Université de Paris-Saclay
Web of data

Project report

An Intelligent Web Application with Semantic Technology

Student

Lilia IZRI

Teachers Mrs. Fatiha SAÏS & Mr. Joe RAAD.

Master 2 - Data Science
Year: 2022 / 2023

Contents

1 Introduction

The project consists in the creation of a web application that allows to answer a complex query using semantic web technologies. The domain chosen for this is *Youtube* or more specifically "Youtubers". The implementation details and techniques chosen will be detailed in this report.

2 Motivation

Youtube communities are facing a continuous growth especially in the last years. It can be interesting to answer some questions that we can ask ourselves about these communities and content creators in general. Some of these questions may be difficult to answer based on the classic web alone, so for this project we will rely on semantic technologies to answer a question chosen among all. The question I chose is:

> Who are the Youtbers born whithin 600km of the french-speaker Youtuber with the biggest ratio Number of views accumulated the last 30 days / Number of videos posted the last 30 days ?

One may wonder why this question ? Well, because a lot a great french speaker Youtubers are born in other than France (Belgium, Switzerland, ...etc). And the number of subscribers is not always significant regarding the audience that a youtuber and his channel can have. Also, I chose a ratio on the last 30 days to take into account only more or less active Youtubers (some Youtube channels accumulate a lot of views but are almost no longer active or even suspended).

In addition of the names of the Youtubers, we will also get their ratio (mentionned previously) and a global ratio (where we consider everything since the beginning of the channel, not only the last 30days) to make some comparisons.

3 Quick Presentation of the archive's project

The archive of the project is composed of 3 folders that contain every file or things like queries mentionned in this report:

◦ Data: Contains the tabular file and RDF files to import in *GraphDB*.

◦ Queries: Contains the query to make links with the external knowledge graph and the query to answer to question.

◦ Application: Contains the Web application.

4 Implementation details

4.1 Tabular data

I used a tabular file that I created myself **"ViewsYoutubers.csv"**. For that, I collected information of 40 French-speaking or not Youtubers, by privileging the more or less known and active Youtubers who are specialized in various domains (humor, gaming, beauty, ... etc).

The file is composed of 5 columns:

- Person: The name of the youtuber.
- Number of views last 30 days: The number of views accumulated these last 30 days.
- Number of videos last 30 days: The number of videos posted these last 30 days.
- Youtube Channel: The name of the main youtube channel of the person (because the youtuber can have several channels).

- Language: The language used by the youtuber.

To know the number of views accumulated the last 30 days I used this site: youtubers.me that allowed me to check for each youtuber the statistics of its youtube channel.
For the rest, I checked directly on youtube.com the missing information.

Remark
◦ The file is the result of statistics gathered December 20^{th}, 2022.

4.2 Creation of ontology

Using the free ontology editor `Protégé`, I created the ontology and the domain to describe my csv file.

The namespace for naming classes and properties is: http://example.fr/upsaclay/wod/izri/. And the prefix for this is: **"wodiz"**.

To represent each of the Youtubers we've got in the tabular file, a class named "Youtuber" was created. And, each youtuber is linked to his information (litterals) with properties:

- lastVideos: The number of the videos posted in the last 30 days.
- lastViews: The number of views accumulated in the last 30 days.
- channel: The name of the main youtube channel.
- language: The language of the youtuber.

We export the resultant ontology into a Turtle file **"youtubers.ttl"**.

```
#################################################################
#    Data properties
#################################################################

###  http://example.fr/upsaclay/wod/izri/channel
wodiz:channel rdf:type owl:DatatypeProperty ;
              rdfs:label "Youtube channel's name"@en .


###  http://example.fr/upsaclay/wod/izri/lastVideos
wodiz:lastVideos rdf:type owl:DatatypeProperty ;
                 rdfs:label "Number of videos in the last 30days"@en .


###  http://example.fr/upsaclay/wod/izri/lastViews
wodiz:lastViews rdf:type owl:DatatypeProperty ;
                rdfs:label "Number of views in the last 30days"@en .


###  http://example.fr/upsaclay/wod/izri/language
wodiz:language rdf:type owl:DatatypeProperty ;
               rdfs:label "Language of the channel"@en .


#################################################################
#    Classes
#################################################################

###  http://example.fr/upsaclay/wod/izri/Youtuber
wodiz:Youtuber rdf:type owl:Class ;
               rdfs:label "Youtuber"@en .
```

Figure 1: Illustration of the file "youtubers.ttl"

4.3 Creation of graph data

For the purpose of this project, a new repository has been created to represent our data in `graphDB`. The repository is called **"project-WOD-IZRI-Youtubers"**.

Setup ▷ *Repositories* ▷ *Create new repository*

After the creation, we will then import the file created previously "youtubers.ttl" in it.

Import ▷ *Upload RDF files*

Plus, in order to be able to import RDF data later, we need to activate the autocomplete index option.

Setup ▷ *Autocomplete*

4.4 Conversion of the tabular file

We use `OntoText Refine` to convert the tabular dataset to RDF file that we will then import in `graphDB` using the same process than previously. We need to precise in `OntoText Refine`, in "Setup", the ID of our repository which is in our case "project-WOD-IZRI-Youtubers".

Then, we need to import the tabular file "ViewsYoutubers.csv":

Projets ▷ *Create project* ▷ *Select files*

After that, we need to edit RDF mappings so we can precise the function of each column of the file. In the picture below we can see the mappings in details:

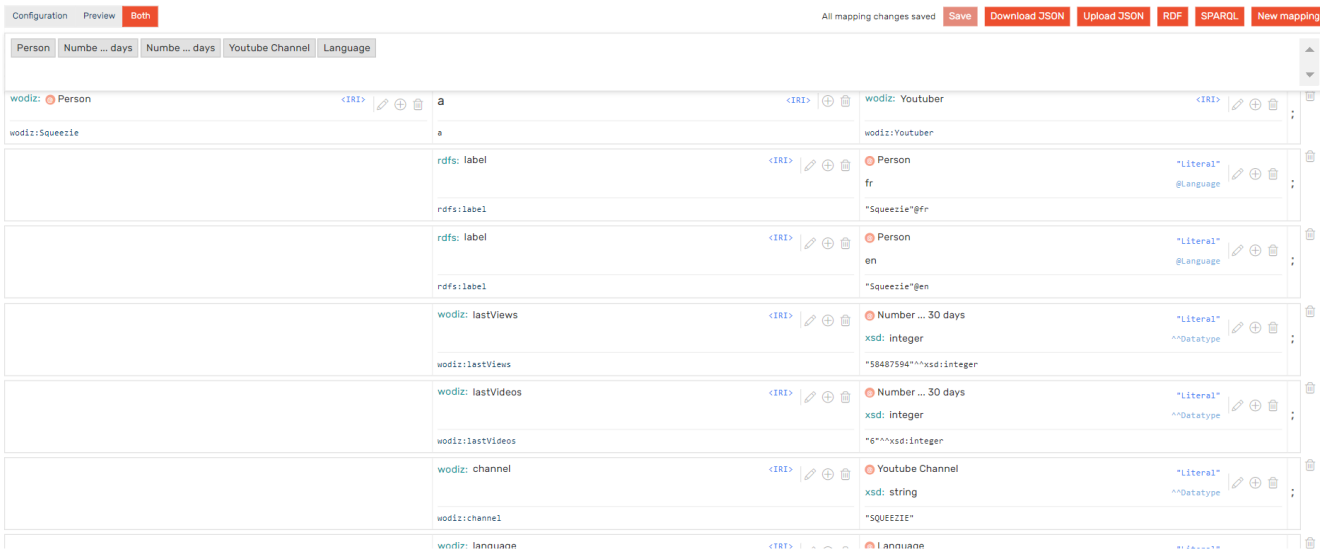

Figure 2: Illustration of the RDF mappings

Finally, we need to export into RDF and import the file **"result-triples-youtubers.ttl"** into *GraphDB* just like we have done it before.

4.5 Linking data

To answer our query we need to link entities of our graph data to external knowledg graph. *Wikidata* knowledge graph was used in this case to get the place of birth of each Youtuber (name of the place and its coordinates).

In addition of that, we also get the number of total videos the Youtuber posted and the number of total views since the beginning of its channel, so we can compute a **global ratio** $= \frac{number\ total\ views}{number\ total\ videos}$ to compare it with the **ratio of the last 30 days** $= \frac{number\ views\ last\ 30days}{number\ videos\ last\ 30days}$ of our query.

◦ The insert query below can be found in the file **InsertQuery.rq.ttl**.

```
INSERT {
    # Create new links
    ?youtuber owl:sameAs ?wdYoutuber ;
              wodiz:allViews ?wdChannelAllViews ;
              wodiz:allVideos ?wdChannelAllVideos ;
              wodiz:birthPlace ?wdYoutuberCityLabel ;
              wodiz:birthPlaceCoordinates ?wdYoutuberCityLocation .
}
WHERE
{
    # Querying wikidata: Get youtubers missing information
    SERVICE <https://query.wikidata.org/sparql> {
        # A youtuber is a type of human, has a label, a place of birth and youtube channel Ids
        ?wdYoutuber wdt:P31 wd:Q5 ;
                    rdfs:label ?wdYoutuberLabel ;
                    wdt:P19 ?wdYoutuberCity ;
                    wdt:P2397 ?wdChannelYtID .

        ?wdYoutuberCity wdt:P625 ?wdYoutuberCityLocation ;
                        wdt:P1448 ?wdYoutuberCityLabel .

        # Get infos of YT channel corresponding to the Id
        ?ytChannel wds:P2397 ?wdChannelYtID ;
                   wdq:P1810 ?wdChannelYtName ;
                   wdq:P3740 ?wdChannelAllVideos ;
                   wdq:P5436 ?wdChannelAllViews .
        FILTER ( LANG(?wdYoutuberLabel) = 'en' || LANG(?wdYoutuberLabel) = 'fr')

    }
    # Youtuber on our local graphdb
    ?youtuber rdf:type wodiz:Youtuber ;
              rdfs:label ?youtuberLabel ;
              wodiz:channel ?channelName .

    # Filter such that the name of the youtuber and its yt channel are the same than the ones on wikidata
    FILTER (STR(?wdChannelYtName) = STR(?channelName)
         && STR(?wdYoutuberLabel) = STR(?youtuberLabel))
}
```

Figure 3: Query to create links with Wikidata

4.6 Query to answer the question

In order to answer the question, we create the query that we can see below. This query is composed of a sub-query to retrieve first the french-speaker Youtuber that has the biggest ratio of number of views/ number of videos for the last 30 days (especially we need the coordinates of its birth place "?cityYTMax-Coordinates"). Then, we filter all the youtubers such as the distance between the coordinates of their birth place ("?cityCoordinates") and the ones of the 1st youtuber we retrieved is less than 600km. To compute the distance we use *GeoSparql* function **distance**.

The query returns the name of the youtubers, their ratio (taking into account only values of the last 30 days), their global ratio (taking into account values since his beginning) and their language. The youtubers are sorted according to their ratio and we take as a limit 10 youtubers.

◦ The query below can be found in the file **FinalQuery.rq.ttl**

Remarks

- The first Youtuber returned by the query is the Youtuber with the biggest ratio himself since I haven't exculeded him.

- All the variables with "Max" in their name, refer to variables related to the french-speaker youtuber with the biggest ratio.

```
SELECT DISTINCT ?youtuberName ?ratio ?ratioGlobal ?language
WHERE{
    # Get all the youtubers we got in our graphdb and their info
    ?youtuber rdf:type wodiz:Youtuber ; rdfs:label ?youtuberLabel ; wodiz:language ?language ; wodiz:lastViews ?nbLastViews ;
              wodiz:lastVideos ?nbLastVideos ; wodiz:birthPlaceCoordinates ?cityCoordinates ; wodiz:allViews ?allViews ; wodiz:allVideos ?allVideos .
    # Computes ratio of the last 30 days of views/videos
    BIND(round(?nbLastViews/?nbLastVideos) AS ?ratio)
    # Computes global ratio of views/videos
    BIND(round(?allViews/?allVideos) AS ?ratioGlobal)

    # SubQuery to get THE youtuber FR that has the biggest ratio nb views/nb videos
    {
        SELECT ?youtuberMax ?ratioMax ?ratioGlobalMax ?cityYTMaxCoordinates
        WHERE {
            ?youtuberMax rdf:type wodiz:Youtuber ; wodiz:language ?languageYTMax ;
                         wodiz:lastViews ?nbLastViewsYTMax ; wodiz:lastVideos ?nbLastVideosYTMax ;
                         wodiz:birthPlaceCoordinates ?cityYTMaxCoordinates ; wodiz:allViews ?allViewsYTMax ; wodiz:allVideos ?allVideosYTMax .
            # Computes ratio of the last 30 days of views/videos
            BIND(round(?nbLastViewsYTMax/?nbLastVideosYTMax) AS ?ratioMax)
            # Computes global ratio of views/videos
            BIND(round(?allViewsYTMax/?allVideosYTMax) AS ?ratioGlobalMax)
            # Filter such that we get French-speaking youtubers
            FILTER (?languageYTMax = 'FR')
        }
        # Order and limit so we get only the youtuber with the biggest ratio
        ORDER BY DESC(?ratioMax)
        LIMIT 1
    }
    # Compute the distance between the place of birth of the youtuber FR with the biggest ratio
    #   and all other youtubers
    BIND(geof:distance(?cityYTMaxCoordinates, ?cityCoordinates)*100 AS ?distance)
    BIND(STR(?youtuberLabel) AS ?youtuberName)
    # We keep only the ones whose distance is less than X km
    FILTER (?distance <= 600)
}
ORDER BY DESC(?ratio) DESC (?ratioGlobal)
LIMIT 10
```

Figure 4: Query to answer the question

4.7 Web application

To illustrate the result of the query, I used the HTML and JavaScript codes shared on GitHub modified a bit so we can plot a graph bar showing both the ratio and global ratio of each youtuber.

5 Result analysis

The result of the query are illustrated in the graph below.

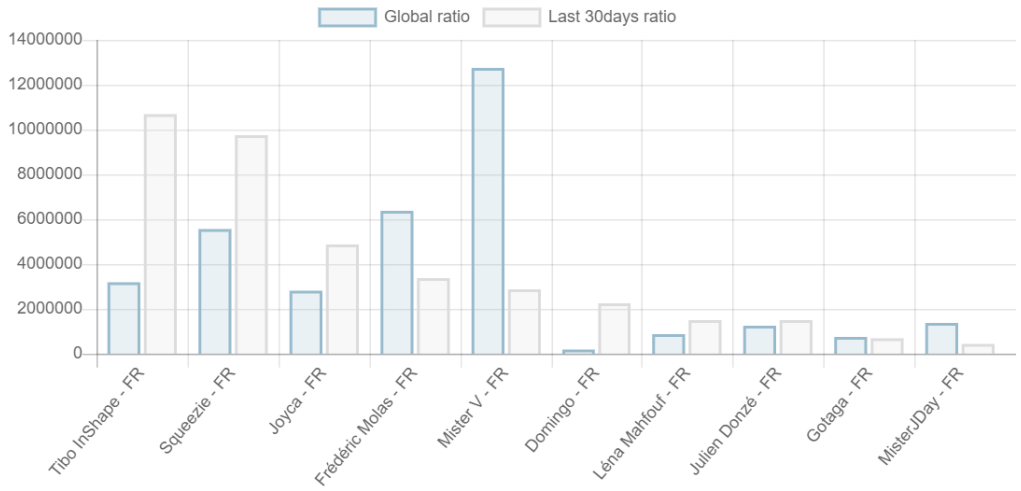

Figure 5: Graph bar illustrating the result of the query

	youtuberName	birthPlace
1	"Tibo InShape"	"Toulouse"@fr
2	"Squeezie"	"Vitry-sur-Seine"@fr
3	"Joyca"	"Grenoble"@fr
4	"Frédéric Molas"	"Perpignan"@fr
5	"Mister V"	"Grenoble"@fr
6	"Domingo"	"Paris"@fr
7	"Léna Mahfouf"	"Asnières-sur-Seine"@fr
8	"Julien Donzé"	"Genève"@fr
9	"Gotaga"	"Mantes-la-Jolie"@fr
10	"MisterJDay"	"Haute-Savoie"@fr

Figure 6: Birth place of each of the previous youtuber

Not really surprising: all the Youtubers are French speakers. Plus, in figure 6, we can see that all of them are born in France except "Julien Donzé" (I modified slightly the query to see the birth place name thanks to *Graphdb*, the query can be found in "AdditionalQueryToSeeBirthPlace.rq.ttl" file).

However, what is a bit surprising is that some Youtubers have a much bigger global ratio than all the others. I'm thinking in particular of "Mister V". This can be due to the fact that his channel is a bit oldest than the others (google research), but still ! The result is quite impressive in my opinion.

What's also interesting is, playing a bit with the query (like for example increasing a bit the distance or choosing the global ratio rather than the ratio of last 30 days to select the youtuber of reference) can give us completely different results : different youtubers from different countries (German, UK, ...etc).

6 Conclusion

As a conclusion, a lot of improvements can be done like for example creating classes to represent the place of birth and Youtube channels related to youtubers, so we can get more information for each instance and make more complexe queries and deeper analysis on Youtube communities (taking into account more statistics). But this project was an opportunity to discover different tools and manipulate them in order to create a functional Semantic Web Application.