Efficient and Effective Search on Wikidata

Improving the QLever Search Engine

Johannes Kalmbach

University of Freiburg

johannes.kalmbach@gmail.com

October 2018

Knowledge Base

- A database of facts represented as triples of (subject, predicate, object)

Neil_Armstrong	is_a	Astronaut .
Neil_Armstrong	country	USA .
	⋮	
Alexander_Gerst	is_a	Astronaut .
Alexander_Gerst	country	Germany .
Alexander_Gerst	nickname	"Astro_Alex" .

- Standard for knowledge bases: RDF (resource description framework)
- SPARQL: standardized query language for RDF

Demo

- Simple SPARQL queries . . .

Wikidata

- General-purpose knowledge base
- Creative-Commons license
- Launched by the Wikimedia Foundation in 2012
- $7.0B$ triples
- Uses abstract ids to identify entities:

Neil Armstrong	country of citizenship	USA
wd:Q1615	wdt:P27	wd:Q30 .

SPARQL on Wikidata

```
PREFIX wd:<http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?dateOfBirth ?astronautLabel WHERE {
    ?astronaut wdt:P106 wd:Q11631 .
    ?astronaut wdt:P27 wd:Q183 .
    ?astronaut wdt:P569 ?dateOfBirth .
    ?astronaut rdfs:label ?astronautLabel .
    FILTER langMatches(lang(?astronautLabel), "en") .
}
ORDER BY DESC(?dateOfBirth)
```

"All German Astronauts and their dates of birth, youngest first "

Demo

- The EntityFinder and the Wikidata Frontend . . .

EntityFinder

- Search on names and aliases/synonyms of Wikidata Entities.
- Interactive prefix search.
- Ranking: Exact matches always before prefix matches.
- Second Step: Sort according to number of sitelinks.
- Future Work: Combine EntityFinder and SPARQL auto completion.

QLever

- SPARQL (+ Text) engine developed at chair of algorithms and data structures, University of Freiburg
- Open source (Apache License): `https://github.com/ad-freiburg/QLever`
- Running instance (with UI): `http://qlever.informatik.uni-freiburg.de/`
- Originally not able to use QLever with Wikidata, RAM usage too high.
- Goal: Reduce QLever's RAM footprint.

Reducing QLever's Memory Footprint

Implemented several mechanisms (details in thesis), general ideas:

- Externalization of data to disk. Important questions:
  - Is the access pattern sequential (disk-friendly)?
  - How frequently is data used?
- Implemented MmapVector (dynamic array like std::vector, allocates memory on (hard) disk).
- Compression of Data.

Vocabulary

Neil_Armstrong is_a Astronaut .
Alexander_Gerst is_a Astronaut .

Id	token
0	Alexander_Gerst
1	Astronaut
2	is_a
3	Neil_Armstrong

- Vocabulary: Set of all of all tokens in KB
- Each token is mapped to an Id
- Internal query processing performed in Id space

Wikidata's Vocabulary Is Big

- ca. 1.2B tokens / 80 GB
- name and description literals in many languages
- fully-qualified URIs:

Neil Armstrong	country of citizenship	USA
wd:Q1615	wdt:P27	wd:Q30 .

is actually

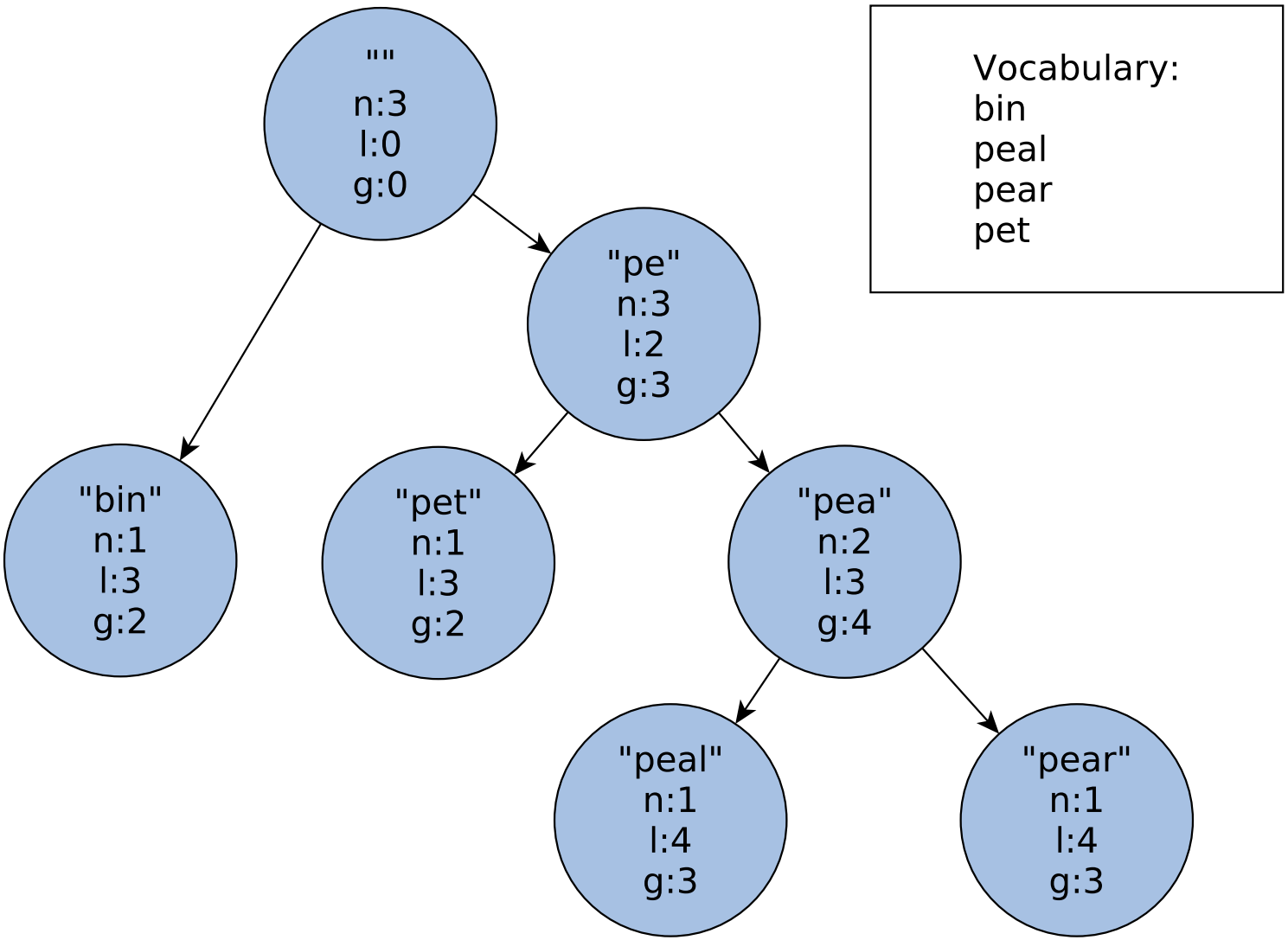
```
<http://www.wikidata.org/entities/Q1615>
<http://www.wikidata.org/prop/direct/P27>
<http://www.wikidata.org/entities/Q30> .
```

Prefix Compression of Vocabulary

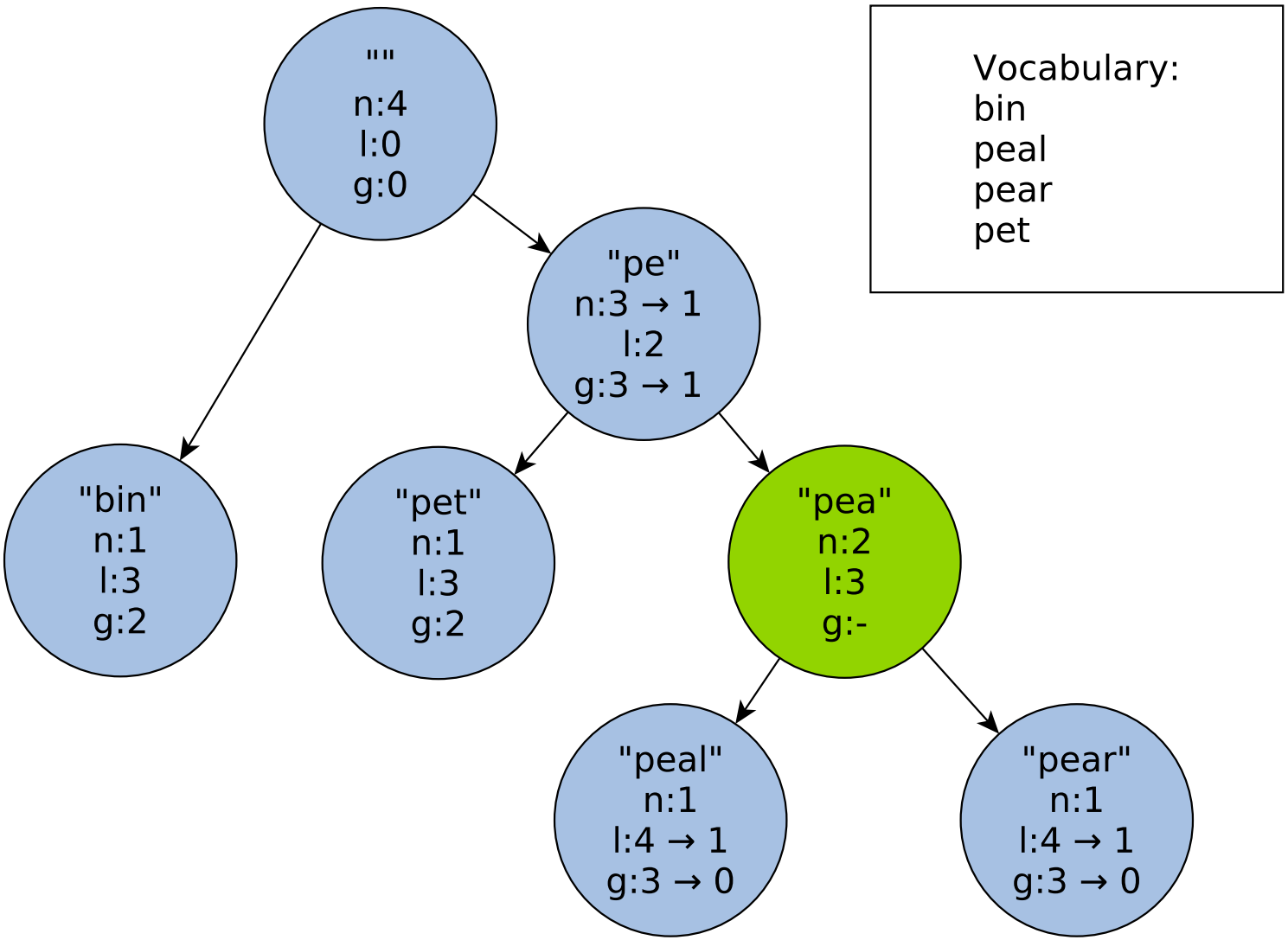
Id	token: uncompressed / compressed	
0	<http://www.wikidata.org/entities/Q42>	☺Q42>
1	<http://www.wikidata.org/entities/Q45>	☺Q45>
2	<http://www.wikidata.org/entities/Q522>	☺Q522>

- Many URIs have a common prefix (e.g. `http://www.wikidata.org/`)
- Find the most common prefixes and compress them.
- Implemented fast greedy heuristic algorithm

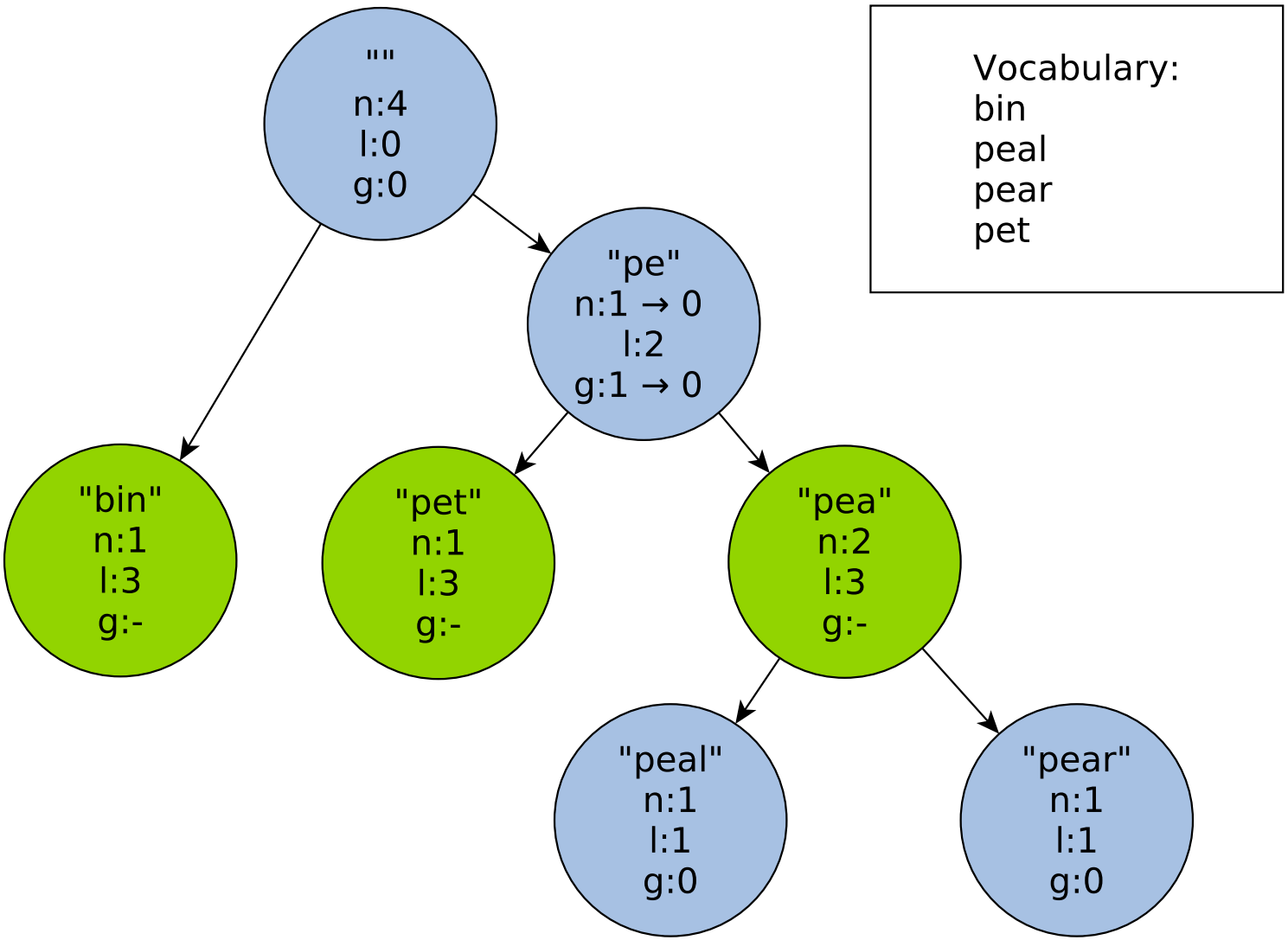
Prefix Compression (assuming 1 byte code length)



Prefix Compression (assuming 1 byte code length)



Prefix Compression (assuming 1 byte code length)



Results

- Prefix compression reduces vocabulary size 45% on Wikidata.
- Total RAM footprint of QLever instance running full Wikidata: 24 GB (originally not fitting on machine with 200GB)

Further Work

- Make auto completion of queries work on Wikidata.
- Systematically evaluate query execution speed of QLever (this project focused on RAM usage).