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

September 18, 2020

Contents

List of Figures

Abstract

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

1. Introduction

The topic of this internship is about image search. Current techniques of image search (google, bing, and others) use annotation of the images to provide results to users. However the computer vision techniques are not always used to index the images, and there are techniques from the Semantic Web domain that could also improve the search, especially advanced Question-Answering techniques based on linked data.

We will see in this report some limitations of current techniques and our innovative approach for semantic image search based on Computer Vision techniques, Semantic Web techniques and Question-Answering techniques. The structure of this report is as follows : the next section will present the current image search techniques. Then we will proceed to describe QAnswer. Thereafter, we will talk about the research problem and state of the art. Following this, we would discuss about our contributions and limitations. Finally, we would conclude by displaying some results.

2. Current image search techniques

Google: give me pictures of bicycle

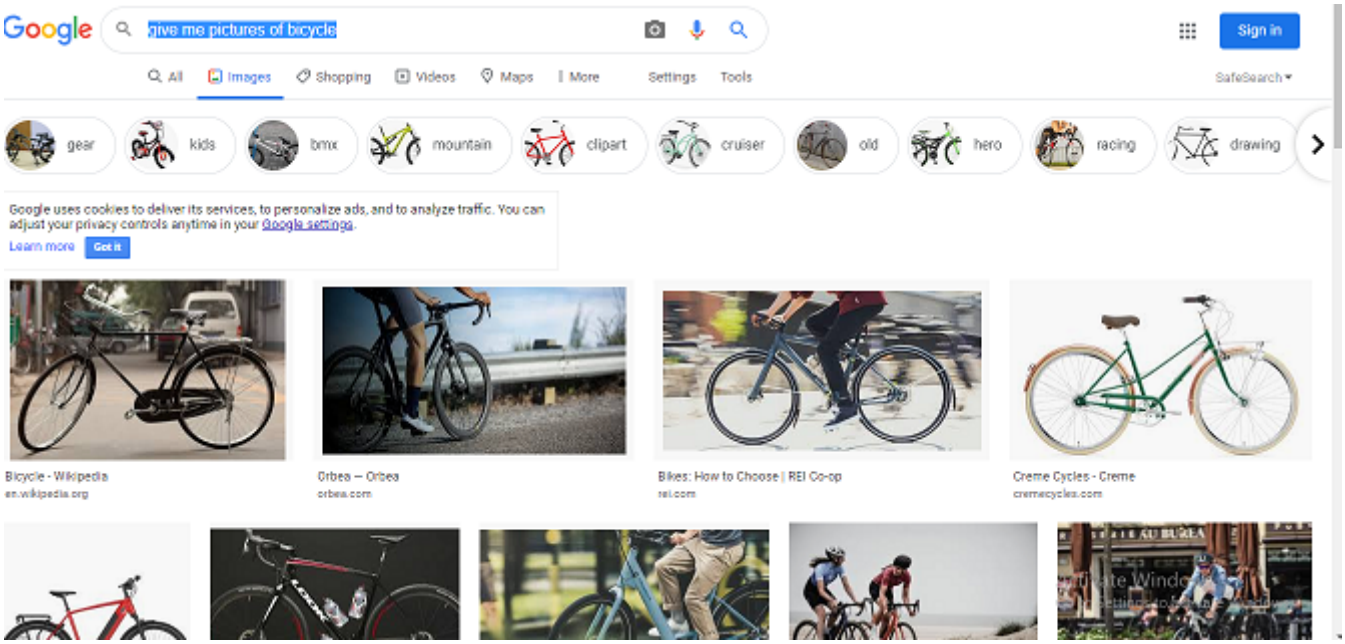

Figure 1. pictures of bicycle

From the above, we can see that, there are just too many results.Visual Search Engines like Google and Bing use image recognition to provide users with the best search results.Picsearch is also a traditional visual search engine that offers a massive image

archive[1].Now,Image recognition is the process of identifying and detecting an object or a feature in a digital image or video[2].The modeling process for image recognition is shown in Step 1 through 4.

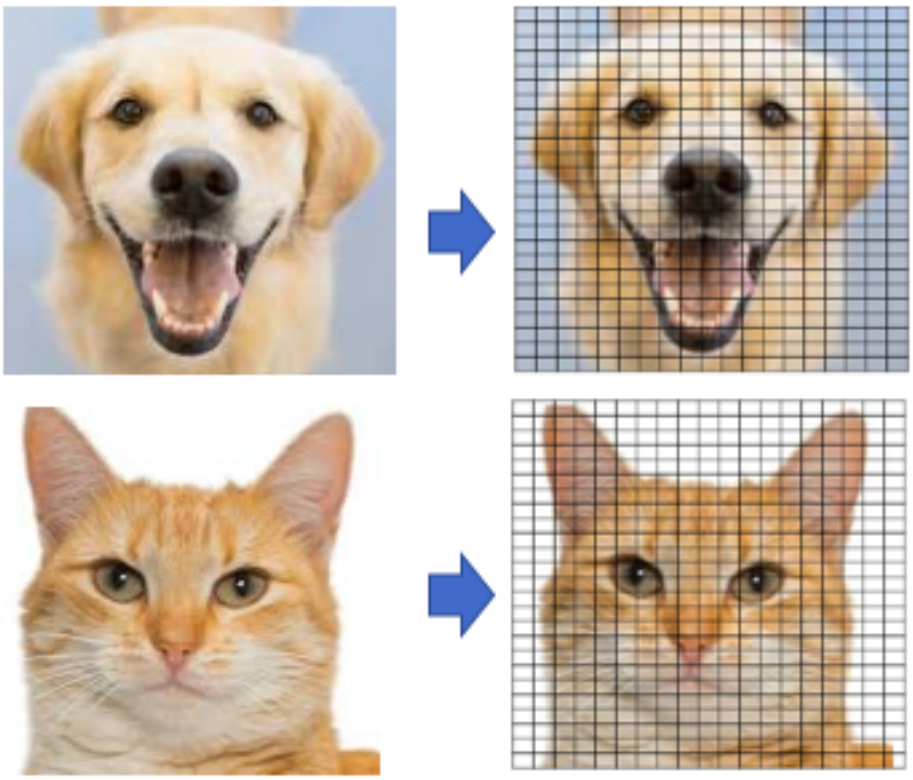

Figure 2. Modeling Step 1: Extract pixel features from an image

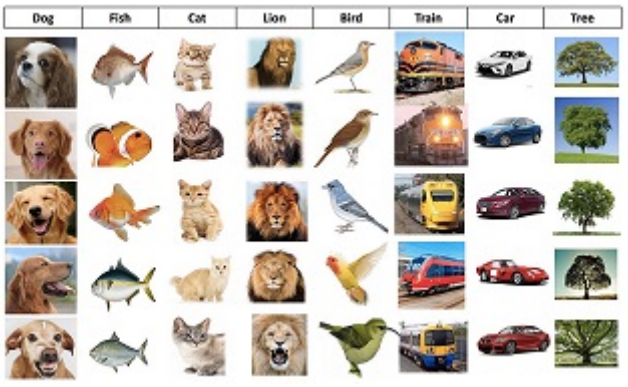

Figure 3. Modeling Step 2: Prepare labeled images to train the model

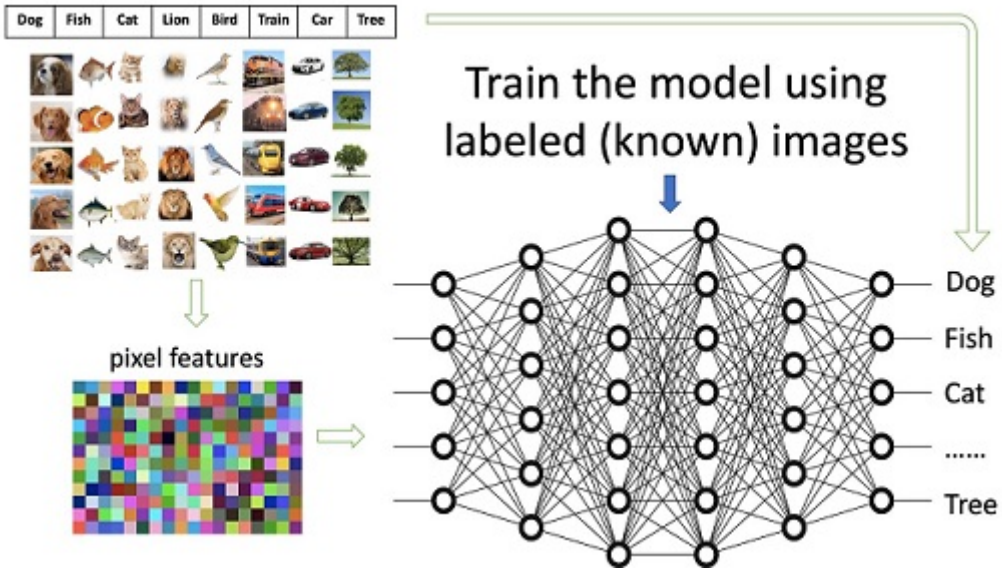

Figure 4. Modeling Step 3: Train the model to be able to categorize images

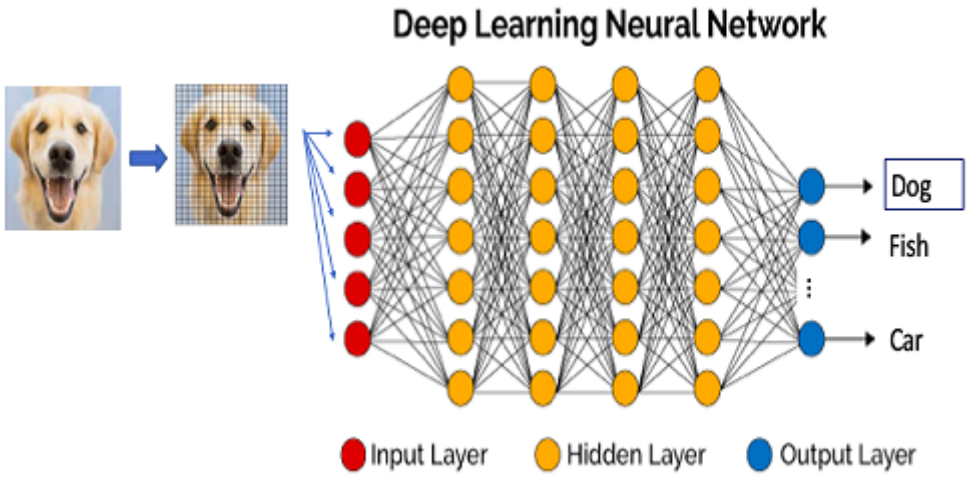

Figure 5. Modeling Step 4: Recognize (or predict) a new image to be one of the categories

Convolution Neural Networks — the algorithm for image recognition The networks in (Modeling Step 3: Train the model to be able to categorize images) or (Modeling Step 4: Recognize (or predict) a new image to be one of the categories) have implied the popular models are neural networks models. Convolutional Neural Networks (CNNs or ConvNets) have been widely applied in image classification, object detection or image recognition[3].

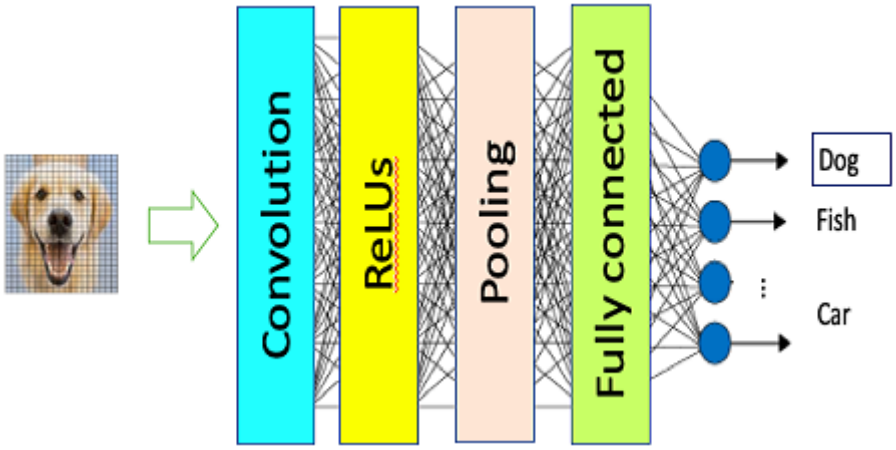

Figure 6. A brief pictorial explanation for Convolution Neural Networks

3. QAnswer

Users' experience is an important factor for the success of a given application[4]. We aim to provide the best possible user's experience by introducing QAnswer. QAnswer is knowledge (or ontology) based QA system.A knowledge base is a collection of facts that can be interpreted by a machine. Such a fact can look like this:"Alan Turing" "student of" "Alonzo Church".We have million of such facts and we use them to find an answer to your question.

The front-end of QAnswer which highly impacts the users' experience, is an important part for a image base query system[5]. 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[6]. 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).

A regular search just gives you a randomly ordered list of items in which your search term occurs. A smart search will provide you a ranked list of items that are strongly related to the search terms you entered, even if they do not match exactly. This means that even though some of the results will doubtless be unrelated, and sometimes absurd, there is a very good chance that there is data you can use pretty high up in those results, even if you didn't ask quite the right question. And there may well be relevant data you didn't even think to ask for.

Being smart in this way is a key benefit of search. Queries cannot be smart. Queries must always give you exactly what you asked for. There can be no tolerance

for serendipity in query results. Search can be smart, but query must be dumb and strictly obedient[7].

There are mainly two reasons for not getting the right query result. The first is that we simply do not have the data to answer it. We generally try to find the best interpretation of your question based on the data we use. If we do not have the data also the interpretation will be wrong. The second reason for wrong query result is that we have the data but we are not able to correctly interpret your question.Here's an instance of the same. We are trying to improve over time the quality of the QA system by adding new datasets and by refining the algorithm that interprets your question. So hopefully next time it works.

A big advantage over traditional search engines is that different information can be combined so that we can answer questions like 'Who was a student of Alonzo Church?'.QA system makes a formal database query,which is addressed in formal terms to a specific dataset.Our underlying datasets are Wikidata and openstreetmap data. Wikidata contains structured knowledge about many existing entities like the European Union. It contains information about the capital is Brussel. This information is converted into the triple"European Union" "capital" "Brussel" allowing us to answer a question like 'What is the capital of the EU?'.

We have an open API: curl –data "query=Who is the wife of Barack Obama" http://QAnswer-core1.univ-st-etienne.fr/api/gerbil which support the following parameters:query: for the questions the language supported currently are en, fr, de, it, es, zh.Moreover, the knowledge-base that are currently supported are dbpedia, wikidata, dblp, and freebase.

4. Presentation of the research problem

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 innovative methods for making this knowledge available.We hope to change this trend by Semantic Web techniques for querying the knowledge made available by computer vision. My work focuses on bridging the two disciplines here.

5. State of the art

Let's try to understand how does a Google image search engine work. There are three 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)Usage of object identification techniques and annotating the images with the name of these objects. 3)Linking all visually similar images to the image with the same text. e.g. consider an image 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[8].

Hence, the current indexing and search techniques provide results on the presence of a given type of object, without considering more details such as: the relative or absolute position of the objects in the image, the number of given objects, and other characteristics that could be inferred from the image analysis (such as ongoing actions). An example of this would be, if 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 with the current techniques 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 and other characteristics from the image analysis.

In the domain of search engines on structured data, QAnswer is a recent technology that converts the natural language into triples and use the best ranked SPARQL query to query structured data sources. The objective of this work is to combine the results from the image analysis processes with the query capabilities of QAnswer in order to improve the state of the art of image search engines.The details are explained in the upcoming section.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 improve the state of the art in the image search engine domain.

6. Contribution/Proposal description and implementation

We have implemented a 2 steps approach that first identifies objects into images, and then generates the topological semantic description of the scene in the picture.

- 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.

6.1. Implementation of an Algorithm for object extraction.

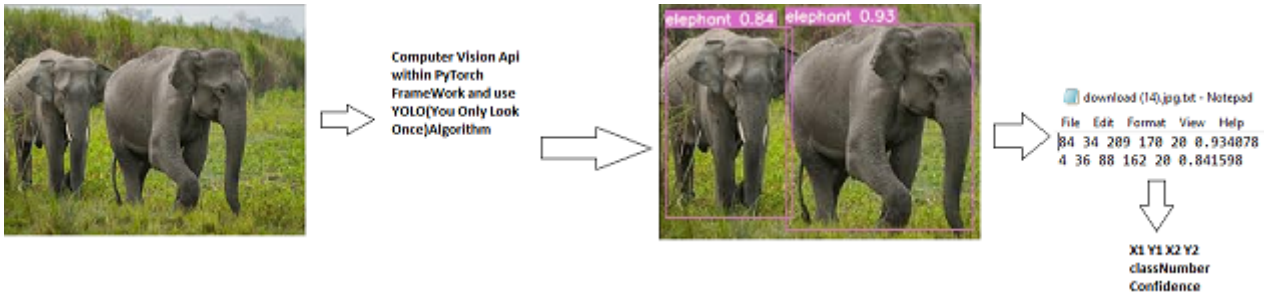

Figure 7. Diagram demonstrating a flowchart

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:

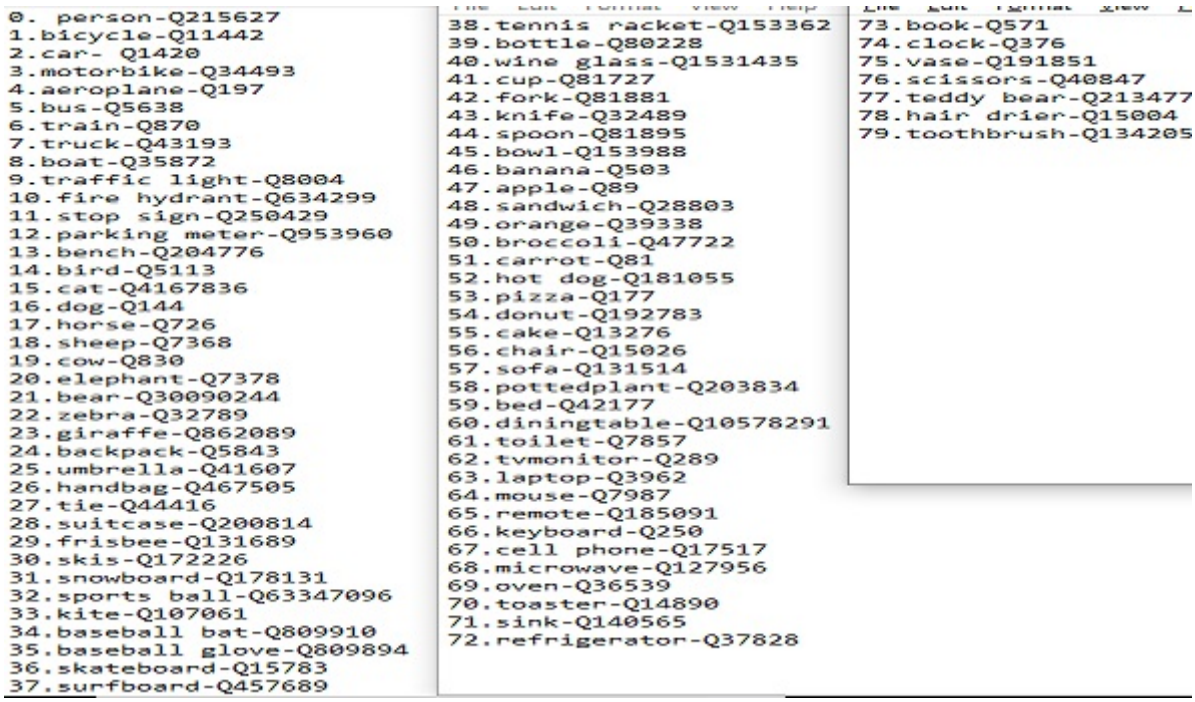

Figure 8. Class number, Class name, QID

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[9]
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	

Figure 9. Subject, Predicate, Object

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

Algorithm 1 has on the left and right

if $X - centre \leq 0.3 * X - ImageDimentions$ **then**
  $hasontheleft \leftarrow object$
else
  if $X - centre \geq 0.6 * X - ImageDimentions$ **then**
    $hasontheright \leftarrow object$
  end if
end if

Algorithm 2 has on the top and bottom

if $Y - centre \leq 0.3 * Y - ImageDimentions$ **then**
  $hasonthetop \leftarrow object$
else
  if $Y - centre \geq 0.6 * Y - ImageDimentions$ **then**
    $hasonthebottom \leftarrow object$
  end if
end if

Algorithm 3 has in the center

1: **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**
2: $hasinthecenter \leftarrow object$

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:

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

After implementation of object extraction and determining the object position within images, we create a csv file containing these details.

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

Figure 10. A csv file that has been created for 4.airplane-Q197

Finally, we try to build a sematic web model.

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

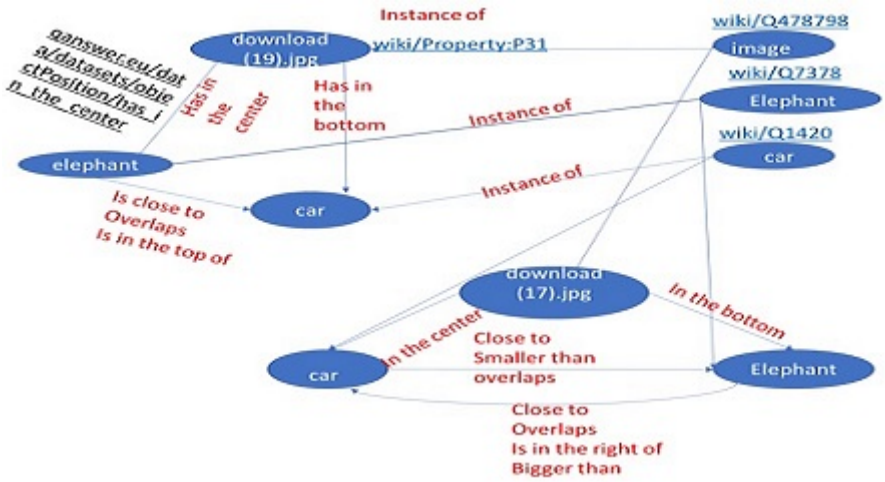

Figure 11. Diagram demonstrating a knowledge graph

The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The first steps in weaving the Semantic Web into the structure of the existing Web are already under way. In the near future, these developments will usher in significant new functionality as machines become much better able to process and ”understand” the data that they merely display at present[10].
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

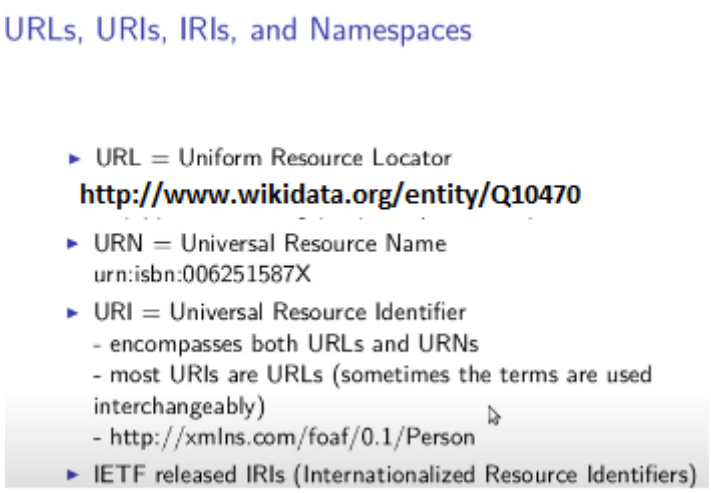

Figure 12. Diagram Demonstrating URI

- 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.

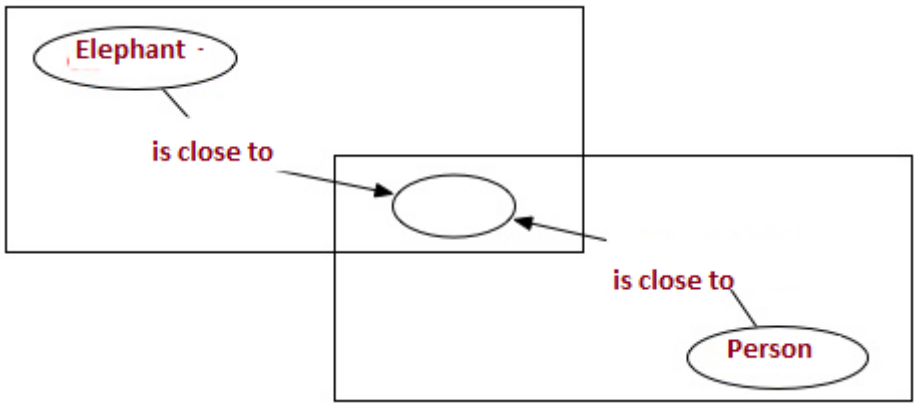

Figure 13. Diagram Demonstrating Blank Node

If, we refer again to the csv file which is shown in Figure 10, the image names are subjects; the relations such as ”has on the left/right/top/bottom/center” are predicates; and objects are object names that has been mentioned under the relations. The csv file that has been referred to, is based on Image-Object Relation. We use a Java program to convert this csv file into a RDF file, by taking the subject-predicate-object into consideration. Here’s a glimpse of an RDF file.

<https://upload.wikimedia.org/wikipedia/commons/6/60/ACE_EMD_F40PH_Fremont_-_San_Jose.jpg> <http://www.w3.org/2000/01/rdf-schema#label>
"https://upload.wikimedia.org/wikipedia/commons/6/60/ACE_EMD_F40PH_Fremont_-_San_Jose.jpg" .
<https://upload.wikimedia.org/wikipedia/commons/6/60/ACE_EMD_F40PH_Fremont_-_San_Jose.jpg> <http://www.wikidata.org/prop/direct/P31>
<http://www.wikidata.org/entity/Q478798> .
<https://upload.wikimedia.org/wikipedia/commons/6/60/ACE_EMD_F40PH_Fremont_-_San_Jose.jpg> <http://qanswer.eu/data/datasets/objectPosition/has_on_the_right>
<http://www.wikidata.org/entity/Q870> .
<https://upload.wikimedia.org/wikipedia/commons/6/60/ACE_EMD_F40PH_Fremont_-_San_Jose.jpg> <http://qanswer.eu/data/datasets/objectPosition/has_on_the_bottom>
<http://www.wikidata.org/entity/Q870> .
<https://upload.wikimedia.org/wikipedia/commons/6/60/ACE_EMD_F40PH_Fremont_-_San_Jose.jpg> <http://qanswer.eu/data/datasets/objectPosition/has_in_the_center>
<http://www.wikidata.org/entity/Q870> .
<https://upload.wikimedia.org/wikipedia/commons/f/f2/L%C3%B6%C3%9Fnitzgrundbahn_991777-4.jpg> <http://www.w3.org/2000/01/rdf-schema#label>
"https://upload.wikimedia.org/wikipedia/commons/f/f2/L%C3%B6%C3%9Fnitzgrundbahn_991777-4.jpg" .
<https://upload.wikimedia.org/wikipedia/commons/f/f2/L%C3%B6%C3%9Fnitzgrundbahn_991777-4.jpg> <http://www.wikidata.org/prop/direct/P31>
<http://www.wikidata.org/entity/Q478798> .
<https://upload.wikimedia.org/wikipedia/commons/f/f2/L%C3%B6%C3%9Fnitzgrundbahn_991777-4.jpg> <http://qanswer.eu/data/datasets/objectPosition/has_on_the_right>
<http://www.wikidata.org/entity/Q870> .
<https://upload.wikimedia.org/wikipedia/commons/f/f2/L%C3%B6%C3%9Fnitzgrundbahn_991777-4.jpg> <http://qanswer.eu/data/datasets/objectPosition/has_on_the_bottom>
<http://www.wikidata.org/entity/Q870> .
<https://upload.wikimedia.org/wikipedia/commons/c/cc/Locomotive_ChS4-109_2012_G1.jpg> <http://www.w3.org/2000/01/rdf-schema#label>
"https://upload.wikimedia.org/wikipedia/commons/c/cc/Locomotive_ChS4-109_2012_G1.jpg" .
<https://upload.wikimedia.org/wikipedia/commons/c/cc/Locomotive_ChS4-109_2012_G1.jpg> <http://www.wikidata.org/prop/direct/P31>
<http://www.wikidata.org/entity/Q478798> .
<https://upload.wikimedia.org/wikipedia/commons/c/cc/Locomotive_ChS4-109_2012_G1.jpg> <http://qanswer.eu/data/datasets/objectPosition/has_on_the_right>
<http://www.wikidata.org/entity/Q870> .
<https://upload.wikimedia.org/wikipedia/commons/c/cc/Locomotive_ChS4-109_2012_G1.jpg> <http://qanswer.eu/data/datasets/objectPosition/has_on_the_bottom>
<http://www.wikidata.org/entity/Q870> .

Figure 14. Diagram Demonstrating resultant output RDF file for 6.train-Q870

6.3. Results

Finally,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.

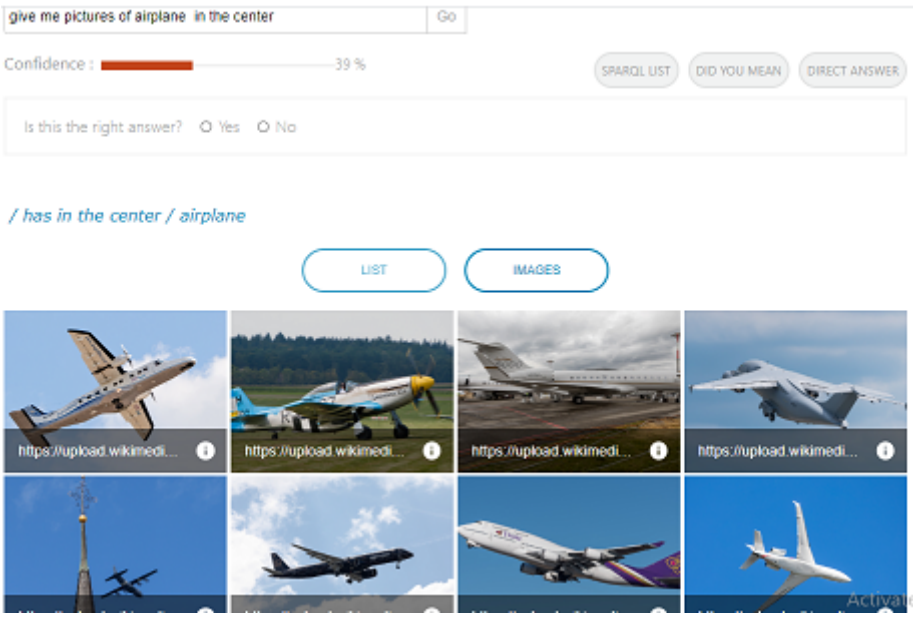

Figure 15. QAnswer: give me pictures of airplane in the center

Let's try to query some more, but this time, we would like to query for bicycles in the right.

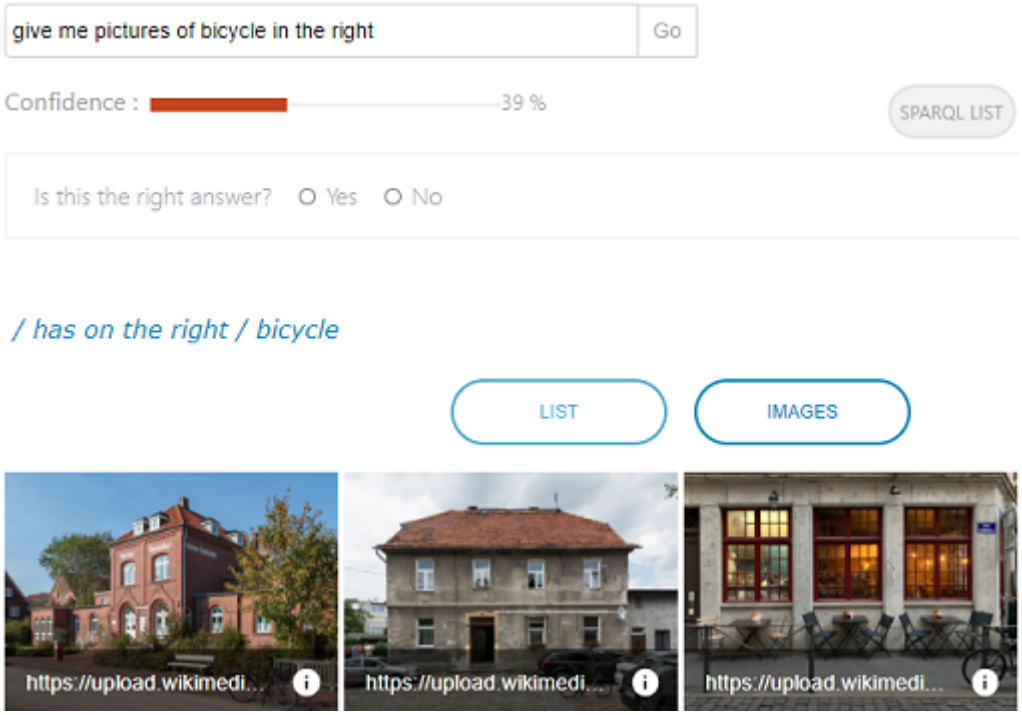

Figure 16. QAnswer: give me pictures of bicycle in the right

Here's a small set of images, consisting of bicycles.

Figure 17. QAnswer: a set of 10 images showing bicycles

So, we can see that, out of 10 pictures of bicycles; only 3 where chosen which fits the query of bicycles in the center

We would now like to demo for images in the left. Here's an URL: `https://qanswer-frontend.univ-st-etienne.fr/qa/full?query=train%20in%20the%20left&tags=%5B%5D&lang=en&kb=onto&user=anindamaulik`

You can just visit the homepage of QAnswer by click on the above hyperlink and type in "train in the left". You will get an image of a train on the left of the photo. You can go ahead to try it, right now.

Query for bench in the bottom.So, we look closely and locate a bench at the bottom of the picture.

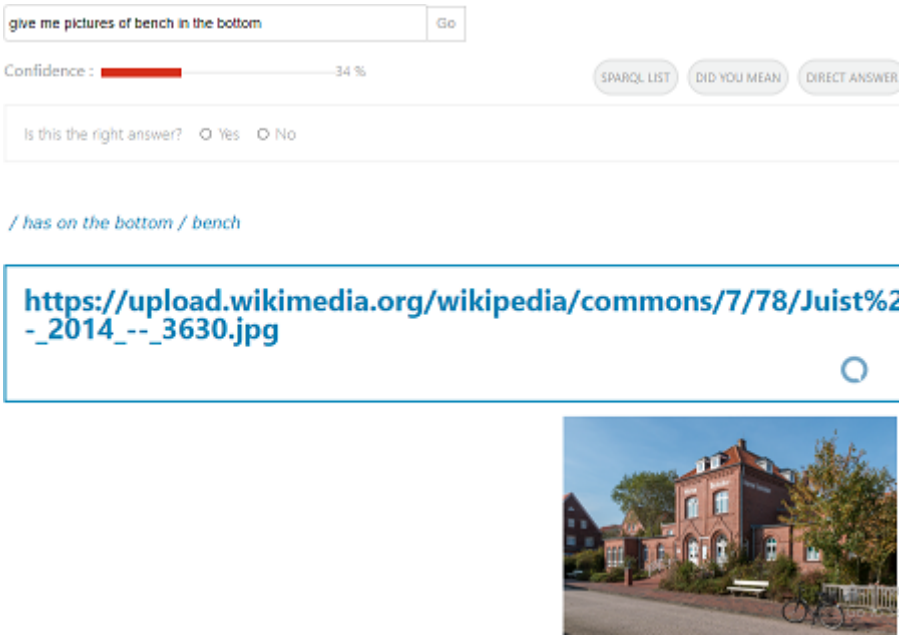

Figure 18. QAnswer: give me pictures of bench in the bottom

6.4. Application to wikimediacommons images

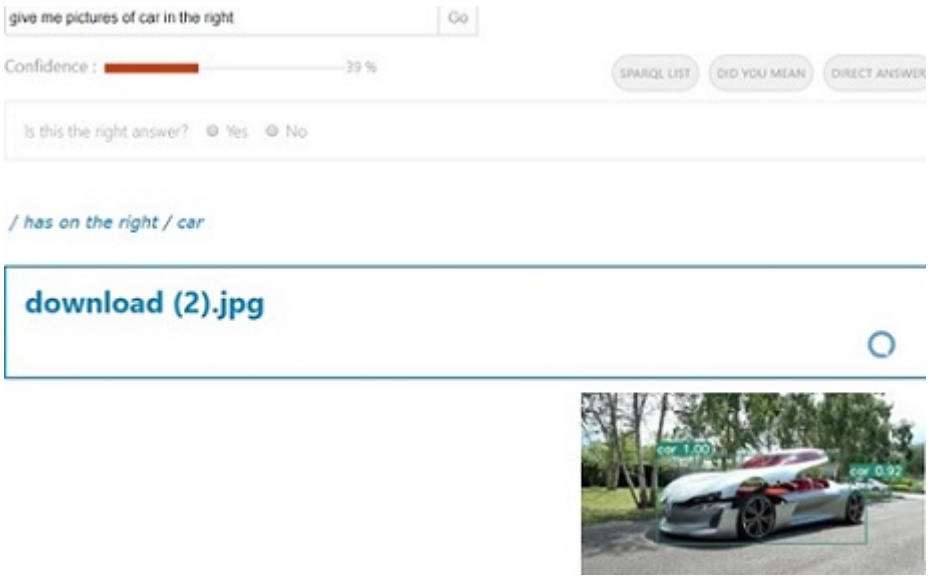

Figure 19. give me pictures of car on the right

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 [11]. 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.

6.5. Automation pipeline

We decided to automate the entire process, starting from downloading the images from wikimediacommons api, containing hand-annotated structured data.Then, YOLO runs over this downloaded data, giving out text files and images with bounding boxes around the detected objects. Following this, a csv file containing Image-Object relation comes into existence. Therafter, a java program is triggerred which has a sole purpose of converting the csv file to rdf file. Finally, we have a rdf file, ready to uploaded to QAnswer for queries on object position.

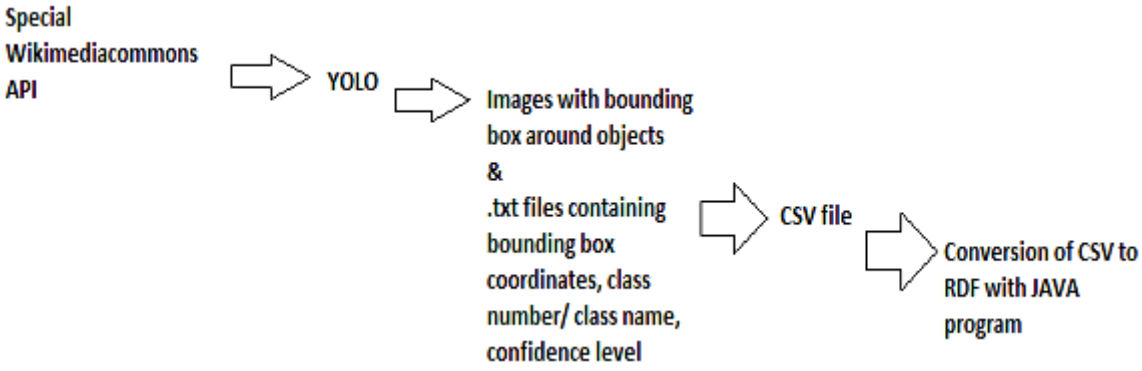

Figure 20. Automation Flowchart

7. Limitations and future work

7.1. Query for images on the top

7.1.1. Issue with confidence of detection by YOLO

Query for clock in the top.So, we look closely and not able locate a clock at the top or anywhere

Figure 21. QAnswer: give me pictures of clock in the top

We are trying to understand that what just happened here. In order to get a bit of an insight, let's try to check the photo with bounding box by YOLO.

Figure 22. QAnswer: give me pictures of clock in the top

So we see that YOLO has detected a clock with a confidence of 86 percent, when there is just a sign and not a clock. We could have increased the confidence level higher that 86, in order to avoid this error. But this error, could have also come up for a confidence above 90. Therefore such errors just cannot be avoided.

7.1.2. Issue with the object name assigned by YOLO

Query for a traffic light in the top. If we check the query in QAnswer, traffic light has become traffic. This is not a fault of QAnswer. Infact, QAnswer is able to find the closest word to traffic light in the uploaded rdf file, and give us the result. The reason, we have traffic in the query is because YOLO identifies traffic light as traffic.

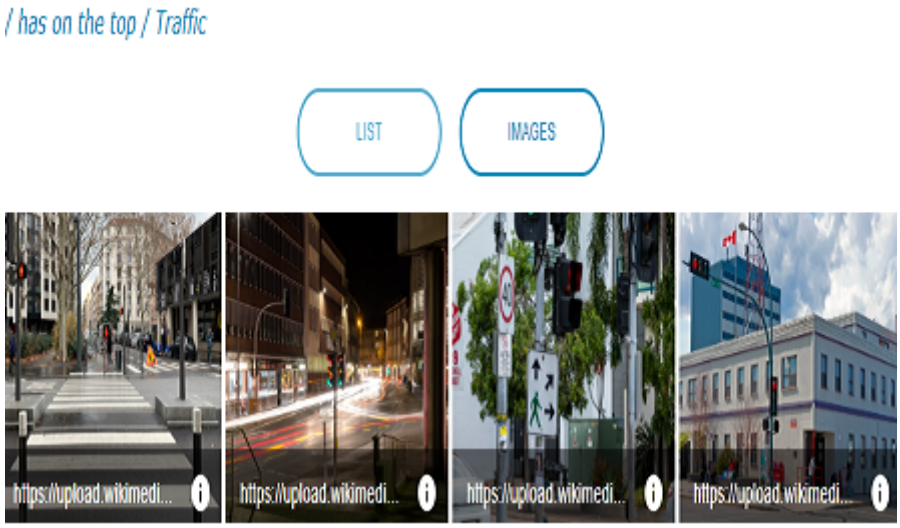

Figure 23. QAnswer: give me pictures of traffic light in the top

7.2. Object-Object Relations

Let us to understand 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.

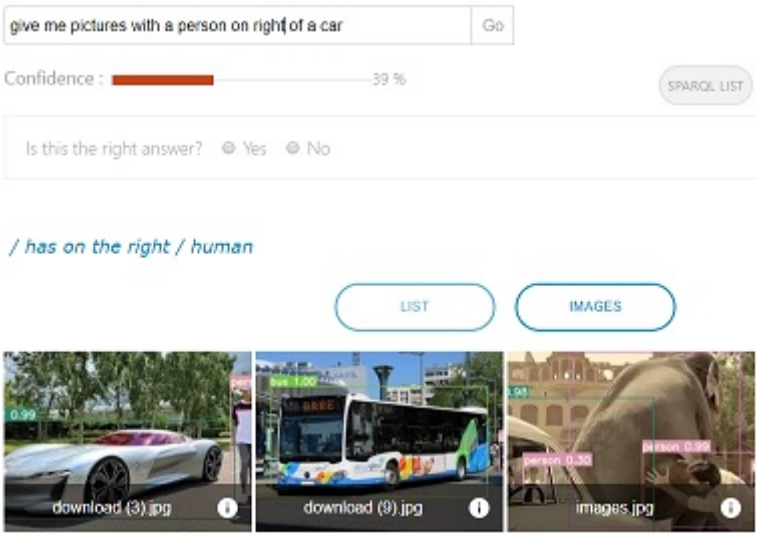

Figure 24. Reified Triple not being generated

So, as we see from the figure, QAnswer is not able to query in form of reified triples.
Our algorithms are ready for future use in regards to Object-Object relation. We would like to present the algorithms for the following Object-Object Relations.

Object	relation	property value
	left of	object
	right of	object
	top of	object
	buttom of	object
	close to	object
	far from	object
	overlaping wit	object
	greater than	object
	smaler than	object
	%of image	value

Figure 25. Object-Object Relations

We would begin with presenting our algorithm for an object to find out, if it is to the left or right of the second object

Algorithm 4 is on the left and right of

1: **if** $X - centreOfObject_1 \leq X - centreOfObject_2$ **then**
2: $Object_1IsOnTheLeft \leftarrow Object_2$
3: **else**
4: **if** $X - centreOfObject_1 \geq X - centreOfObject_2$ **then**
5: $Object_1IsOnTheRight \leftarrow Object_2$
6: **end if**
7: **end if**

We would continue with presenting our algorithm for an object to find out, if it is on the top or bottom of the second object

Algorithm 5 is on the top and bottom of

1: **if** $Y - centreOfObject_1 \leq Y - centreOfObject_2$ **then**
2: $Object_1IsOnTheTop \leftarrow Object_2$
3: **else**
4: **if** $Y - centreOfObject_1 \geq Y - centreOfObject_2$ **then**
5: $Object_1IsOnTheBottom \leftarrow Object_2$
6: **end if**
7: **end if**

Following this, here we are presenting our algorithm for an object to find out, if it is close or far from the second object

Algorithm 6 close and far from

1: **if** $distance(Center1, Center2) \leq meansofdiagonalofthe2objects$ **then**
2: $Object_1IsCloseTo \leftarrow Object_2$
3: **else**
4: **if** $distance(Center1, Center2) \geq meansofdiagonalofthe2objects$ **then**
5: $Object_1IsFarFrom \leftarrow Object_2$
6: **end if**
7: **end if**

Thereafter, here we are presenting our algorithm for an object to find out, if it is smaller or greater than the second object

Algorithm 7 greater and smaller than

1: **if** $AreaOfObject_1 \leq AreaOfObject_2$ **then**
2: $Object_1IsSmallerThan \leftarrow Object_2$
3: **else**
4: **if** $AreaOfObject_1 \geq AreaOfObject_2$ **then**
5: $Object_1IsGreaterThan \leftarrow Object_2$
6: **end if**
7: **end if**

Now, we would take into account a scenario to find out the percentage occupancy of an object in a picture. We try to find the percentage by multiplying 100 to the ratio of the area of the object and the picture.
Finally, we want to conclude by explaining our way of finding out if an object overlaps another object; in other words, we are trying to find out if one object's bounding box overlaps the other.
Let's introduce ourselves to a spatial data model which is accompanied by a natural language relationships between geometric objects – intersects-and a theoretical framework for understanding that using the 3x3 matrix of the mutual intersections of their component point sets.The following code shows how you can test for intersection:

```
from shapely.geometry import Polygon
p1 = Polygon([(0,0), (1,1), (1,0)])
p2 = Polygon([(0,1), (1,0), (1,1)])
print(p1.intersects(p2))
```

```
True
```

Figure 26. Simple Python code demonstration

7.3. QAnswer

7.3.1. Not getting the right query result

As mentioned in the QAnswer section, there can two reasons for not getting the right query result. The first is that we simply do not have the data to answer it. We generally try to find the best interpretation of your question based on the data we use. If we do not have the data also the interpretation will be wrong.

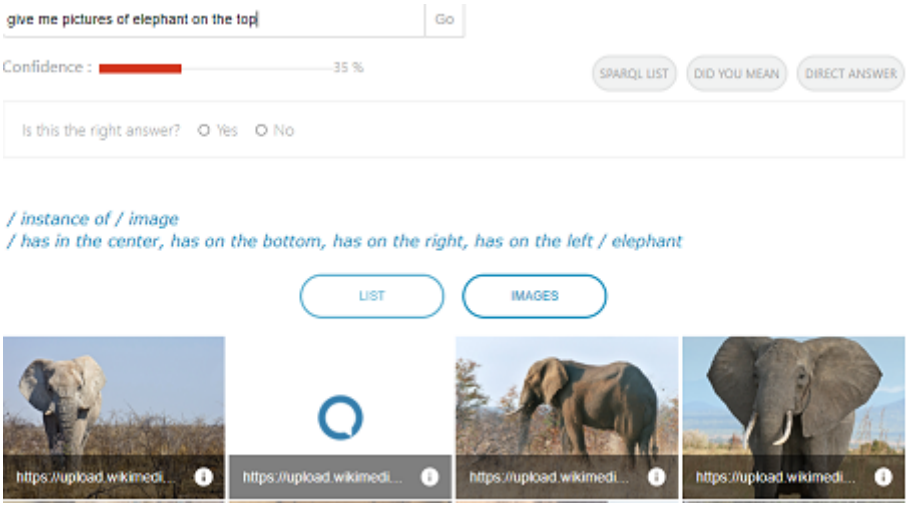

Figure 27. The wrong interpretation made due to non-availability of data

X1	Y1	X2	Y2	object (	Image name	hasinthetop
367	934	2320	3663	elephant	African_Bush_Elephant.jpg	na
660	332	2170	1659	elephant	African_bush_elephant_in_San_Diego_Zoo.jpg	na
48	383	4239	3146	elephant	Asian_Elephant_10.jpg	na
1230	452	3745	3044	elephant	Barcelona_-_Mammoth_sculpture.JPG	na
2248	1390	4036	3079	elephant	Elefantes%2C_Ayutthaya%2C_Tailandia%2C_2013-08-23%2C_DD_10.jpg	na
169	1495	1788	2820	elephant	Elefantes%2C_Ayutthaya%2C_Tailandia%2C_2013-08-23%2C_DD_10.jpg	na
3975	1579	4466	2618	elephant	Elefantes%2C_Ayutthaya%2C_Tailandia%2C_2013-08-23%2C_DD_10.jpg	na
1545	1037	2116	1412	umbrella	Elefantes%2C_Ayutthaya%2C_Tailandia%2C_2013-08-23%2C_DD_10.jpg	umbrella
3252	1038	3825	1397	umbrella	Elefantes%2C_Ayutthaya%2C_Tailandia%2C_2013-08-23%2C_DD_10.jpg	umbrella
2925	701	3505	1095	umbrella	Elefantes%2C_Ayutthaya%2C_Tailandia%2C_2013-08-23%2C_DD_10.jpg	umbrella
2338	1155	3143	1527	bench	Elefantes%2C_Ayutthaya%2C_Tailandia%2C_2013-08-23%2C_DD_10.jpg	bench
0	1297	5668	8517	elephant	Elefante_africano_de_sabana_%28Loxodonta_africana%29%2C_parque_	na
388	125	1382	881	elephant	Elephant_Kruger_2003.jpg	na
323	168	1039	1328	elephant	Etosha_elefant.jpg	na
88	373	2969	2483	elephant	Loxodonta_africana_-_drinking.jpg	na

Figure 28. In the csv file snapshot, we can see that there's data available, but not for elephant

The second reason for wrong query result is that we have the data but we are not able to correctly interpret your question.Here's an instance of the same.

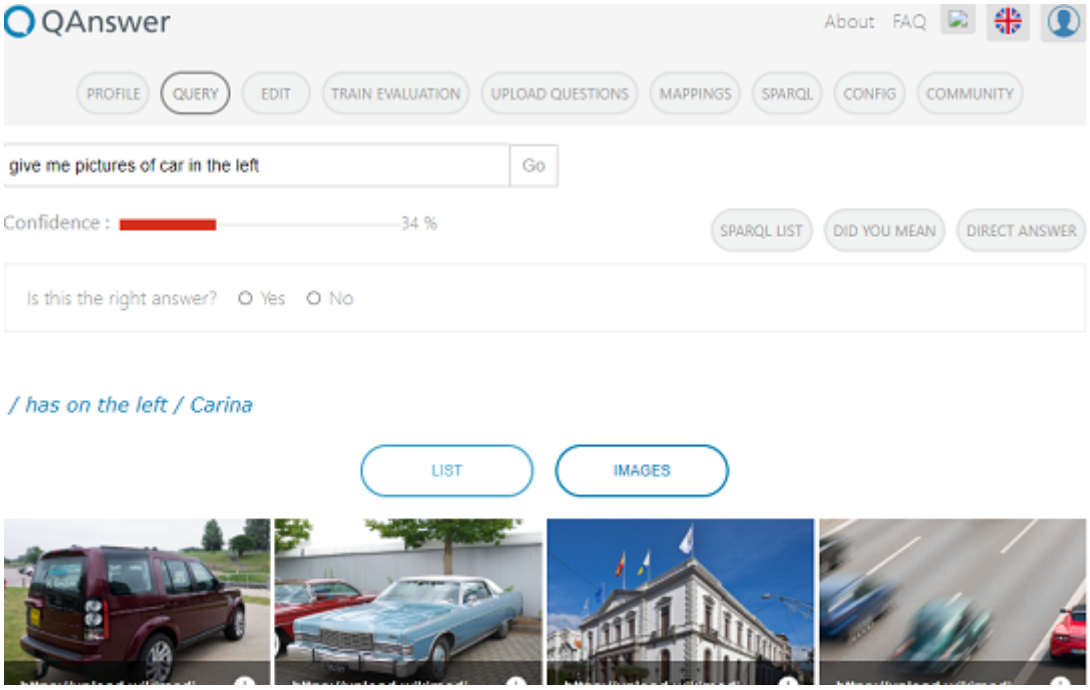

Figure 29. Failed to interpret the question correctly

7.3.2. Handling reified triples

QAnswer team is working on handling reified data, that has been discussed in the above section, so that we could answer question related to object to object relations into the images.

7.4. RDF data from wikimediacommons api

The wiki media API which we chose to work with to get images from wiki media commons, can be used to retrieve human annotated structured data. We're currently working on using the structured data. Here's the API for getting RDF data for images containing objects annotated as "elephant" which is coded Q7378.: `https://commons.wikimedia.org/w/api.php?action=querylist=searchsrsearch=haswbstatement:P180=Q7378srnamespace=6format=json`

7.5. Usage by others

Our technique can be included into search engines for better results We can also improve on computer vision techniques to identify more objects and we can also introduce the identification of background scenes.

8. Conclusion

we have worked on improving image search engines by combining Computer Vision techniques with Semantic Web techniques and Question Answering techniques. Computer Vision techniques is able to identify objects into images, and Semantic Web techniques give a semantic representation of the images that can be queried with QA engine, namely QAnswer.

QAnswer was not able to answer questions about image content. Now, we have an automated Python program 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.

This work can be easily used by any search or query engine to give results based on image-object relation and in a near future on object-object relation.

References

[1] 8 uses cases of image recognition that we see in our daily lives. https://rb.gy/3bf7hi.

[2] Recognition methods in image processing. https://rb.gy/mvezbh.

[3] What is image recognition. https://rb.gy/gaxy5m.

[4] Dennis Diefenbach, Shanzay Amjad, Andreas Both, Kamal Singh, and Pierre Maret. Trill: A reusable front-end for qa systems. In *European Semantic Web Conference*, pages 48–53. Springer, 2017.

[5] Dennis Diefenbach. https://qanswer-frontend.univ-st-etienne.fr/faq.

[6] https://en.wikipedia.org/wiki/SPARQL.

[7] Mark Baker. Search vs. query. https://rb.gy/gba3ev.

[8] Nitin Thokare. How does a google image search engine work. https://rb.gy/mb2vqp, 2013.

[9] class number and name. https://rb.gy/ofdhb0.

[10] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

[11] api parameters. https://www.mediawiki.org/wiki/API:Search.