



**GHENT
UNIVERSITY**

A Simple Approach to Accurately Convert Tabular Data into Semantic Knowledge



Gilles Vandewiele
(PhD student)



Bram Steenwinckel
(PhD student)



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



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

Problem statement

col0	col1	col2	col3
1946 Roussillon Grand Prix	Maserati 4CL	France	H. Louveau
1946 Nice Grand Prix	Alfa Romeo 308	France	R. Sommer
1946 Marseille Grand Prix	Maserati	France	E. Platé
1937 San Remo Grand Prix	Maserati in motorsport	Italy	P. Dusio
1935 Italian Grand Prix	Alfa romeo in motorsport	Italy	R. Dreyfus

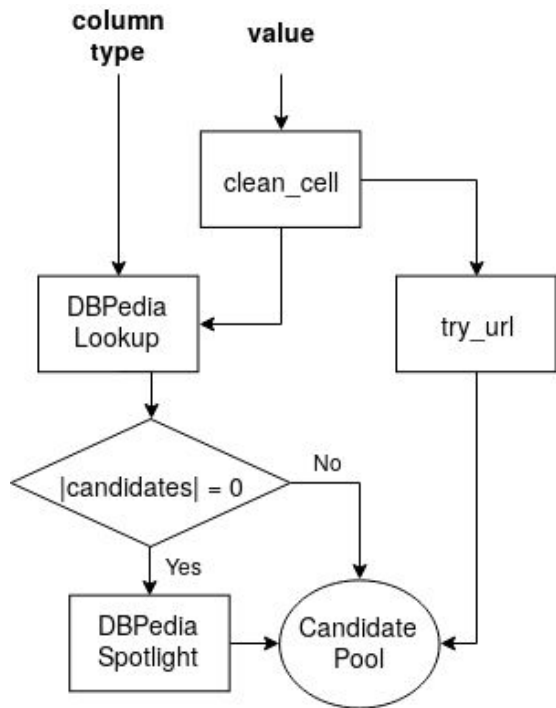
Diagram annotations:

- A dashed blue box labeled **CEA** encloses the first two columns (col0 and col1) for the 1946 Nice Grand Prix row.
- A dashed blue box labeled **CTA** encloses the last column (col3) for all rows.
- A dashed blue box encloses the last two columns (col2 and col3) for the 1946 Nice Grand Prix row.
- A dashed blue arrow labeled **CPA** points from the 1946 Nice Grand Prix row in the CTA box back to the 1946 Nice Grand Prix row in the CEA box.

High-level overview



Phase 1: using lookups to create initial annotations



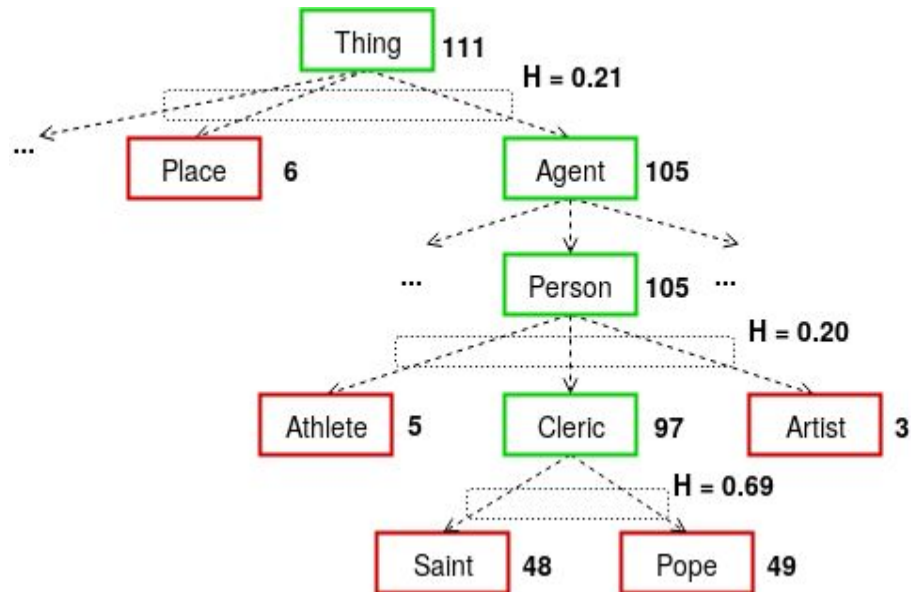
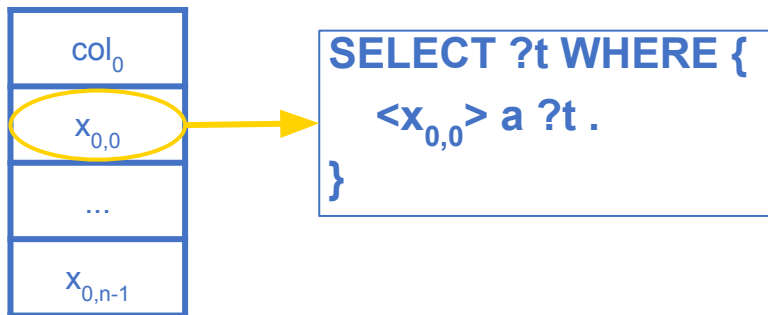
→ detect names & only use family names
REGEX: `"^(\w\ .)+([\w\-']+)$"`

→ 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> .  
}
```

col ₀	col ₁
x _{0,0}	x _{1,0}
...	
x _{0,n-1}	x _{1,n-1}

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.

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}



Phase 5: annotate all other cells

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

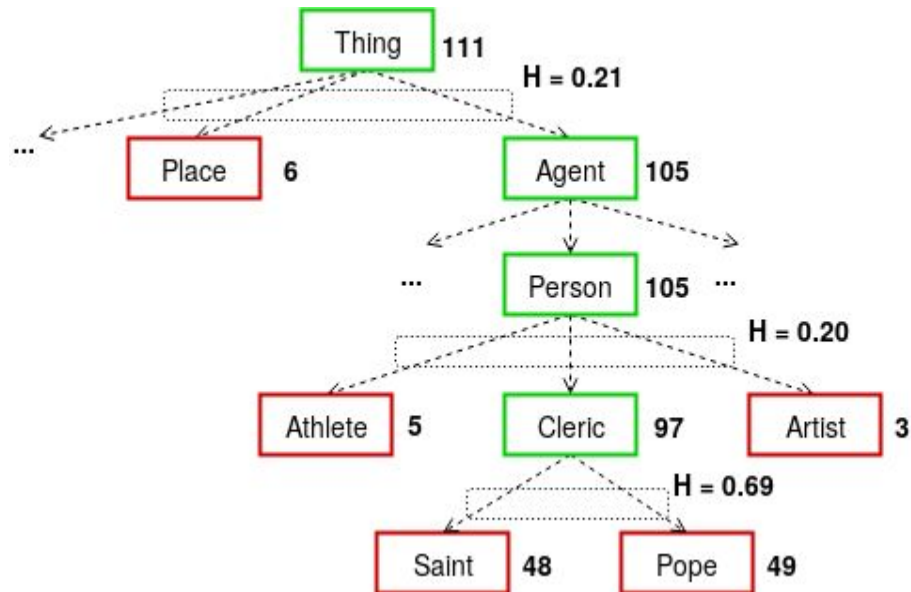
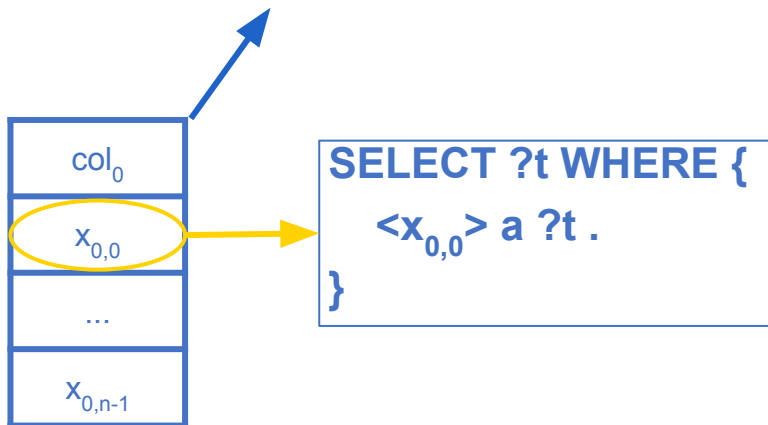
col ₀	col ₁	...	col _{n-1}
x _{0,0}	x _{1,0}	...	x _{n-1,0}
...		...	
x _{0,n-1}	x _{1,n-1}	...	x _{n-1,n-1}

→ 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

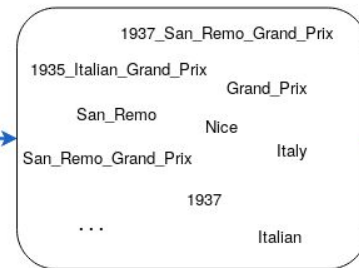
Clustering of lookup candidates using jaccard distances between their rdf types.



Candidate generation

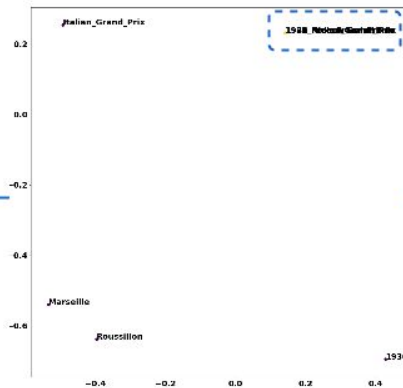


Candidate pool (size = N)



type	count
SocietalEvent	7
SportsEvent	7
GrandPrix	7

Majority voting



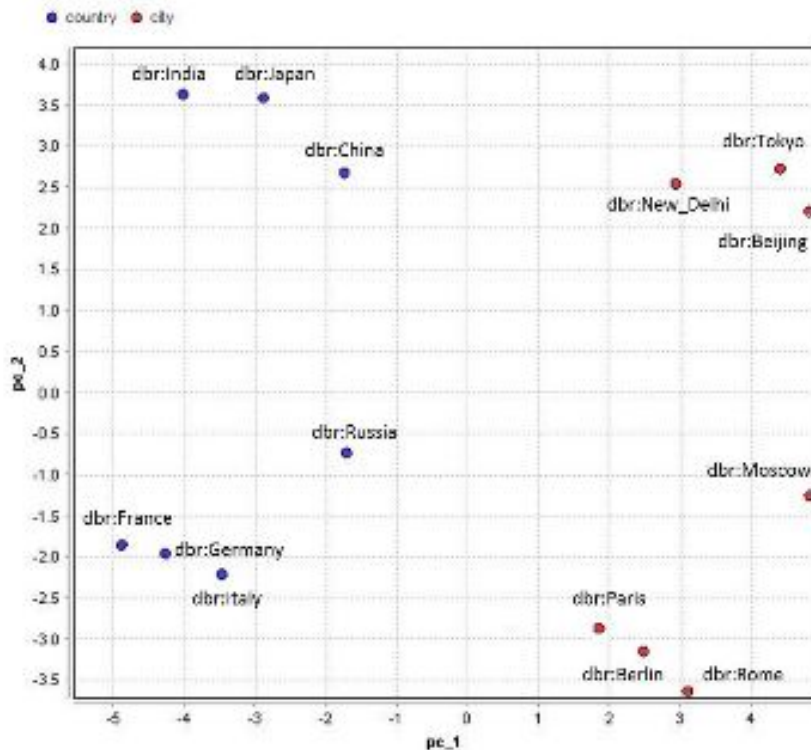
DBSCAN / outlier removal

	1935_Italian_Grand_Prix	1937_San_Remo_Grand_Prix	...	Italy	Nice
1935_Italian_Grand_Prix	NaN	0	...	1	1
1937_San_Remo_Grand_Prix	0	NaN	...	1	1
...
Italy	1	1	...	NaN	0.6
Nice	1	1	...	0.6	NaN

Pairwise Jaccard distances
N x N matrix

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	<small>IDLab</small> IDLab	0.883	0.893
---	-----------	----------------------------	--------------	--------------

CTA

●	02	<small>IDLab</small> IDLab	1.376	0.257
---	-----------	----------------------------	--------------	--------------

CPA

●	02	<small>IDLab</small> 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>