

The CaLiGraph Ontology as a Challenge for OWL Reasoners

Nicolas Heist¹, Heiko Paulheim¹

¹*Data and Web Science Group, University of Mannheim, Germany*

Abstract

CaLiGraph is a large-scale cross-domain knowledge graph generated from Wikipedia by exploiting the category system, list pages, and other list structures in Wikipedia, containing more than 15 million typed entities and around 10 million relation assertions. Other than knowledge graphs such as DBpedia and YAGO, whose ontologies are comparably simplistic, CaLiGraph also has a rich ontology, comprising more than 200,000 class restrictions. Those two properties – a large A-box and a rich ontology – make it an interesting challenge for benchmarking reasoners. In this paper, we show that a reasoning task which is particularly relevant for CaLiGraph, i.e., the materialization of `owl:hasValue` constraints into assertions between individuals and between individuals and literals, is insufficiently supported by available reasoning systems. We provide differently sized benchmark subsets of CaLiGraph, which can be used for performance analysis of reasoning systems.

Keywords

Knowledge Graph, Reasoning, Scalability, `owl:hasValue` Restrictions, Materialization

1. Introduction

In the recent decade, open cross-domain knowledge graphs have been recognized as an interesting ingredient to build intelligent systems. This gave rise to the development of various open knowledge graphs, such as DBpedia [1], YAGO [2], and Wikidata [3]. While those knowledge graphs come with large-scale A-boxes comprising millions of entities, their T-boxes are usually not very expressive, defining mainly a class hierarchy and relations with domains and ranges, but not using complex class constructors [4].

CaLiGraph is a comparatively new knowledge graph, which is constructed from categories, list pages, and other lists in Wikipedia. It uses DBpedia as a training set to derive interpretations of lists and categories [5]. This allows to extract more entities from list pages [6] and other listings [7], making the A-box comprise more than twice as many entities as DBpedia. At the same time, the definitions of the derived classes are also maintained in the T-box, making it more expressive than the ontologies of other knowledge graphs.

With those characteristics, CaLiGraph creates some interesting tasks to reasoners, since it requires the materialization of individual and literal assertions, and, at the same time, poses


SemREC'21: Semantic Reasoning Evaluation Challenge, ISWC'21, Oct 24 – 28, Albany, NY

✉ nico@informatik.uni-mannheim.de (N. Heist); heiko@informatik.uni-mannheim.de (H. Paulheim)

🌐 <http://www.uni-mannheim.de/dws/people/researchers/phd-students/nicolas-heist/> (N. Heist); <http://www.heikopaulheim.com/> (H. Paulheim)

🆔 0000-0002-4354-9138 (N. Heist); 0000-0002-4354-9138 (H. Paulheim)

© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

Category:Swedish rock musicians

 Help

From Wikipedia, the free encyclopedia



Subcategories

This category has the following 2 subcategories, out of 2 total.

G

► Swedish rock guitarists (33 P)

S

► Swedish rock singers (40 P)

Pages in category "Swedish rock musicians"

The following 14 pages are in this category, out of 14 total. This list may not reflect recent changes ([learn more](#)).

B

• Nicke Borg

D

• Dregen

E

• John Eriksson (musician)

F

• Johan Frandsen

G

• Kristian Gidlund

H

• Niklas Hellberg

L

• Laleh (singer)

• Pär Lindh

• Boris Lindqvist

M

• Peter Morén

N

• Totta Näslund

P

• Pekkanini

S

• Pär Sundström

W

• Anders Wendin

Categories: [Rock musicians by nationality](#) | [Swedish musicians by genre](#)
Hidden categories: [Commons category link is on Wikidata](#)

Figure 1: Example of a Category in Wikipedia

high scalability requirements when asking a reasoner to materialize 10 million A-box assertions. Since those requirements are hardly met by currently available reasoners, the materialization of the CaLiGraph A-box is carried out by custom code, not by running a reasoner, albeit being a standard materialization task which an OWL2 EL reasoner should be capable of performing.

In this paper, we report on experiments with three reasoning systems on the materialization of CaLiGraph. Furthermore, we introduce differently sized subsets of CaLiGraph, which allow for performing scalability experiments.

The rest of this paper is structured as follows. Section 2 gives a short introduction into CaLiGraph. In section 3, we describe a set of experiments we conducted to investigate the fitness of existing reasoners for processing CaLiGraph. We close with a short summary and an outlook on future work.

2. The CaLiGraph Ontology

As described above, CaLiGraph processes categories and list pages in Wikipedia, and derives classes and instances from those. For example, when processing the Wikipedia category Swedish Rock Musicians,¹ shown in Fig. 1, it will create a class `clgo:Swedish_rock_musician`,²

¹https://en.wikipedia.org/wiki/Category:Swedish_rock_musicians

²The following name space conventions are used throughout this paper:

`clgo`=<http://caligraph.org/ontology/>

`clgr`=<http://caligraph.org/resource/>

and heuristically assign the following subclass axioms to it:

```
clgo:Swedish_rock_musician
  a owl:Class ;
  rdfs:subClassOf
    clgo:Rock_musician ,
    clgo:Swedish_musician ,
    [
      a owl:Restriction ;
      owl:onProperty clgo:birthPlace ;
      owl:hasValue clgr:Sweden
    ] ,
    [
      a owl:Restriction ;
      owl:onProperty clgo:genre ;
      owl:hasValue clgr:Rock_music
    ] ,
    [
      a owl:Restriction ;
      owl:onProperty clgo:occupation ;
      owl:hasValue clgr:Musician
    ] .

clgo:Swedish_rock_guitarist rdfs:subClassOf clgo:Swedish_rock_musician .
clgo:Swedish_rock_singer rdfs:subClassOf clgo:Swedish_rock_musician .
```

The subclass axioms are derived from the subcategory relations in Wikipedia, the restrictions are created using a mix of statistic and textual heuristics [5, 6].

As shown in this example, restrictions using `owl:hasValue` are quite frequently used in CaLiGraph. Overall, the current release has more than 200,000 of such restrictions, most of them using a resource as their object, but some also using literals (e.g., restrictions on birth years).

When populating the A-box, CaLiGraph creates entities for Wikipedia pages, as well as entities found on list pages. The only axioms created for those entities are the class memberships for the classes derived from the corresponding categories and list pages; the remaining axioms are materialized according to the class definitions. In the example above, given that the entity Dennis Lyxzén is found in the category Swedish rock singers (which is a subcategory of the above mentioned Swedish rock musicians, CaLiGraph will produce the following axiom:

```
clgr:Dennis_Lyxzén a clgo:Swedish_rock_singer .
```

Given the T-box above, the derived axioms according to the definition above should include:

```
clgr:Dennis_Lyxzén a clgo:Swedish_rock_musician .
```

No. of classes	1,061,598
No. of relations	338
No. of subclass assertions	1,711,270
No. of restrictions	
...with individuals	214,248
...with literals	12,925
No. of classes with direct restrictions	223,552
No. of classes with direct or inherited restrictions	488,364
No. of instances	14,452,393
No. of direct type assertions	50,169,052
No. of transitive type assertions	138,713,499
No. of relation assertions	
...with individuals	9,637,354
...with literals	969,022

Table 1
Statistics of the CaLiGraph ontology

```

clgr:Dennis_Lyxzén a clgo:Rock_musician .
clgr:Dennis_Lyxzén a clgo:Swedish_musician .
clgr:Dennis_Lyxzén clgo:birthPlace clgr:Sweden .
clgr:Dennis_Lyxzén clgo:genre clgr:Rock_music .
clgr:Dennis_Lyxzén clgo:occupation clgr:Musician .

```

The first three axioms are obtained by materializing the subclass relations, the latter three by materializing the restrictions based on `owl:hasValue`. Note that the example only depicts a subset of the information on the instance `clgr:Dennis_Lyxzén` in CaLiGraph, which also contains information on his birthplace and birth year.³

While CaLiGraph uses a custom implementation to materialize those A-box axioms, this is a task which should, in principle, be possible to carry out for an OWL DL reasoner. Moreover, since restrictions with `owl:hasValue` are possible in OWL2 EL and OWL2 RL [8], we would expect that reasoners supporting those fragments should be capable of carrying out that materialization as well [9].

The characteristics of the most recent version 2.1.0 of CaLiGraph are shown in table 1. The table shows that a reasoner, provided with the 14 million individuals and their 50M direct type assertions, should in principle be able to derive 139M transitive type relations, as well as 10M relation and literal assertions.

This kind of reasoning – at least at that scale – is rather underrepresented in current reasoning benchmarks, and often not evaluated at all. For example, in the ORE2015 benchmark, the task of materialization is defined as *the computation of all instances for all named classes in the ontology* [10]. In other words: in that benchmark, the task is to infer all class assertions for an instance given all its literal and individual assertions, while for CaLiGraph, we require the reverse direction.

Among the 1,920 test cases in the ORE2015 benchmarks [10], there are only 311 which use `owl:hasValue` constraints with an individual (max. 6,455 in a single file), and 279 which use

³http://caligraph.org/resource/Dennis_Lyxzén

`owl:hasValue` with a literal (max. 1,104 in a single file). The more recent OWL2Bench dataset [11], proposed in 2020, contains exactly one `owl:hasValue` restriction, mainly for checking whether or not a reasoner supports that construct, but does not allow for evaluating that support at scale. In contrast, the CaLiGraph ontology makes much heavier use of those constructs, as shown above, and therefore imposes more challenging scalability requirements with respect to that reasoning task.

3. Experiments

To validate how far reasoning systems actually meet those challenges, we conducted a series of experiments, both with CaLiGraph as a whole and with differently-sized subsets of CaLiGraph.

3.1. Reasoning Systems

In our experiments, we chose three commonly known reasoners: ELK [12], HermiT [13], and Pellet [14]. All of them claim to be able to process OWL2 EL ontologies. We used the latest non-commercial versions, i.e.:

- ELK version 0.4.3
- Pellet version 2.3.6
- HermiT version 1.4.5.456

The reasoning systems were called through OWL API⁴ using the respective connectors to the reasoners. As the reasoners rely on different versions of OWL API, we provide an individual dependency configuration for each of them in our implementation. Note that since OWL API does not support restrictions with explicit URIs and labels, we transformed them to blank nodes (as shown in Fig. 2) before importing them with OWL API.

3.2. Sandbox Example

As a first sanity check, we extracted a simple sandbox example from CaLiGraph, which we coin `clg_10` in the following. The sandbox example contains two classes with one `owl:hasValue` restriction each, one using an individual, one using a literal. For each of the two classes, there is an individual, so that an OWL2EL reasoner should be capable of inferring one individual and one literal assertion. The sandbox example is shown in Fig. 2.

From our three reasoners, only Pellet is capable of inferring both assertions. HermiT can infer the resource-valued assertion, but not the literal-valued one, while ELK does not infer any of the two. Table 2 summarizes the results of the sandbox experiment. Here, *Disjointness materialization* refers to the addition of disjointness axioms to subclasses. In the example above, if `clgo:Swedish_Rock_Musician` was defined as a subclass of `clgo:Person`, and `clgo:Person` is disjoint with `clgo:Organization`, the reasoner would also infer that `clgo:Swedish_Rock_Musician` is disjoint with `clgo:Organization` (and its subclasses). However, we just inspected that out of curiosity, and removed disjointness axioms from the larger scale datasets.

⁴<https://github.com/owlcs/owlapi>

```

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix clgo: <http://caligraph.org/ontology/> .
@prefix clgr: <http://caligraph.org/resource/> .

clgo:Agent a owl:Class;
  rdfs:subClassOf owl:Thing .

clgo:Organization a owl:Class;
  rdfs:subClassOf clgo:Agent .

clgo:Person a owl:Class;
  rdfs:subClassOf clgo:Agent;
  owl:disjointWith clgo:Organization .

clgo:International_organization a owl:Class;
  rdfs:subClassOf clgo:Organization .

clgo:Organization_based_in_Asia a owl:Class;
  rdfs:subClassOf clgo:Organization .

clgo:Organization_based_in_China a owl:Class;
  rdfs:subClassOf clgo:Organization_based_in_Asia .

clgo:headquarter a owl:ObjectProperty .

clgr:China a owl:NamedIndividual .

_:b1 a owl:Restriction;
  owl:onProperty clgo:headquarter;
  owl:hasValue clgr:China .

clgo:International_organization_based_in_China a owl:Class;
  rdfs:subClassOf clgo:Organization_based_in_China, clgo:International_organization, _:b1 .

clgr:International_Center_on_Small_Hydro_Power a owl:NamedIndividual,
  clgo:International_organization_based_in_China .

clgo:Organization_disestablished_in_1939 a owl:Class;
  rdfs:subClassOf clgo:Organization .

clgo:activeYearsEndYear a owl:DatatypeProperty .

_:b2 a owl:Restriction;
  owl:onProperty clgo:activeYearsEndYear;
  owl:hasValue "1939"^^xsd:integer .

clgo:Military_unit_or_formation_disestablished_in_1939 a owl:Class;
  rdfs:subClassOf clgo:Organization_disestablished_in_1939, _:b2 .

clgr:46th_Mixed_Brigade a owl:NamedIndividual, clgo:Military_unit_or_formation_disestablished_in_1939 .

```

Figure 2: Sandbox example for testing the reasoners' capabilities of processing restrictions (with individuals and literals) and disjointness axioms

Reasoner	ELK	HermiT	Pellet
Subclass materialization	X	X	X
Disjointness materialization		X	X
Restrictions with individuals		X	X
Restrictions with literals			X

Table 2
Reasoner capabilities as evaluated in the sandbox experiments

Dataset	Triples	Classes	Restrictions	Instances	Inferable Assertions		
					Trans. Types	Individuals	Literals
c1g_10	35	9	2	3	10	1	1
c1g_10e2	510	101	3	70	425	20	1
c1g_10e3	43,120	1,000	79	18,072	4,500	514	5,706
c1g_10e4	297,266	10,006	1,299	115,501	73,683	12,085	10,369
c1g_10e5	4,641,400	99,923	12,147	1,968,282	42,820	187,929	7,120
c1g_full	54,914,982	1,061,598	227,173	14,452,393	138,713,499	9,637,354	969,022

Table 3
Overview of the different CaLiGraph subsets used for the evaluation

We tried a couple of other reasoners, which were not capable of completely processing the Sandbox example, including

- jcel [15] version 0.24.121
- Snorocket [16] version 4.0.17
- Konclude [17] version 0.7.0

Those did either not produce any of the desired statements, or reported by themselves that they do not support inference based on `owl:hasValue` constructs. Thus, we decided to only keep ELK as a representative of those reasoners which cannot produce the inferences sought to compare its performance to those which can, and discarded the rest.

3.3. Scalability Experiments

To conduct experiments for scalability, we extracted differently sized subsets of CaLiGraph, each of which should have around 10^n classes for different values of n . The resulting datasets are depicted in table 3. The datasets were designed so that they create meaningful subtrees of the overall hierarchy in CaLiGraph. The corresponding SPARQL query is depicted in figure 3. The different datasets are generated by varying the inner SELECT statement which retrieves the leaf classes. Datasets up to `c1g_10e4` are drawn from subclasses of `c1go:Organization`, while `c1g_10e5` is drawn from the complete ontology. The relatively low number of transitive types of the latter dataset is explained by the fact that the average type depth of subclasses of `c1go:Organization` is much higher than the overall average type depth.

We conducted experiments on a system with 32 Intel Xeon CPUs and 200GB RAM, and we set a timeout of 72h. Overall, ELK was the only system capable of processing all datasets (but

```

PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX clgo: <http://caligraph.org/ontology/>
CONSTRUCT {
  ?leaf ?leafPred ?leafObj .
  ?super ?superPred ?superObj .
  ?ind a owl:NamedIndividual, ?leaf .
  ?resPred ?resPredPred ?resPredObj .
  ?resObj ?resObjPred ?resObjObj .
} WHERE {
  # Collect set of leaf classes to bootstrap sample from
  # Sample subgraph can be changed by picking something else than clgo:Organization as basis
  # Use a specific LIMIT for N to restrict the number of leaf classes in the sample
  {
    SELECT ?leaf {
      ?leaf rdfs:subClassOf+ clgo:Organization .
      FILTER NOT EXISTS {?sub rdfs:subClassOf ?leaf}
    }
    LIMIT N
  }
  # Collect all relevant superclasses of the leaves
  ?leaf rdfs:subClassOf+ ?super ;
  ?leafPred ?leafObj .
  ?super ?superPred ?superObj .
  # Collect individuals for the leaves
  ?ind a ?leaf .
  # Add restrictions, if available
  OPTIONAL {
    ?leaf rdfs:subClassOf+ ?res .
    ?res a owl:Restriction ;
      owl:onProperty ?resPred ;
      owl:hasValue ?resObj .
    # Add information about the predicates and objects of restrictions to the sample
    ?resPred ?resPredPred ?resPredObj .
    OPTIONAL {
      ?resObj ?resObjPred ?resObjObj .
    }
  }
  # Discard disjointnesses as this would blow up the sample size
  FILTER (?superPred != owl:disjointWith)
  FILTER (?leafPred != owl:disjointWith)
}

```

Figure 3: SPARQL query to generate datasets for scalability experiments

not returning any of the inferable individual and literal assertions, as discussed above). The processing of the largest dataset with ELK took about 3.5 hours.

HermiT and Pellet, on the other hand, could only process `clg_10e2` and `clg_10e3`, respectively, and timed out on the larger ones (with Pellet running out of memory on `clg_full`). This shows that those reasoning systems are still three to four orders of magnitude away from the reasoning capabilities required for processing CaLiGraph.

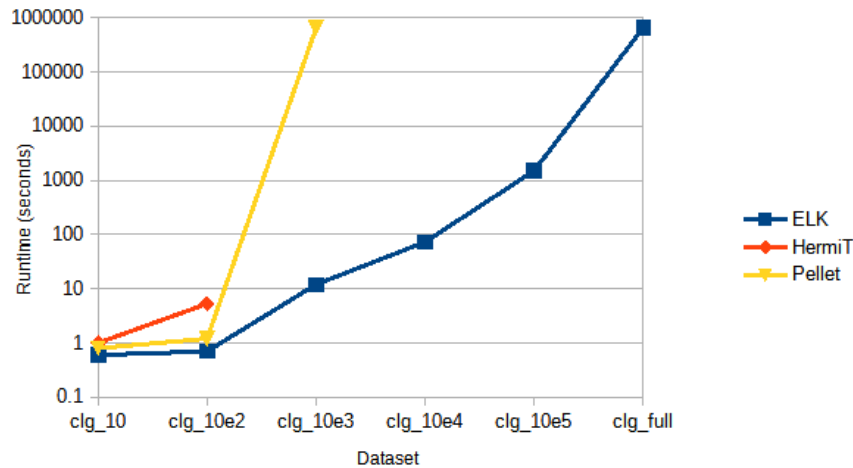


Figure 4: Scalability of the Reasoning Systems investigated

4. Conclusion and Outlook

In this paper, we investigated three state of the art OWL2EL reasoners, i.e., ELK, HermiT, and Pellet. The task for those reasoners was to materialize the A-box of the CaLiGraph knowledge graph, particularly the creation of around 10M A-box assertions from `owl:hasValue` definitions. As discussed above, this kind of reasoning at scale is currently not evaluated in benchmarks such as ORE2015 and OWL2Bench.

The outcome of our experiments is that the reasoners evaluated are either (a) not capable of performing that inference at all, as in the case of ELK, or (b) not scalable enough to process knowledge graphs with 10k instances or more – which is some orders of magnitude below the size of popular knowledge graphs [4].

With this work, we have also provided a benchmark suite of differently sized test sets.⁵ This allows for running scalability experiments also with respect to materializing a large amount of individual and literal A-box assertions. Moreover, the benchmark is suitable for running evaluations with approximate reasoning systems, since recall and precision of the generated A-box assertions can be quantified. Therefore, the datasets are also suitable for measuring the trade-off between scalability and accuracy of approximate reasoning systems.

CaLiGraph is generated heuristically. Although the quality is high [7], it is not perfect. In the future, we want to further analyze the impact of wrong statements in the graph, as well as experiment with artificially adding noise to the different datasets.

While we have generated subsets of CaLiGraph using the SPARQL query in figure 3 and were therefore experimenting with manual limits to get the desired amount of classes for the sample, more sophisticated methods for generating subsets of knowledge graphs exist. For example, Melo and Paulheim [18] propose a synthesis model for the generation of knowledge graph

⁵The datasets and the code for running the experiments are available at <https://github.com/nheist/CaLiGraph-for-SemREC>.

benchmark datasets which strives to preserve characteristics like class, relation and instance distributions. For future work, we consider to generate such more realistic benchmark datasets with the help of synthesis models.

References

- [1] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al., Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia, *Semantic web 6 (2015)* 167–195.
- [2] F. M. Suchanek, G. Kasneci, G. Weikum, Yago: a core of semantic knowledge, in: *Proceedings of the 16th international conference on World Wide Web, 2007*, pp. 697–706.
- [3] D. Vrandečić, Wikidata: A new platform for collaborative data collection, in: *Proceedings of the 21st international conference on world wide web, 2012*, pp. 1063–1064.
- [4] N. Heist, S. Hertling, D. Ringler, H. Paulheim, Knowledge graphs on the web-an overview, in: P. H. Ilaria Tiddi, Freddy Lécué (Ed.), *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*, volume 47 of *Studies on the Semantic Web*, IOS Press/AKA, 2020, pp. 3–22.
- [5] N. Heist, H. Paulheim, Uncovering the semantics of wikipedia categories, in: *International semantic web conference, Springer, 2019*, pp. 219–236.
- [6] N. Heist, H. Paulheim, Entity extraction from wikipedia list pages, in: *European Semantic Web Conference, Springer, 2020*, pp. 327–342.
- [7] N. Heist, H. Paulheim, Information extraction from co-occurring similar entities, in: *Proceedings of the Web Conference 2021, 2021*, pp. 3999–4009.
- [8] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz, et al., Owl 2 web ontology language profiles, 2009. URL: <https://www.w3.org/TR/owl2-profiles/>.
- [9] G. Antoniou, S. Batsakis, R. Mutharaju, J. Z. Pan, G. Qi, I. Tachmazidis, J. Urbani, Z. Zhou, A survey of large-scale reasoning on the web of data, *The Knowledge Engineering Review* 33 (2018).
- [10] B. Parsia, N. Matentzoglou, R. S. Gonçalves, B. Glimm, A. Steigmiller, The owl reasoner evaluation (ore) 2015 competition report, *Journal of Automated Reasoning* 59 (2017) 455–482.
- [11] G. Singh, S. Bhatia, R. Mutharaju, Owl2bench: A benchmark for owl 2 reasoners, in: *International Semantic Web Conference, Springer, 2020*, pp. 81–96.
- [12] Y. Kazakov, M. Krötzsch, F. Simančík, The incredible elk, *Journal of automated reasoning* 53 (2014) 1–61.
- [13] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, Z. Wang, Hermit: an owl 2 reasoner, *Journal of Automated Reasoning* 53 (2014) 245–269.
- [14] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical owl-dl reasoner, *Journal of Web Semantics* 5 (2007) 51–53.
- [15] J. Mendez, jcel: A modular rule-based reasoner., in: *ORE, 2012*.
- [16] M. J. Lawley, C. Bousquet, Fast classification in protégé: Snorocket as an owl 2 el reasoner, in: *Proc. 6th Australasian Ontology Workshop (IAOA'10). Conferences in Research and Practice in Information Technology*, volume 122, 2010, pp. 45–49.

- [17] A. Steigmiller, T. Liebig, B. Glimm, Konclude: system description, *Journal of Web Semantics* 27 (2014) 78–85.
- [18] A. Melo, H. Paulheim, Synthesizing knowledge graphs for link and type prediction benchmarking, in: *European Semantic Web Conference*, Springer, 2017, pp. 136–151.