GHENT
UNIVERSITY

A Simple Approach to Accurately Convert Tabular Data into Semantic Knowledge

Gilles Vandewiele
(PhD student)

Bram Steenwinckel
(PhD student)

prof. dr. Femke Ongenae
(assistant professor, promotor)

prof. dr. Filip De Turck
(professor, promotor)

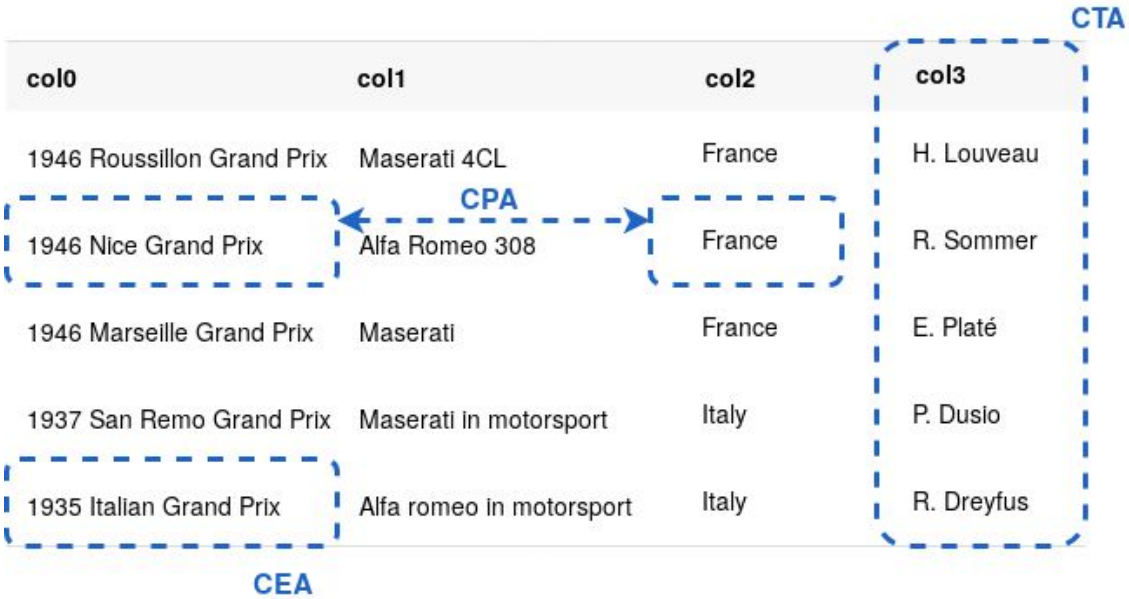
GHENT UNIVERSITY

IDLab
INTERNET & DATA LAB

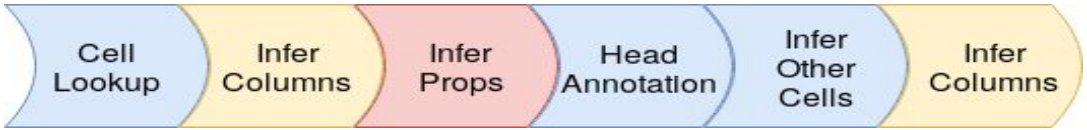
imec

Auckland 2019
ISWC

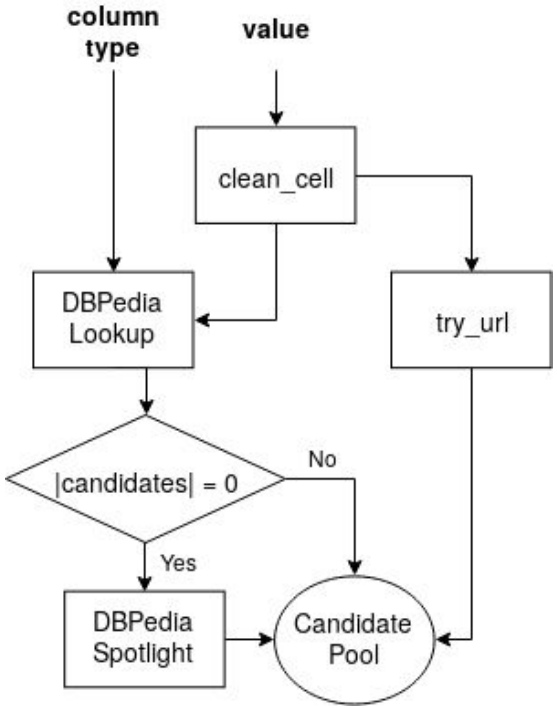
Problem statement



High-level overview



Phase 1: using lookups to create initial annotations



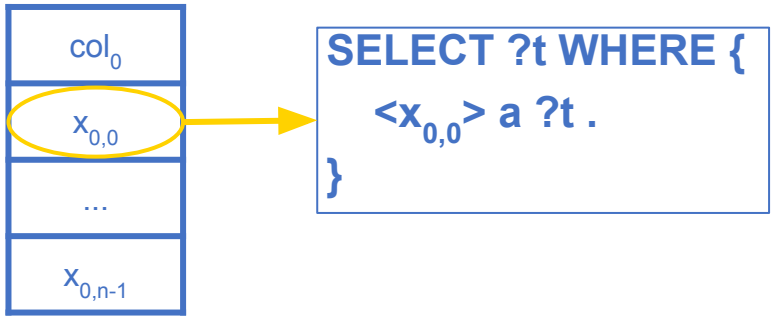
→ detect names & only use family names

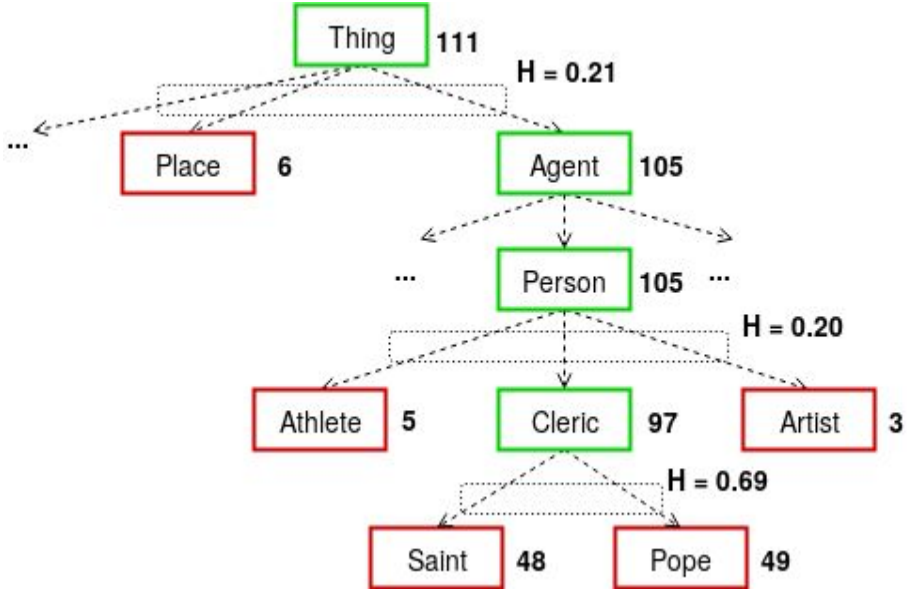
REGEX: "^(\w\.)+([\w\-']+)$"

→ disambiguation is done with Levenshtein distance for non-names & whoswho library for person names

https://github.com/rliebz/whoswho

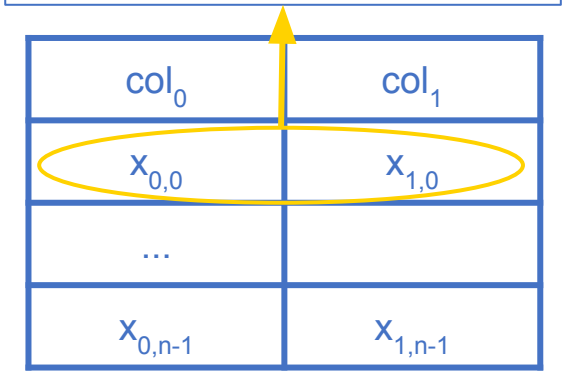
Phase 2: infer columns based on cell annotations





Phase 3: infer properties based on cell annotations and disambiguate with column annotations

```
SELECT ?p WHERE {
    <x0,0> ?p <x1,0> .
}
```



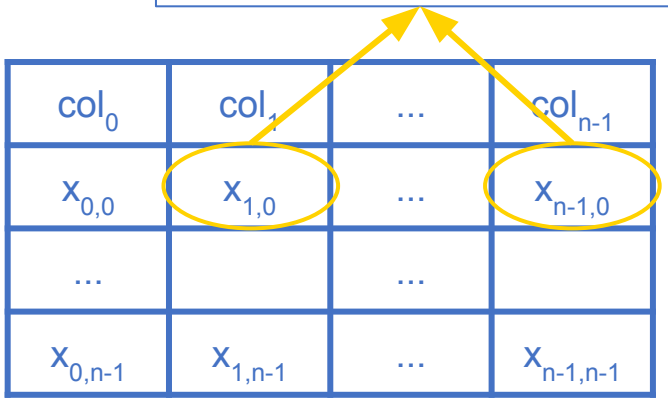
Disambiguation:

Look for domain & range in column types

```
SELECT ?domain ?range WHERE {
    <pred> rdfs:domain ?domain .
    <pred> rdfs:range ?range .
}
```

Phase 4: annotate the head cells with the properties

```
SELECT ?s WHERE {
   ?s <pred> <x1,0> .
}
```



→ Take ?s with highest counts. In case of ex aequo, use Levenshtein.

Phase 5: annotate all other cells

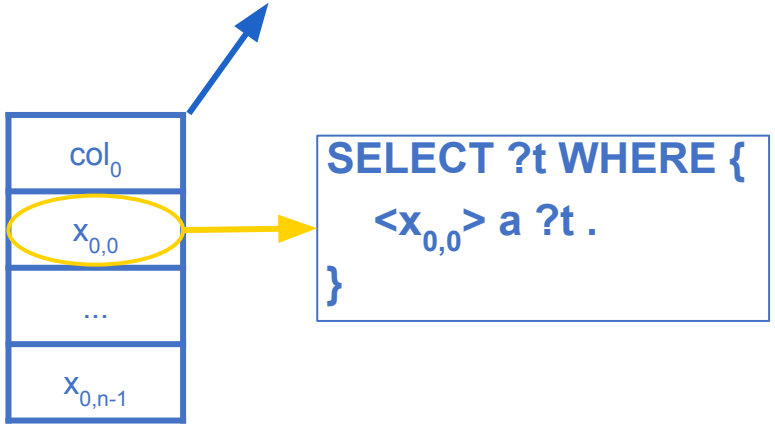
```
SELECT ?o WHERE {
    <x0,0> <pred> ?o .
}
```

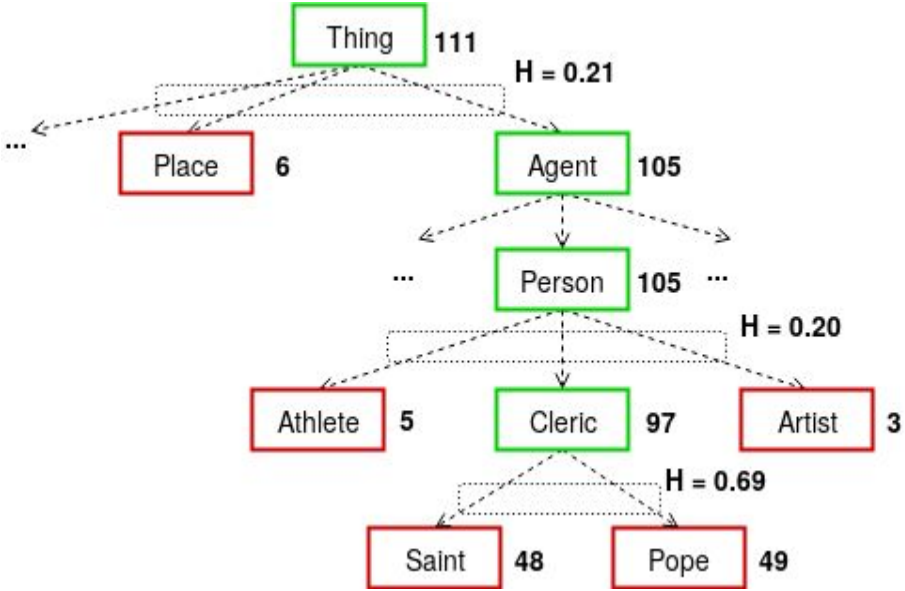
col_0	col_1	…	col_{n-1}
$x_{0,0}$	$x_{1,0}$	…	$x_{n-1,0}$
…		…	
$x_{0,n-1}$	$x_{1,n-1}$	…	$x_{n-1,n-1}$

→ **Disambiguate with Levenshtein**

Phase 6: final column annotation

Higher quality cell annotations



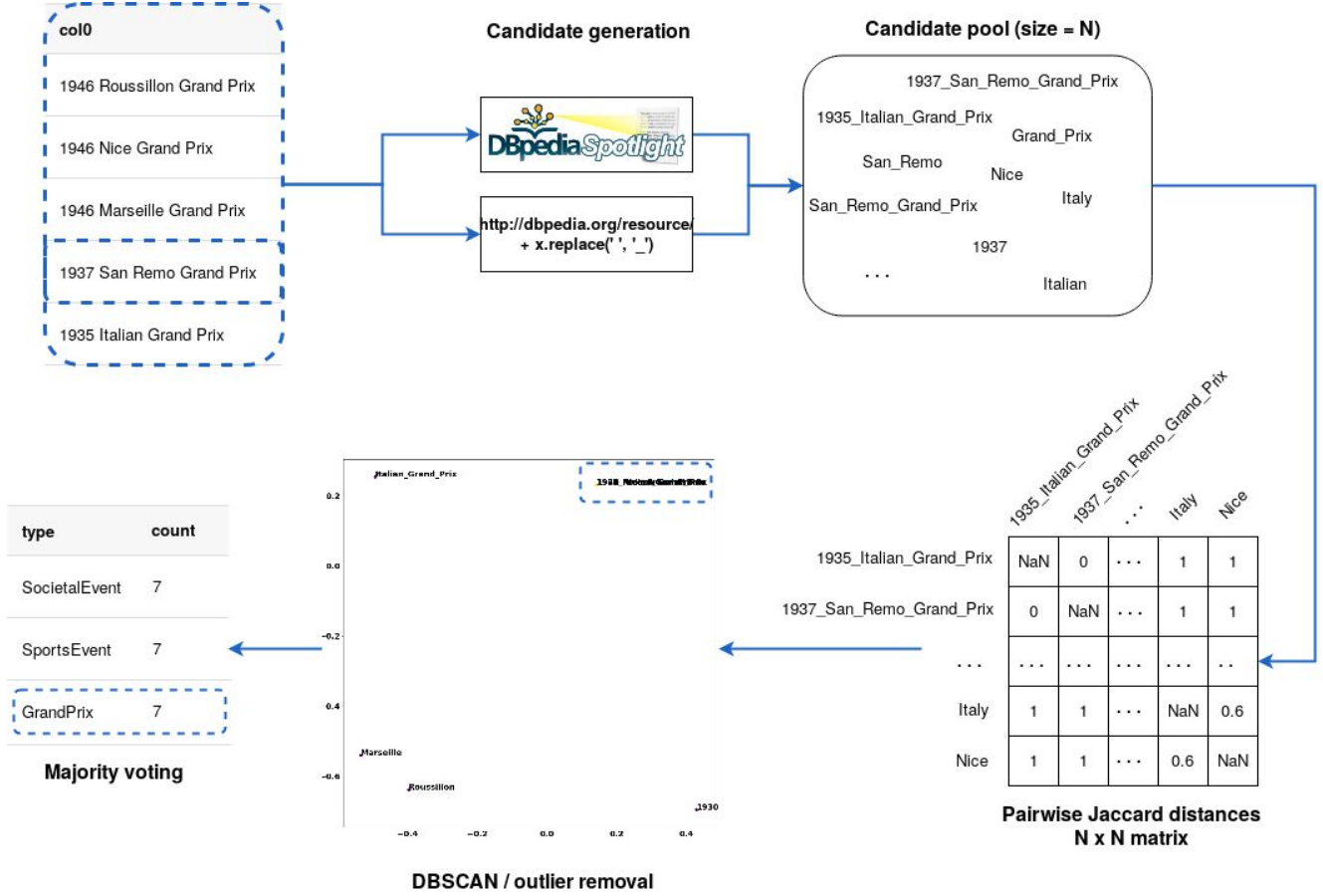


Some sly tricks to boost our score

- Many names (e.g. G. Vandewiele, B. Steenwinckel)
  → custom code for these
- CTA score is not bounded by 1! Add all the parents to the column annotation
  → Max score per row if perfect type is on depth d:
  $1 + (d - 1) * 0.5$
- Reasoning to find equivalent classes and add these as well
- Find tables that are very similar (in earlier rounds the CSV headers often matched) and apply majority voting

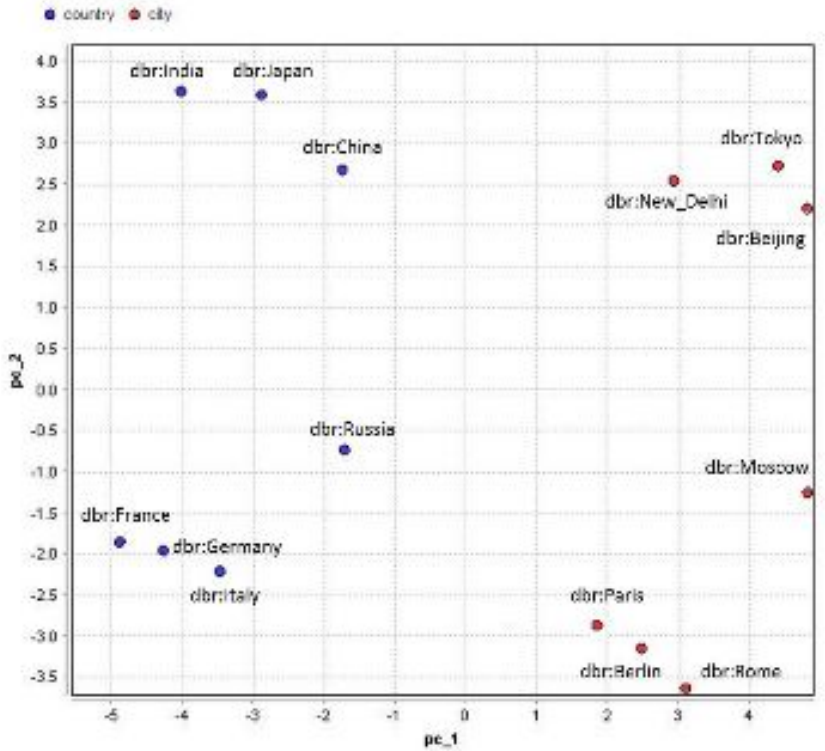
Things we tried, but didn't work well

Clustering of lookup candidates using jaccard distances between their rdf types.



Things we tried, but didn't work well

Playing around (outlier removal, clustering, …) with pre-made RDF2Vec embeddings for DBPedia



a) *DBpedia vectors*

https://github.com/IBCNServices/pyRDF2Vec

Results: Round 1

CTA

▼	06	IDLab IDLab	0.833	0.833

Results: Round 2

CEA

●	02	IDLab IDLab	0.883	0.893

CTA

●	02	IDLab IDLab	1.376	0.257

CPA

●	02	IDLab IDLab	0.877	0.926

Results: Round 3

CEA

▼	02	IDLab IDLab	0.962	0.964

CTA

▼	02	IDLab IDLab	1.864	0.247

CPA

●	02	IDLab IDLab	0.841	0.843

Results: Round 4

CEA

▼	03	IDLab IDLab	0.907	0.912

CTA

●	02	IDLab IDLab	1.846	0.274

CPA

●	02	IDLab IDLab	0.830	0.835

Conclusion & future work

- We first tried more sophisticated approaches, they were all subpar → KISS

- Simple approach performs really well (second place overall)

- The iterative approach can easily be replaced by a better approach that jointly learns to annotate properties, column types and cells (keeping track of all possible candidates)

Thank you!


gilles.vandewiele@ugent.be


https://twitter.com/Gillesvdwiele


https://www.linkedin.com/in/gillesvandewiele/


www.gillesvandewiele.com

Paper:
http://www.cs.ox.ac.uk/isg/challenges/sem-tab/papers/IDLab.pdf
Code (WIP):
https://github.com/IBCNServices/CSV2KG