UNIVERSITY OF JEAN MONNET

LABORATOIRE HUBERT CURIEN

CONNECTED INTELLIGENCE TEAM

Object extraction techniques and visual image search with Semantic web techniques

Submitted by:
Aninda MAULIK,
CPS2

Supervisor:
Prof. Pierre MARET
Dennis DIEFENBACH

June 26, 2020

Abstract

This internship is about exploration of object detection and extraction techniques with a state of the art computer vision api. Thereafter, we design a semantic web model for the extracted data and finally implement a visual image search engine through Qanswer.

1. Introduction

Users' experience is an important factor for the success of a given application. Thus, the front-end of Qanswer which highly impacts the users' experience, is an important part for a image base query system. Qanswer, well handles the translation from a natural language question to correct SPARQL queries.SPARQL has emerged as the standard RDF query language. An RDF query language is able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. RDF data model is based on the idea of making statements about web resources in expressions of the form subject–predicate–object, known as triples. The subject denotes the resource, and the predicate denotes traits or aspects of the resource, and expresses a relationship between the subject and the object. We generate the rdf data model from a csv file, in which each line includes information for a triplet and all its components. The csv file is generated by consolidating the information and details about the required images. We primarily get the information of the required images by running the state-of-the-art, real-time object detection system; YOLO(You Look Only Once).

2. Presentation of the research problem

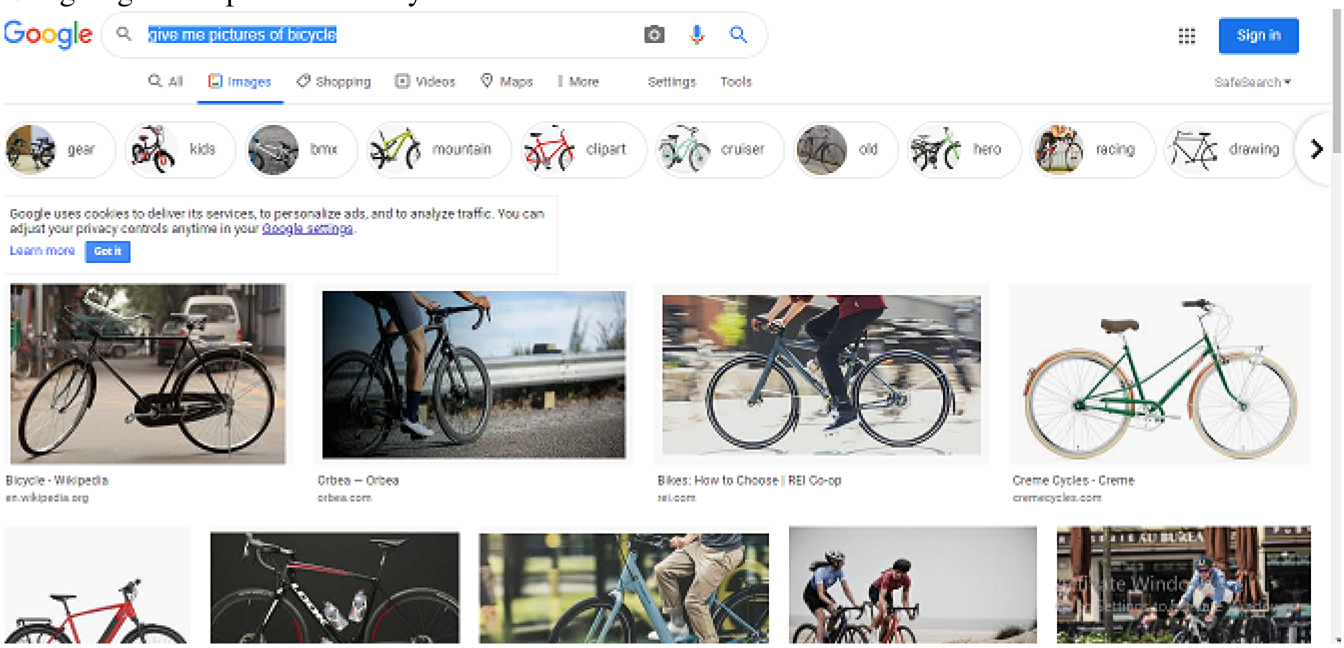
There is no way to use the knowledge generated by computer vision techniques, to query image bases.The research community has made a lot of efforts to use the computer vision techniques for extracting knowledge from images. On the other side, not much attention has been paid to the implementation of methods for making this knowledge available.We hope to change this trend by presenting Semantic Web techniques for querying the knowledge made available by computer vision. My work focuses on bridging the two disciplines here.

3. State of the art

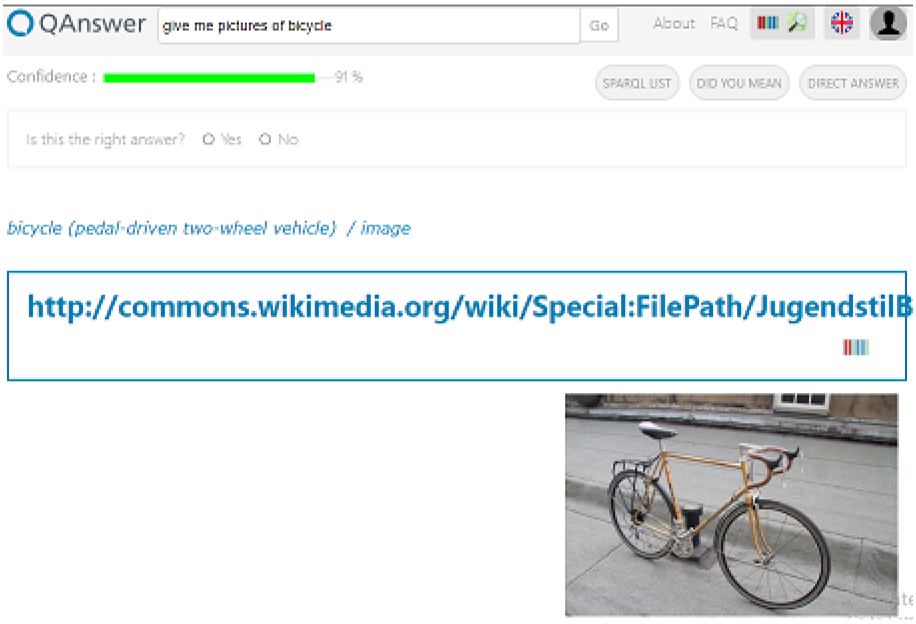
Let's try to understand how does a Google image search engine work. There are two ways in this. 1) Indexing the text surrounding any image and matching it with the given query. If query matches, the corresponding linked image is retrieved. 2) Going ahead, in addition to linking text surrounding an image to that image, we can link all visually similar images to that image with the same text. e.g. consider an image/photo Img1 on any site with it's surrounding text Txt1. And lets say there are some other images Img2,Img3,Img4 etc. which may or may not have text but their (visual) content matches with the contents of Img1. Now for given query, if Txt1 is a good match, the retrieved result can contain Img1 in addition to Img2, Img3, Img4, etc. This is just one factor in addition to many other like matching query with text, features used to represent an image, page-rank of page containing an image, relevance, indexed database size available with search engine, etc. Huge indexed database availability with Google is one of the reasons why Google can give you best search results. Hence, when we ask for pictures of bicycle, we get many photos. However, when we try to search for images of bicycles on the left part of the photo, we get all the bicycle images which may or may not contain a photo of a left hand sided bicycle. Thus, we can conclude that google doesn't index their image base, based on object position. Qanswer,on the other hand, converts the natural language into triples and use the best ranked SPARQL query to query structured data sources. Hence, when we try to query/search for images of bicycles, we don't get much results; because the query is being made over structured data sources only and the existence of structured data is limited. But, the results are reliable. Now, if we try to search for pictures of bicycle on the left of the image, over Qanswer, then we're also going to get accurate results. The details are explained in the upcoming section. We would just like to say that, as discussed before, it would all start from converting the natural language text into triples; the triples would be then matched with an RDF file embedded within Qanswer. Thereafter, SparQL queries would be generated to make queries over structured data sources based on the RDF file information. All our efforts goes to the creation of this one RDF file which makes the difference. This RDF file gives the ability to Qanswer, to do object position detection in an image.This ability makes Qanswer, the best question answering system.

4. Contribution/Proposal description and implementation

In this section, we would first try to compare the results of Qanswer with google's. Thereafter, we would make an attempt to explain the key element that has been introduced, that allows us to detect object positions in an image.
Google: give me pictures of bicycle

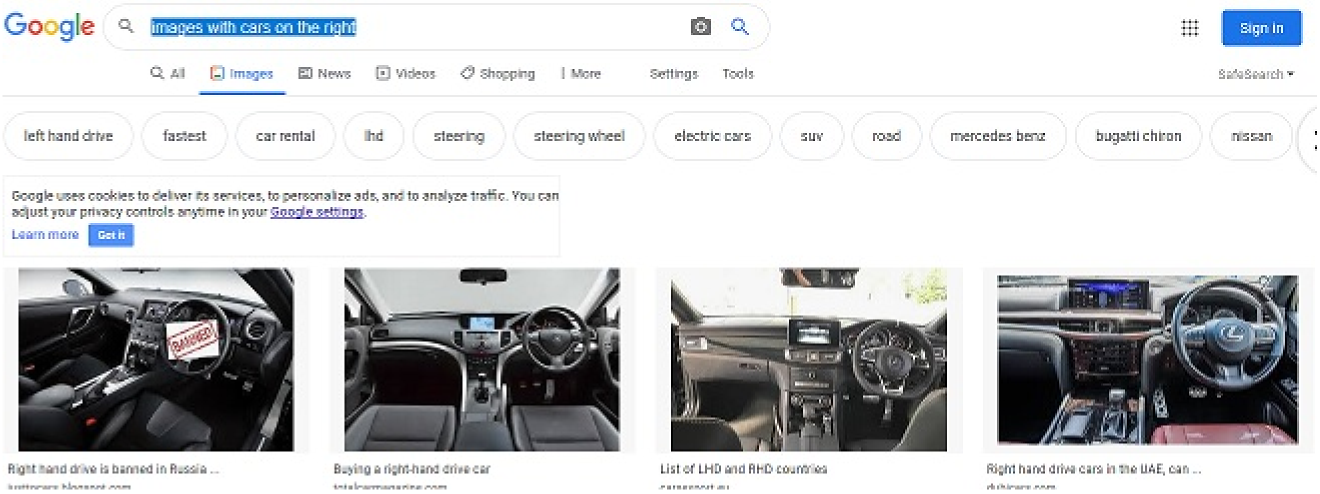


From the above, we can see that, there are just too many results. Now, we would try do the same with Qanswer.



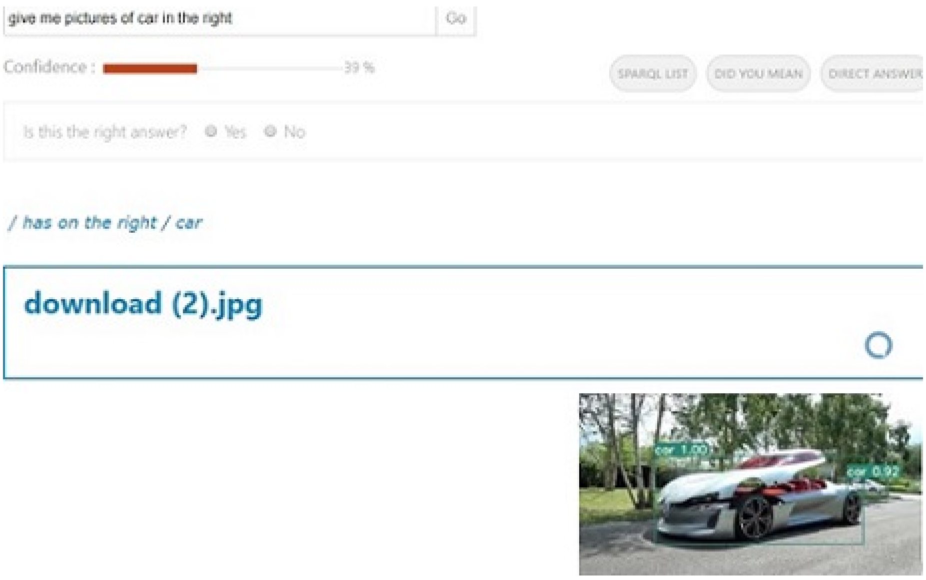
We see that there's just one image that showed up from Wikidata. If, we look closely, we will see that the structured data source from where the image appears is present.

Let's google: images with cars on the right



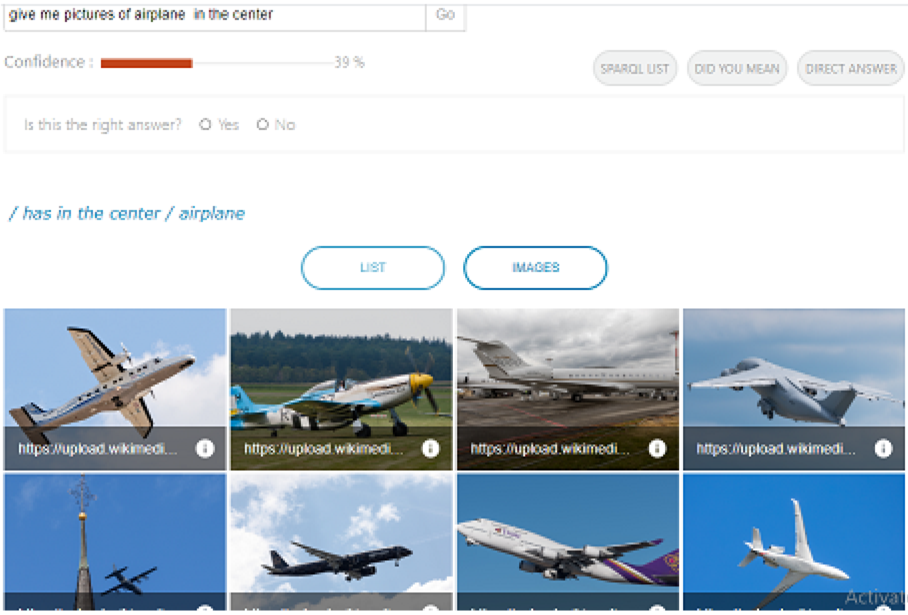
The resultant photos we get,from google, shows the inside of car.

Let's do the same search on Qanswer: give me pictures of car on the right



We began our work, by downloading a very small set of images. Now, from this very small set, Qanswer was able to find successfully that one picture which has a car on the right. On the first glance, it seems that there’s a car in the center but if we look closely then we can see that there’s indeed another car on the right side of the image.
Since, the initial work was based on a very small random dataset, hence we decided to work on a significantly larger structured dataset. The first option that seemed fair at the time was Wikimedia which contains more than a million images. But, we wanted to broaden our area of work by including images and human hand annotated structured data.Hence we chose to work on a particular wikimedia api. The api is: https://commons.wikimedia.org/w/api.php?action=querylist=searchsrsearch=haswbstatement:P180=Q7378 srnamespace=6format=json ..With this api, we are querying for images with QID: Q7378, which is the QID of an elephant. If, we just copy and paste this api in our browser, we’ll get a json containing the information about 10 images of elephant. In the api, mentioned we can add a parameter srlimit=500, and if we paste this new url: https://commons.wikimedia.org/w/api.php?action=querylist=searchsrsearch=haswbstatement:P180=Q7378 srnamespace=6srlimit=500format=json in our browser then we can get json data of 500 images, given that it is available in the api. Please note that the new parameter introduced,srlimit, has a default value of 10 and hence we pasted the url without this parameter we got json data for 10 images. There are many such parameters like this and the details of these parameter can be found in the documentation under this link: https://www.mediawiki.org/wiki/API:Search . Let’s talk about the QID: Q7378, which is the QID of an elephant, for a moment. We get the json data of 227 images of elephant in one go by using the api containing the additional parameter,srlimit=500, mentioned above. Now, a different scenario arises, where the json data availability is more than 500 like in the case of QID:Q1420-car, then we can use a while loop and iterate over the parameter sroffset.
Let’s try to query on the Qanswer using the api, that has been discussed above for airplane-Q197. Here’s a snapshot of the first 8 images.

Qanswer: give me pictures of airplane in the center



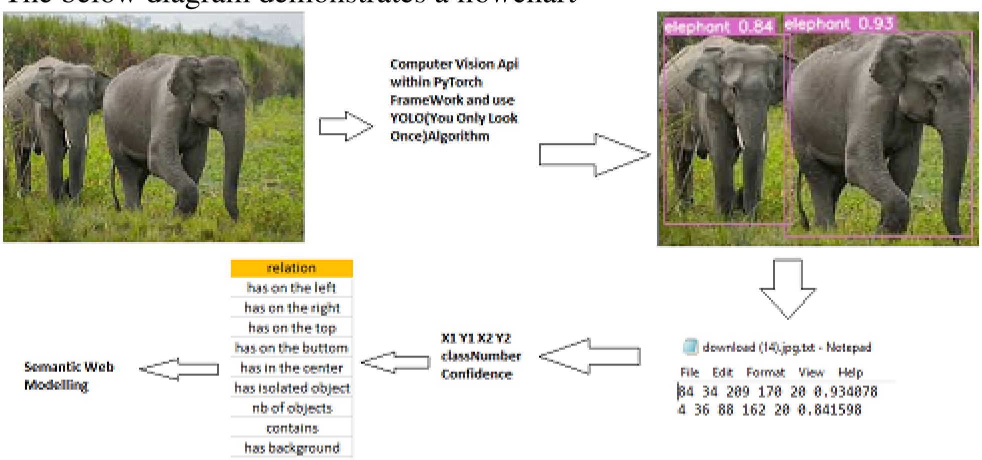
Now, the objective of my work can be subdivided into 3 parts. They are:

- Implementation of an Algorithm for object extraction.
- Design of a semantic web modelling for extracted data.
- Implementation of a visual image search engine through Qanswer.

Let's discuss each part separately.

4.1. Implementation of an Algorithm for object extraction.

The below diagram demonstrates a flowchart



We initiate our work over a desired image base which could random downloaded set, structured data like wikimedia or specialised wikimedia api. After choosing a set of images, we select the state-of-the-art computer vision api(Yolo-You Only Look Once) with a pre-trained model,within PyTorch Framework, to detect objects in our image. Now, our favourite computer vision api YOLO, is able to identify 80 classes of objects, here's a list:

```
0. person-Q215627
1.bicycle-Q11442
2.car- Q1420
3.motorbike-Q34493
4.aeroplane-Q197
5.bus-Q5638
6.train-Q870
7.truck-Q43193
8.boat-Q35872
9.traffic light-Q8004
10.fire hydrant-Q634299
11.stop sign-Q250429
12.parking meter-Q953960
13.bench-Q204776
14.bird-Q5113
15.cat-Q4167836
16.dog-Q144
17.horse-Q726
18.sheep-Q7368
19.cow-Q830
20.elephant-Q7378
21.bear-Q30090244
22.zebra-Q32789
23.giraffe-Q862089
24.backpack-Q5843
25.umbrella-Q41607
26.handbag-Q467505
27.tie-Q44416
28.suitcase-Q200814
29.frisbee-Q131689
30.skis-Q172226
31.snowboard-Q178131
32.sports ball-Q63347096
33.kite-Q107061
34.baseball bat-Q809910
35.baseball glove-Q809894
36.skateboard-Q15783
37.surfboard-Q457689
38.tennis racket-Q153362
39.bottle-Q80228
40.wine glass-Q1531435
41.cup-Q81727
42.fork-Q81881
43.knife-Q32489
44.spoon-Q81895
45.bowl-Q153988
46.banana-Q503
47.apple-Q89
48.sandwich-Q28803
49.orange-Q39338
50.broccoli-Q47722
51.carrot-Q81
52.hot dog-Q181055
53.pizza-Q177
54.donut-Q192783
55.cake-Q13276
56.chair-Q15026
57.sofa-Q131514
58.pottedplant-Q203834
59.bed-Q42177
60.diningtable-Q10578291
61.toilet-Q7857
62.tvmonitor-Q289
63.laptop-Q3962
64.mouse-Q7987
65.remote-Q185091
66.keyboard-Q250
67.cell phone-Q17517
68.microwave-Q127956
69.oven-Q36539
70.toaster-Q14890
71.sink-Q140565
72.refrigerator-Q37828
73.book-Q571
74.clock-Q376
75.vase-Q191851
76.scissors-Q40847
77.teddy bear-Q213477
78.hair drier-Q15004
79.toothbrush-Q134205
```

The above list contains the class number, class name and also QID. We can find the same information in this url, without the QID. The link is:
https://github.com/pjreddie/darknet/blob/master/data/coco.names
We introduce a dataset of images to our YOLO program. YOLO gives a corresponding text files, containing the co-ordinates of bounding box(X1,Y1,X2,Y2), class number or class name,confidence percentage with which it detects an object in those images. It also returns all the images along with bounding box marked around every object, which YOLO has identified.
Then we try to use the bounding box co-ordinates to understand the following.

Image	relation	property value
	has on the left	
	has on the right	
	has on the top	
	has on the buttom	
	has in the center	

We would try to explain the algorithm used for each of them.

Algorithm 1 has on the left and right

1: **if** $X - centre \leq 0.3 * X - ImageDimentions$ **then**
2: $\quad hasontheleft \leftarrow object$
3: **else**
4: $\quad$ **if** $X - centre \geq 0.3 * X - ImageDimentions$ **then**
5: $\quad\quad hasontheright \leftarrow object$
6: $\quad$ **end if**
7: **end if**

We would want to explain X-Image Dimentions,Y-Image Dimentions refer to the breadth and length respectively. Moreover, we calculate the X-centre and Y-centre as follows:

Algorithm 2 has on the top and bottom

---

if $Y - centre \leq 0.3 * Y - ImageDimentions$ **then**
  $hasonthetop \leftarrow object$
else
  if $Y - centre \geq 0.3 * Y - ImageDimentions$ **then**
    $hasonthebottom \leftarrow object$
  end if
end if

Algorithm 3 has in the center

---

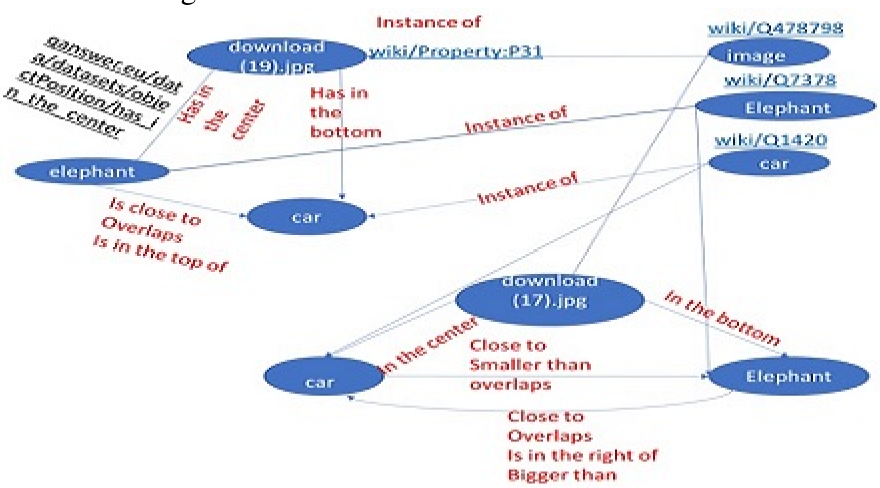
if $X - centre \geq 0.3 * X - ImageDimentions$,
$X - centre \leq 0.66 * X - ImageDimentions$,
$Y - centre \geq 0.3 * Y - ImageDimentions$,
$Y - centre \leq 0.66 * Y - ImageDimentions$ **then**
  $hasinthecenter \leftarrow object$

```
X-centre = (X1+X2)/2
and Y-centre= (Y1+Y2)/2
```

Therafter, we try to build a sematic web model.

4.2. Design of a semantic web modelling for extracted data.

The below diagram demonstrates a flowchart



After implementation of object extraction and determining the object position within images, we create a csv file containing these details. Here's a glimpse of a csv file that has been created for 4.airplane-Q197

X1	Y1	X2	Y2	object nar	Image nar	X-centre	Y-centre	URLs	dimentior	Y-Image D	X-Image D	has on the left	has on the right	has on the	has on the bottom	has in the center
190	813	3897	1932	airplane	Antonov_	2043.5	1372.5	https://uç	(4096, 273	2734	4096	na	airplane	na	airplane	airplane
220	596	5021	1673	airplane	EBACE_20	2620.5	1134.5	https://uç	(5241, 294	2948	5241	na	airplane	na	airplane	airplane
742	1303	3933	2099	airplane	Embraer_I	2337.5	1701	https://uç	(4771, 318	3181	4771	na	airplane	na	airplane	airplane
889	1035	1378	1374	airplane	Kirchturm	1133.5	1204.5	https://uç	(2021, 192	1920	2021	na	airplane	na	airplane	airplane
172	278	4018	1722	airplane	Lufthansa_	2095	1000	https://uç	(4216, 198	1980	4216	na	airplane	na	airplane	airplane
331	532	2362	1245	airplane	North_Am	1346.5	888.5	https://uç	(2800, 157	1575	2800	na	airplane	na	airplane	airplane
1355	704	1444	813	person	North_Am	1399.5	758.5	https://uç	(2800, 157	1575	2800	na	person	na	person	person
833	251	2202	1761	airplane	Paris_Air_	1517.5	1006	https://uç	(3002, 200	2004	3002	na	airplane	na	airplane	airplane
1460	1207	1579	1738	person	Playing_ir	1519.5	1472.5	https://uç	(3000, 200	2000	3000	na	person	na	person	na
224	1063	466	1813	person	Playing_ir	345	1438	https://uç	(3000, 200	2000	3000	person	na	na	person	na
756	1286	845	1562	person	Playing_ir	800.5	1424	https://uç	(3000, 200	2000	3000	person	na	na	person	na
2374	990	2486	1359	person	Playing_ir	2430	1174.5	https://uç	(3000, 200	2000	3000	na	person	na	person	na
468	1194	546	1264	frisbee	Playing_ir	507	1229	https://uç	(3000, 200	2000	3000	frisbee	na	na	frisbee	na
264	273	2284	1571	airplane	RUAG_Avi	1274	922	https://uç	(2800, 186	1866	2800	na	airplane	na	airplane	airplane
582	687	3826	1588	airplane	Thai_Airw	2204	1137.5	https://uç	(4096, 230	2304	4096	na	airplane	na	airplane	airplane

Let's talk a little bit about triples in semantic web.
Triples

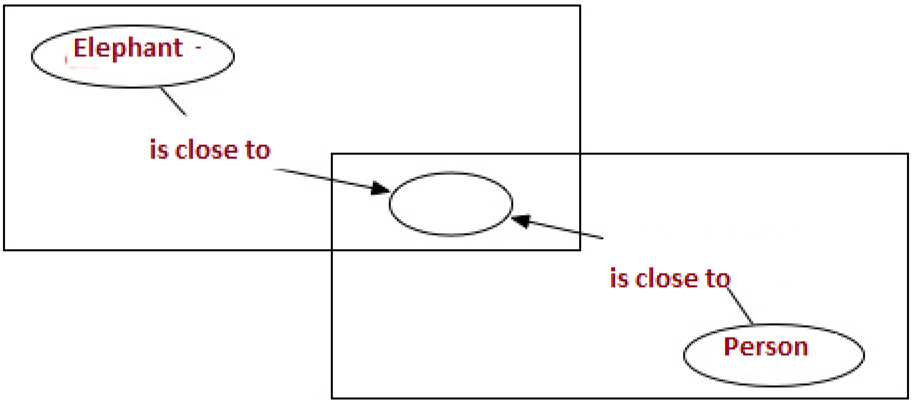
- Subject could be URI or blank node
- Predicate could be URI, but never be a blank node
- Object could be URI, blank node or literal

Now, let's try to define the important terms that has been just mentioned.

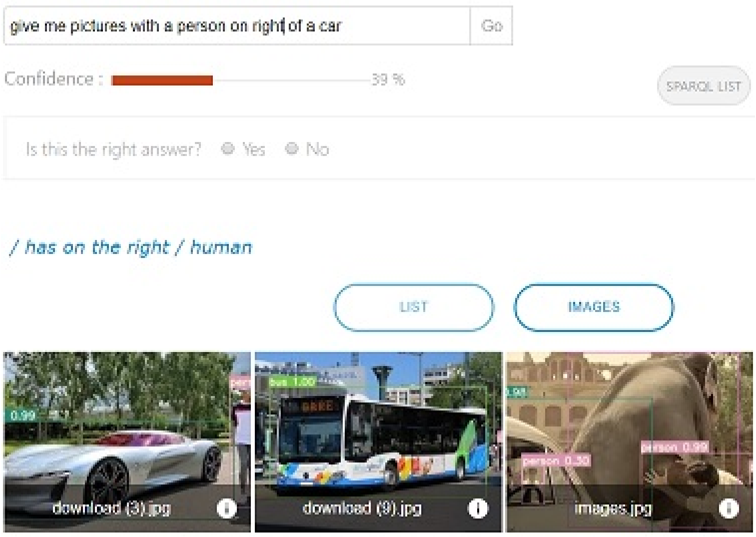
- URI
  URLs, URIs, IRIs, and Namespaces

  - URL = Uniform Resource Locator
    http://www.wikidata.org/entity/Q10470
  - URN = Universal Resource Name
    urn:isbn:006251587X
  - URI = Universal Resource Identifier
    - encompasses both URLs and URNs
    - most URIs are URLs (sometimes the terms are used interchangeably)
    - http://xmlns.com/foaf/0.1/Person
  - IETF released IRIs (Internationalized Resource Identifiers)

- blank node- In RDF, a blank node (also called bnode) is a node in an RDF graph representing a resource for which a URI or literal is not given.The resource represented by a blank node is also called an anonymous resource.



If, we refer again to the csv file, the image names are subjects; the relations such ”has on the left/right/top/bottom/center” are predicates; and objects are object names that has been mentioned under the predicates. The csv file that has been refered to, is based on Image-Object Relation. Let us now move on to the Object-Object Relation which is an attempt to establish the relationship between two objects in an image. Eg: car is on the left of a person in the image1. Here, ”car is on the left of a person” is a triplet and this triplet becomes the subject of the second triplet, followed by the predicate-”in” and object-”image1”. Such kind of triplets are called reified triplets. This features is still not available in Qanswer. Here's a glimpse of what would we get if we try to query a reified triple.



5. Future Work

- Qanswer is working on handling reified images
- The wikimedia api which we chose to work with, can be used to retrieve human annotated structured data. We're working on using the structured data. Here's an example of the api:
  https://commons.wikimedia.org/w/api.php?
  action=querylist=searchsrsearch=haswbstatement:P180=Q7378
  srnamespace=6format=json

6. Conclusion

We would want to conclude stating that an automated Python program is already in place which takes the special wikimedia api to download images, runs YOLO over it, creates a Image-Object relation based csv file and then converts the same into a RDF file which gets readily available for Qanswer's use. During csv generation phase, another csv file comes into existence, which contains Object-Object relation. It's just a matter of time, in which Qanswer would be able to handle reified triples. This work can be easily used by any search or query engine to give results based on image-object relation and object-object relation.

References