Tabular Information Extraction

Master Thesis
Tobias Matysiak

18.12.2019
city | country | population

1. PREFIX wd: <http://www.wikidata.org/entity/>
2. PREFIX wdt: <http://www.wikidata.org/prop/direct/>
3. PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

4. SELECT DISTINCT ?city_name ?country_name ?population
   WHERE {
   10.  FILTER (lang(?city_name) = "en") .
   11.  FILTER (lang(?country_name) = "en") .
   12.}

18.12.2019
Table Definition

- **Column filters**
  - city(="Berlin"@en) | country | population
  - city | country | population(>= 100000)

- **Column orders**
  - city | country | population [DESC]
  - city [ASC, 1] | country | population [DESC, 2]

- **Exact IDs**
  - Q515 | country | population

- **Explicit linking**
  - city | country | population -> 1
Table Matching - Problem Definition

- Input: table definition
- Output: SPARQL query

- Column matching
  - Find a class or property for each column

- Relations between columns
  - Each column must relate to another (except one “master column”)
Table Matching - Problem Definition

<table>
<thead>
<tr>
<th>city</th>
<th>country</th>
<th>population</th>
</tr>
</thead>
</table>

Q515 ("city") matches P17 ("country") matches P1082 ("population")

relates to

relates to

relates to
Approach

- **Search index**
  - find class or property for each column

- **Column graph**
  - find relations between two columns
  - SPARQL templates

- **Pairwise matching of columns** (Search index + Column graph)
Document-term matrix (tf-idf features)

- Documents: labels of classes and properties
- Terms: tri-grams of the labels

Python library *scikit-learn*
1. SELECT ?o WHERE {
2.   ?s <is-a> [CLASS] .
4. }

- `<is-a>` in Freebase: fb:type.object.type
- `<is-a>` in Wikidata: wdt:P31

- **Example**
CC Template (Class – Class)

1. SELECT ?p WHERE {
2.   ?s <is-a> [CLASS1] .
3.   ?o <is-a> [CLASS2] .
5. }

- Example
Column Graph

- A node for each class or property
- Edges between template matching nodes

- Created from RDF dump
  - Entity dict
    - <Entity> <is-a> <Class> .
  - CC Template
    - <Entity1> <Property> <Entity2> .
  - CP Template
    - <Entity> <Property> <Value or other Non-entity> .
Column Pair Matching

- **Input**: a pair of column definitions

<table>
<thead>
<tr>
<th>city</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>location.citiy-town</td>
<td>location.country</td>
</tr>
<tr>
<td>user.brendan.default_domain.top_architectural_city</td>
<td>sports.sport_country</td>
</tr>
<tr>
<td>base.aareas.schema.earth.city-town</td>
<td>film.film.country</td>
</tr>
</tbody>
</table>

- **Output**: ranked list of best matching classes and properties
  - e.g. (location.citiy-town, location.country)
  - e.g. (location.citiy-town, film.film.country)
Baseline Table Matching Algorithm

1. For each column find best matching class or property using the search index
2. Pairwise matching of all best matches
3. Find best related column for each column
4. Master column: The column which is the best related column the most often

<table>
<thead>
<tr>
<th>city</th>
<th>country</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q515 (“city”)</td>
<td>Q6256 (“country”)</td>
<td>P1082 (“population”)</td>
</tr>
</tbody>
</table>

- City: Q515
- Country: Q6256
- Population: P1082
KBTE Table Matching Algorithm

1. Pairwise matching of all columns
2. Based on pairwise matchings
   - Rank columns → master column candidates
   - Create a ranking of the best classes for each candidate
3. Try each candidate and class
   - If each other column has a related column without contradiction → table matching found
Evaluation

- Evaluation on 5000 Wikipedia tables
- Evaluation on 15 handcrafted tables for Freebase and Wikidata
Wikipedia Tables – Dataset Creation

- Tables created from Wikipedia dump
  - 14,051,148 articles (18,665,935 pages)
    - Python library WikiTextParser
  - 2,372,431 tables
    - Filtering & Preprocessing
  - 1,066,292 tables

- Removing of references (<ref></ref>) and line breaks (<br>)
- Links ([[target|text]]) are replaced by their text
- Only tables with row span and col span 1
- Only tables where >50% of the columns are known to search index
### Wikipedia Tables – Results

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Baseline FB</th>
<th>KBTE FB</th>
<th>Baseline WD</th>
<th>KBTE WD</th>
</tr>
</thead>
<tbody>
<tr>
<td>kbte-error</td>
<td>3345 (85.4%)</td>
<td>2938 (75.0%)</td>
<td>3793 (96.9%)</td>
<td>3379 (86.3%)</td>
</tr>
<tr>
<td>sparql-error</td>
<td>46 (1.2%)</td>
<td>66 (1.7%)</td>
<td>2 (0.1%)</td>
<td>17 (0.4%)</td>
</tr>
<tr>
<td>sparql-empty</td>
<td>210 (5.4%)</td>
<td>436 (11.1%)</td>
<td>26 (0.7%)</td>
<td>247 (6.3%)</td>
</tr>
<tr>
<td>sparql-full</td>
<td>314 (8.0%)</td>
<td>475 (12.1%)</td>
<td>94 (2.4%)</td>
<td>272 (6.9%)</td>
</tr>
<tr>
<td># tabledefs</td>
<td></td>
<td></td>
<td>3915</td>
<td></td>
</tr>
</tbody>
</table>

- **kbte-error**: Error during table matching
- **sparql-error**: Error during execution of SPARQL query
- **sparql-empty**: Execution of SPARQL query yields no results
- **sparql-full**: Execution of SPARQL yields results
## Handcrafted Tables – Wikidata (1)

<table>
<thead>
<tr>
<th>Table number</th>
<th>Wikidata table definition &amp; Ground truth</th>
</tr>
</thead>
</table>
| 1 (Persons)  | human | birth | death | spouse  
|              | Q5   | P569 | P570 | P26     |
| 2 (Cities 1) | city | country | population | location 
|              | Q515 | P17 | P1082 | P625 |
| 3 (Cities 2) | city | country | capital  
|              | Q515 | P17 | P1376 |
| 4 (Mountains)| mountain | country | elevation | mountain range 
|              | Q8502 | P17 | P2044 | P4552 |
| 5 (Movies 1) | film | publication date | genre | country | director 
|              | Q11424 | P577 | P136 | P495 | P57 |
| 6 (Movies 2) | film | date | genre | country | director 
|              | Q11424 | P577 | P136 | P495 | P57 |
| 7 (Books 1) | book | author | publication date 
|              | Q571 | P50 | P577  |
| 8 (Books 2) | book | author | date  
<p>|              | Q571 | P50 | P577  |</p>
<table>
<thead>
<tr>
<th>Table number</th>
<th>Wikidata table definition &amp; Ground truth</th>
</tr>
</thead>
</table>
| 9 (Space missions) | mission | human | occupation(="astronaut"@en)  
Q2133344 | Q5 | P106(="astronaut"@en) |
| 10 (Politicians) | human | occupation(="politician"@en) | gender | country | date of birth  
Q5 | P106("politician"@en) | P21 | P17 | P569 |
| 11 (Super Bowls) | super bowl | point in time | location | winner  
Q32096 | P585 | P276 | P1346 |
| 12 (Sports teams) | club | city | sport | league | venue  
Q847017 | Q515 | P641 | P118 | P115 |
| 13 (Buildings) | building | architect | country | height | floors  
Q41176 | P84 | P17 | P2048 | P1101 |
| 14 (Airports) | airport | city | country | patronage  
Q1248784 | Q515 | P17 | P3872 |
| 15 (Chemical elements) | element | number | symbol | mass  
Q11344 | P1086 | P246 | P2067 |
## Handcrafted Tables – Freebase

<table>
<thead>
<tr>
<th>Table number</th>
<th>Freebase table definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Persons)</td>
<td>person</td>
</tr>
<tr>
<td>2 (Cities 1)</td>
<td>city</td>
</tr>
<tr>
<td>3 (Cities 2)</td>
<td>city</td>
</tr>
<tr>
<td>4 (Mountains)</td>
<td>mountain</td>
</tr>
<tr>
<td>5 (Movies 1)</td>
<td>film</td>
</tr>
<tr>
<td>6 (Movies 2)</td>
<td>film</td>
</tr>
<tr>
<td>7 (Books 1)</td>
<td>book</td>
</tr>
<tr>
<td>8 (Books 2)</td>
<td>book</td>
</tr>
<tr>
<td>9 (Space missions)</td>
<td>mission</td>
</tr>
<tr>
<td>10 (Politicians)</td>
<td>politician</td>
</tr>
<tr>
<td>11 (Super Bowls)</td>
<td>super bowl</td>
</tr>
<tr>
<td>12 (Sports teams)</td>
<td>sports teams</td>
</tr>
<tr>
<td>13 (Buildings)</td>
<td>building</td>
</tr>
<tr>
<td>14 (Airports)</td>
<td>airport</td>
</tr>
<tr>
<td>15 (Chemical elements)</td>
<td>element</td>
</tr>
</tbody>
</table>
# Handcrafted Tables - Results

<table>
<thead>
<tr>
<th>Table number</th>
<th>Baseline FB</th>
<th>KBTE FB</th>
<th>Baseline WD</th>
<th>KBTE WD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Persons)</td>
<td>3 / 4 (75%)</td>
<td>4 / 4 (100%)</td>
<td>4 / 4 (100%)</td>
<td>4 / 4 (100%)</td>
</tr>
<tr>
<td>2 (Cities 1)</td>
<td>4 / 5 (80%)</td>
<td>5 / 5 (100%)</td>
<td>3 / 4 (75%)</td>
<td>3 / 4 (75%)</td>
</tr>
<tr>
<td>3 (Cities 2)</td>
<td>2 / 3 (67%)</td>
<td>3 / 3 (100%)</td>
<td>1 / 3 (33%)</td>
<td>2 / 3 (67%)</td>
</tr>
<tr>
<td>4 (Mountains)</td>
<td>2 / 4 (50%)</td>
<td>4 / 4 (100%)</td>
<td>2 / 4 (50%)</td>
<td>4 / 4 (100%)</td>
</tr>
<tr>
<td>5 (Movies 1)</td>
<td>2 / 5 (40%)</td>
<td>5 / 5 (100%)</td>
<td>4 / 5 (80%)</td>
<td>5 / 5 (100%)</td>
</tr>
<tr>
<td>6 (Movies 2)</td>
<td>2 / 5 (40%)</td>
<td>5 / 5 (100%)</td>
<td>4 / 5 (80%)</td>
<td>5 / 5 (100%)</td>
</tr>
<tr>
<td>7 (Books 1)</td>
<td>1 / 3 (33%)</td>
<td>3 / 3 (100%)</td>
<td>2 / 3 (67%)</td>
<td>3 / 3 (100%)</td>
</tr>
<tr>
<td>8 (Books 2)</td>
<td>1 / 3 (33%)</td>
<td>3 / 3 (100%)</td>
<td>2 / 3 (67%)</td>
<td>3 / 3 (100%)</td>
</tr>
<tr>
<td>9 (Space missions)</td>
<td>kbt-e-error</td>
<td>4 / 4 (100%)</td>
<td>2 / 3 (67%)</td>
<td>2 / 3 (67%)</td>
</tr>
<tr>
<td>10 (Politicians)</td>
<td>2 / 4 (50%)</td>
<td>4 / 4 (100%)</td>
<td>3 / 5 (60%)</td>
<td>5 / 5 (100%)</td>
</tr>
<tr>
<td>11 (Super Bowls)</td>
<td>kbt-e-error</td>
<td>2 / 6 (33%)</td>
<td>kbt-e-error</td>
<td>4 / 4 (100%)</td>
</tr>
<tr>
<td>12 (Sports teams)</td>
<td>kbt-e-error</td>
<td>1 / 5 (20%)</td>
<td>4 / 5 (80%)</td>
<td>4 / 5 (80%)</td>
</tr>
<tr>
<td>13 (Buildings)</td>
<td>kbt-e-error</td>
<td>3 / 5 (60%)</td>
<td>3 / 5 (60%)</td>
<td>4 / 5 (80%)</td>
</tr>
<tr>
<td>14 (Airports)</td>
<td>kbt-e-error</td>
<td>4 / 4 (100%)</td>
<td>3 / 4 (75%)</td>
<td>3 / 4 (75%)</td>
</tr>
<tr>
<td>15 (Chemical elements)</td>
<td>kbt-e-error</td>
<td>3 / 4 (75%)</td>
<td>kbt-e-error</td>
<td>4 / 4 (100%)</td>
</tr>
<tr>
<td>overall</td>
<td>19 / 64 (30%)</td>
<td>53 / 64 (83%)</td>
<td>37 / 61 (61%)</td>
<td>55 / 61 (90%)</td>
</tr>
</tbody>
</table>
Conclusion

- KBTE algorithm is more robust and more accurate than baseline algorithm
- KBTE has difficulties with Wikipedia tables
  - Wikipedia tables often lack context
  - Table content may not be contained in FB & WD
- Possible improvements for KBTE (Wikidata)
  - Usage of alternative labels
  - Class hierarchy
  - Allow professions as columns
Summary

Table Definition

Search Index

Column Graph (SPARQL Templates)

Column Pair Matching

Table Matching

KBTE

SPARQL query
Thank you for your attention!
Tabular Description Format

tabledef = columnndef (separator columnndef)*
columnndef = string [filter] [order] [link]
filter = "(" comparator (string | number) ")"
order = "[" ("asc" | "desc") ("," number) "]"
link = ":" number
separator = "|"
string = CHAR+
number = DIGIT+
comparator = "!=" | "<=" | "<" | ">=" | ">" | ">="
## Search Index

<table>
<thead>
<tr>
<th>Documents (classes and properties)</th>
<th>n_p</th>
<th>n_r</th>
<th>nge</th>
<th>nta</th>
<th>oon</th>
<th>pas</th>
<th>ran</th>
<th>tai</th>
<th>unt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q8502 (“mountain”)</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4552 (“mountain range”)</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q133056 (“mountain pass”)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 3.1:** Term frequencies of three exemplary Wikidata documents

### Example Calculation

\[
A = \begin{pmatrix}
0 & 0 & 0.41 & 0 & 0 & 0.41 & 0 & 0 & 0.41 & 0 & 0 & 0.41 \\
0 & 0.36 & 0.21 & 0.36 & 0 & 0.28 & 0.21 & 0 & 0.36 & 0.36 & 0.21 & 0.21 \\
0.39 & 0 & 0.23 & 0 & 0.39 & 0.29 & 0.23 & 0.39 & 0 & 0.23 & 0.23 & 0.39
\end{pmatrix}
\]

\[
q_{range}^\top = \begin{pmatrix}
0 & 0 & 0.58 & 0 & 0 & 0 & 0.58 & 0 & 0 & 0.58 & 0 & 0
\end{pmatrix}
\]

\[
q_{range} \cdot A^\top = \begin{pmatrix}
0 & 0.63 & 0
\end{pmatrix}
\]

**Figure 3.2:** Example calculation of the relevance scores for the input “range”