



SPARQL Hands-on session

Gilles Sérasset (Université Grenoble-Alpes, France)
Max Ionov (University of Cologne, Germany)

SPARQL is a QUERY language

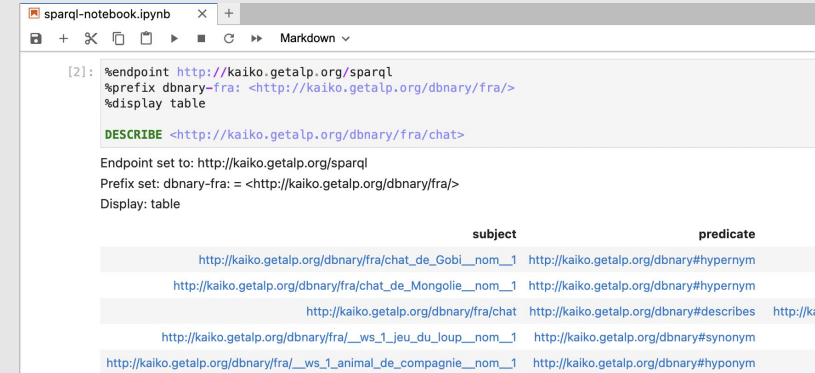
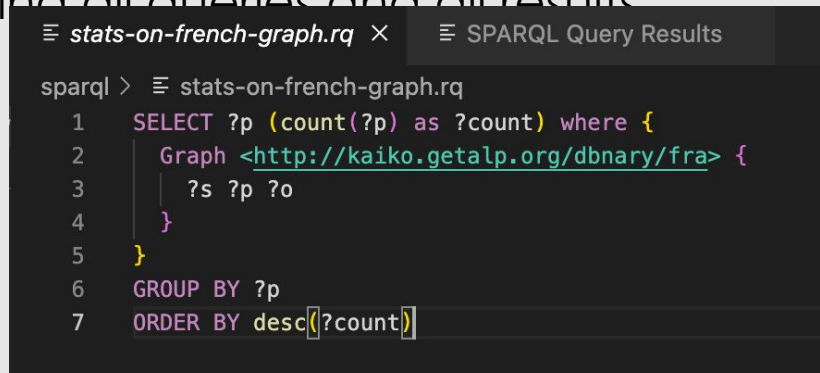
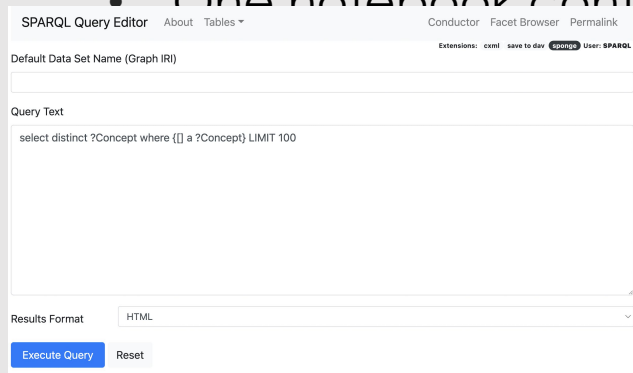
We will query the Dbnary dataset

Tools necessary for this hands-on

Most of the processing will be done on DBnary public “SPARQL endpoint” (i.e. a public server)

What you need on your computer (installed at install-fest)

- A minima:
 - A browser (but you'll lose your former queries, so use an editor and copy-paste)
- A bit better: Visual Studio Code + SPARQL Executor plugin
 - Syntax colouring on queries
 - Choosing the endpoint (^ + shift + E)
 - Execute them on an endpoint (^+ shift + X)
 - Drawback: You will have 1 file per query
- Deluxe setup: Jupyter + SPARQL kernel (packed in a docker container)
 - One notebook containing all queries and all results



Data necessary for this hands-on

Download and decompress :

<https://tinyurl.com/sdllod-dbnary>

Contains 2 folders:

- Jupyter-sparql for people using the jupyter container (docker)
- Sparql folder for the others

Before going further

DBnary:

- short story: Wiktionary data as Linked data
- Extracted from 22 different Wiktionary language editions
- Uses ontolex (+ a very light addition for Translations)

What should I know before querying some dataset ?

Your very first concern should be: “what is available here, and how is it modelled ?”

- Go to <http://kaiko.getalp.org/fct> and try to query something (it's a factetted browser, quite complex to master, but we will use it just to get a grasp of the data and its modelling)
- Never hesitate to draw a graph from the data you see here...

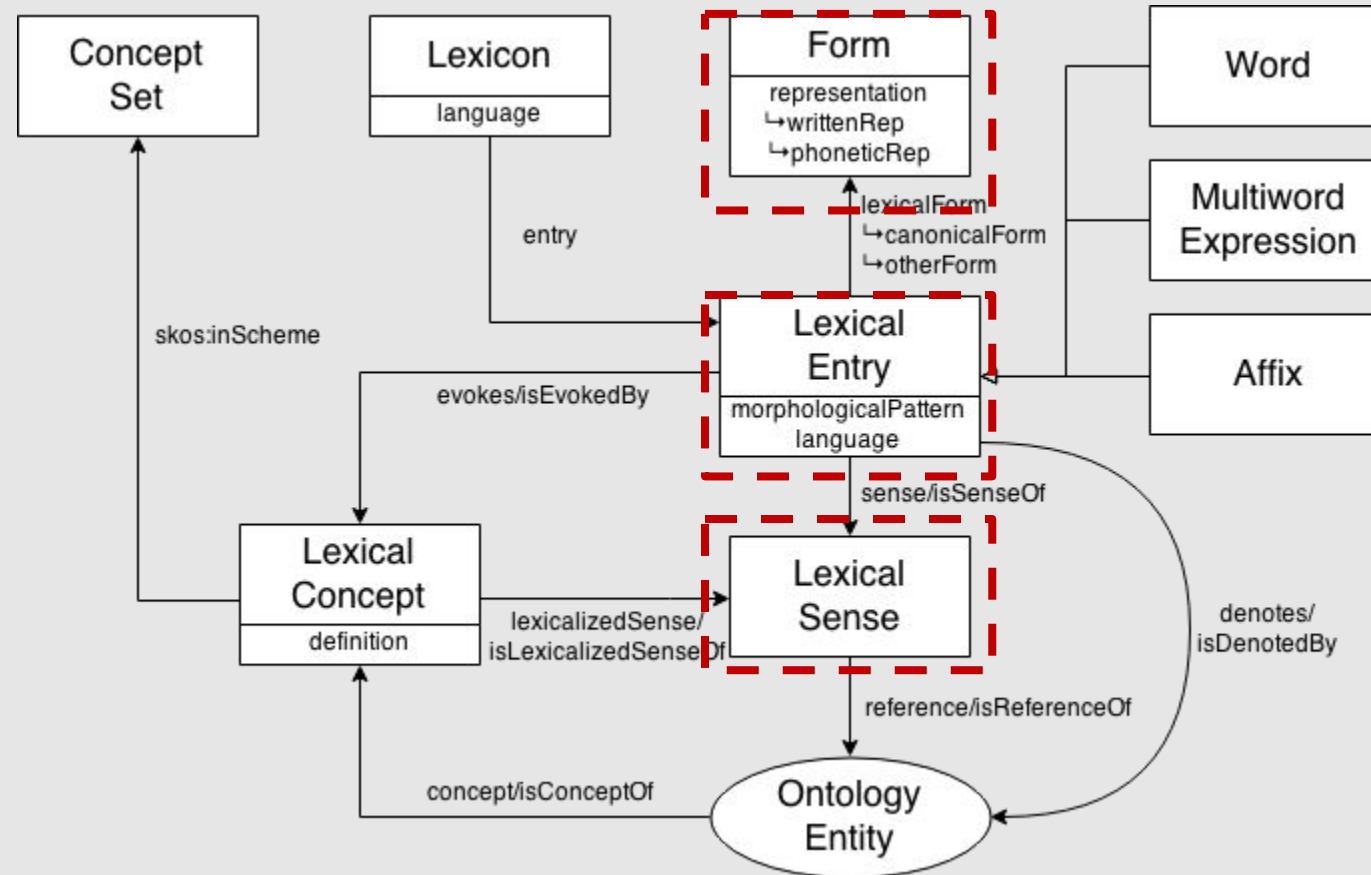
What should I know before querying some dataset ?

Your very first concern should be: “what is available here, and how is it modelled ?”

- Data is available as LLOD:
- <http://kaiko.getalp.org/dbnary/eng/tutorial>

LLD – The Ontolex lemon model

Core of the model



<https://www.w3.org/2016/05/ontolex/>

What should I know before querying some dataset ?

Your very first concern should be: “what is available here, and how is it modelled ?”

```
SELECT *  
WHERE {  
  | ?s ?p ?o .  
}
```

```
SELECT (COUNT(*) as ?count)  
WHERE {  
  | ?s ?p ?o .  
}
```

Not so informative... it's looking for needles in a haystack

What should I know before querying some dataset ?

Your very first concern should be: “what is available here, and how is it modelled ?”

```
select distinct ?Concept
where {
  [] a ?Concept
} LIMIT 200
```

```
select distinct ?p
where {
  ?s ?p ?o
} LIMIT 200
```

A little bit better, but there is more...

What should I know before querying some dataset ?

Remember, data may be organised as several different graphs

```
SELECT DISTINCT ?g
WHERE {
  GRAPH ?g {?s ?p ?o} .
}
```

What should I know before querying some dataset ?

What is available in a particular graph

```
SELECT distinct ?p (COUNT(?s) as ?count)
FROM <http://kaiko.getalp.org/dbnary/fra>
where {
  |    ?s ?p ?o
}
GROUP BY ?p
ORDER BY DESC(?count)
```

A Word about prefixes...

Usually, you should specify the prefix in all queries, but virtuoso (the database software used by DBnary already knows some of them...)

```
PREFIX ontollex: <http://www.w3.org/ns/lemon/ontollex#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX lexvo: <http://lexvo.org/id/iso639-3/>
PREFIX lime: <http://www.w3.org/ns/lemon/lime#>
PREFIX dbnary: <http://kaiko.getalp.org/dbnary#>
PREFIX dbnary-fra: <http://kaiko.getalp.org/dbnary/fra/>
PREFIX lexinfo: <http://www.lexinfo.net/ontology/2.0/lexinfo#>
PREFIX olia: <http://purl.org/olia/olia.owl#>
```

1. Get Lexical Entries with canonical form “chat”

This will not work... do you know why ?

```
PREFIX ontollex: <http://www.w3.org/ns/lemon/ontollex#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX lexvo: <http://lexvo.org/id/iso639-3/>
PREFIX lime: <http://www.w3.org/ns/lemon/lime#>

SELECT ?le
WHERE {
    ?le a ontollex:LexicalEntry ;
        rdfs:label "chat" ;
}
```

1. Get Lexical Entries with canonical form “chat”

You know the language :

- You need to tell the language of the label (or canonical form)...

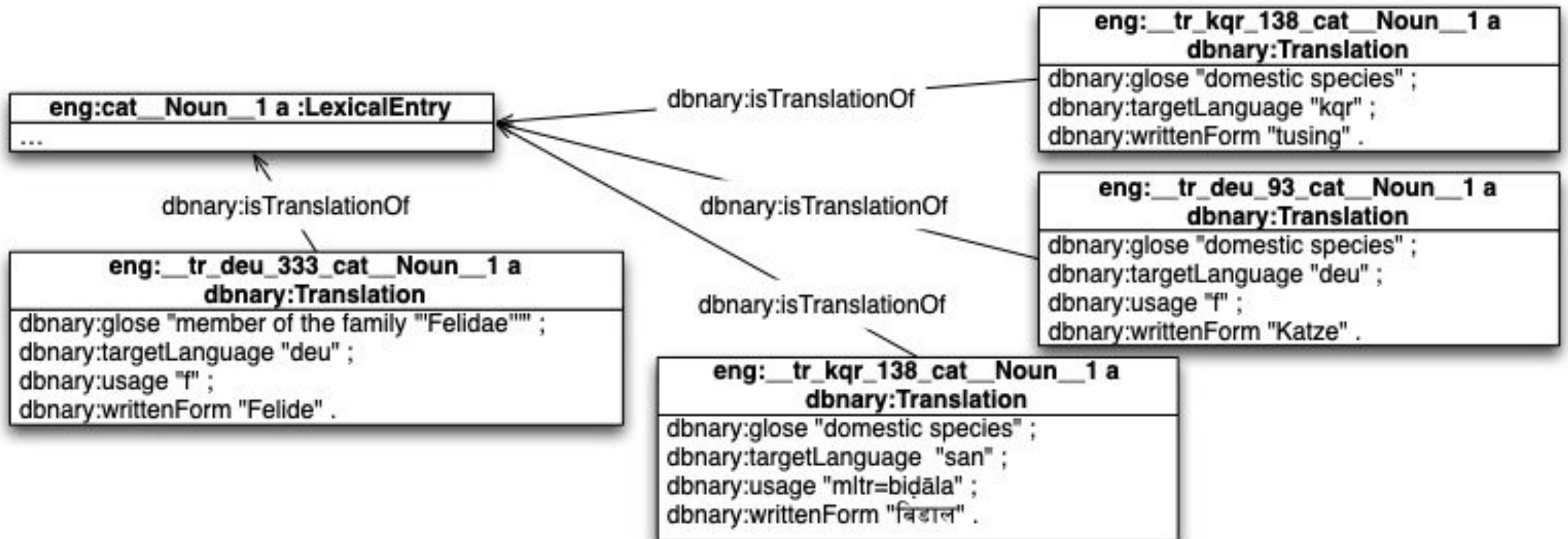
You want to query all languages :

- (hint: you should use a FILTER clause...)

You'll get some strange entries: i.e. the one coming from “Exolexica”

- (hint: you can filter them out using lime metadata)

1. Get all translations from the Lexical Entry you retrieved



Exercises

1. Count number of entries in each dictionary
2. Count number of translations to Quechua
3. Get lexical entries with the most translations in the Latin dictionary
4. Get all the prepositions, hint: 2 steps
 - a. Find a preposition in your language of choice
 - b. Find all entries with the same lexinfo:partOfSpeech property
5. Choose one of the following:
 - a. Compare two languages of your choice in their frequencies of nouns vs. verbs in the dictionary
 - b. Compare two languages of your choice in the frequency of a grammatical category values, e.g. distribution of grammatical genders
 - c. Is it possible to translate a French Entry to Quechua by pivoting through another language ?

1. Count number of translations to Quechua
2. Is it possible to translate a French Entry to Quechua by pivoting through another language ?