
```

**Target: Subject**

```sql
SELECT ?s WHERE {
    ?s <relation> <entity> .
}
```
Questions?
Approach: Pipeline

question → entity linking → candidate generation → relation matching → ranking → predicted SPARQL query
Question: In which city was Leonhard Euler born?

Identified entities for each subsequence:

\[
\begin{array}{c|c}
  s & E_s \\
  "Leonhard Euler" & \{Q7604, Q58118685, \ldots\} \\
  "city" & \{Q515, \ldots\} \\
  \ldots & \ldots \\
\end{array}
\]
Question: In which city was Leonhard Euler born?

Identified entities for each subsequence:

\[
\begin{array}{c|c}
  s & E_s \\
  "Leonhard Euler" & \{Q7604, Q58118685, \ldots\} \\
  "city" & \{Q515, \ldots\} \\
  \ldots & \ldots \\
\end{array}
\]

Get final set \( E' \) by combining all \( E_s \) and by dropping less promising entities

\[
E' = \{Q7604, Q58118685, Q515, \ldots\}
\]
### Approach: Candidate generation

For each entity in $E'$, we generate all possible query candidates:

<table>
<thead>
<tr>
<th>Entity</th>
<th>Relations Target: Object</th>
<th>Relations Target: Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q7604</td>
<td>{P19, P937, ...}</td>
<td>{P138, ...}</td>
</tr>
<tr>
<td>Q515</td>
<td>{P135, ...}</td>
<td>{P31, P1813, ...}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

In this case 930 query candidates are generated, including the correct query:

```sql
SELECT ?o WHERE {
  wd:Q7604 wdt:P19 ?o .
}
```
Approach: Candidate generation

- For each entity in $E'$, we generate all possible query candidates:

<table>
<thead>
<tr>
<th>Entity</th>
<th>Relations Target: Object</th>
<th>Relations Target: Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q7604</td>
<td>{P19, P937, ...}</td>
<td>{P138, ...}</td>
</tr>
<tr>
<td>Q515</td>
<td>{P135, ...}</td>
<td>{P31, P1813, ...}</td>
</tr>
</tbody>
</table>

- In this case 930 query candidates are generated, including the correct query:

```
SELECT ?o WHERE {
  wd:Q7604 wdt:P19 ?o .
}
```
Illustration of relation scorer for the correct candidate:

**Question**

In which city was Leonhard Euler born?

**Relation**

P19: place of birth
Illustration of relation scorer for the correct candidate:

**Question**

In which city was Leonhard Euler born?

↓ *entity masking*

In which city was `<entity>` born?

**Relation**

P19: place of birth

answer type string

- big city
- birthplace; birth place;
- born in; location born;
- born; birth city; location of birth; location born; born at

relation aliases
Approach: Relation Matching

Illustration of relation scorer for the correct candidate:

**Question**
In which city was Leonhard Euler born?

**Relation**
P19: place of birth

answer type string
- big city
- birthplace
- birth place
- born in
- location born
- born
- birth city
- location of birth
- location born
- born at

Sentence-BERT

BERT

mean pooling

cosine similarity

0.997
Approach: Relation Matching

- Fine-tune relation scorer with the Multiple Negatives Ranking (MNR) loss function:
  - Create batches without duplicates, $q_1, \ldots, q_b$ question representations, $r_1, \ldots, r_b$ relation representations
  - Use cross entropy loss
Approach: Relation Matching

- Fine-tune relation scorer with the Multiple Negatives Ranking (MNR) loss function:
  - Create batches without duplicates, $q_1, \ldots, q_b$ question representations, $r_1, \ldots, r_b$ relation representations
  - Use cross entropy loss
  - Alternative if few relations: contrastive loss function
Approach: Ranking

- Create feature vector for each candidate. Vector of correct candidate: [1, 2, 174, 1, 2, 1, 4, 0, 0.997, 0.57, 3288499]
Approach: Ranking

- Create feature vector for each candidate. Vector of correct candidate: [1, 2, 174, 1, 2, 1, 4, 0, 0.997, 0.57, 3288499]
- Use random forest model for binary classification to infer a pairwise ranking
Approach: Ranking

- Create feature vector for each candidate. Vector of correct candidate: 
  \[ [1, 2, 174, 1, 2, 1, 4, 0, 0.997, 0.57, 3288499] \]
- Use random forest model for binary classification to infer a pairwise ranking
- Compare each pair of candidates and sort candidates by number of "won" comparisons
Questions?
Evaluation: Datasets

- Three different benchmarks, all provide simple questions together with the corresponding gold query
Three different benchmarks, all provide simple questions together with the corresponding gold query

- SimpleQuestions-Wikidata: Translated from SimpleQuestions dataset, low variety in questions
Evaluation: Datasets

- Three different benchmarks, all provide simple questions together with the corresponding gold query
- SimpleQuestions-Wikidata: Translated from SimpleQuestions dataset, low variety in questions
- LC-QuAD 2.0 SQ: Simple questions of LC-QuAD 2.0 dataset
Three different benchmarks, all provide simple questions together with the corresponding gold query

- **SimpleQuestions-Wikidata**: Translated from SimpleQuestions dataset, low variety in questions
- **LC-QuAD 2.0 SQ**: Simple questions of LC-QuAD 2.0 dataset
- **Own questions**: 50 own questions, high variety
Evaluation: Results

- **Accuracy**: Fraction of questions, for which the answers of the predicted query are the same as the answers of the gold query.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
<th>AD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleQuestions-Wikidata</td>
<td>0.816</td>
<td>0.49</td>
</tr>
<tr>
<td>LC-QuAD 2.0 SQ</td>
<td>0.825</td>
<td>0.57</td>
</tr>
<tr>
<td>Own questions</td>
<td>0.820</td>
<td>0.46</td>
</tr>
</tbody>
</table>
Evaluation: Results

- **Accuracy**: Fraction of questions, for which the answers of the predicted query are the same as the answers of the gold query.
- **Main results on the three benchmarks (AD is the average duration per question):**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
<th>AD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleQuestions-Wikidata</td>
<td>0.816</td>
<td>0.49</td>
</tr>
<tr>
<td>LC-QuAD 2.0 SQ</td>
<td>0.825</td>
<td>0.57</td>
</tr>
<tr>
<td>Own questions</td>
<td>0.820</td>
<td>0.46</td>
</tr>
</tbody>
</table>
### Evaluation: Results

- **Accuracy on SimpleQuestions-Wikidata compared to the accuracies of other QA systems:**

<table>
<thead>
<tr>
<th>QA System</th>
<th>SimpleQuestions (FB2M)</th>
<th>SimpleQuestions-Wikidata</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yu et al. (2017)</td>
<td>0.787</td>
<td>-</td>
</tr>
<tr>
<td>Petrochuk et al. (2018)</td>
<td>0.781</td>
<td>-</td>
</tr>
<tr>
<td>Oliya et al. (2021)</td>
<td>-</td>
<td>0.682</td>
</tr>
<tr>
<td>Goette (2021)</td>
<td>-</td>
<td>0.586</td>
</tr>
<tr>
<td>Aqqu Wikidata (2023)</td>
<td>-</td>
<td>0.816</td>
</tr>
</tbody>
</table>
Questions?
## Appendix: All features

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Exact entity match</td>
</tr>
<tr>
<td>2</td>
<td>Exact entity token matches</td>
</tr>
<tr>
<td>3</td>
<td>Entity popularity score</td>
</tr>
<tr>
<td>4</td>
<td>Exact relation match</td>
</tr>
<tr>
<td>5</td>
<td>Literal score</td>
</tr>
<tr>
<td>6</td>
<td>Content literal score</td>
</tr>
<tr>
<td>7</td>
<td>Exact token matches</td>
</tr>
<tr>
<td>8</td>
<td>Similarity score</td>
</tr>
<tr>
<td>9</td>
<td>Relation score</td>
</tr>
<tr>
<td>10</td>
<td>Proportion matched/total tokens</td>
</tr>
<tr>
<td>11</td>
<td>Occurrences relation KG</td>
</tr>
</tbody>
</table>
Appendix: MNR loss

\[ L_{MNR}(q_i, r_1, \ldots, r_b) = -\log \left( \frac{\exp(s \cdot \text{sim}(q_i, r_i))}{\sum_{j=1}^{b} \exp(s \cdot \text{sim}(q_i, r_j))} \right), \]

with \( \text{sim}(q, r) = \frac{q \cdot r}{\|q\|\|r\|} \)
Appendix: Contrastive loss

Loss for single question-relation pair (embeddings $q_i, r_i$) and label $y_i$ can be computed with

$$L_{CL}(q_i, r_i, y_i) = y_i \frac{1}{2} \| q_i - r_i \|_2 + (1 - y_i) \frac{1}{2} \max(0, m - \| q_i - r_i \|_2)^2.$$ 

with $m$ being a parameter that controls the influence of negative pairs.
Appendix: Results for different loss functions

<table>
<thead>
<tr>
<th></th>
<th>SimpleQuestions-Wikidata</th>
<th>LC-QuAD 2.0 SQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNR loss fine-tuning</td>
<td>0.799</td>
<td>0.825</td>
</tr>
<tr>
<td>contrastive loss fine-tuning</td>
<td>0.816</td>
<td>0.807</td>
</tr>
</tbody>
</table>
## Appendix: Detailed evaluation

<table>
<thead>
<tr>
<th></th>
<th>SimpleQuestions-Wikidata</th>
<th>LC-QuAD 2.0 SQ</th>
<th>Own questions</th>
<th>AD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Pipeline</td>
<td>0.816</td>
<td>0.825</td>
<td>0.820</td>
<td>0.50</td>
</tr>
<tr>
<td>w/o rel score</td>
<td>0.673</td>
<td>0.808</td>
<td>0.760</td>
<td>0.44</td>
</tr>
<tr>
<td>w/o rel occs, w/o sim score</td>
<td>0.811</td>
<td>0.823</td>
<td>0.760</td>
<td>0.40</td>
</tr>
<tr>
<td>only rel and popularity score</td>
<td>0.792</td>
<td>0.785</td>
<td>0.740</td>
<td>0.38</td>
</tr>
<tr>
<td>entity sentence: marking</td>
<td>0.795</td>
<td>0.826</td>
<td>0.820</td>
<td>0.59</td>
</tr>
<tr>
<td>fine-tuning WikiQuestions</td>
<td>0.813</td>
<td>0.823</td>
<td>0.820</td>
<td>0.52</td>
</tr>
<tr>
<td>entity pruning: 200/500</td>
<td>0.818</td>
<td>0.819</td>
<td>0.820</td>
<td>1.76</td>
</tr>
<tr>
<td>no candidate pruning</td>
<td>0.816</td>
<td>0.825</td>
<td>0.820</td>
<td>2.01</td>
</tr>
<tr>
<td>Dataset</td>
<td>Accuracy</td>
<td>Top-2</td>
<td>Top-3</td>
<td>Top-5</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>SimpleQuestions-Wikidata</td>
<td>0.816</td>
<td>0.863</td>
<td>0.879</td>
<td>0.889</td>
</tr>
<tr>
<td>LC-QuAD 2.0 SQ</td>
<td>0.825</td>
<td>0.860</td>
<td>0.865</td>
<td>0.873</td>
</tr>
<tr>
<td>Own questions</td>
<td>0.820</td>
<td>0.880</td>
<td>0.920</td>
<td>0.960</td>
</tr>
</tbody>
</table>