Wikidata analysis and constraint validation with KGTK

Daniel Garijo

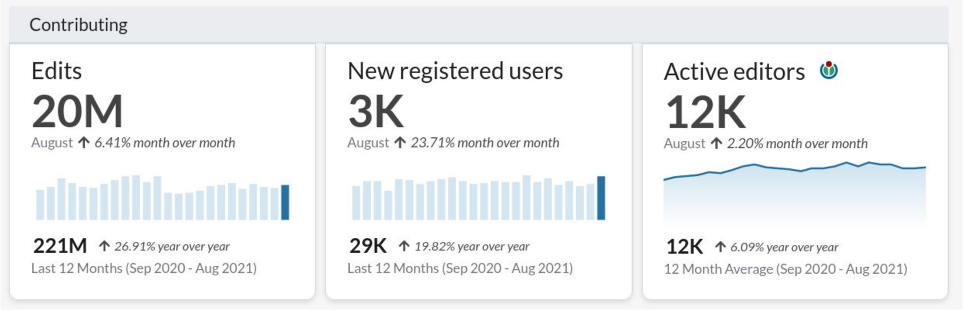
Universidad Politécnica de Madrid, daniel.garijo@upm.es ✉

@dgarijov 🐦

KGTK: Tools for Creating and Exploiting Large Knowledge Graphs. The Web Conference, 2022

A short introduction to Wikidata

- Free
- Collaborative
- Multilingual
- A secondary database
- Collecting structured data
- Support for Wikimedia wikis
- For anyone in the world
- With many applications
- A thriving community of contributors



Contributing

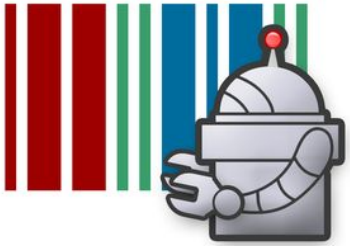
Edits
20M
August ↑ 6.41% month over month
221M ↑ 26.91% year over year
Last 12 Months (Sep 2020 - Aug 2021)

New registered users
3K
August ↑ 23.71% month over month
29K ↑ 19.82% year over year
Last 12 Months (Sep 2020 - Aug 2021)

Active editors
12K
August ↑ 2.20% month over month
12K ↑ 6.09% year over year
12 Month Average (Sep 2020 - Aug 2021)

> 90M items
> 8,000 properties
> 2,000,000 classes
> 4,000 external id types
> 1B triples

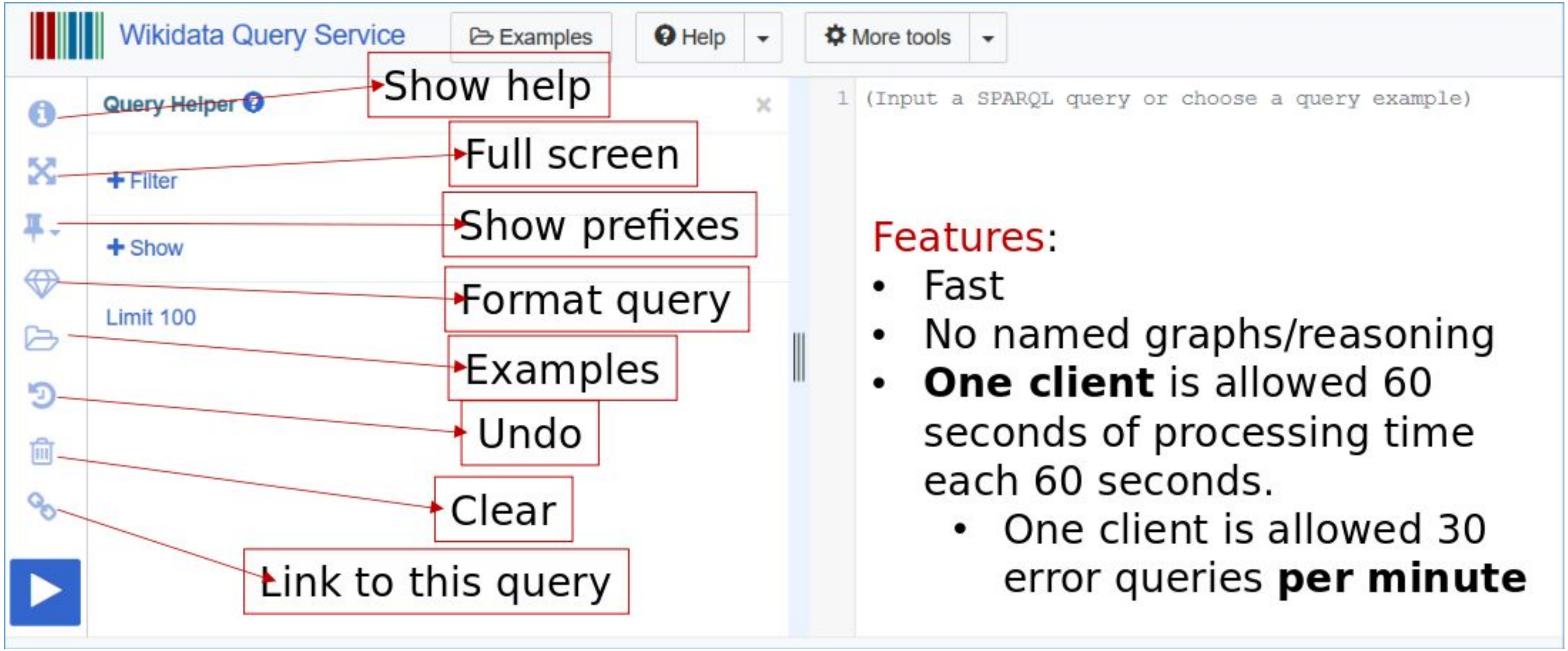
Public SPARQL endpoint



Bots allowed

Working with Wikidata: Endpoint

https://query.wikidata.org



(Input a SPARQL query or choose a query example)

Features:

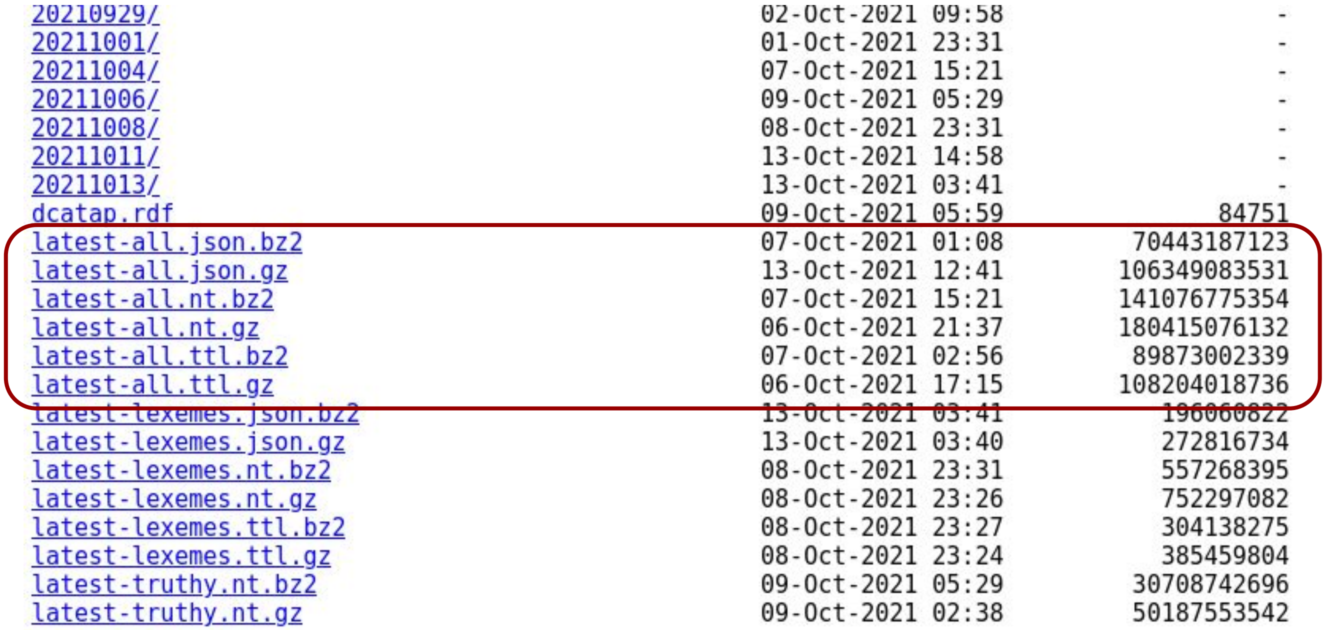
- Fast
- No named graphs/reasoning
- **One client** is allowed 60 seconds of processing time each 60 seconds.
  - One client is allowed 30 error queries **per minute**

+ Easy to use and setup
+ Works reasonably well with many results

- Lack of support for complex queries (time outs)

Working with Wikidata: Dumps

https://dumps.wikimedia.org/wikidatawiki/entities/



```
20210929/                   02-Oct-2021 09:58                   -
20211001/                   01-Oct-2021 23:31                   -
20211004/                   07-Oct-2021 15:21                   -
20211006/                   09-Oct-2021 05:29                   -
20211008/                   08-Oct-2021 23:31                   -
20211011/                   13-Oct-2021 14:58                   -
20211013/                   13-Oct-2021 03:41                   -
dcatap.rdf                  09-Oct-2021 05:59               84751
latest-all.json.bz2         07-Oct-2021 01:08         70443187123
latest-all.json.gz          13-Oct-2021 12:41        106349083531
latest-all.nt.bz2           07-Oct-2021 15:21        141076775354
latest-all.nt.gz            06-Oct-2021 21:37        180415076132
latest-all.ttl.bz2          07-Oct-2021 02:56         89873002339
latest-all.ttl.gz           06-Oct-2021 17:15        108204018736
latest-lexemes.json.bz2     13-Oct-2021 03:41           196060822
latest-lexemes.json.gz      13-Oct-2021 03:40           272816734
latest-lexemes.nt.bz2       08-Oct-2021 23:31           557268395
latest-lexemes.nt.gz        08-Oct-2021 23:26           752297082
latest-lexemes.ttl.bz2      08-Oct-2021 23:27           304138275
latest-lexemes.ttl.gz       08-Oct-2021 23:24           385459804
latest-truthy.nt.bz2        09-Oct-2021 05:29         30708742696
latest-truthy.nt.gz         09-Oct-2021 02:38         50187553542
```

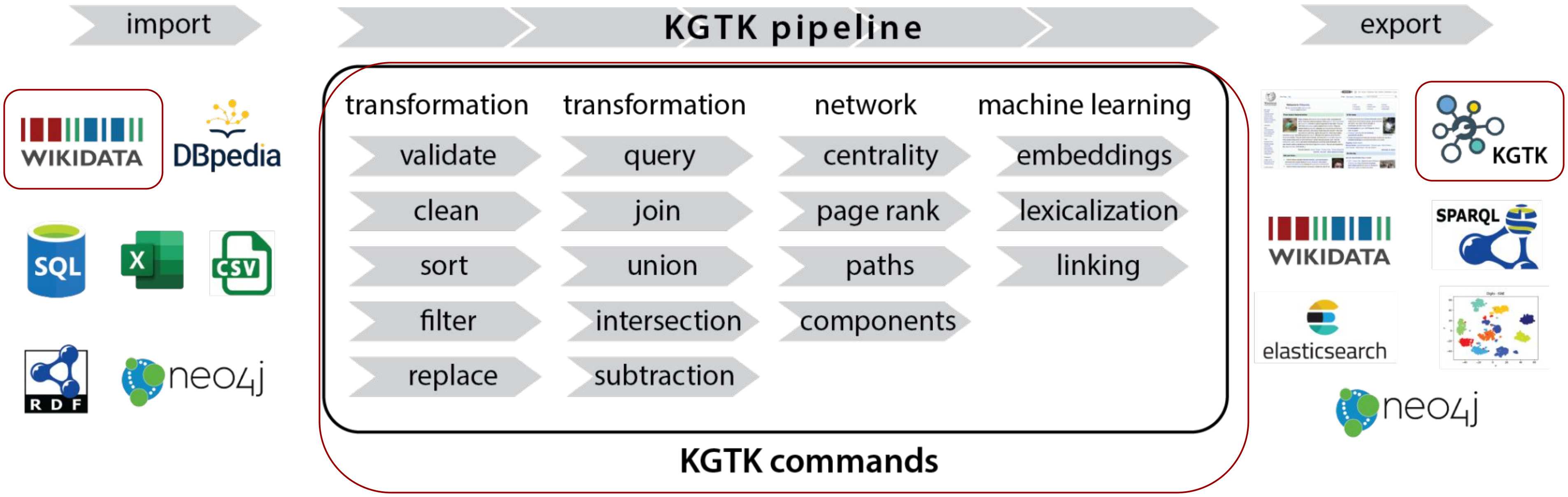
> 100 GB (compressed)
> 150GB uncompressed

+ Once loaded, support for complex queries

- Time needed for loadouts (days-week)
- Operations over the data are costly and slow

KGTK to the rescue

jupyter



import

WIKIDATA

DBpedia

SQL

X

CSV

RDF

neo4j

KGTK pipeline

transformation	transformation	network	machine learning
validate	query	centrality	embeddings
clean	join	page rank	lexicalization
sort	union	paths	linking
filter	intersection	components	
replace	subtraction		

KGTK commands

export

KGTK

WIKIDATA

SPARQL

elasticsearch

neo4j

Outline

1) **Evolution of Wikidata: Analyzing > 300 dumps**
   - But… why?
   - Data collection
   - Analysis with KGTK
   - Results

2) Wikidata quality analysis
   - Defining quality in Wikidata
   - Anatomy of a constraint
   - Constraint validation with KGTK
   - Results
   - Playing around with the sample KG

Why look at the evolution of Wikidata? (2014-2021)

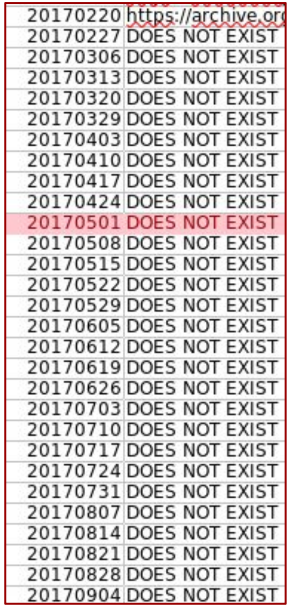
- How do large KGs (Wikidata) get populated:
  - When were classes added
  - When were new properties added?
  - Stable terms (highly used properties and classes)
  - “Complete” classes (few new individuals added)
  - Are highly edited classes more or less connected? (pagerank)

- Timeliness of large KGs (propagation of changes)
  - Lag: how much time is there between a qualifier is added to its respective statement?

Work with Pedro Szekely and FIlip Ilievski

Data collection: Challenges

- Weekly dumps in https://dumps.wikimedia.org/wikidatawiki/entities/ only go back a few months
- Internet Archive has many Wikidata dumps
  - Unfortunately, there are gaps for many months!
  - Reached out the community for help
- Size of dataset
  - First releases of Wikidata are a few GB compressed
  - Last releases are > 100 GB (compressed)
- Total dumps collected: **311** (Oct 2014 - Jan 2021)
  - > 15 TB (compressed)



20170220	https://archive.or
20170227	DOES NOT EXIST
20170306	DOES NOT EXIST
20170313	DOES NOT EXIST
20170320	DOES NOT EXIST
20170329	DOES NOT EXIST
20170403	DOES NOT EXIST
20170410	DOES NOT EXIST
20170417	DOES NOT EXIST
20170424	DOES NOT EXIST
20170501	DOES NOT EXIST
20170508	DOES NOT EXIST
20170515	DOES NOT EXIST
20170522	DOES NOT EXIST
20170529	DOES NOT EXIST
20170605	DOES NOT EXIST
20170612	DOES NOT EXIST
20170619	DOES NOT EXIST
20170626	DOES NOT EXIST
20170703	DOES NOT EXIST
20170710	DOES NOT EXIST
20170717	DOES NOT EXIST
20170724	DOES NOT EXIST
20170731	DOES NOT EXIST
20170807	DOES NOT EXIST
20170814	DOES NOT EXIST
20170821	DOES NOT EXIST
20170828	DOES NOT EXIST
20170904	DOES NOT EXIST

Data analysis with KGTK

For each dump

- Import from JSON -> KGTK format
- Sort
- Calculate deltas (external script)
- Count classes and instances (and qualifiers)
- Pagerank, hubs
- Compress results and save them

Challenges:

- Errors on import (JSON format has changed in Wikidata)
- Errors with problematic dumps
  - Problems recognizing some quantities, literals, etc.

Data extraction and analysis: example in KGTK: sh scripts

```
while read -r line ; do
        # Initialize folder, unpack and sort. $line has the full path
         echo "Processing file: $line"
         folder_new_name=$(basename $line)
         folder_new_name="${folder_new_name%%.*}"
        echo "Name of folder: $folder_new_name"

         mkdir $folder_new_name
         echo "Importing file..."
         TEMPDIR=$folder_new_name
         # Import the Wikidata dump file, getting labels, aliases, and descriptions
         # in English and in all languages.
         kgtk  --debug --timing --progress import-wikidata \
        -i $line \
        --node ${TEMPDIR}/nodefile.tsv \
        --edge ${TEMPDIR}/edgefile.tsv \
        --qual ${TEMPDIR}/qualfile.tsv \
        --use-mgzip-for-input True \
        --use-mgzip-for-output True \
        --use-shm True \
        --procs 6 \
        --mapper-batch-size 5 \
        --max-size-per-mapper-queue 3 \
        --single-mapper-queue True \
        --collect-results True \
        --collect-seperately True\
        --collector-batch-size 10 \
        --collector-queue-per-proc-size 3 \
        --progress-interval 500000 --fail-if-missing False

         echo "Sorting file ..."
         kgtk sort -i "$folder_new_name"/edgefile.tsv -c 'id' -o "$folder_new_name"/edgefile_sorted.tsv
         # Remove edge file (to save a little space)
         rm "$folder_new_name"/edgefile.tsv
done < dumps_to_import.txt
```

Import + sort

```
folder=$PWD
for entry in "$folder"/*
do
  #echo "Processing: $entry"
        file_name=$(basename $entry)
        FILE="$entry"/edgefile_sorted.tsv
        if [ -f "$FILE" ]; then
  echo "Processing $FILE"
                kgtk query --debug --graph-cache /wikidata.sqlite3.db -i
"$entry"/edgefile_sorted.tsv -i datatypes.tsv -o "$entry"/claims.wikibase-item.tsv.gz \
 --match 'edgefile_sorted: (n1)-[l {label: p}]->(n2), datatypes:
(p)-[:datatype]->(:`wikibase-item`)' \
 --return 'l as id, n1 as node1, p as label, n2 as node2' \
 --order-by 'l'
```

1

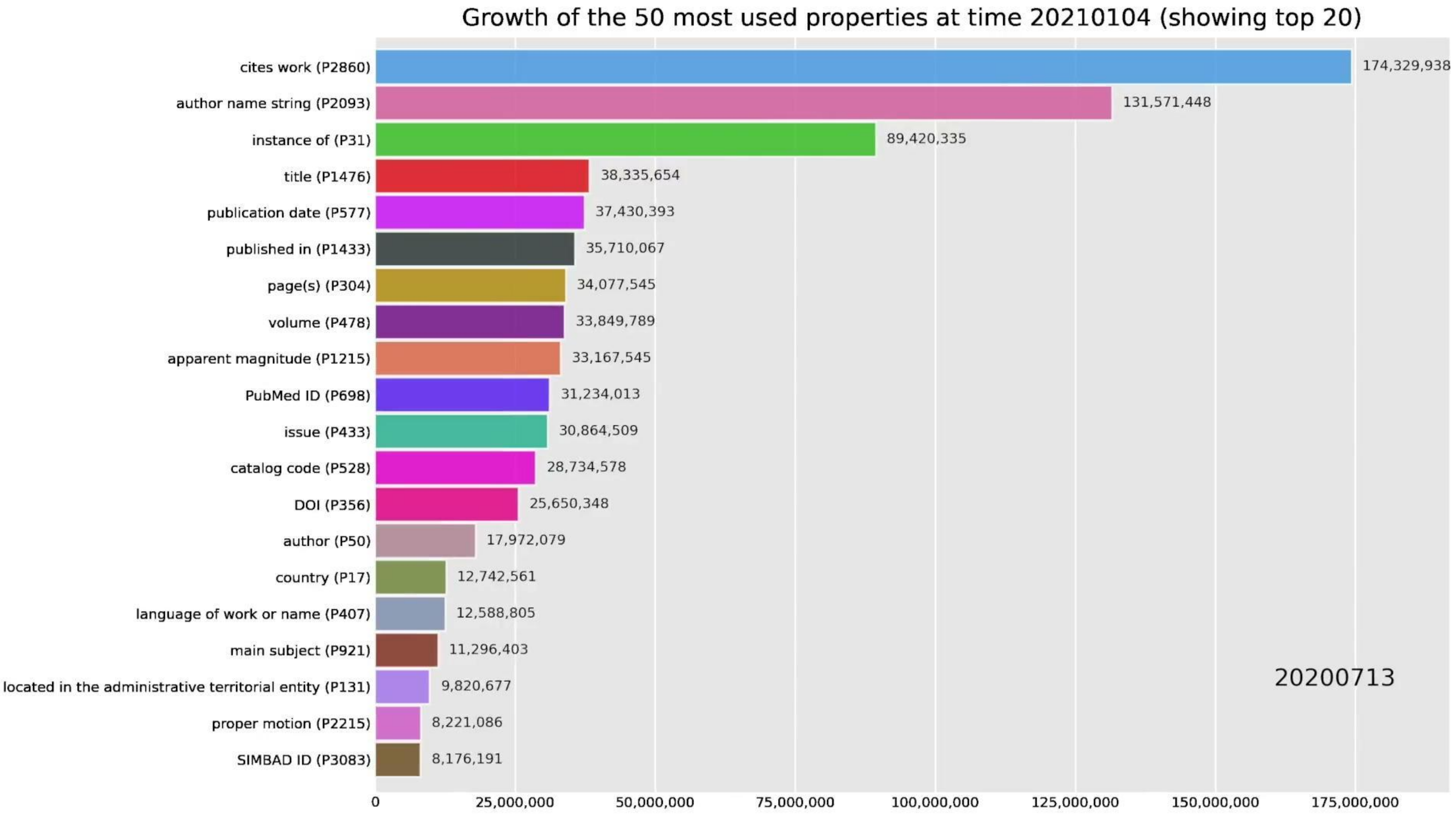
```
kgtk graph-statistics -i "$entry"/claims.wikibase-item.tsv.gz -o
"$entry"/pagerank.undirected.tsv.gz \
  --page-rank-property undirected_pagerank \
  --pagerank --statistics-only --hits\
  --log "$entry"/pagerank.undirected.summary.txt --print-top-n 100
```

2

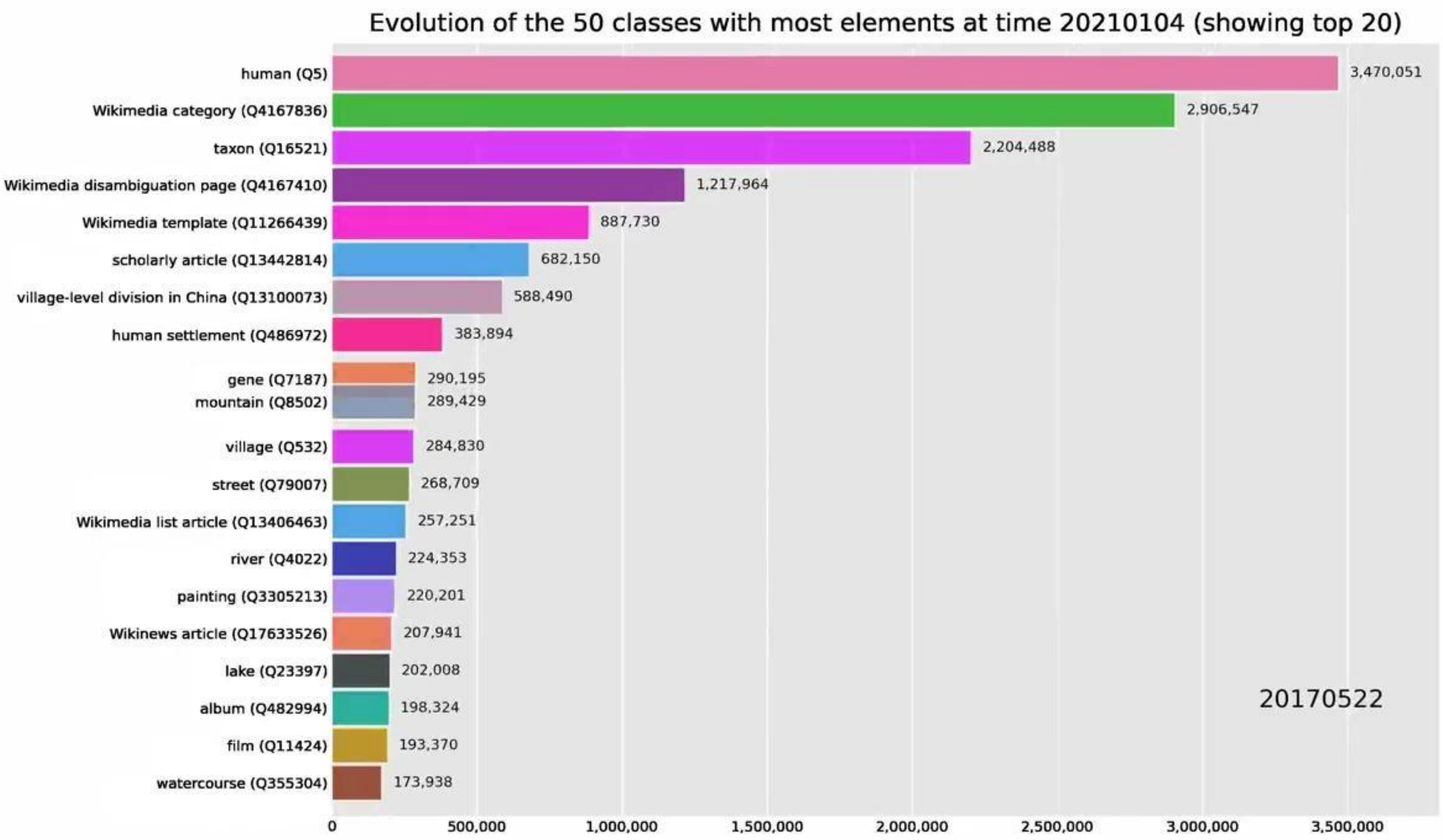
```
rm /wikidata.sqlite3.db
        fi
        #cp "$entry"/modified_prop_count.tsv
"$target"/"$file_name"_modified_prop_count.tsv
done
```

1. Retrieve just wikibase items
2. Calculate top 100 entities in the page rank

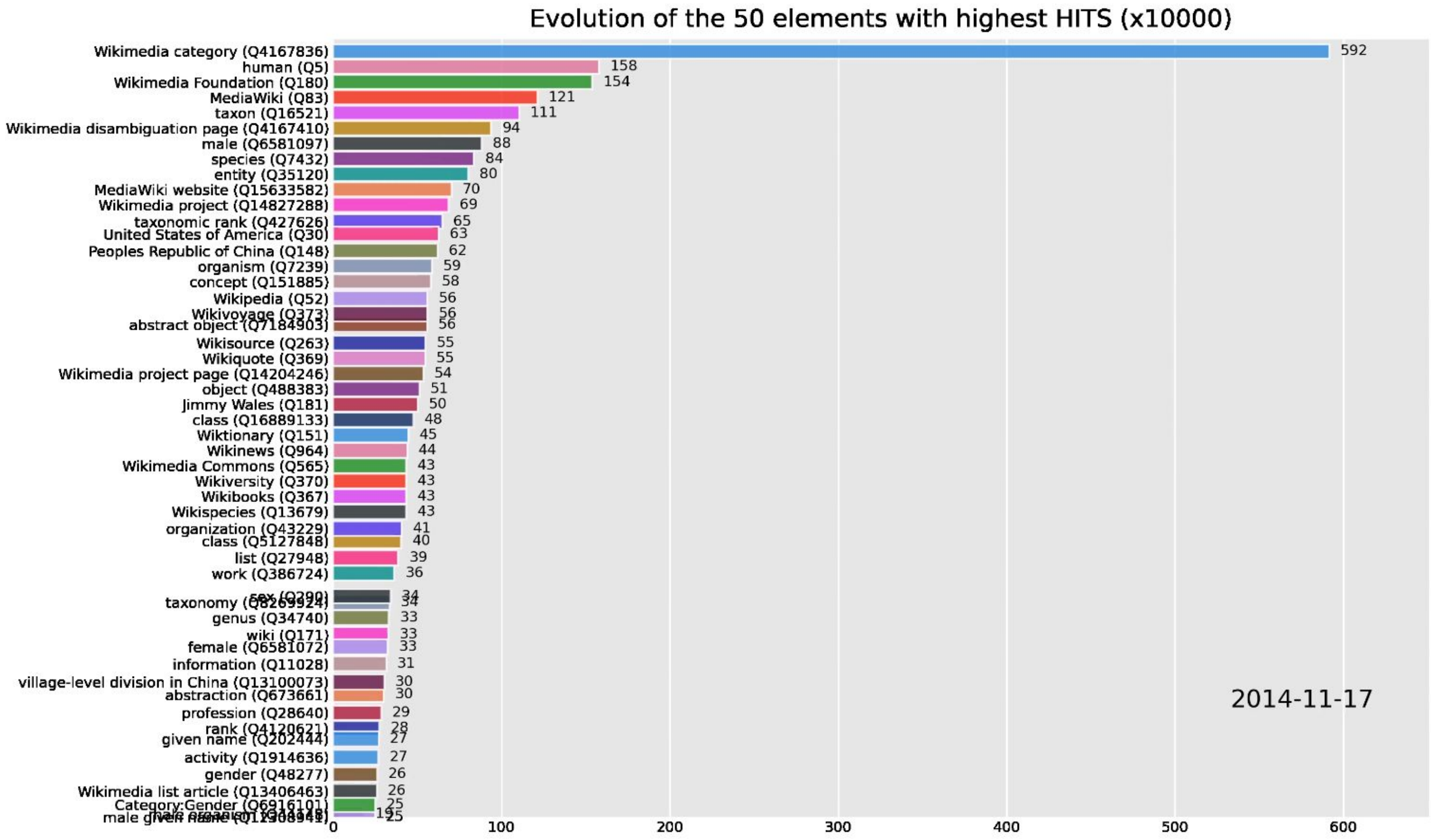
Results: Evolution of the most popular properties



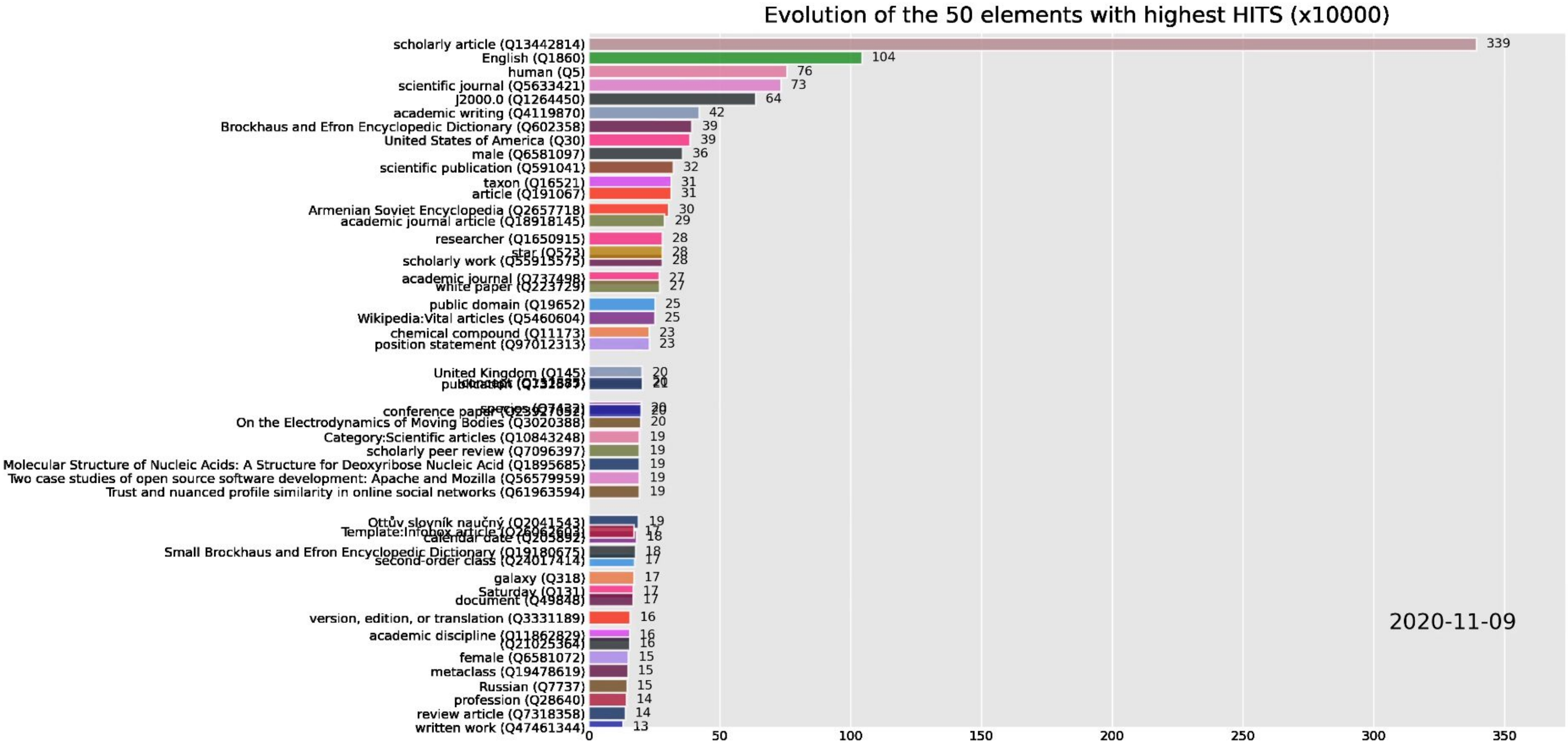
Results: Evolution of the most popular classes



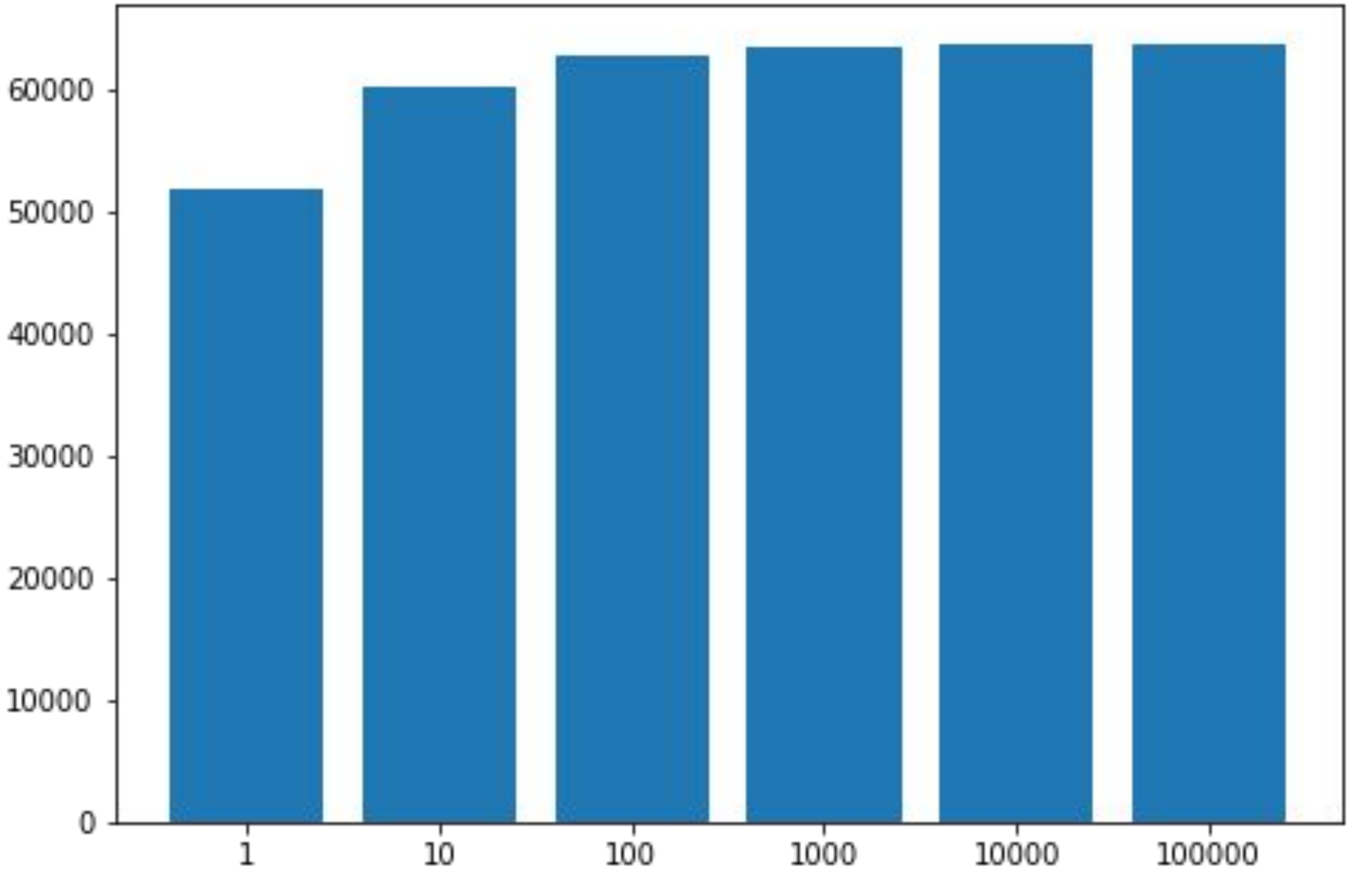
Results: Evolution of the entities with biggest page rank



Results: Evolution of the entities with biggest page rank



Results: How “stable” are Wikidata classes?

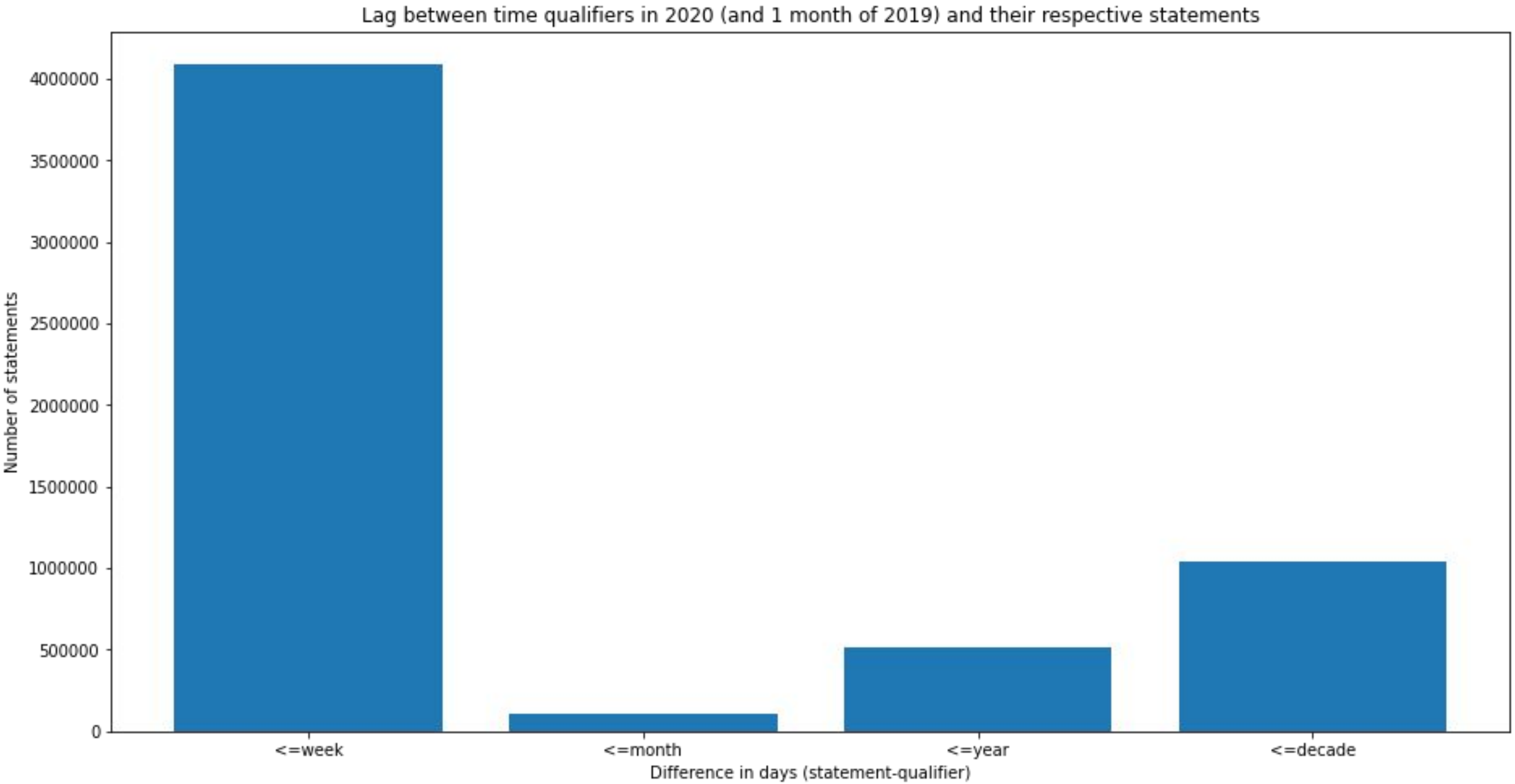


X axis: variance
Y axis: number of classes with that variance.

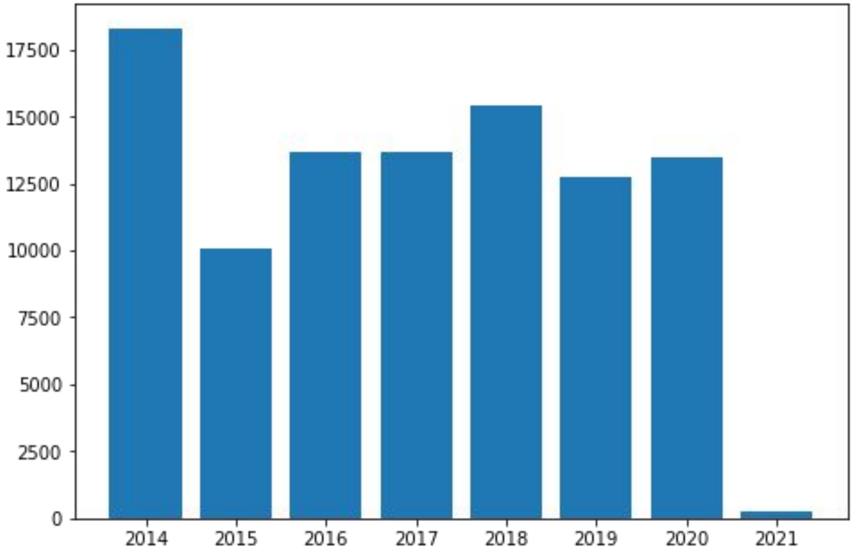
e.g., number of classes that had a difference of 10 **<u>or less</u>** instances with respect the previous week for any week (some weeks could be 0) during 2020 is 60292 (very stable).

Depopulated classes have not been included

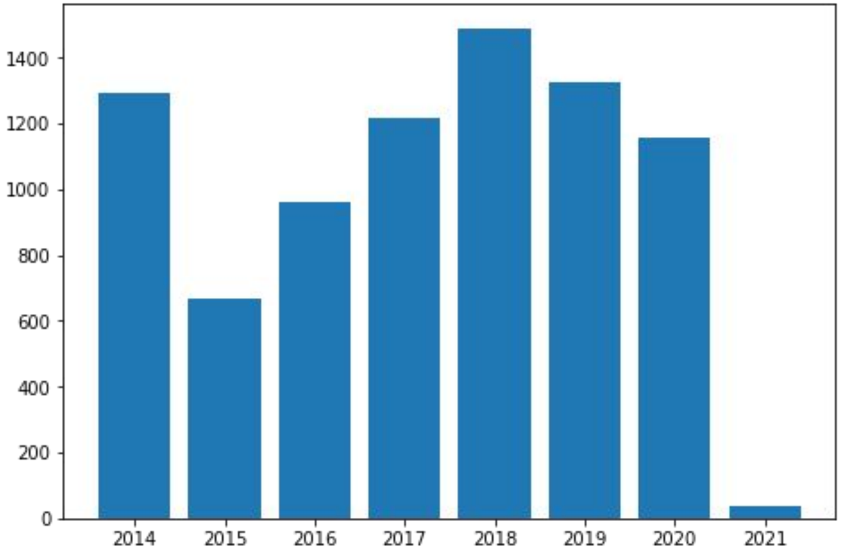
Results: Lag between qualifiers and the entity they describe (2020)



Results: How alive is Wikidata (new terms)



When were classes first populated?



When were properties first used?

More results soon (ongoing paper)

Outline

1) Evolution of Wikidata: Analyzing > 300 dumps
   - But… why?
   - Data collection
   - Analysis with KGTK
   - Results

2) **Wikidata quality analysis**
   - Defining quality in Wikidata
   - Anatomy of a constraint
   - Constraint validation with KGTK
   - Results
   - Playing around with the sample KG

Data Quality in Wikidata

Crowdsourced data is great, but:

- Conceptual issues
  - Conflicting real-world models
  - Mixup of meta-levels
  - Conceptual ambiguity
  - Subclass of cycles
  - Entity vs class distinction
  - Messy upper ontology
- Constraint violations
  - Inconsistent modeling
- Duplicate entities
- ...

Based on

Ontology issues in Wikidata

An overview

By lydia.pintscher

https://commons.wikimedia.org/wiki/File:DataQualityDaysontologyissues.pdf

What questions did we want to explore regarding quality?

- Q1: Are entities being deduplicated?
- Q2: Can the community distinguish classes from instances?
- Q3: Are property types and value types respected?
- Q4: Can we detect missing triples?
- Q5: Are constraints correct and complete?
- Q6: What statements get deprecated?
- Q7: Are constraint violations getting fixed?

Work with Kartik Shenoy, Filip Ilievski, Pedro Szekely and Daniel Schwabe. https://arxiv.org/abs/2107.00156

KGTK Analysis: Three indicators of low quality statements

Sources:

- Permanently deleted statements **(76.5 M)**
  - Q1 (calculate redirections in entity deduplication)
  - Q2 (which entities have switched from class to instance?)
  - Q7 (constraint violation correction)
- Deprecated statements **(10 M)**
  - Q6 (what statements get deprecated)
- Constraint violations **(symmetric, inverse, etc.)**
  - Q3 (property types getting respected)
  - Q4 (infer new triples)
  - Q5 (number of constraint violations)
  - Q7 (constraint violation correction)

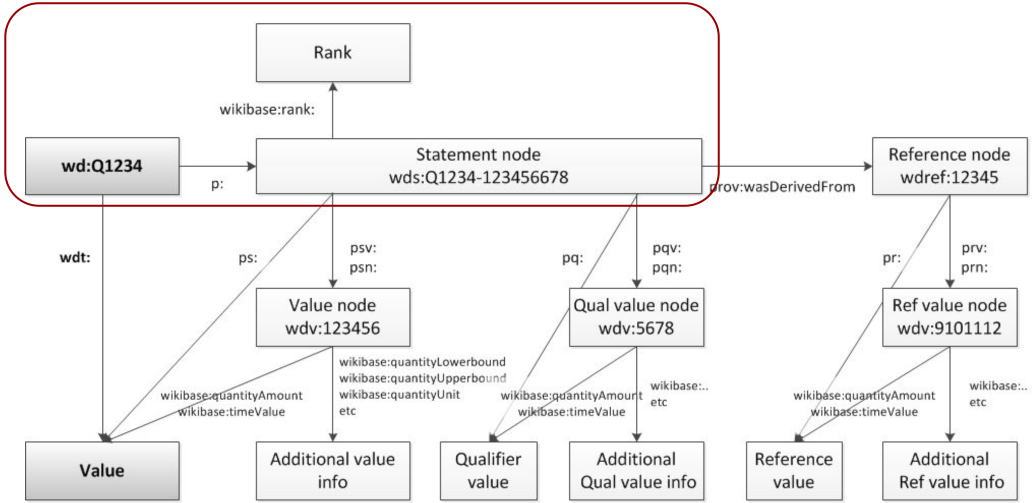
KGTK Analysis: Data sources

Permanently deleted statements: extracted from the evolution analysis

```
kgtk ifnotexists -i $removed --filter-on "$entry"/added.tsv -o $removed_aux
# Aggregate the difference into the deleted file
echo "Concatenating statements at the end of $removed"
kgtk cat -i $removed_aux "$entry"/deleted.tsv  -o $removed
```

Deprecated terms

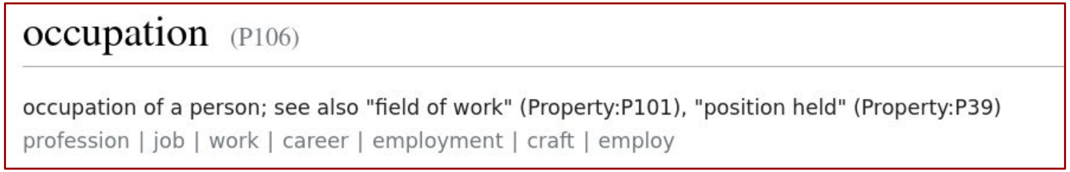
- Filter by rank “deprecated”



Constraint violations

- Kgtk query

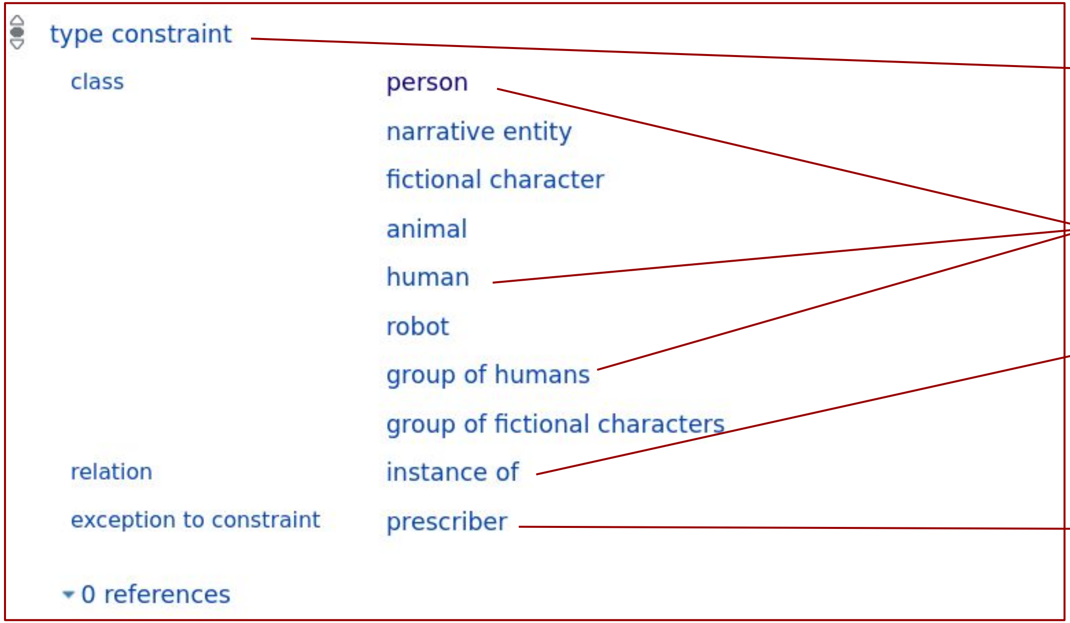
Anatomy of a Wikidata constraint



occupation (P106)

occupation of a person; see also "field of work" (Property:P101), "position held" (Property:P39)

profession | job | work | career | employment | craft | employ



type constraint

class	person narrative entity fictional character animal human robot group of humans group of fictional characters
relation	instance of
exception to constraint	prescriber

▾ 0 references

Constraint type (there are over 30). Similar to rdfs:domain

Described entity should be instance of one of these classes

Exceptions to the rule (either classes or instances)

Work with Kartik Shenoy, Filip Ilievski, Pedro Szekely and Daniel Schwabe. https://arxiv.org/abs/2107.00156

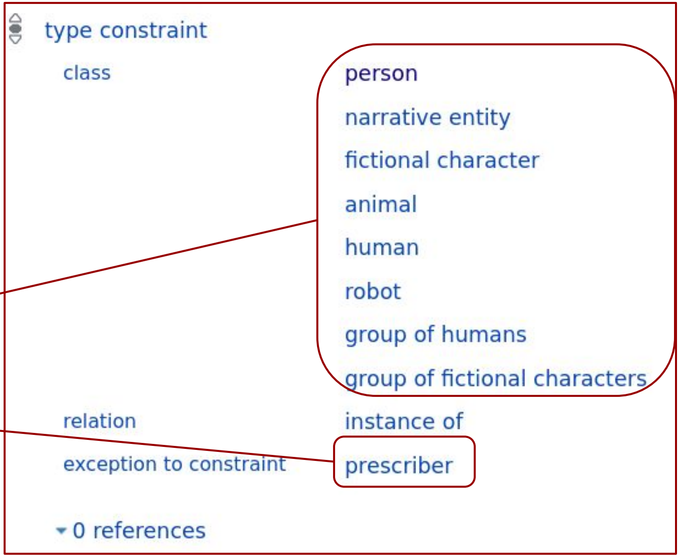
Constraint validation with KGTK

We limited ourselves to five common types of constraints: type, value-type, item-requires-statement, symmetric and inverse.

Example for type constraint template:

```
kgtk query
-i statements_with_property instance_of subclass_of_star \
--match statements: (subject)-[id {label:property}]->(object), \
        instance_of: (subject)-[]->(class), \
        subclass_of_star: (class)-[]->(parent)' \
--where 'parent in expected_parents or subject in exceptions' \
--return 'distinct id, subject, property, object' \
-o statements_correct.tsv

kgtk ifexists -i statements_with_property --filter-on statements_correct.tsv \
-o statements_incorrect.tsv
```



type constraint

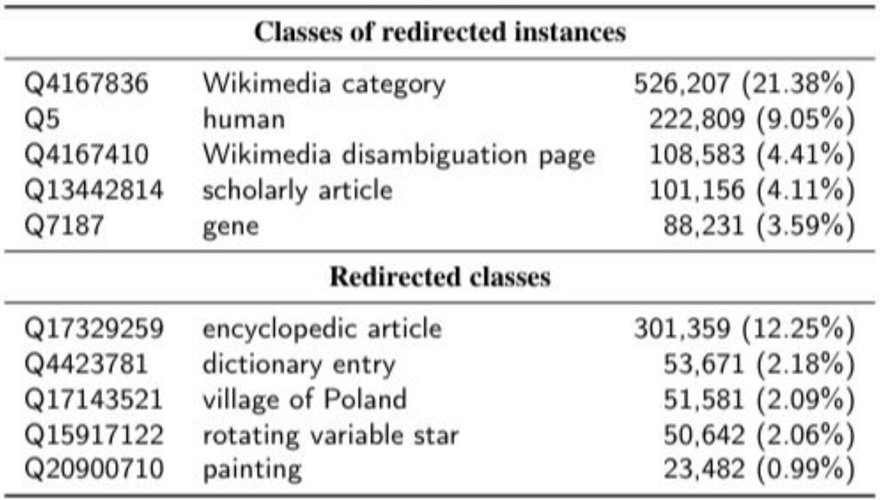
class	person narrative entity fictional character animal human robot group of humans group of fictional characters
relation	instance of
exception to constraint	prescriber

▾ 0 references

Work with Kartik Shenoy, Filip Ilievski, Pedro Szekely and Daniel Schwabe. https://arxiv.org/abs/2107.00156

Result summary

- **Q1: Are entities being deduplicated?**
  - Some are through redirects.
  - 2 million redirected nodes, affecting over 21.3 million statements (27.8% of the removed statements)



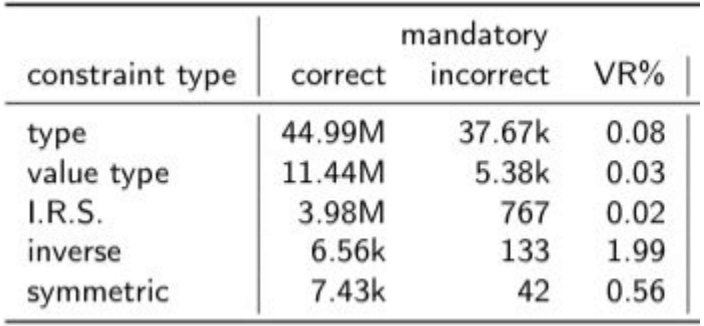
Classes of redirected instances		
Q4167836	Wikimedia category	526,207 (21.38%)
Q5	human	222,809 (9.05%)
Q4167410	Wikimedia disambiguation page	108,583 (4.41%)
Q13442814	scholarly article	101,156 (4.11%)
Q7187	gene	88,231 (3.59%)
Redirected classes		
Q17329259	encyclopedic article	301,359 (12.25%)
Q4423781	dictionary entry	53,671 (2.18%)
Q17143521	village of Poland	51,581 (2.09%)
Q15917122	rotating variable star	50,642 (2.06%)
Q20900710	painting	23,482 (0.99%)

Work with Kartik Shenoy, Filip Ilievski, Pedro Szekely and Daniel Schwabe. https://arxiv.org/abs/2107.00156

Result summary

- **Q2: Can the community distinguish classes from instances?**
  - More than 500.000 changes (class -> instance or vice versa)
  - 440k go from class -> instance

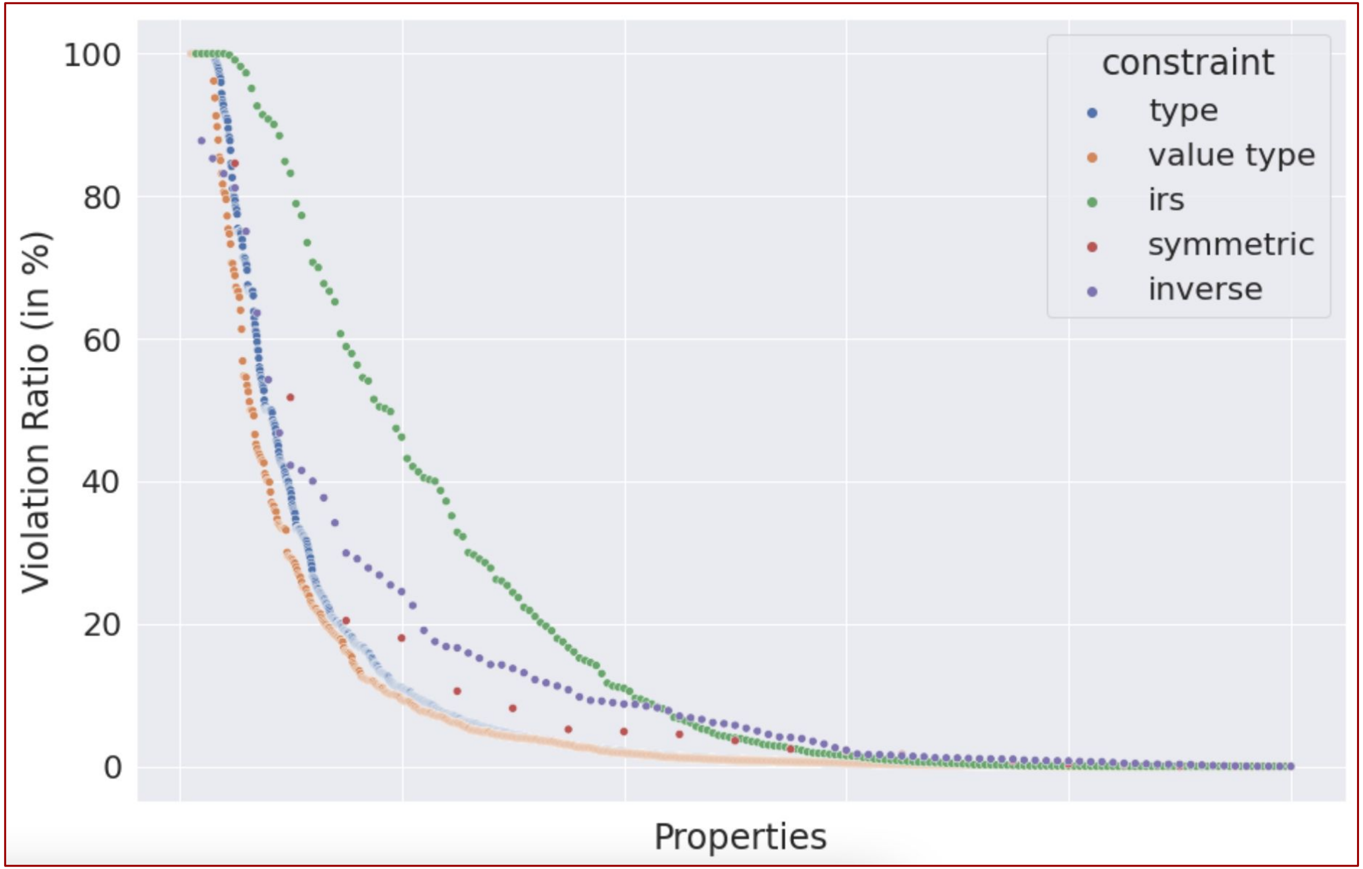
- **Q3: Are property types and value types constraints respected?**
  - Most violation ratios are on “suggested” constraints (20%)
  - Small number of violation ratios for mandatory constraints (1%, but more than 40K statements!)



	mandatory		
constraint type	correct	incorrect	VR%
type	44.99M	37.67k	0.08
value type	11.44M	5.38k	0.03
I.R.S.	3.98M	767	0.02
inverse	6.56k	133	1.99
symmetric	7.43k	42	0.56

Work with Kartik Shenoy, Filip Ilievski, Pedro Szekely and Daniel Schwabe. https://arxiv.org/abs/2107.00156

Constraint violation ratios



Q4: Can we detect missing triples?

Item-require-statement and inverse property violations can be used to suggest candidates.

Q5: Are constraints correct and complete?

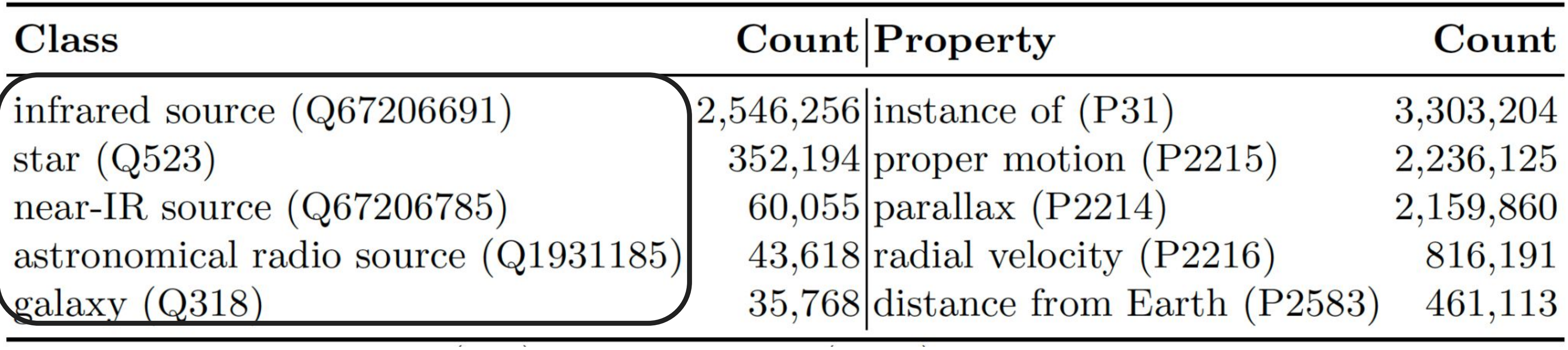
Not always. E.g., those constraints with high violation ratios may need to be reviewed

Each dot is a property with that constraint type

Work with Kartik Shenoy, Filip Ilievski, Pedro Szekely and Daniel Schwabe. *https://arxiv.org/abs/2107.00156*

Result summary

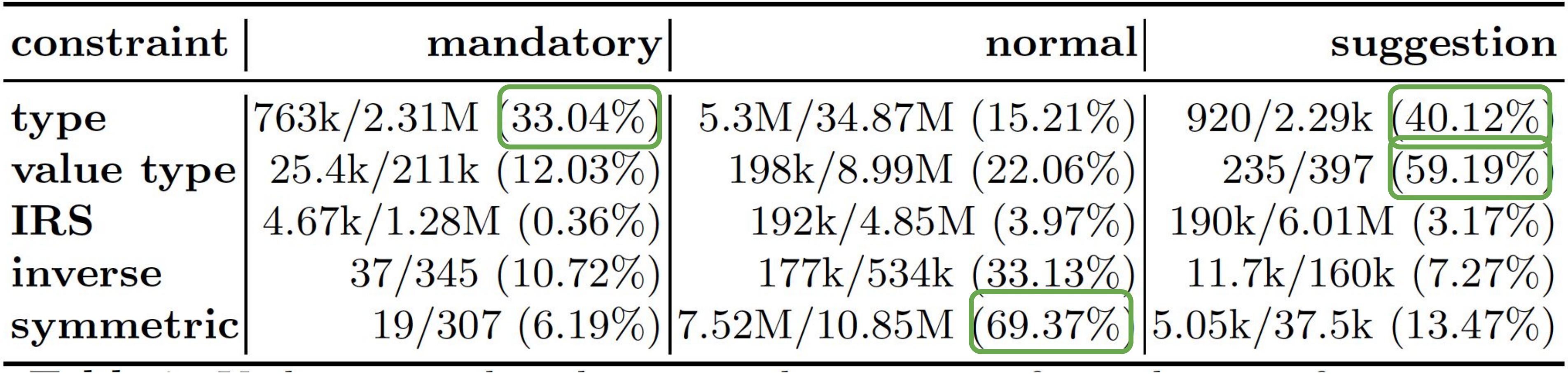
- **Q6: What statements get deprecated?**
  - Largely, in the Astronomy domain
  - Top 5 classes with instances and properties being deprecated:



Class	Count	Property	Count
infrared source (Q67206691)	2,546,256	instance of (P31)	3,303,204
star (Q523)	352,194	proper motion (P2215)	2,236,125
near-IR source (Q67206785)	60,055	parallax (P2214)	2,159,860
astronomical radio source (Q1931185)	43,618	radial velocity (P2216)	816,191
galaxy (Q318)	35,768	distance from Earth (P2583)	461,113

Result summary

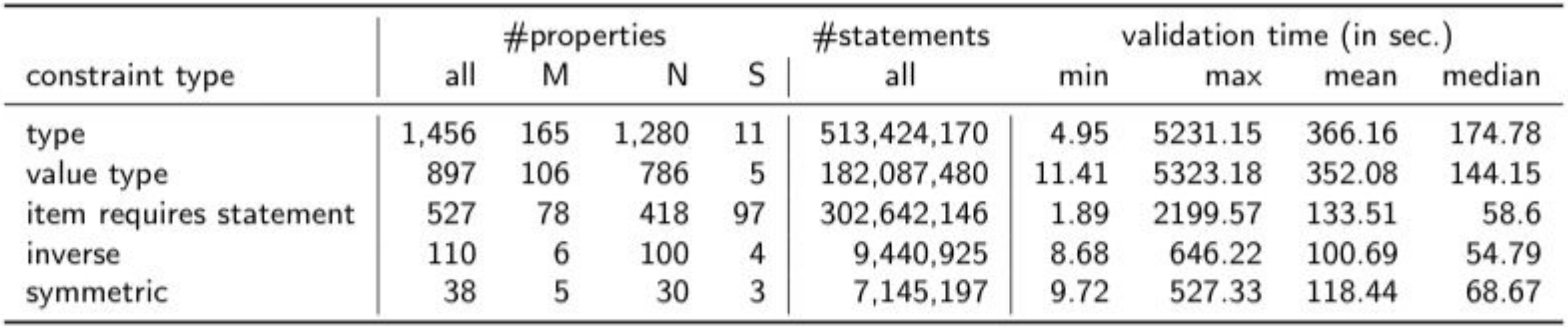
- **Q7: Are constraint violations getting fixed?**
  - Yes. By analyzing the deleted statements, many included deleted constraint violations. E.g.,
    - 30% type constraint (mandatory), 15 % (normal), 40% (suggestion)
    - 12% value type constraint (mandatory), 22% (normal), 59% (suggestion)



constraint	mandatory	normal	suggestion
type	763k/2.31M (33.04%)	5.3M/34.87M (15.21%)	920/2.29k (40.12%)
value type	25.4k/211k (12.03%)	198k/8.99M (22.06%)	235/397 (59.19%)
IRS	4.67k/1.28M (0.36%)	192k/4.85M (3.97%)	190k/6.01M (3.17%)
inverse	37/345 (10.72%)	177k/534k (33.13%)	11.7k/160k (7.27%)
symmetric	19/307 (6.19%)	7.52M/10.85M (69.37%)	5.05k/37.5k (13.47%)

How long did it take?

- There are more than 8000 properties, each with different constraints.
- Analysis covered only wikibase item-based properties.
- Median of 2 min per constraint, avg of 5 min.
- Time does not include importing wikidata, generation of filtered files.



	#properties				#statements	validation time (in sec.)			
constraint type	all	M	N	S	all	min	max	mean	median
type	1,456	165	1,280	11	513,424,170	4.95	5231.15	366.16	174.78
value type	897	106	786	5	182,087,480	11.41	5323.18	352.08	144.15
item requires statement	527	78	418	97	302,642,146	1.89	2199.57	133.51	58.6
inverse	110	6	100	4	9,440,925	8.68	646.22	100.69	54.79
symmetric	38	5	30	3	7,145,197	9.72	527.33	118.44	68.67

Want to see more details?

Check our paper

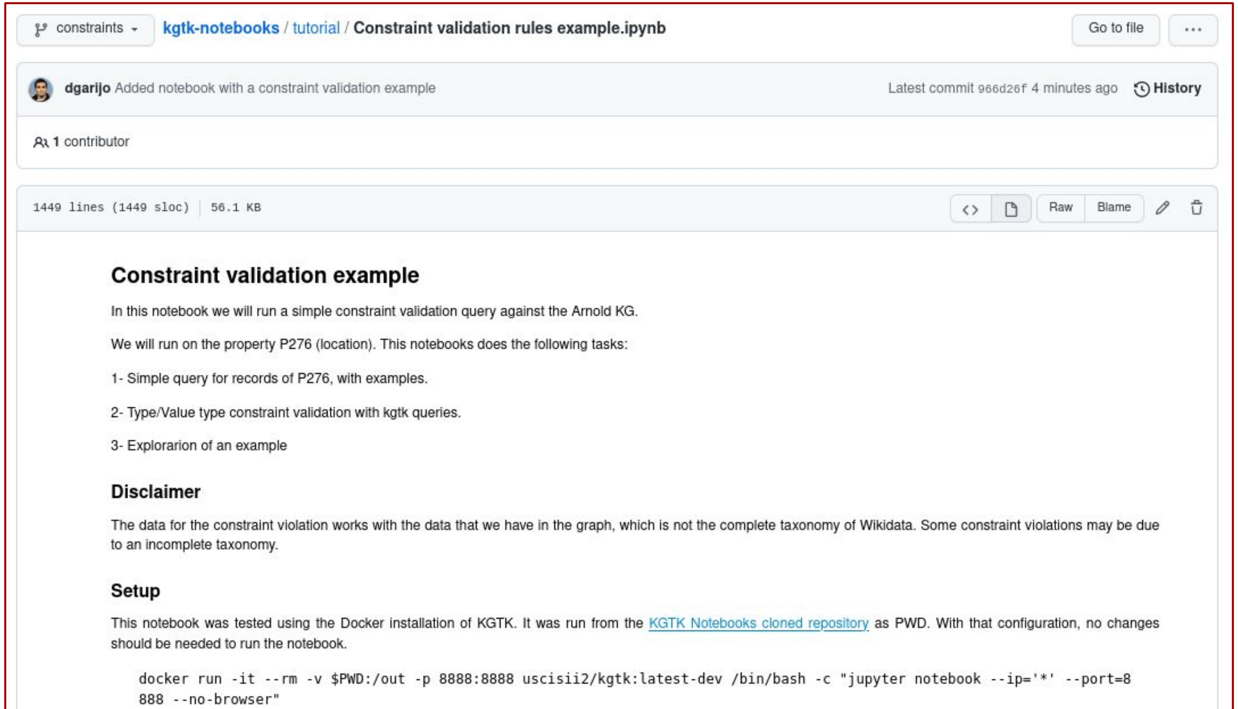
https://doi.org/10.1016/j.websem.2021.100679

(arxiv version: *https://arxiv.org/abs/2107.00156*)

Live example (Notebook)

Let’s try and validate one of the constraints in the KG shown in previous sessions (Arnold Schwarzenegger’s KG): https://github.com/usc-isi-i2/kgtk-notebooks/blob/main/tutorial/07-kg-constraint-validation.ipynb or

https://colab.research.google.com/drive/182ikHCeGDhyPs8WuLQC03c-ET84p0ozM?usp=sharing



kgtk-notebooks / tutorial / Constraint validation rules example.ipynb

dgarijo Added notebook with a constraint validation example

Latest commit 966d26f 4 minutes ago

1 contributor

1449 lines (1449 sloc) | 56.1 KB

Constraint validation example

In this notebook we will run a simple constraint validation query against the Arnold KG.

We will run on the property P276 (location). This notebooks does the following tasks:

1- Simple query for records of P276, with examples.

2- Type/Value type constraint validation with kgtk queries.

3- Explorarion of an example

Disclaimer

The data for the constraint violation works with the data that we have in the graph, which is not the complete taxonomy of Wikidata. Some constraint violations may be due to an incomplete taxonomy.

Setup

This notebook was tested using the Docker installation of KGTK. It was run from the KGTK Notebooks cloned repository as PWD. With that configuration, no changes should be needed to run the notebook.

```
docker run -it --rm -v $PWD:/out -p 8888:8888 uscisii2/kgtk:latest-dev /bin/bash -c "jupyter notebook --ip='*' --port=8
888 --no-browser"
```