

Public Knowledge Graphs

IE650 Knowledge Graphs



Previously on “Knowledge Graphs”

- Principles:
 - RDF, RDF-S, SPARQL & co
 - Linked Open Data
- Today
 - What knowledge graphs exists and which to choose
 - Knowledge Graph Construction
 - Knowledge Graph Validation

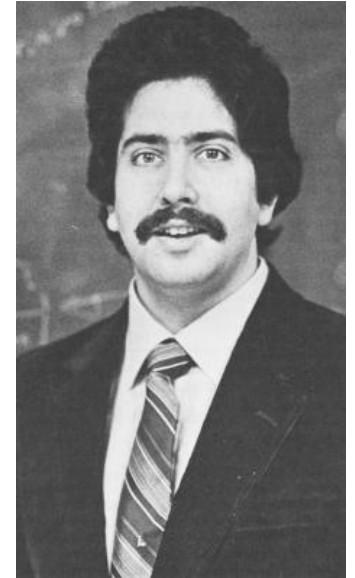
Introduction

- Knowledge Graphs out there
(not guaranteed to be complete)

Name	Instances	Facts	Types	Relations	
DBpedia (English)	4,806,150	176,043,129	735	2,813	public
YAGO	4,595,906	25,946,870	488,469	77	
Freebase	49,947,845	3,041,722,635	26,507	37,781	
Wikidata	15,602,060	65,993,797	23,157	1,673	
NELL	2,006,896	432,845	285	425	
OpenCyc	118,499	2,413,894	45,153	18,526	
Google's Knowledge Graph	570,000,000	18,000,000,000	1,500	35,000	private
Google's Knowledge Vault	45,000,000	271,000,000	1,100	4,469	
Yahoo! Knowledge Graph	3,443,743	1,391,054,990	250	800	

Knowledge Graph Creation: CyC

- The beginning
 - Encyclopedic collection of knowledge
 - Started by Douglas Lenat in 1984
 - Estimation: 350 person years and 250,000 rules should do the job of collecting the essence of the world's knowledge
- The present (as of June 2017)
 - ~1,000 person years, \$120M total development cost
 - 21M axioms and rules
 - Used to exist until 2017

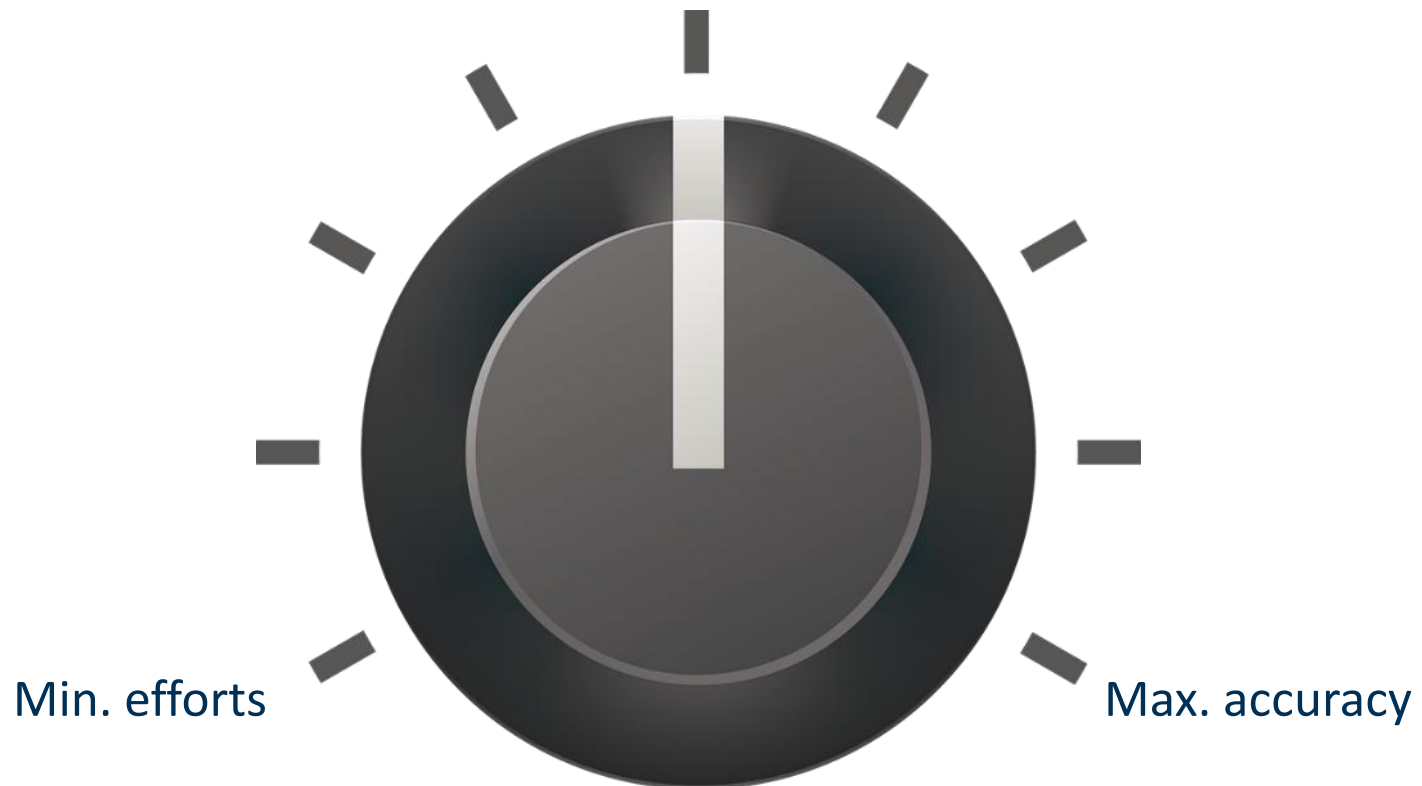


UNIVERSITY
OF MANNHEIM
Data and Web Science Group



Knowledge Graph Creation

- Lesson learned no. 1:
 - Trading efforts against accuracy



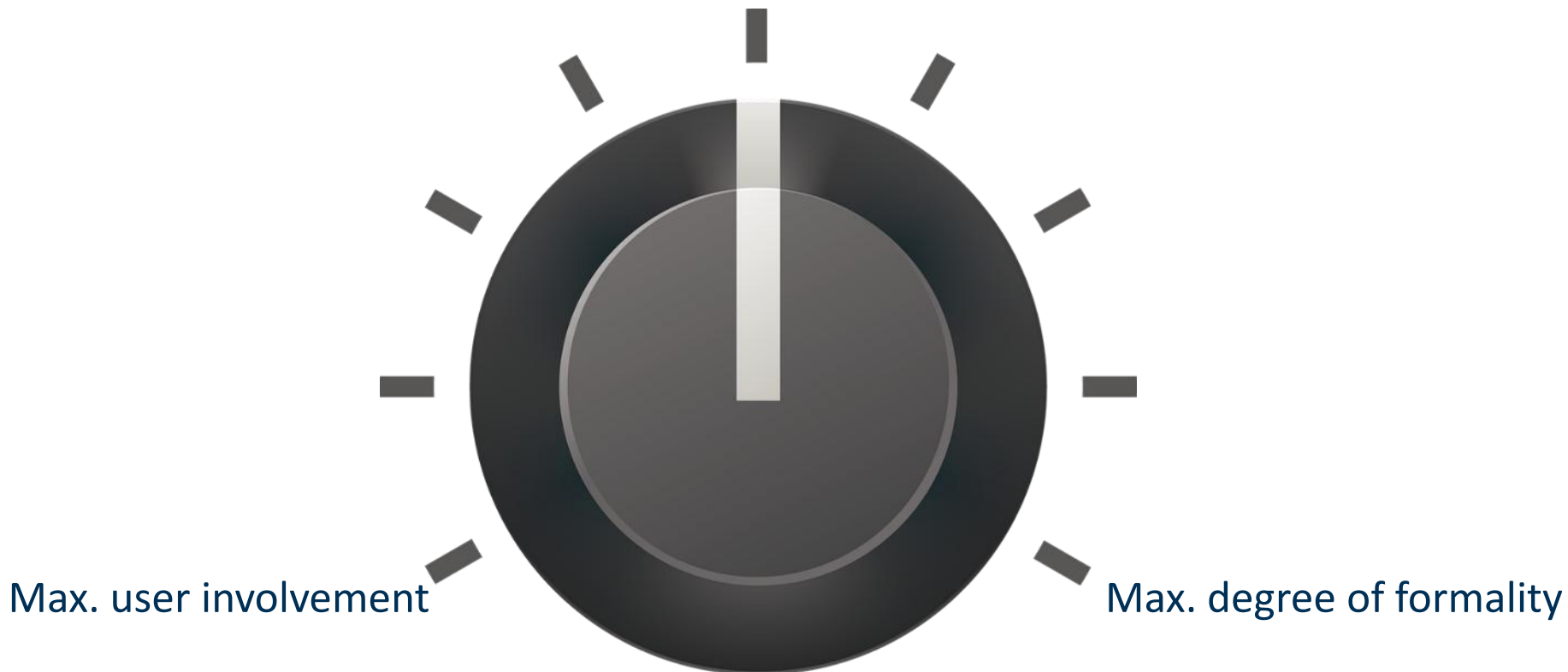
Knowledge Graph Creation: Freebase

- The 2000s
 - Freebase: collaborative editing
 - Schema not fixed
- Present
 - Acquired by Google in 2010
 - Powered first version of Google's Knowledge Graph
 - Shut down in 2016
 - Partly lives on in Wikidata



Knowledge Graph Creation

- Lesson learned no. 2:
 - Trading formality against number of users

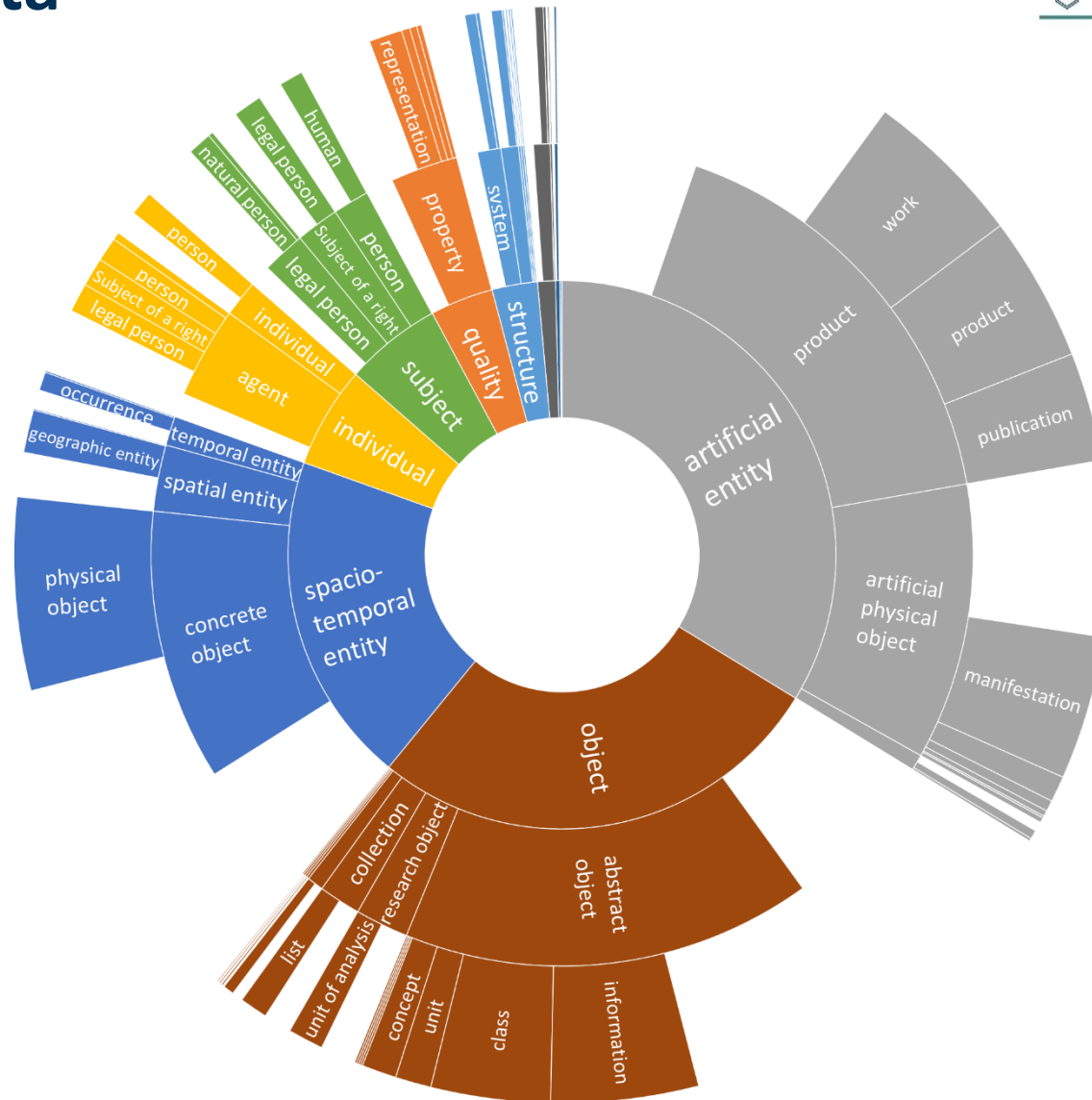


Knowledge Graph Creation: Wikidata

- The 2010s
 - Wikidata: launched 2012
 - Goal: centralize data from Wikipedia languages
 - Collaborative
 - Imports other datasets
- Present
 - One of the largest public knowledge graphs (see later)
 - Includes rich provenance

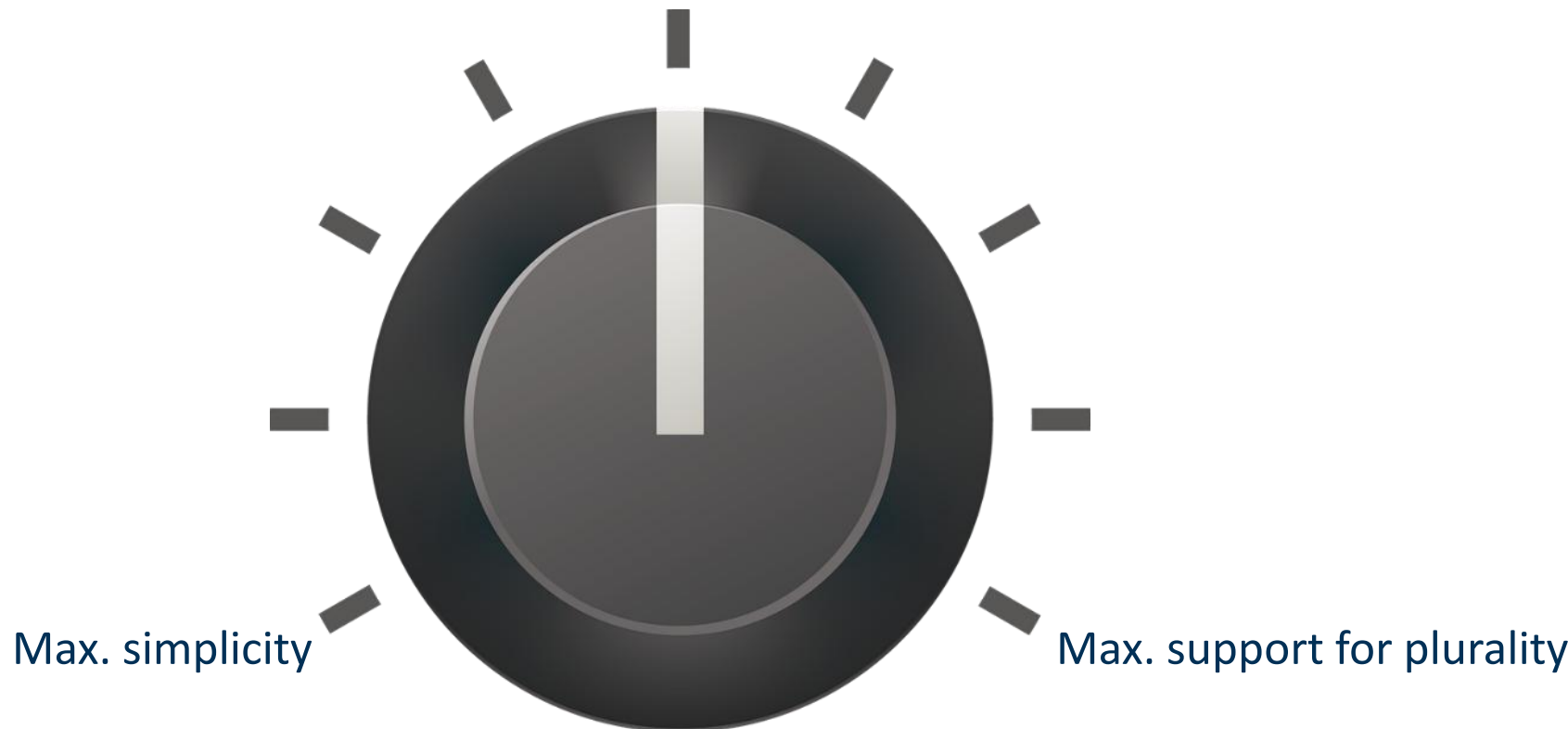


Wikidata



Knowledge Graph Creation

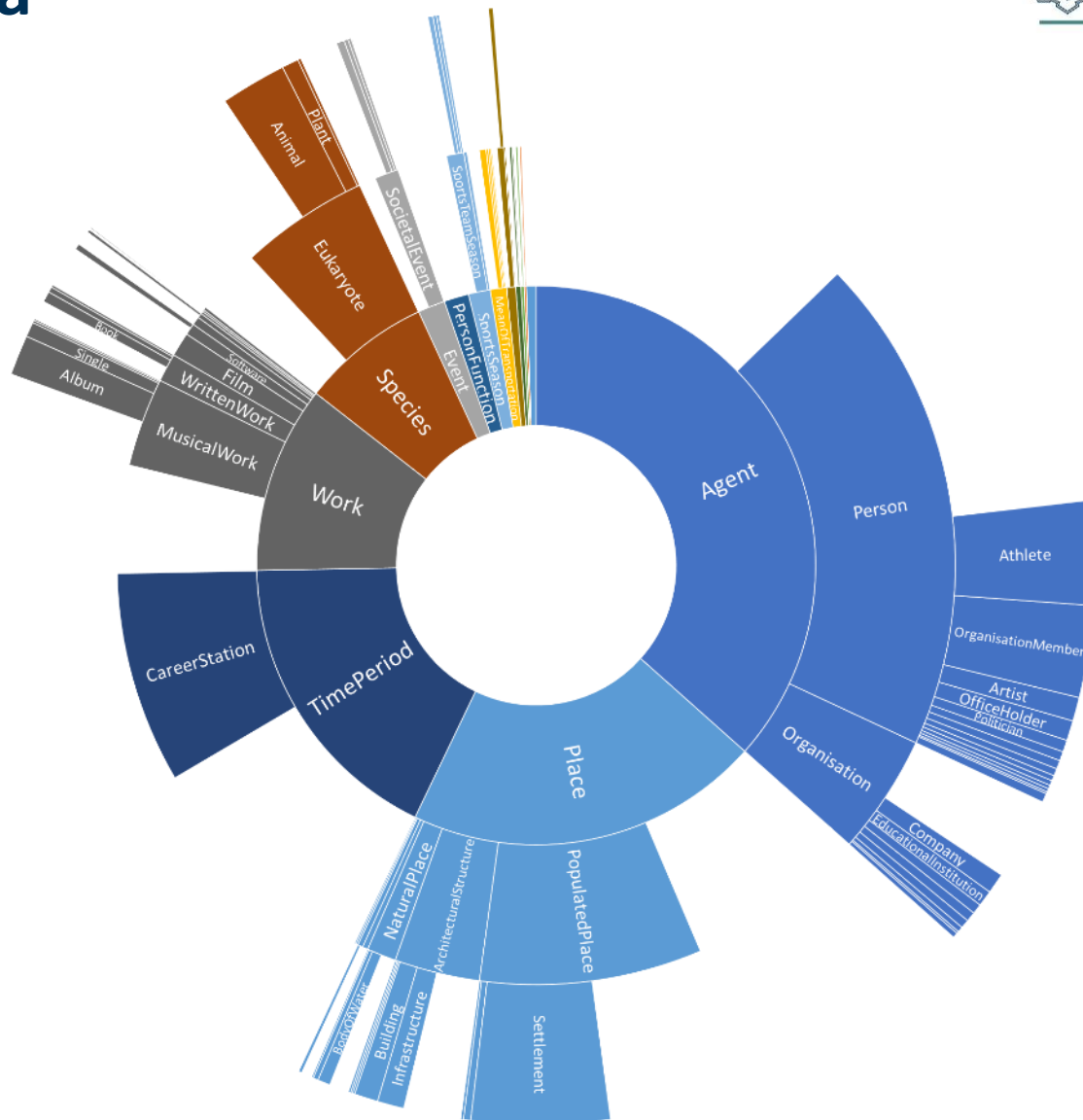
- Lesson learned no. 3:
 - There is not one truth (but allowing for plurality adds complexity)



Knowledge Graph Creation: DBpedia & YAGO

- The 2010s
 - DBpedia: launched 2007
 - YAGO: launched 2008
 - Extraction from Wikipedia using mappings & heuristics
- Present
 - Two of the most used knowledge graphs
 - Next to Wikidata

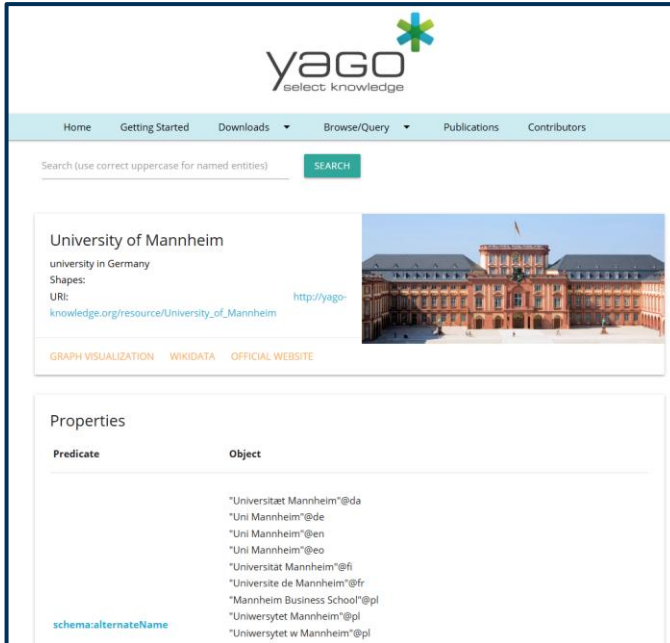




YAGO

- YAGO 1
 - Wikipedia categories for types
 - Plus WordNet as upper structure
- YAGO 2
 - Anchored in time and space
- YAGO 3
 - Further improvements + multiple languages
- YAGO 4
 - Based on Wikidata
 - Refines taxonomy and properties by using schema.org

<Mannheim>		
← <Reinhard_Bütikofer> ← <Umit_Davala> ← <Hans_Martin_Pippart> ← <de/Klaus_May> ← <Werner_Gatel> ← <de/Fritz_Röbling> ← <Reinhold_Fanz> ← <Peter_Dreher> ← <Hans-Jürgen_Boysen> ← <Albert_Speer> ← <Manuel_Guido> ← <Caroline_Augusta_of_Bavaria> ← <Reiner_Hollich> ← <Ludwig_Landmann> ← <Timo_Zahleiter> ← <Jochen_Hecht> ← <Carolin_Leonhardt> ← <de/Mark_Etz> ← <Max_Noether> ← <de/Marc_Hemmerich>		"Metropolregion Rhein-Neckar"@de "World's 15 Most Inventive Cities" "The Manhattan of Germany: the i "The rise of the smart city"@eng "Germany and the Second World" "Ausgabe der Klimadaten. Monats "Press release announcing the me "FEI European Jumping Champion "Partner und Freundsstädte"@en "Grase infräkte (Twin cities of Min "Swansea - Wales .Mannheim.de"
		<hasCitationTitle>
		<Sepp_Herberger> <Jiangsu> <Prince-electo> <Karl_Benz> <Frankfurt> <Dennis_Siver> <Uwe_Rahn> <Jochen_Hecht> <Norwich_Duff> <SAP_Arena> <Oceanic_climate> <Ontario> <Lithuania>
<2010_IJHF_World_Championship> ← <de/Schlacht_bei_Seckenheim> <Firstadion am Friedhofsbereich>	<wasBornIn> <happenedIn>	



The screenshot shows the YAGO website interface. At the top is the YAGO logo with the tagline "select knowledge". Below the logo is a navigation bar with links: Home, Getting Started, Downloads, Browse/Query, Publications, and Contributors. A search bar is present with the placeholder text "Search (use correct uppercase for named entities)".

The main content area displays the entry for "University of Mannheim". It includes a description: "university in Germany", a list of shapes: "Shapes: URI: knowledge.org/resource/University_of_Mannheim", and a URL: "http://yago-". To the right of the text is a photograph of the University of Mannheim building.

Below the main entry, there are links for "GRAPH VISUALIZATION", "WIKIDATA", and "OFFICIAL WEBSITE".

At the bottom, there is a section titled "Properties" with a table showing predicates and their corresponding objects.

Predicate	Object
schema:alternateName	"Universitaet Mannheim"@da
	"Uni Mannheim"@de
	"Uni Mannheim"@en
	"Uni Mannheim"@eo
	"Universitat Mannheim"@fi
	"Universite de Mannheim"@fr
	"Mannheim Business School"@pl
	"Uniwerytet Mannheim"@pl
	"Uniwerytet w Mannheim"@pl

YAGO 3



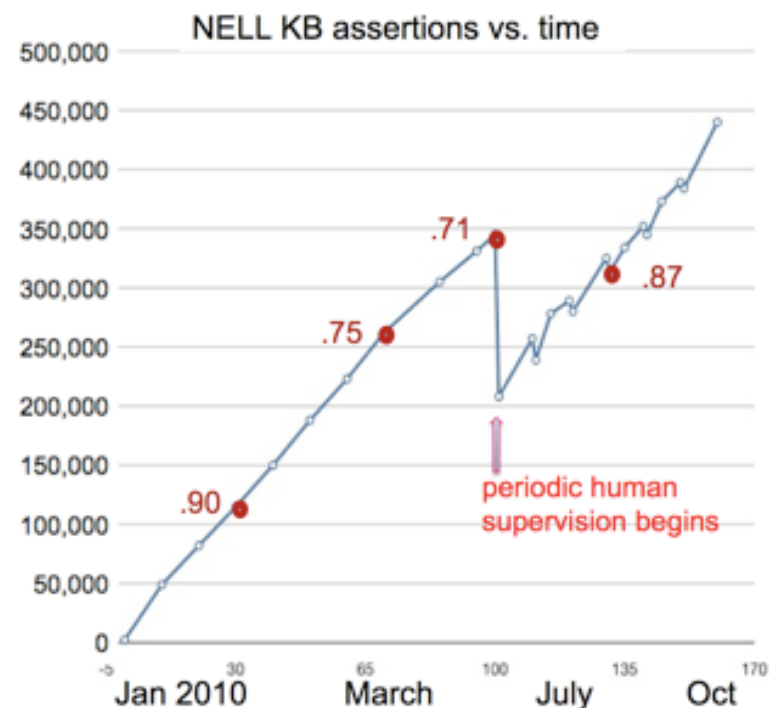
Knowledge Graph Creation

- Lesson learned no. 4:
 - Heuristics help increasing coverage (at the cost of accuracy)



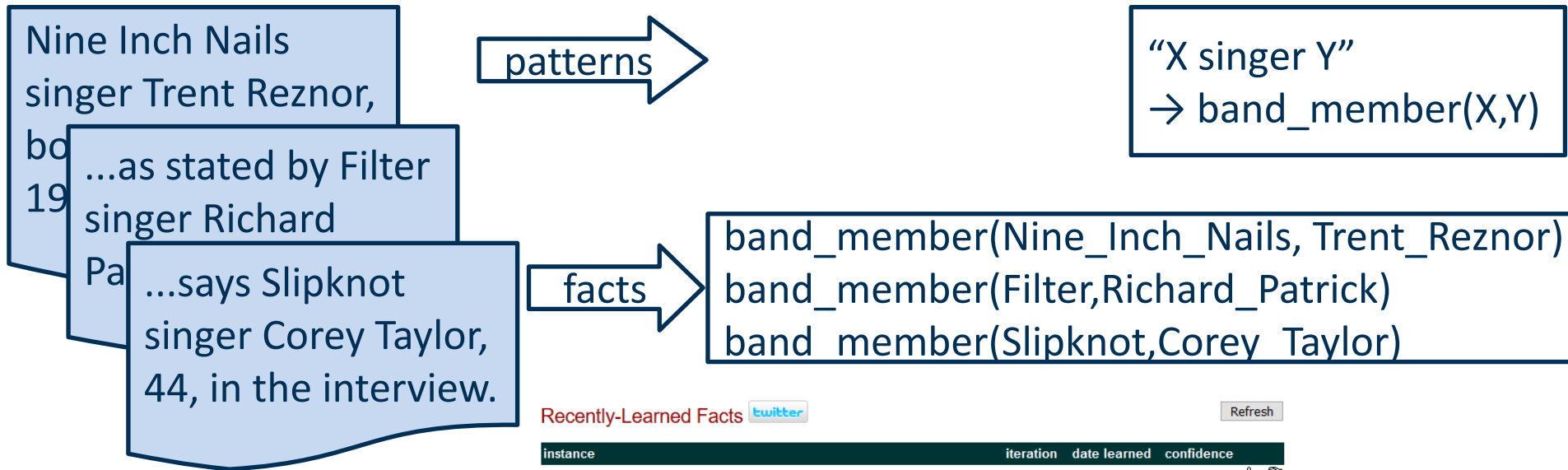
Knowledge Graph Creation: NELL

- The 2010s
 - NELL: Never ending language learner
 - Input: ontology, seed examples, text corpus
 - Output: facts, text patterns
 - Large degree of automation, occasional human feedback
- Until 2018
 - Continuously ran for ~8 years
 - New release every few days



Knowledge Graph Creation: NELL

- Extraction of a Knowledge Graph from a Text Corpus



Recently-Learned Facts

[Refresh](#)

instance	iteration	date learned	confidence
conversion_table is an item found on the floor	1111	06-jul-2018	99.8  
arlene_martel is a comedian	1111	06-jul-2018	94.0  
cigar_rights is a socio-political term	1111	06-jul-2018	95.7  
linton_zoological_gardens is an aquarium	1111	06-jul-2018	100.0  
robb_miller_coaches a sports team	1111	06-jul-2018	91.4  
eric_e_schmidt is a person who was written about in new york times	1111	06-jul-2018	100.0  
rodin is a visual artist in the field of sculpture	1115	03-sep-2018	99.6  
the_today_show is a company in the economic sector of news	1114	25-aug-2018	93.0  
china is a country located in the geopolitical location other_countries	1111	06-jul-2018	100.0  
jerusalem is a city located in the geopolitical location israel	1114	25-aug-2018	99.8  

Knowledge Graph Creation: NELL



Knowledge Graph Creation

- Lesson learned no. 5:
 - Quality cannot be maximized without human intervention



Summary of Trade Offs

- (Manual) effort vs. accuracy and completeness
- User involvement (or usability) vs. degree of formality
- Simplicity vs. support for plurality and provenance

→ all those decisions influence the shape of a knowledge graph!



Welcome to GPTKB

A large general-domain knowledge base entirely from a large language model

100 million
triples

6.1 million
entities

10x
less cost

Overview

This interface presents **GPTKB v1.5** (Hu et al. arXiv, 2025b), a large general-domain knowledge base (KB) entirely from a large language model (LLM). It demonstrates the feasibility of large-scale KB construction from LLMs, while highlighting specific challenges arising around entity recognition, entity and property canonicalization, and taxonomy construction (Hu et al. ACL, 2025a).

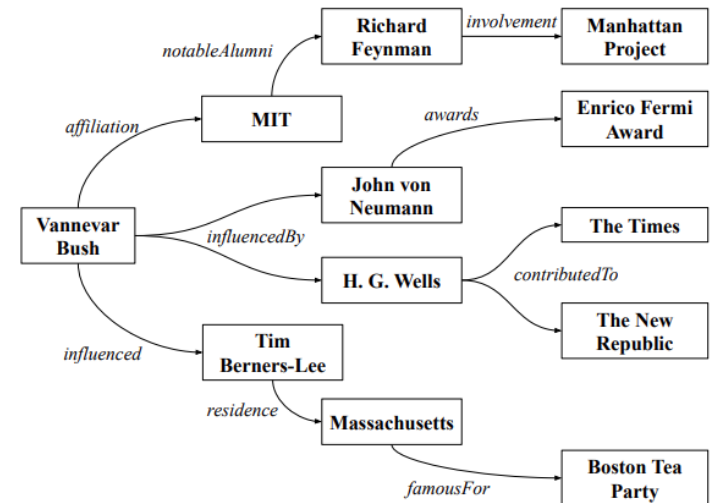
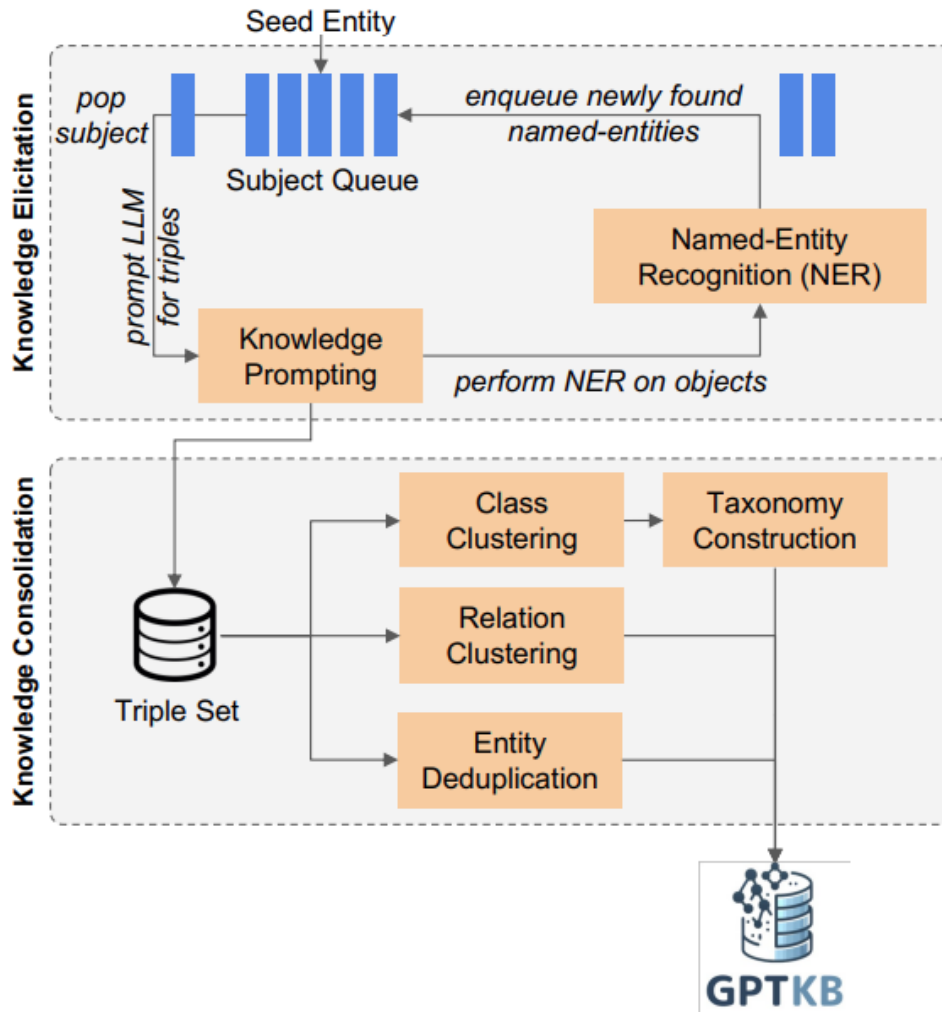
Based on GPT-4.1, **GPTKB v1.5** contains 100 million triples for more than 6.1 million entities, at a cost 10x less than previous KBC projects. We also provide GPTKB v1.1 for download.

GPTKB is a landmark for two fields:

- For NLP, for the first time, it provides constructive insights into the knowledge (or beliefs) of LLMs.
- For the Semantic Web, it shows novel ways forward for the long-standing challenge of general-domain KB construction.

Using the search field, you can [find entities by name](#), and browse their triples. Alternatively, you can [write SPARQL queries](#) or [download](#) the whole KB as a TTL file. [Paper](#) and [code](#) are also available.

GPTKB



You are a knowledge base construction expert. Given a subject entity, return all facts that you know for the subject as a list of subject, predicate, object triples. The number of facts may be very high, between 50 to 100 or more, for very popular subjects. For less popular subjects, the number of facts can be very low, like 5 or 10.

Important:

- If you don't know the subject, return an empty list.
- If the subject is not a named entity, return an empty list.
- If the subject is a named entity, include at least one triple where predicate is "instanceOf".
- Do not get too wordy.
- Separate several objects into multiple triples with one object.

Figure 9: Prompt for knowledge elicitation.

You are an expert on named entity recognition (NER). Your task is to classify if given phrases are named entities (e.g., persons, organizations, works of art), or not (e.g., literals, dates, URLs, verbose phrases). Each phrase is given to you in a line.

Figure 10: Prompt for named-entity recognition (NER).

Entities	2.9M
Triples	101M
Relations	567K (788K before canonicalization)
Classes	4,715 (103K before canonicalization)
Triple objects	37M entities, 64M literals
Avg. triples/entity	35
Avg. label length	21.8 characters
Avg. outlinks	4
Subject-precision*	74% Verifiable, 9% Plausible, 17% Unverifiable
Subjects in Wikidata*	37% in WD, 63% not in WD
Triple-precision*	31% True, 61% Plausible, 1% Implausible, 7% False

** as a weighted average across layers.*

Non-Public Knowledge Graphs

- Many companies have their own private knowledge graphs
 - Google: Knowledge Graph, Knowledge Vault
 - Yahoo!: Knowledge Graph
 - Microsoft: Satori
 - Facebook: Entities Graph
 - Thomson Reuters: permid.org (partly public)
- However, we usually know only little about them



Comparison of Knowledge Graphs

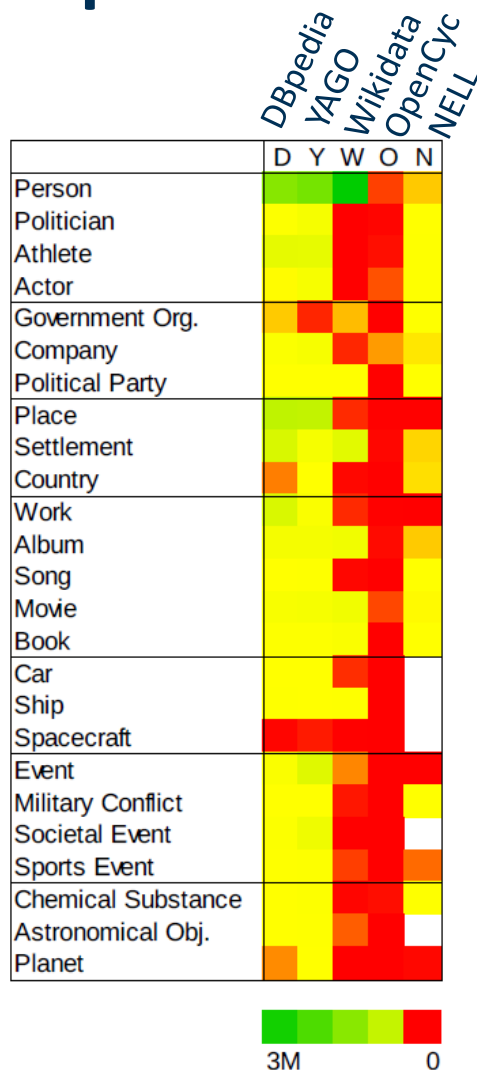
2020

	DBpedia	YAGO	Wikidata	BabelNet
# Instances	5,044,223	6,349,359	52,252,549	7,735,436
# Assertions	854,294,312	479,392,870	732,420,508	178,982,397
Avg. linking degree	21.30	48.26	6.38	0.00
Median ingoing edges	0	0	0	0
Median outgoing edges	30	95	10	9
# Classes	760	819,292	2,356,259	6,044,564
# Relations	1355	77	6,236	22
Avg. depth of class tree	3.51	6.61	6.43	4.11
Avg. branching factor of class tree	4.53	8.48	36.48	71.0
Ontology complexity	SHOFD	SHOIF	SOD	SO
	Cyc	NELL	CaLiGraph	Voldemort
# Instances	122,441	5,120,688	7,315,918	55,861
# Assertions	2,229,266	60,594,443	517,099,124	693,428
Avg. linking degree	3.34	6.72	1.48	0
Median ingoing edges	0	0	0	0
Median outgoing edges	3	0	1	5
# Classes	116,821	1,187	755,963	621
# Relations	148	440	271	294
Avg. depth of class tree	5.58	3.13	4.74	3.17
Avg. branching factor of class tree	5.62	6.37	4.81	5.40
Ontology complexity	SHOIFD	SROIF	SHOD	SH

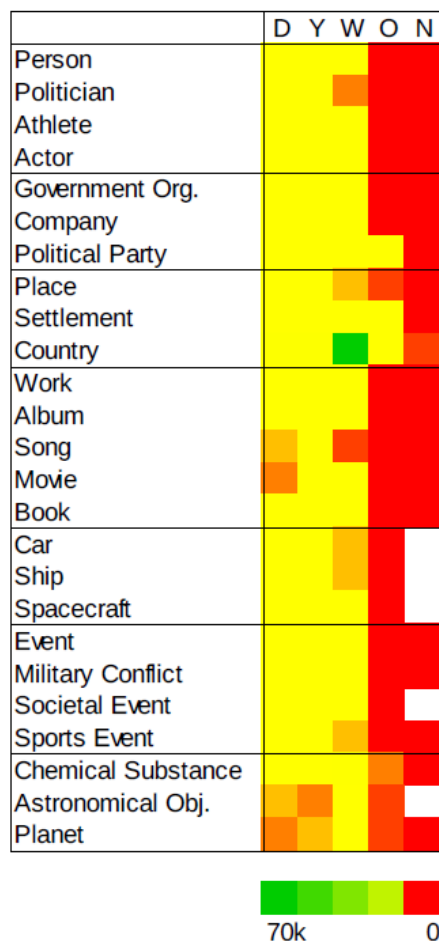
Comparison of Knowledge Graphs

- What do they actually contain?
- Experiment: pick 25 classes of interest
 - And find them in respective ontologies
- Count instances (coverage)
- Determine in and out degree (level of detail)

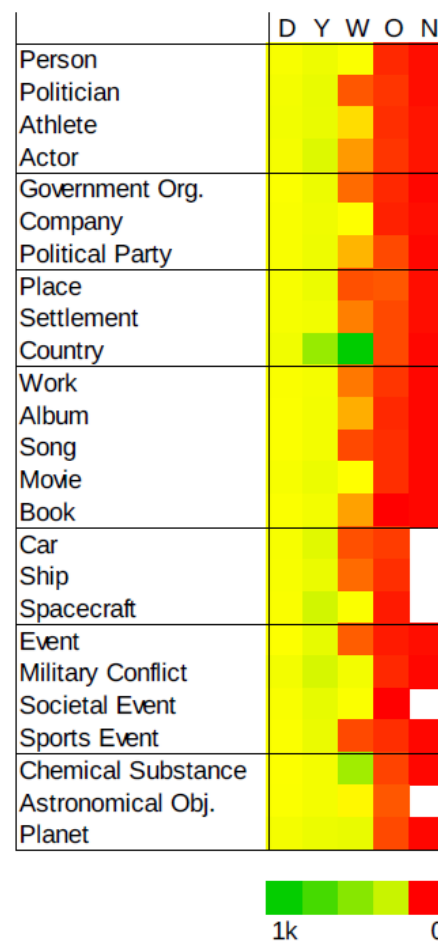
Comparison of Knowledge Graphs



(a) Number of instances



(b) Average indegree



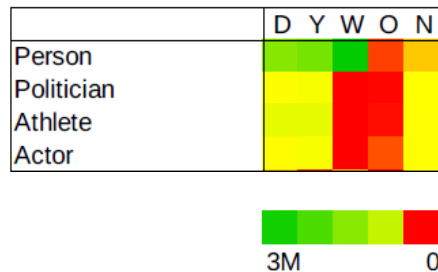
(c) Average outdegree

Comparison of Knowledge Graphs

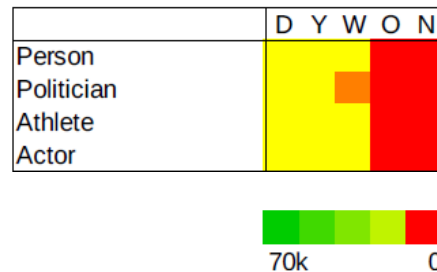
- Summary findings:
 - Persons: more in Wikidata
(twice as many persons as DBpedia and YAGO)
 - Countries: more details in Wikidata
 - Places: most in DBpedia
 - Organizations: most in YAGO
 - Events: most in YAGO
 - Artistic works:
 - Wikidata contains more movies and albums
 - YAGO contains more songs

Caveats

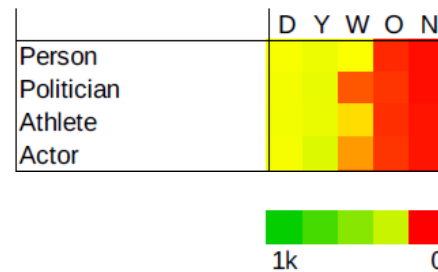
- Reading the diagrams right...



(a) Number of instances



(b) Average indegree

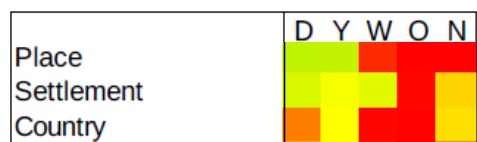


(c) Average outdegree

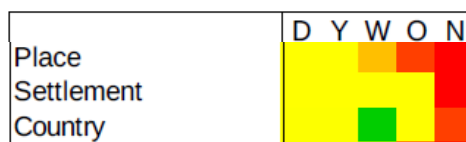
- So, Wikidata contains more persons
 - but less instances of all the interesting subclasses?
- There are classes like *Actor* in Wikidata
 - but they are hardly used
 - rather: modeled using *profession* relation

Caveats

- Reading the diagrams right... (ctd.)



(a) Number of instances



(b) Average indegree

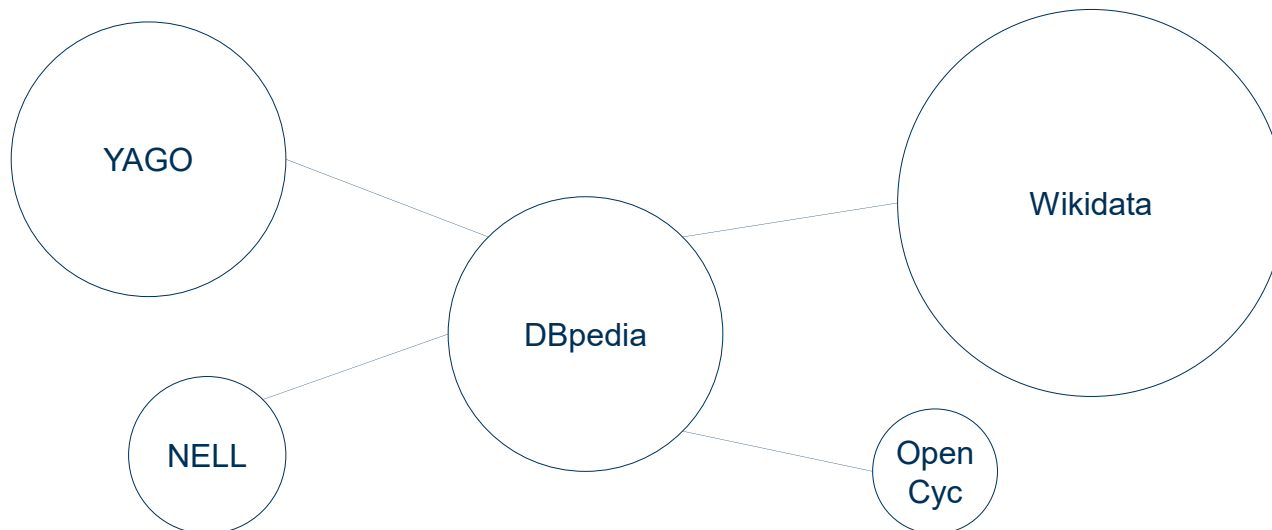
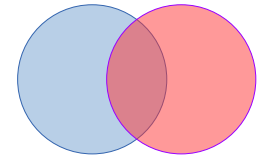


(c) Average outdegree

- So, Wikidata contains more data on countries, but less countries?
- First: Wikidata only counts current, actual countries
 - DBpedia and YAGO also count historical countries
- “KG1 contains less of X than KG2” can mean
 - it actually contains less instances of X
 - it contains equally many or more instances, but they are not typed with X (see later)
- Second: we count single facts about countries
 - Wikidata records some time indexed information, e.g., population
 - Each point in time contributes a fact

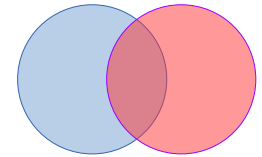
Overlap of Knowledge Graphs

- How largely do knowledge graphs overlap?
- They are interlinked, so we can simply count links
 - For NELL, we use links to Wikipedia as a proxy



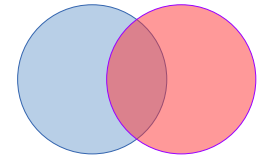
Overlap of Knowledge Graphs

- How largely do knowledge graphs overlap?
- They are interlinked, so we can simply count links
 - For NELL, we use links to Wikipedia as a proxy



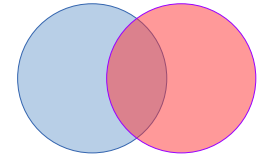
Overlap of Knowledge Graphs

- Links between Knowledge Graphs are incomplete
 - The Open World Assumption also holds for interlinks
- But we can estimate their number
- Approach:
 - Find link set automatically with different heuristics
 - Determine precision and recall on existing interlinks
 - Estimate actual number of links



Overlap of Knowledge Graphs

- Idea:
 - Given that the link set F is found
 - And the (unknown) actual link set would be C
- Precision P : Fraction of F which is actually correct
 - i.e., measures how much $|F|$ is *over*-estimating $|C|$
- Recall R : Fraction of C which is contained in F
 - i.e., measures how much $|F|$ is *under*-estimating $|C|$
- From that, we estimate $|C| = |F| * P * \frac{1}{R}$

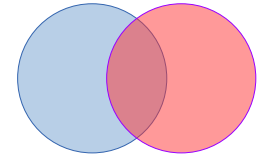


Overlap of Knowledge Graphs

- Mathematical derivation:

- Definition of recall: $R = \frac{|F_{correct}|}{|C|}$

unknown



- Definition of precision: $P = \frac{|F_{correct}|}{|F|}$

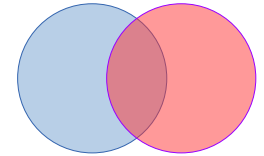
- Resolve both to $|F_{correct}|$, substitute, and resolve to $|C|$

$$|C| = |F| * P * \frac{1}{R}$$

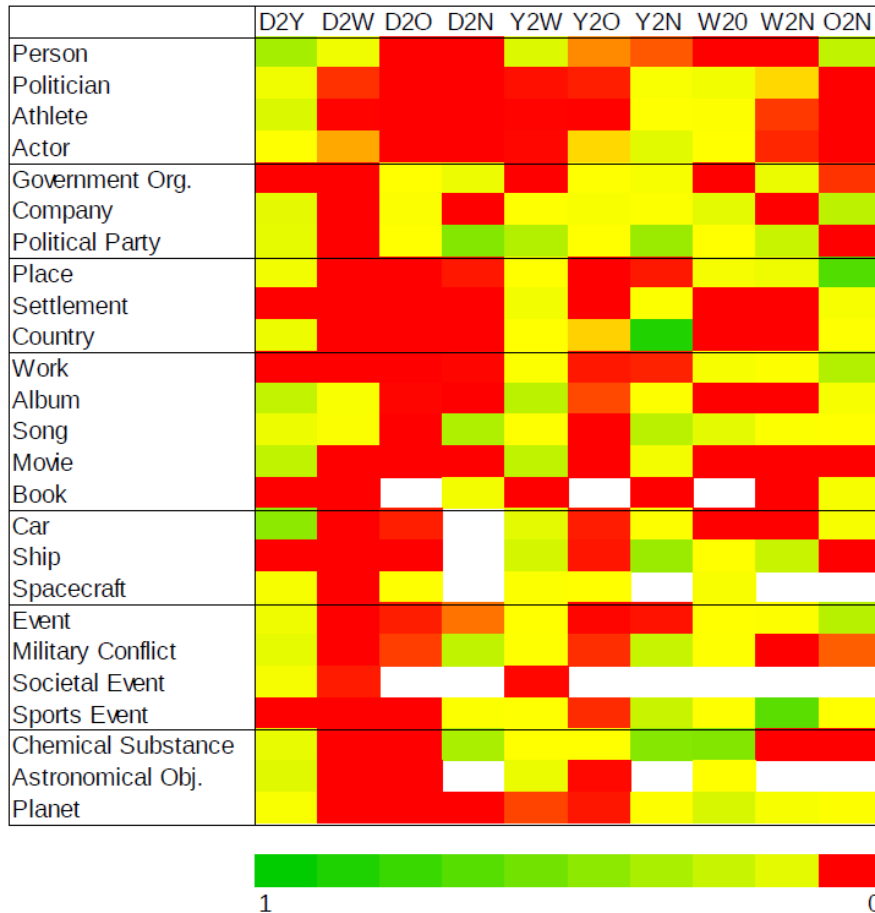
- The existing links are used to compute precision and recall

Overlap of Knowledge Graphs

- Experiment:
 - We use the same 25 classes as before
 - Measure 1: overlap relative to smaller KG (i.e., potential gain)
 - Measure 2: overlap relative to explicit links (i.e., importance of improving links)
- Link generation with 16 different metrics and thresholds
 - Intra-class correlation coefficient for $|C|$: 0.969
 - Intra-class correlation coefficient for $|F|$: 0.646
- Bottom line:
 - Despite variety in link sets generated, the overlap is estimated reliably
 - The link generation mechanisms do not need to be overly accurate



Overlap of Knowledge Graphs



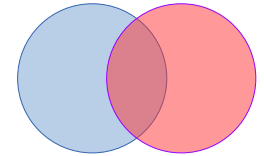
(a) Overlap as potential gain



(b) Overlap relative to existing links

Overlap of Knowledge Graphs

- Summary findings:
 - DBpedia and YAGO cover roughly the same instances (not much surprising)
 - NELL is the most complementary to the others
 - Existing interlinks are insufficient for out-of-the-box parallel usage



Knowledge Graph Construction

- Construction of KGs can be done in many ways
 - Manually
 - Automated
 - **From unstructured text**
 - Open Information Extraction
 - Closed Information Extraction
 - **From structured sources**
 - With a Script (e.g. Python)
 - **With a declarative mapping**

Knowledge Graph Construction

- From unstructured text

The 1957 European Baseball Championship was held in Mannheim and was organized by WBSC Europe.

– Usually involves:

- Named Entity Recognition

– and other NLP techniques like part of speech tagging, dependency parsing etc

```
import spacy
from spacy import displacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("The 1957 European Baseball Championship was held in Mannheim and was organized by WBSC Europe.")
displacy.render(doc, style="ent", jupyter=True)
```

... The 1957 DATE European Baseball Championship EVENT was held in Mannheim GPE and was organized by WBSC Europe ORG .

Knowledge Graph Construction

- From unstructured text

The 1957 European Baseball Championship was held in Mannheim and was organized by WBSC Europe.

– Usually involves:

- Co-Reference Resolution

The 1957 European Baseball Championship was held in Mannheim.
It was organized by WBSC Europe.

Knowledge Graph Construction

- From unstructured text

The 1957 European Baseball Championship was held in Mannheim and was organized by WBSC Europe.

— Usually involves:

- Entity Linking / Entity Disambiguation
 - Which Mannheim

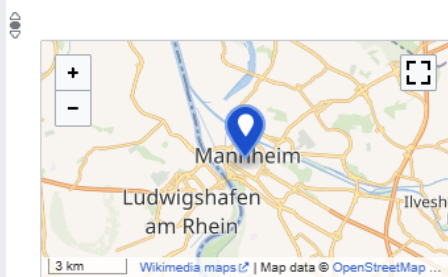
Mannheim (Q2119)

[Item](#) [Discussion](#)

city in Baden-Württemberg, Germany

Mannem | Monnem | Universitätsstadt Mannheim | Mannheim, Germany | Mannheim (Germany) | Mannheim Germany

coordinate location



[49°29'16"N, 8°27'58"E](#)

1 reference

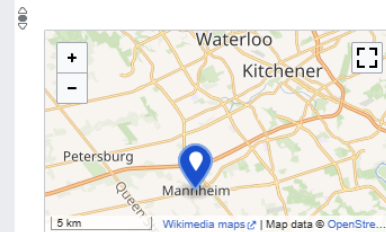
Mannheim (Q111205834)

[Item](#) [Discussion](#)

locality in Wllmot,Ontario

Mannheim, Ontario

coordinate location



[43°23'53"N, 80°32'43"W](#)

1 reference

Knowledge Graph Construction

- From unstructured text

The 1957 European Baseball Championship was held in Mannheim and was organized by WBSC Europe.

- Usually involves:
 - Relation Extraction

The 1957 European Baseball Championship was held in Mannheim.

?

Knowledge Graph Construction

- From unstructured text

The 1957 European Baseball Championship was held in Mannheim and was organized by WBSC Europe.

- Two types of Information Extraction

- **Open Information Extraction**

- Ignores Entity Linking
- Uses text as representation of entities and relations

[1957_European_Baseball_Championship, held_in, Mannheim]
[1957_European_Baseball_Championship, organized_by, WBSC_Europe]

- **Closed Information Extraction**

- Links to given entities and relations

1957_European_Baseball_Championship

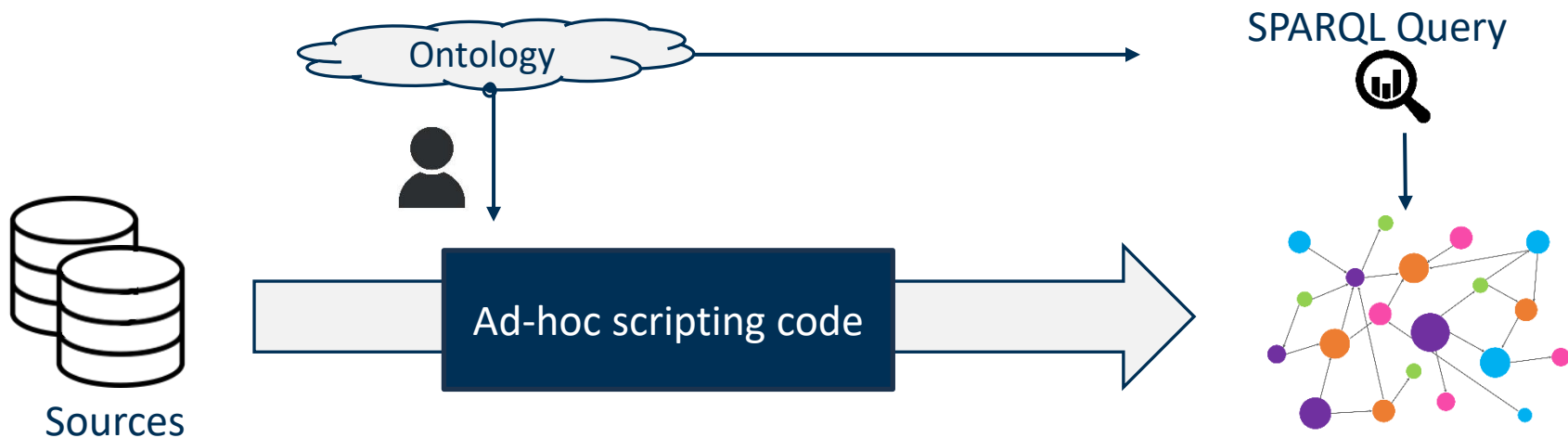
location

Mannheim

[Q4509146, P276, Q2119]
[Q4509146, P664, Q680032]

Knowledge Graph Construction

- From structured sources
 - Like Relational Databases, CSV files, XML, JSON, etc.
 - Can usually be (precisely) mapped
 - With a Script (e.g. Python)

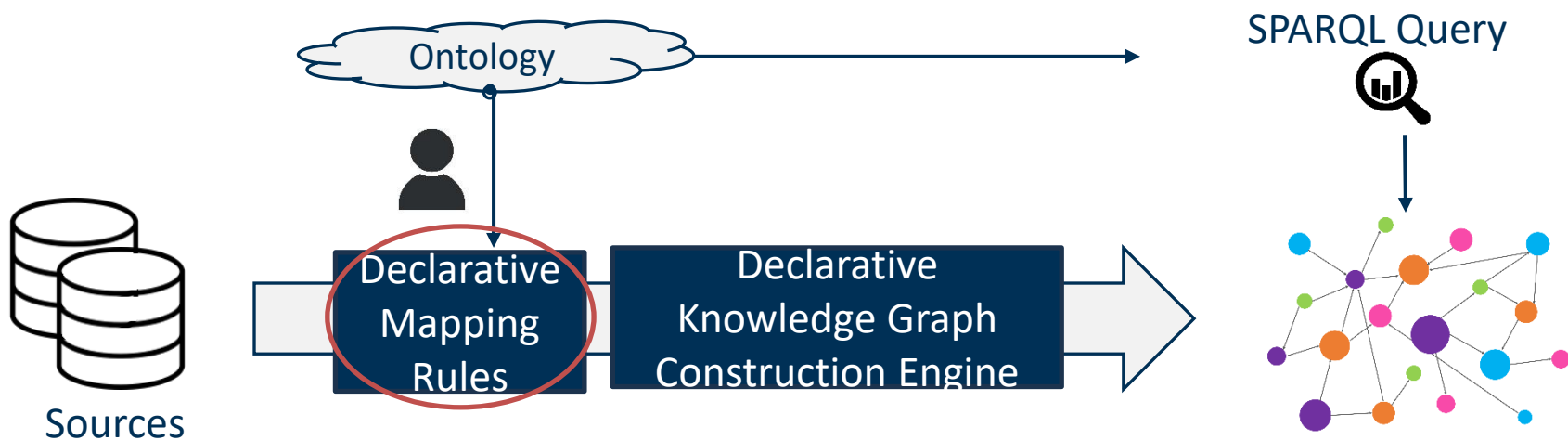


Knowledge Graph Construction

- From structured sources
 - Like Relational Databases, CSV files, XML, JSON, etc.
 - Can usually be (precisely) mapped
 - With a Script (e.g. Python)
 - Allows for arbitrary conversions (e.g. `name = [concat(A,B) if C=5]`)
 - Might or might not be efficient and scalable
 - But not maintainable and robust

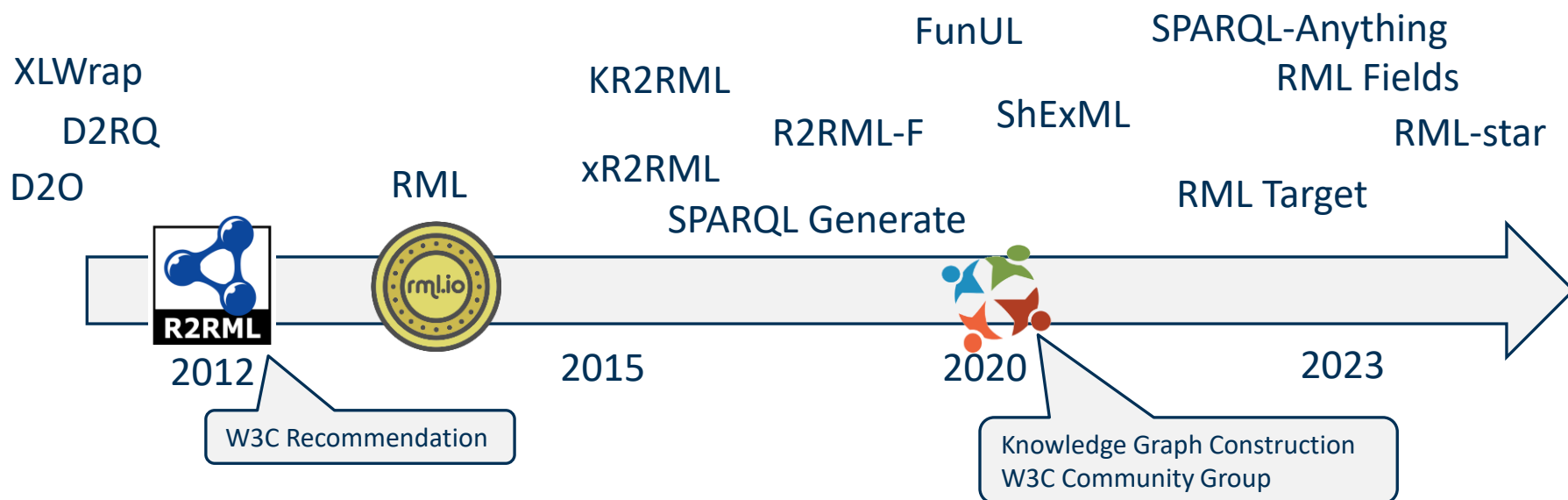
Knowledge Graph Construction

- From structured sources
 - Like Relational Databases, CSV files, XML, JSON, etc.
 - Can usually be (precisely) mapped
 - With declarative mapping (materialization)



Knowledge Graph Construction

- Declarative Mapping languages
 - Allows to specify transformation rules from data sources into RDF
 - Data source and access description
(e.g. password for database, location of CSV file etc)
 - Triple generation
 - Many different languages proposed with different features



RML Example

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

people.csv

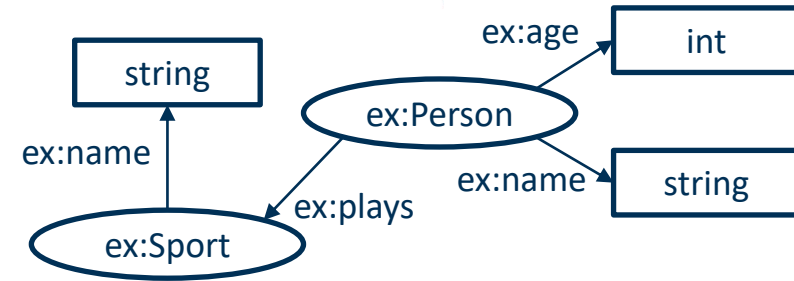
ID	Sport
1	Ice Skating
2	Rugby

sports.csv

```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
```

```
:peopleMap a rr:TriplesMap.
```

```
:peopleMap rml:logicalSource [
  rml:source "people.csv" ;
  rml:referenceFormulation ql:CSV
].
```



RML Example

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

people.csv

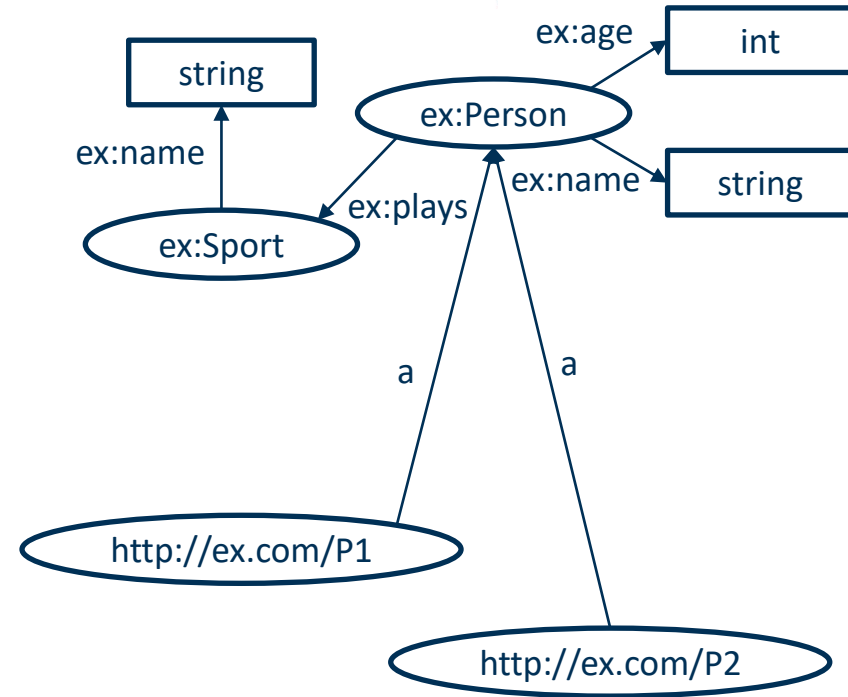
ID	Sport
1	Ice Skating
2	Rugby

sports.csv

```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
```

```
:peopleMap a rr:TriplesMap.
```

```
:peopleMap rml:logicalSource [
  rml:source "people.csv" ;
  rml:referenceFormulation ql:CSV
].
:peopleMap rml:subjectMap [
  rml:template "http://ex.com/P{ID}";
  rml:class ex:Person
].
```



RML Example

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

people.csv

ID	Sport
1	Ice Skating
2	Rugby

sports.csv

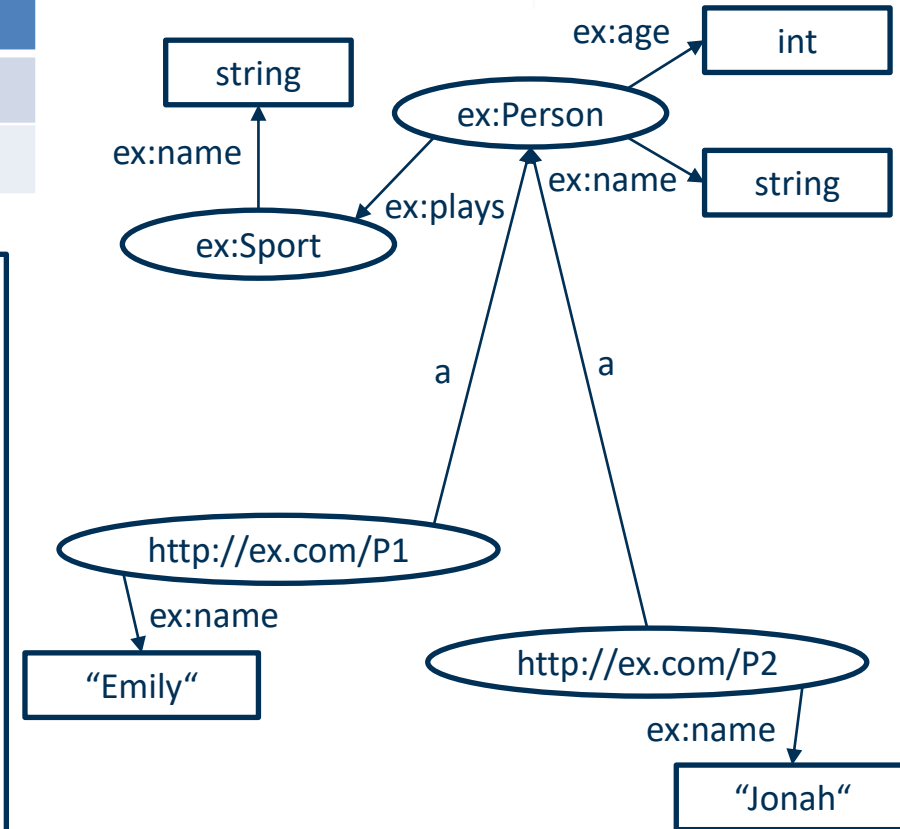
```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
```

```
:peopleMap a rr:TriplesMap.
```

```
:peopleMap rml:logicalSource [
  rml:source "people.csv" ;
  rml:referenceFormulation ql:CSV
].
```

```
:peopleMap rml:subjectMap [
  rml:template "http://ex.com/P{ID}";
  rml:class ex:Person
].
```

```
:peopleMap rml:predicateObjectMap [
  rml:predicate ex:name;
  rml:objectMap [ rml:reference "Name" ]
].
```



RML Example

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

people.csv

ID	Sport
1	Ice Skating
2	Rugby

sports.csv

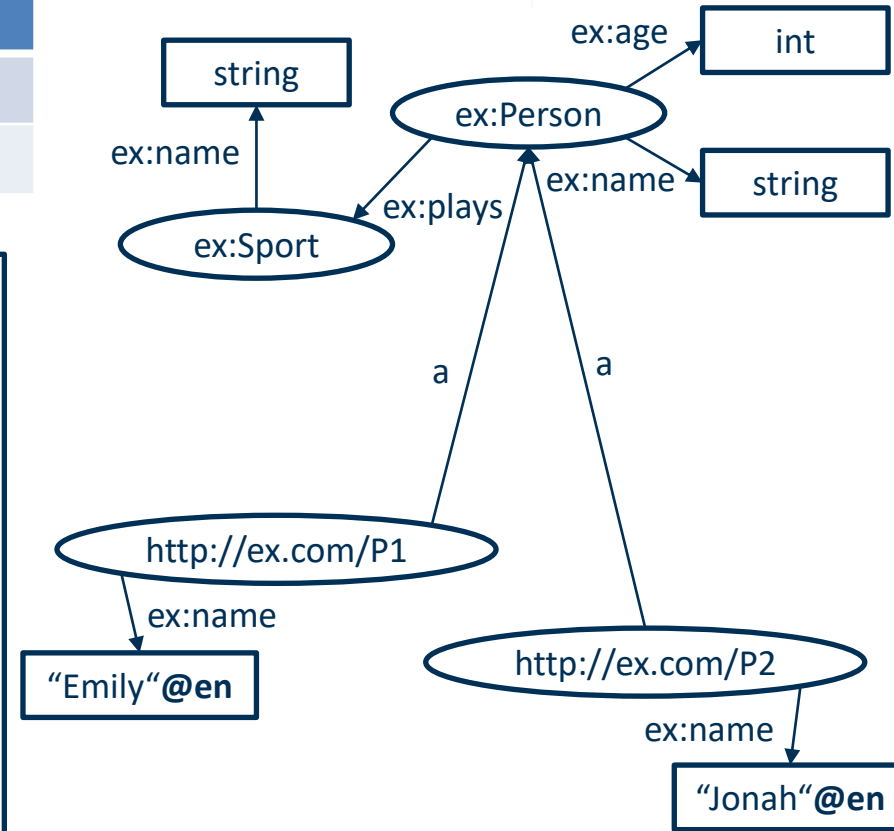
```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
```

```
:peopleMap a rr:TriplesMap.
```

```
:peopleMap rml:logicalSource [
  rml:source "people.csv" ;
  rml:referenceFormulation ql:CSV
].
```

```
:peopleMap rml:subjectMap [
  rml:template "http://ex.com/P{ID}";
  rml:class ex:Person
].
```

```
:peopleMap rml:predicateObjectMap [
  rml:predicate ex:name;
  rml:objectMap [ rml:reference "Name"; rml:language "en" ]
].
```



RML Example

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

people.csv

ID	Sport
1	Ice Skating
2	Rugby

sports.csv

```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
```

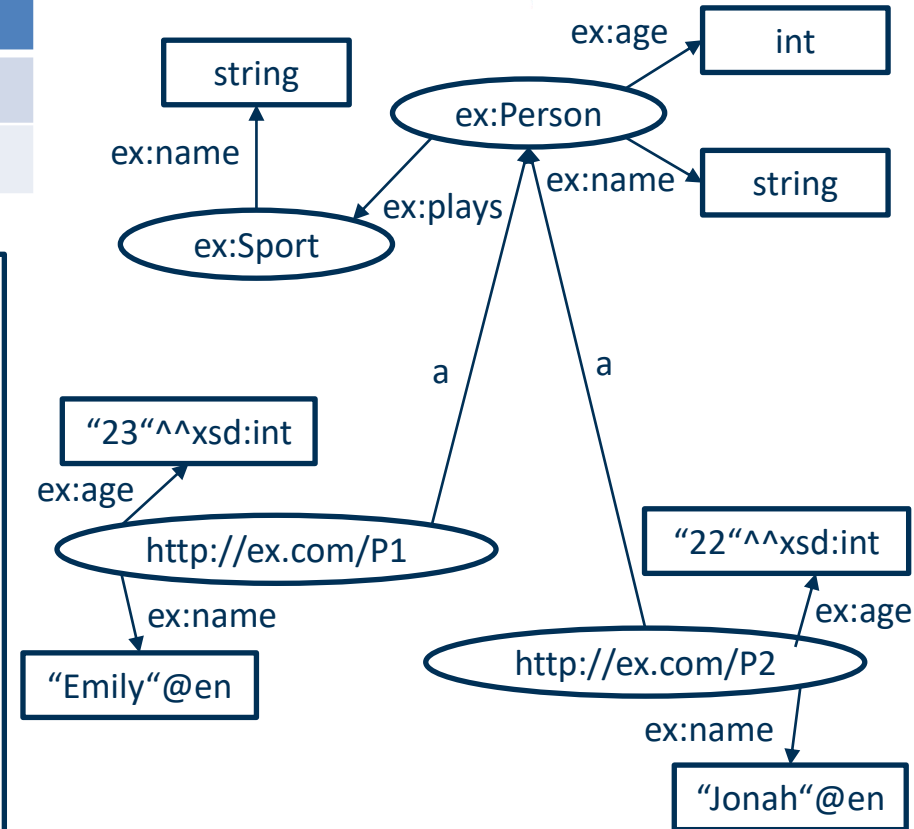
```
:peopleMap a rr:TriplesMap.
```

```
:peopleMap rml:logicalSource [
  rml:source "people.csv" ;
  rml:referenceFormulation ql:CSV
].
```

```
:peopleMap rml:subjectMap [
  rml:template "http://ex.com/P{ID}";
  rml:class ex:Person
].
```

```
:peopleMap rml:predicateObjectMap [
  rml:predicate ex:name;
  rml:objectMap [ rml:reference "Name"; rml:language "en" ]
].
```

```
:peopleMap rml:predicateObjectMap [
  rml:predicate ex:age;
  rml:objectMap [ rml:reference "Age"; rml:datatype xsd:int ]
].
```



RML Example

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

people.csv

ID	Sport
1	Ice Skating
2	Rugby

sports.csv

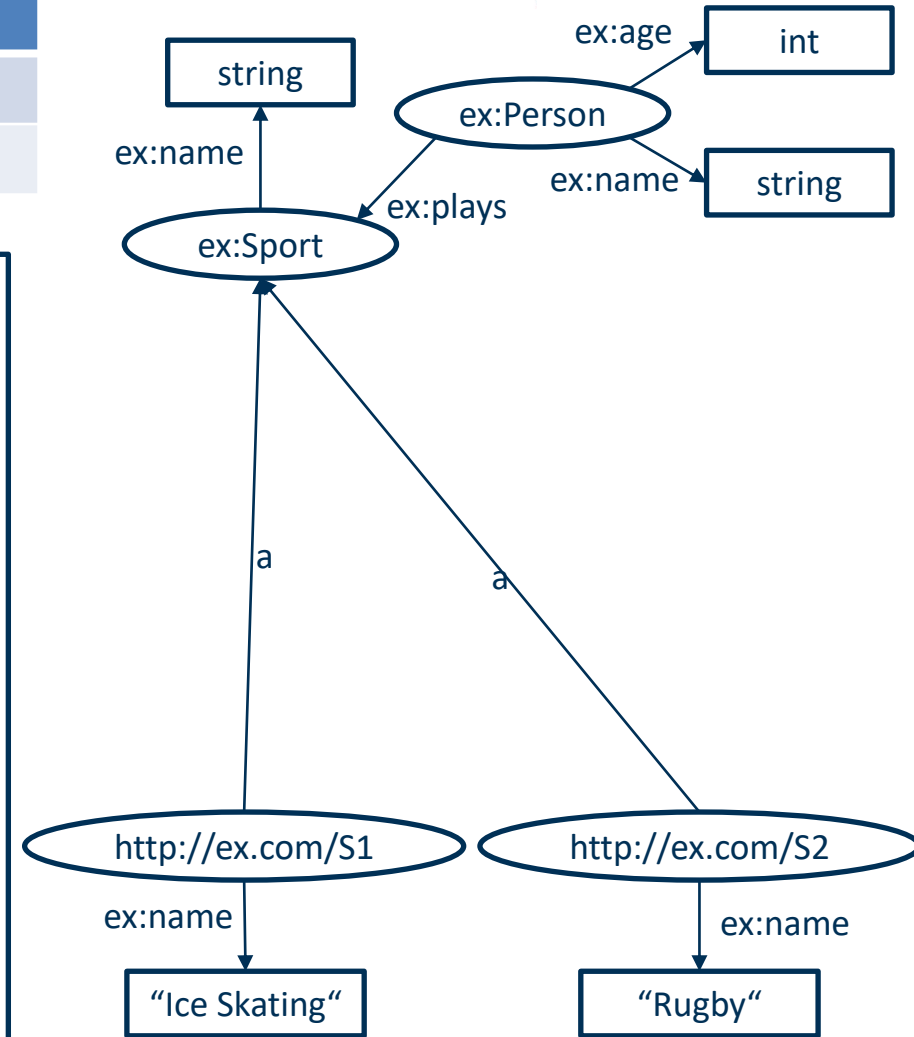
```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
```

```
:sportMap a rr:TriplesMap.
```

```
:sportMap rml:logicalSource [
  rml:source "sports.csv" ;
  rml:referenceFormulation ql:CSV
].
```

```
:sportMap rml:subjectMap [
  rml:template "http://ex.com/S{ID}";
  rml:class ex:Sport
].
```

```
:peopleMap rml:predicateObjectMap [
  rml:predicate ex:name;
  rml:objectMap [ rml:reference "Sport" ]
].
```



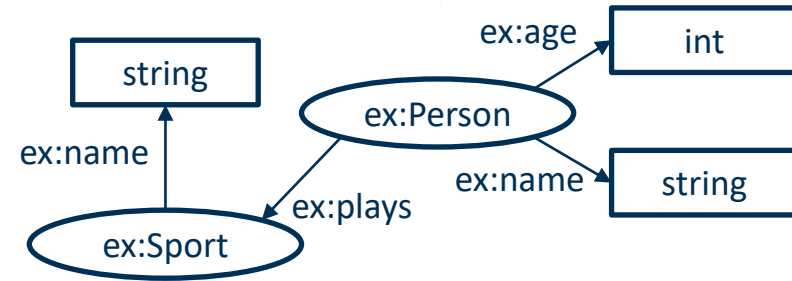
RML Example

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

people.csv

ID	Sport
1	Ice Skating
2	Rugby

sports.csv



```

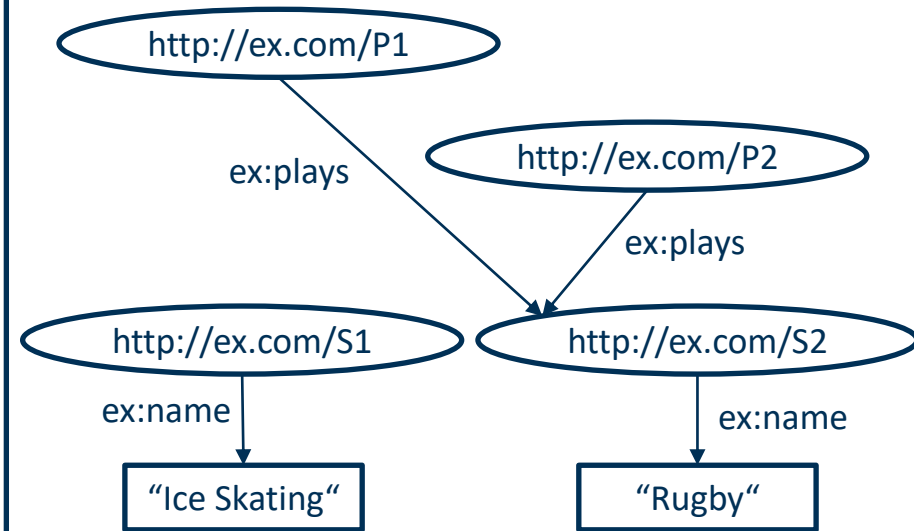
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .

```

```

:peopleMap rml:predicateObjectMap [
  rml:predicate ex:plays;
  rml:objectMap [
    rml:parentTriplesMap :sportsMap;
    rml:joinCondition [
      rml:child "SportID";
      rml:parent "ID" ]
    ];
]

```



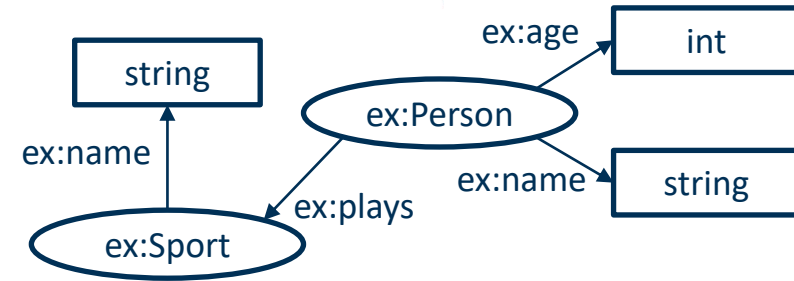
RML Example

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

people.csv

ID	Sport
1	Ice Skating
2	Rugby

sports.csv



- YARRRML is a more user friendly serialization based on YAML
- Tool “Yatter” converts YARRRML to RML

```

mappings:
  PERSON:
    sources:
      - ['data/people.csv~csv']
    s: http://ex.com/P$(ID)
    po:
      - [a, ex:Person]
      - [ex:name, $(Name)]
      - [ex:age, $(age)]
  
```

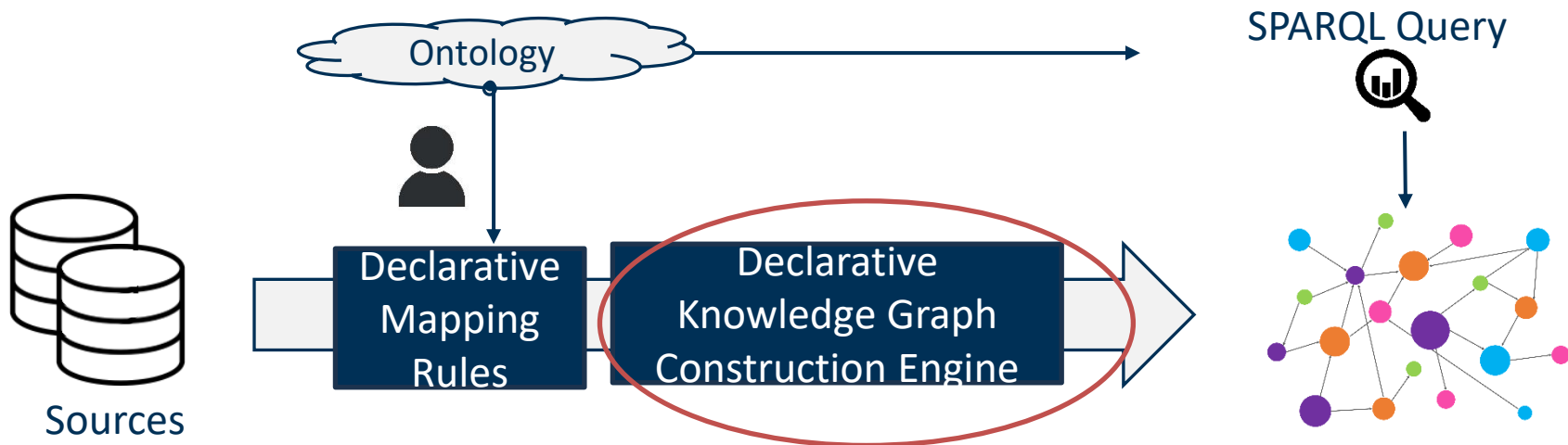
YAML

Knowledge Graph Construction

- Which engines to use?

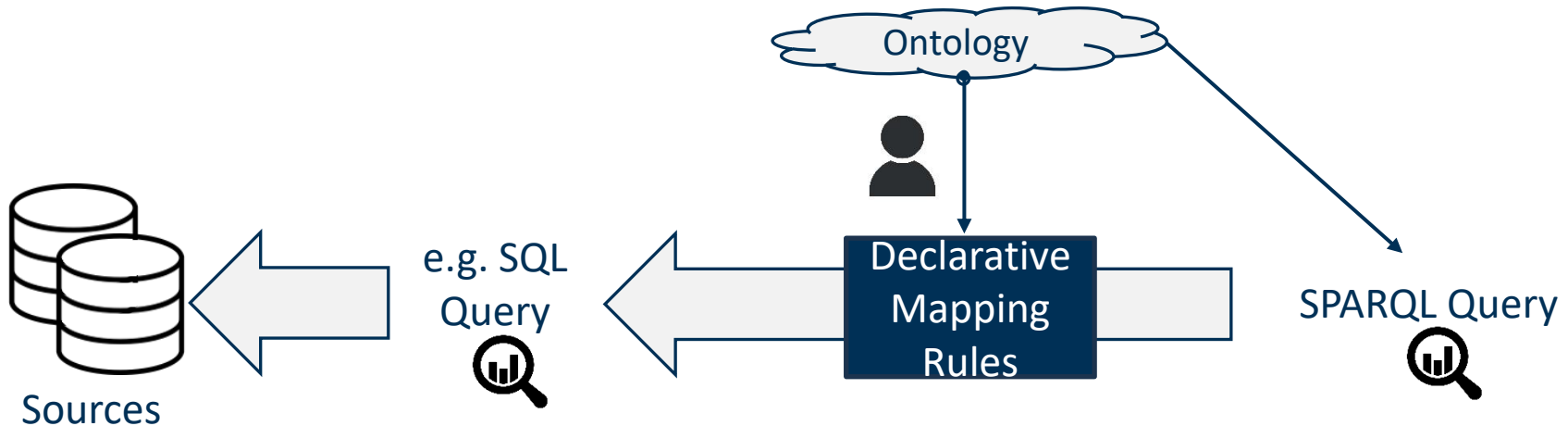
- SDM-RDFizer
- RMLMapper
- RMLStreamer
- FlexRML
- Morph-KGC

They expect RML as input.
YARRRML need to be converted into RML



Knowledge Graph Construction

- With declarative mapping rules you can also virtualize the graph
 - The graph is not materialized
 - SPARQL query (using the vocabulary of the ontology) is translated into queries that the source uses e.g. SQL (Xpath for XML etc)
 - Results of query are returned using identifiers and vocabulary of KG
 - Also called Ontology based data access (OBDA)



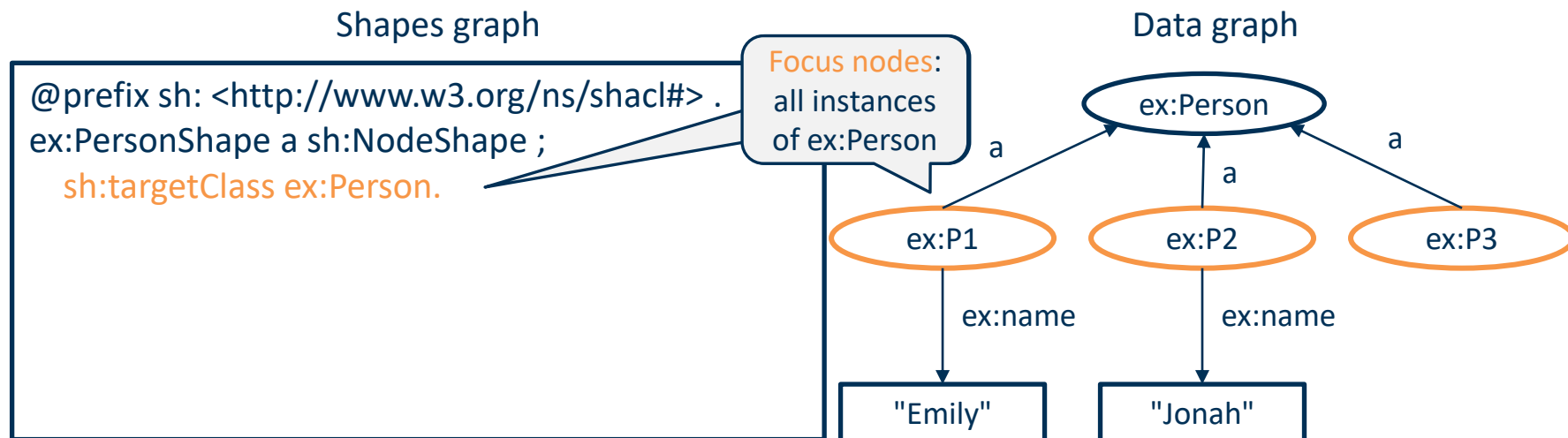
Validation / Quality Assurance of Knowledge Graphs

- Shapes Constraint Language (SHACL)
 - Allows RDF data to be validated/checked against **shapes (conditions)**
 - W3C Recommendation since 2017
 - A shape describes how the data should look like:
classes, properties, datatypes, cardinalities, value ranges, ...
 - Shapes are expressed in RDF themselves



SHACL Example

- Shapes Constraint Language (SHACL)
 - Requires a **target** (which RDF nodes should be checked)
 - sh:targetClass – all instances of a class
 - sh:targetNode – one specific node
 - sh:targetSubjectsOf / sh:targetObjectsOf – all subjects / objects of a property



SHACL Example

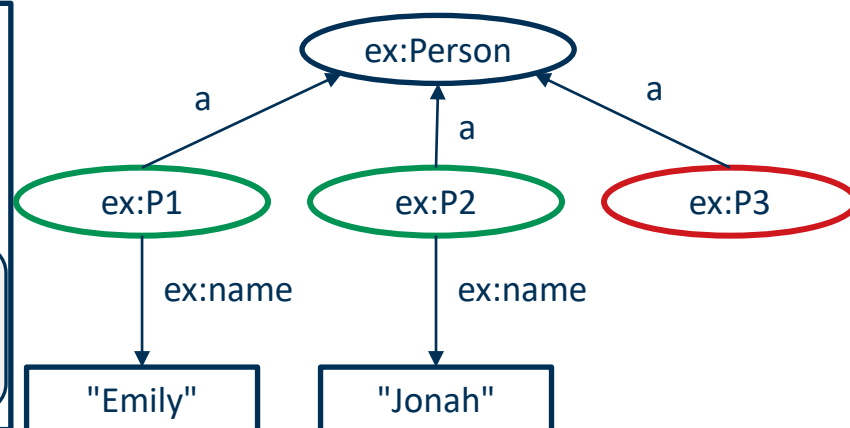
- Shapes Constraint Language (SHACL)
 - Requires a **target** (which RDF nodes should be checked)
 - sh:targetClass – all instances of a class
 - sh:targetNode – one specific node
 - sh:targetSubjectsOf / sh:targetObjectsOf – all subjects / objects of a property
 - And a collection of constraints

Shapes graph

```
@prefix sh: <http://www.w3.org/ns/shacl#> .  
ex:PersonShape a sh:NodeShape ;  
  sh:targetClass ex:Person;  
  sh:property [  
    sh:path ex:name ;  
    sh:minCount 1  
  ] .
```

Each of them needs at least one ex:name
(Cardinality Constraint)

Data graph



SHACL Example

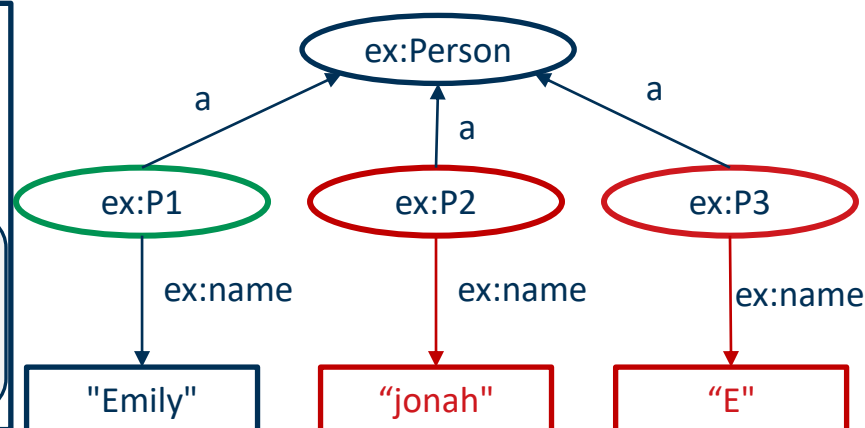
- Shapes Constraint Language (SHACL)
 - Requires a **target** (which RDF nodes should be checked)
 - sh:targetClass – all instances of a class
 - sh:targetNode – one specific node
 - sh:targetSubjectsOf / sh:targetObjectsOf – all subjects / objects of a property
 - And a collection of constraints

Shapes graph

```
@prefix sh: <http://www.w3.org/ns/shacl#> .  
ex:PersonShape a sh:NodeShape ;  
  sh:targetClass ex:Person;  
  sh:property [  
    sh:path ex:name ;  
    sh:minLength 2 ;  
    sh:pattern "^[A-Z]"  
  ] .
```

Name needs at least two characters and start with upper-case letter (String based Constraint)

Data graph



SHACL Example

- Shapes Constraint Language (SHACL)
 - Requires a **target** (which RDF nodes should be checked)
 - sh:targetClass – all instances of a class
 - sh:targetNode – one specific node
 - sh:targetSubjectsOf / sh:targetObjectsOf – all subjects / objects of a property
 - And a collection of constraints

Shapes graph

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
```

```
ex:PersonShape a sh:NodeShape ;
```

```
sh:targetClass ex:Person;
```

```
sh:property [
```

```
sh:path ex:plays ;
```

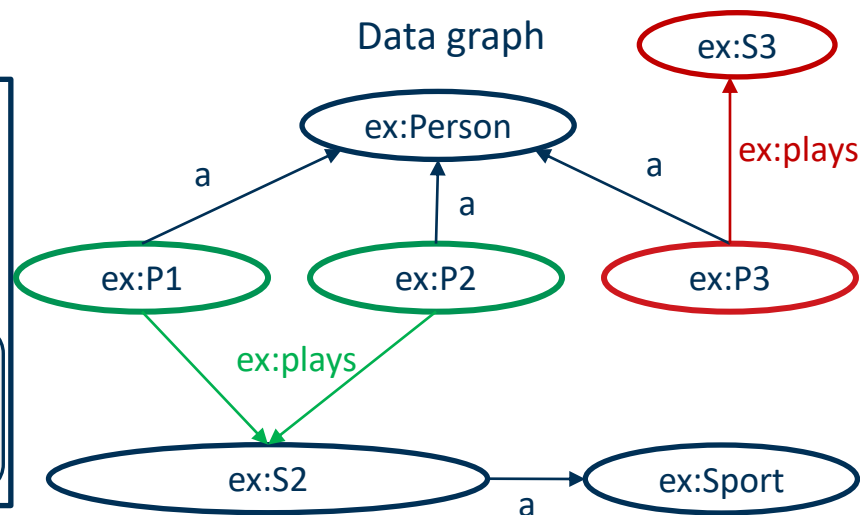
```
sh:class ex:Sport ;
```

```
sh:maxCount 1
```

```
].
```

sh:class: the value must
be an instance of ex:Sport
(Value Type Constraint)

Data graph



SHACL Core Constraint Components

Category	Constraint components
Cardinality	sh:minCount, sh:maxCount
Value type	sh:class, sh:datatype, sh:nodeKind (sh:IRI, sh:Literal, sh:BlankNode, ...)
Value range	sh:minInclusive, sh:maxInclusive, sh:minExclusive, sh:maxExclusive
String-based	sh:minLength, sh:maxLength, sh:pattern (regex), sh:languageIn, sh:uniqueLang
Property pairs	sh:equals, sh:disjoint, sh:lessThan, sh:lessThanOrEquals
Logical	sh:not, sh:and, sh:or, sh:xone
Shape-based	sh:node, sh:property, sh:qualifiedValueShape
Other	sh:closed, sh:ignoredProperties, sh:hasValue, sh:in
Property paths	(ex:plays ex:name), [sh:inversePath ex:plays], sh:alternativePath, sh:zeroOrMorePath, ...

- For constraints that are not (easily) expressible with SHACL Core
 - Use a SPARQL query (\$this is bound to the current focus node)
 - Each result row is one violation
- Example: two persons should not have the same name

```
ex:PersonShape sh:sparql [  
  a sh:SPARQLConstraint ;  
  sh:message "Another person has the same name: {?value}" ;  
  sh:select ""  
    SELECT $this ?value WHERE {  
      $this ex:name ?value .  
      ?other ex:name ?value .  
      FILTER (?other != $this)  
    }"" ;  
].
```

- Validators
 - pySHACL (Python), Apache Jena SHACL
- Use in knowledge graph construction
 - Validate the output of the mapping engine before publishing the KG
 - Run automatically whenever sources or mappings change
 - Severity levels: sh:Violation, sh:Warning, sh:Info
- Where do the shapes come from?
 - Written manually / derived from the ontology
 - Extracted from existing data or generated with LLMs

Questions?

