Table of Contents

1. Introduction

With videos becoming the biggest big data, there has been increasing need to summarize, index and browse the large corpus of video content. As a common practice, videos are often summarized by key frames, i.e., a set of representative video frames [19, 20, and 21]. In this work, we propose to summarize videos into key objects instead of key frames, Comparing with key frames. Video can be summarized by three different ways which are as follows, for the work flow of our progress we have chosen the key frame based one.

1) Key Frame Based Video Summarization.
2) Video Skim Based Video Summarization.
3) Short Segmentation Based Video Summarization.

Here is the short segment wise module approach of our work flow which we have done throughout of our project,

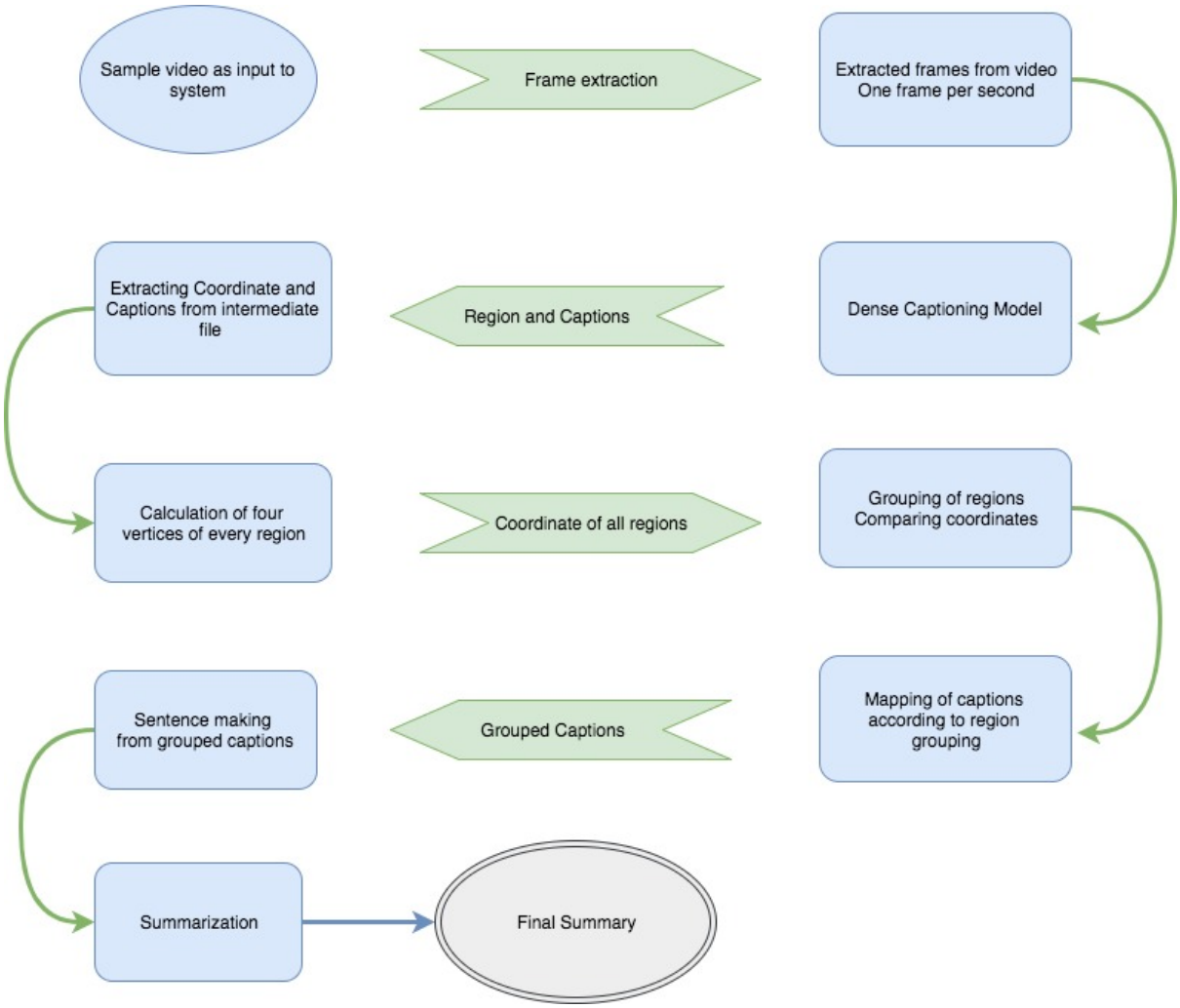

Image 1.1 Work flow diagram of the project

In our present task, we have introduced about event identification from images which are also an important source of information, specially we can collect lots of information from a single image. Every image has a story behind it. We have used the Dense Captioning task, which requires a computer vision system to both localize and describe salient regions in images in natural language. The dense captioning task generalizes object detection when the descriptions consists of a single word, and Image Captioning when one predicted region covers the full image.

2. Review of Literature

Video Shortening has been a field of active research since a long time. However, the focus has mainly been on either decreasing storage space using compression or removing redundant frames without loss of actual content. The latter is based on extracting key-frames from the video that can best represent a sequence of frames. One of the common approaches to do this is based on frame content changes computed by features, such as colour histogram or motion activity [13]. Another very common technique is to cluster the frames using supervised or unsupervised learning by the similarity of their content.

Zhuang proposed an unsupervised clustering scheme to adaptively extract key- frames from shots [15]. Other more sophisticated methods include the integration of the motion and spatial activity analysis with face detection technologies, a progressive multi-resolution key- frame extraction technique [1], and object-based approach. The trajectories of objects are used in [11] while user attention is modelled. The linear dynamical system theory is applied. Singular value decomposition is adopted to summarize video content in [4].

Advanced Computer Vision Techniques and Deep learning have only recently found their way into this field. Combines deep CNNs and RBMs to extract key-frames from videos uses web images as a prior to rank frames by their significance.

All these techniques concentrate on reducing redundancy in the video while keeping all the content. Another approach possible for video summarization, the one taken in this work, is to identify the “highlights” or the most important frames of the video and only keep them thresholds frames based on an importance score associated to each. However, the summarization is done on segments of video instead of entire videos, with the segments identified using clustering.

3. Objective

The main and foremost objective of our project is, given a video introduced as input, to produce a shortened video that only keeps the content that is more relevant to the user, so that he or she can get a short summary of the video that gives the most relevant events. That means finally it can identify the event in which human is involve from a video.

In this project, video summarization technology will be designed, developed and evaluated to work in real scenarios through which identification of event can be done. Video summarization is an important topic in the computer vision field since it enables faster browsing of large video collections and efficient content indexing and access. The objective of this technology is to significantly reduce the video footage that needs to be manually inspected by the human operator.

As a result, short videos can be delivered, showing only the relevant content for the operator. This will be done by automatically detecting all the objects, events and activities of interest that are displayed in the video. Based on this, only those fragments of video showing the detected content of interest will be kept in the summary.

4. System Design

Frame extraction is the first module of this system. Frame extraction module takes video as input and it extracts frames from this input video. This frames are output from frame extraction module. This frames are input to the DenseCap module. This module takes frames as input and produces region captioning as output. After we get region captions we map captions to regions. Next we group those captions according to a region. Grouping phase require comparison of coordinate information of regions. Manipulation of coordinates of regions is needed for partially inclusive regions. Grouping of regions need to be mapped with captions which tends to grouping of captions as well. Now we have chunk of sentences for a particular region as output. This chunk of sentences is summarized to get one or two sentences as output. This captions grouping process continues for every regions and then summarization is done on that chunk. Finally, we summarize sentences that we get form a single frame. Every frame has to pass through all those steps after extraction of frames is done. At the the final step of this system we run the summarization algorithm once again to get two or three sentences as output from the system.

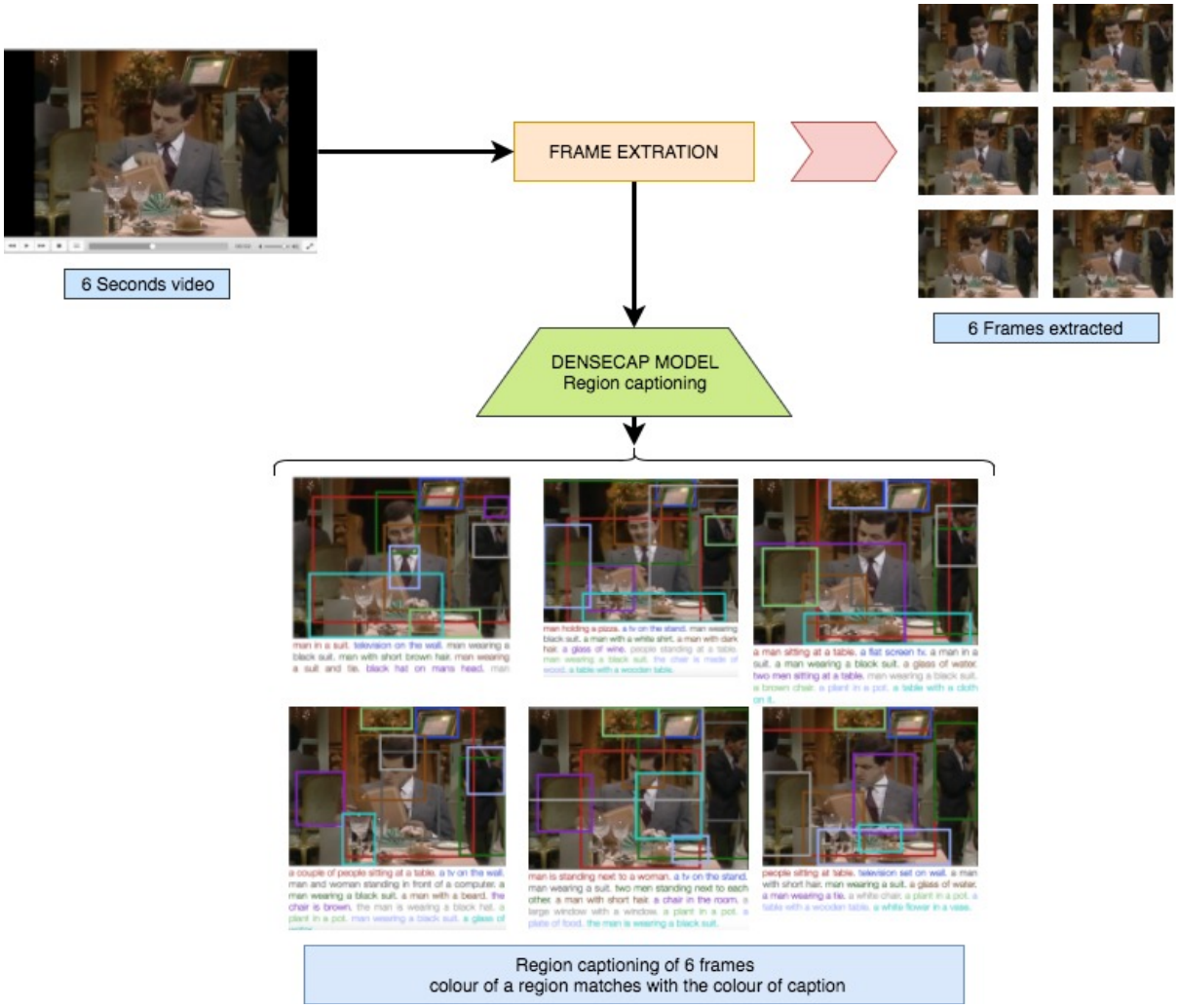

Image 4.1 Frame extraction and dense captioning steps are shown in this image. Video is input to frame extraction module and extracted frames are used as input to dense captioning module which gives region captioning as output.

Above block diagram shows functionality of frame extraction and dense captioning modules. A 6 seconds long and 25 frames per second video is taken as input to the Frame extraction module. This module extracts one frame per second from this video so we get 6 frames as output from this module. This frames goes to DenseCap model as input and we get region captioning as output from it.
Next task is to group regions which result to grouping of captions. Group of captions in other sense chunk of sentences are summarized. Below shown block diagram describes these two steps.

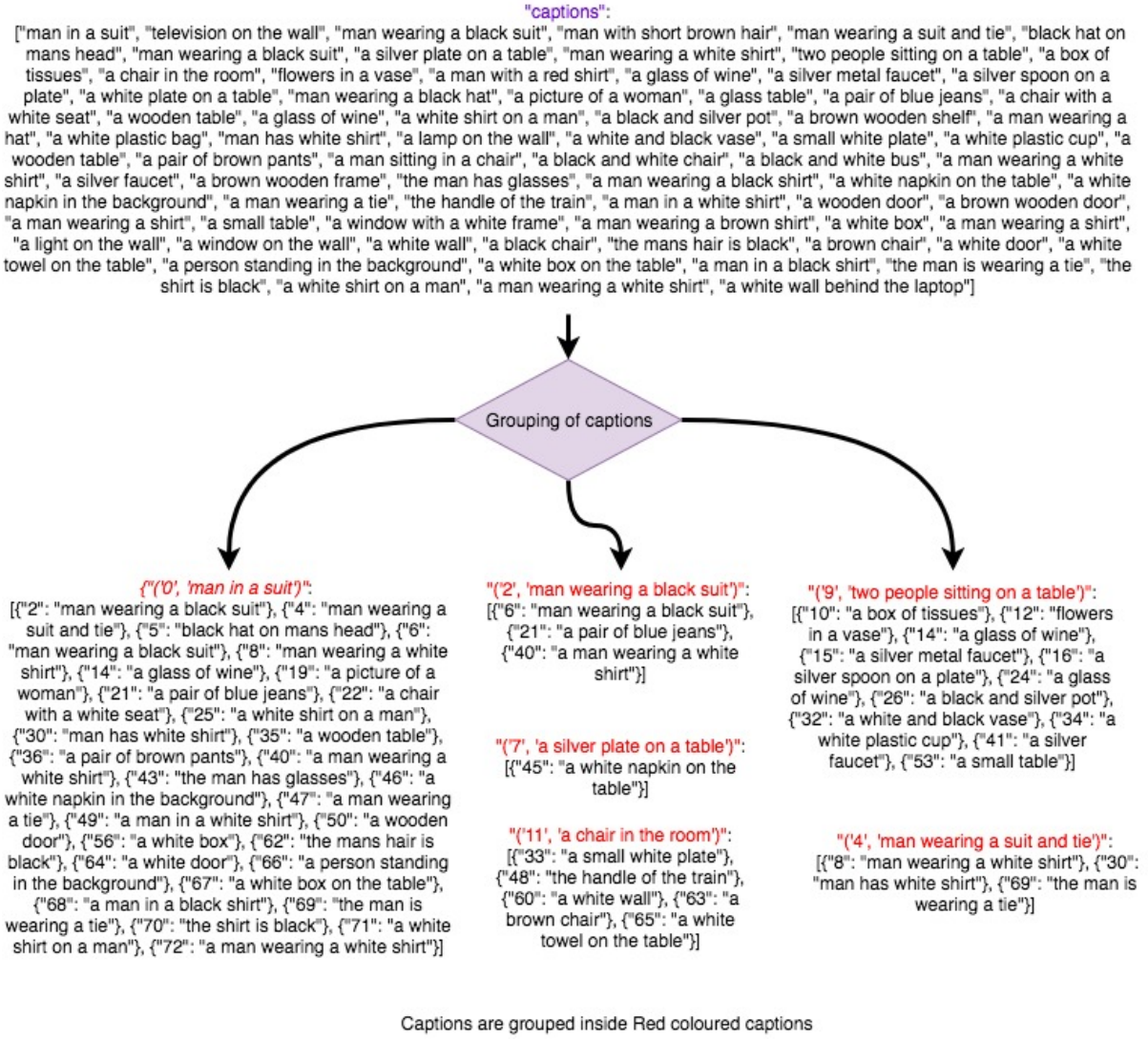

Image 4.2 Captions are grouped under different captions
which gives us details about a particular region.

In the above image, captions are used as input to grouping algorithm. After passing through the grouping algorithm step, captions are grouped under a particular caption. That means grouped captions describe the details of a particular region. Now we summarize this captions using summarization algorithm based on a particular region.

Consider this below text:
“the man has glasses. a man wearing a white shirt. a white door. a pair of blue jeans. the shirt is black. a white shirt on a man. man wearing a suit and tie. a man in a white shirt. a glass of wine. man has white shirt. a wooden table. a white box. a white box on the table. a man in a black shirt. a man wearing a tie. the mans hair is black. a wooden door. man wearing a black suit. a picture of a woman. a person standing in the background. a white napkin in the background. black hat on mans head. man wearing a white shirt. a chair with a white seat. a pair of brown pants. the man is wearing a tie. ”

After summarization we will get:
“A man wearing tie, black suit and white shirt.”

System design gives an over all view of the project. Every steps that we have discussed here are explained elaborately in the next two sections of this report. Method of implementation talks about our approach that we have taken to achieve final result. Along with this mathematical formulas and algorithms are shown in this

section. Implementation details talks about the internal details of our approach and detail discussion of mathematical formula and algorithms.

5. Methodology of Implementation

Previous section of this report shows an overview of the whole process of this project. Now in this section we are going to take a look at the actual input/output, formula and algorithm to accomplish small steps of this project. Our first step is frame extraction from video. As input to this step, we have taken a 6 seconds long and 25 frames per second video. Path to the video is given to take input from it. Next path to the folder where the frames will be stored is given as input. This algorithm extracts one frame per second. That means we get 6 frames after executing this extraction algorithm. Detail algorithm is shown here.

Algorithm 1: Frame extraction algorithm

1. start
2. path to the video file: videoFile
3. location of output frames: frameFolder
4. load the video
5. find video frame rate
6. if video channel is open
    1. get frame id
    2. read a frame
    3. if frame id divisible by frame rate
        i. save frame into frameFolder
7. repeat step 5 till video has correct frame
8. end

Algorithm explanation:
First step is to provide path to the video; *videoFile* is a variable where we store location of video which is taken from using using *input method (for python programming)*. In the second step we have given the path to the location where to store extracted frames in the variable *frameFolder*.

Enter path to video:"/Users/arin/Desktop/final_yr_project/frame_extract/mrbeantrim.mp4"
videoFile = "/Users/arin/Desktop/final_yr_project/frame_extract/mrbeantrim.mp4"

Enter output path for frames:"/Users/arin/Desktop/final_yr_project/frame_extract/frames"
frameFolder = "/Users/arin/Desktop/final_yr_project/frame_extract/frames"

OpenCV VideoCapture() method is used to load the video file at 3rd step. *OpenCV get(5)* method with parameter 5 is used to get the frame rate of the video. Opening of the video channel can be controlled by using *isOpened* method. Once again we have used *get(1)* method with parameter 1 to get the frame id. *OpenCV read()* method is used to read a frame from video and it also returns *boolean* value *true* for correct frames. Frames are written to the local file system with a name for it using *imwrite(framename, frame)* method.

A block diagram is shown below for this frame extraction step,

Image 5.1 Extraction of frames from a video when it is passed through the frame extraction algorithm

Region captioning is the next step after we get frames from video. Here we have used a DenseCap model. Internal working details of this model is discussed in *implementation details* section. We have hardware limitation so we have given 2 frames at a time as input to densecap model for region captioning. To execute

run_model.lua script of densecap model, we have used *Torch* framework which is a scientific computing framework with wide support for machine learning algorithms. Along with this the path to the frame folder is also given and for GPU acceleration we have turned on *–gpu -1* switch. We have given the command to execute *run_model.lua* script:

th run_model.lua –input_dir /path/to/frame/folder –gpu – 1

We can view the captioned frame by pointing to *localhost* on the browser. To do this we have to go to the *vis* folder of densecap model. In this folder we have a *html* file named as *view_results.html* through which we can view the region captioning of frames. Below command is to run localhost and given url is for point to *view_results.html* file.

Command:
$python –m SimpleHTTPServer 8181

Url:
http://localhost:8181/view_results.html

Below is a block diagram which shows dense captioning,

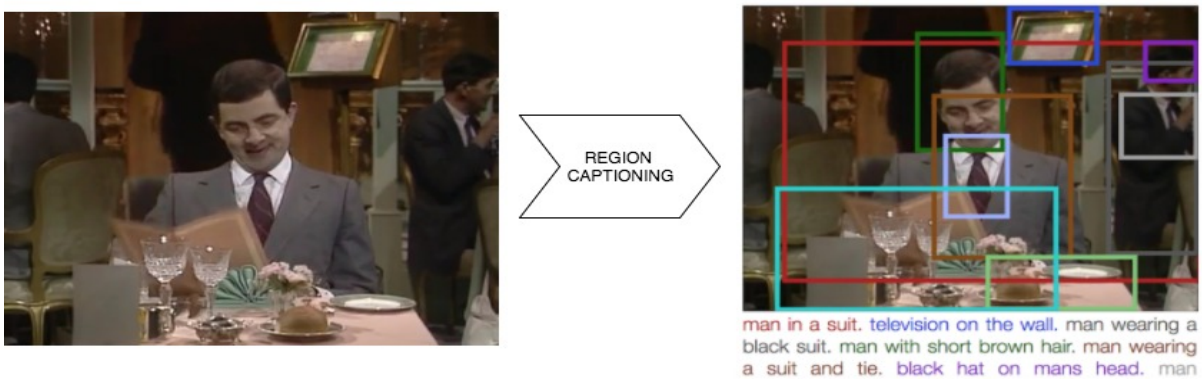

Image 5.2 Output from densecap after region captioning of a frame

Captioning information along with coordinate of regions are stored into *result.json* file inside *vis/data* folder of densecap model. For every image extraction densecap dumps region information and many other data into *result.json* file. We have fetched out coordinate information of regions and captioning from this file. For easy access of the data of a particular image, we have saved that data into a new *.json* file. To achieve this task, we have used a short algorithm which is shown below,

Algorithm 2: Fetching region information

1. start
2. load json file from file system
3. take one result entry from json file
    a. save entry to new file
    b. write new file to the local file system
    c. go to next result entry
4. repeat step 2 till there is a new result entry in the json file
5. end

Algorithm explanation:
Json file is easy to load using *json.load()* method by passing file path to file as parameter. *result.json* file is a dictionary (according to python programming) that means key and value pair. From this dictionary we have have taken list of value for the key *result*. We have iterated through this list and for every result value one new .json file is created. We repeat step 3 to fetch next result until it is empty. Thus we can separate information of different frame into different json file.

After getting two coordinates, height and width of boxes from densecap model, we have calculated four coordinates of boxes using mathematical expression. A small program code is used to find coordinates and the algorithm of this code is written below,

Algorithm 3. Calculate (x1, y1) to (x4 to y4) from x, y, h, w

1. start
2. initialize x1, y1, h, w for every region
3. x2 = x1 + h and y2 = y1
4. x3 = x1 + h and y3 = y1 + w
5. x4 = x1 and y4 = y1 + w
6. repeat 2 to 5 till there is a new region
7. store information into a file
8. end

Algorithm 3 Explanation:
We have x1, y1, h and w of every boxes so we first load them. Then (x2, y2), (x3, y3) and (x4, y4) are calculated using simple addition and subtraction. Final result is stored in a new file.

Now we have to group regions using coordinate information. Initially every large boxes gives generic information about a region. When we deeply caption that region we get more information of that region. Therefore, we have to find which information is relevant to a region or not. Regions are completely inclusive in some case and partially inclusive in other cases. By comparing coordinate information of regions, we have decided which boxes are completely inclusive or partially inclusive to a region. Mathematically it can be shown by the following formulas.

Consider the four coordinates of a large box are,
(X1, Y1), (X2, Y2), (X3, Y3), (X4, X4)

And consider the four coordinates of a small box are,
(X1', Y1'), (X2', Y2'), (X3', Y3'), (X4', Y4')

Condition for complete inclusiveness is,
(X1' >= X1 AND Y1' >= Y1) AND (X2' <= X2 AND Y2' >= Y2)
AND (X3' <= X3 AND Y3' <= Y3) AND (X4' >= X4 AND Y4' <= Y4)

For partially inclusive boxes, we have modified this formula by adding or subtracting pixels from every coordinates. Therefore, the condition for partially inclusive boxes becomes,

Consider the four coordinates of a large box are,
(X1, Y1), (X2, Y2), (X3, Y3), (X4, X4)

And consider the four coordinates of a partially inclusive box are,
(X1', Y1'), (X2', Y2'), (X3', Y3'), (X4', Y4')

Mathematical formula for grouping small boxes,

(X1' + 10 >= X1 AND Y1' + 10 >= Y1) AND (X2' – 10 <= X2 AND Y2' + 10 >= Y2)
AND (X3' – 10 <= X3 AND Y3' – 10 <= Y3) AND (X4' + 10 >= X4 AND Y4' – 10 < = Y4)

Final result after performing this step is shown in below diagram,

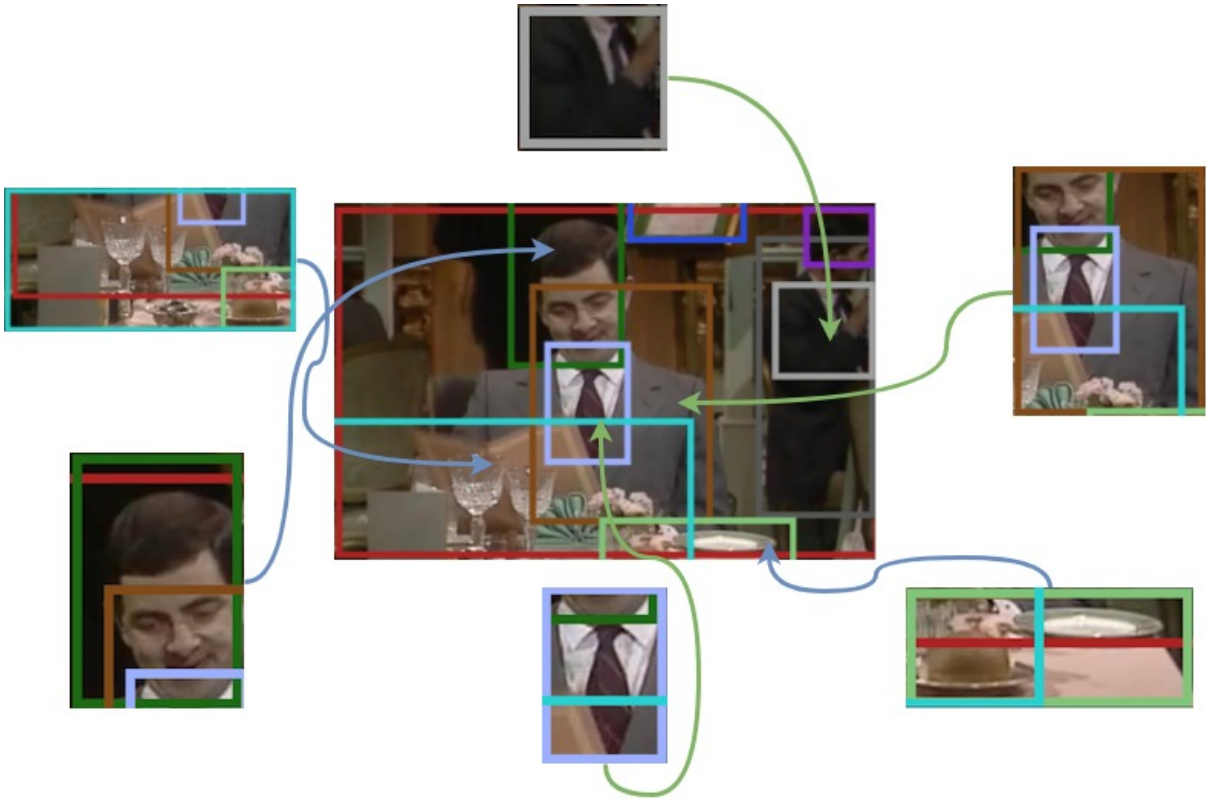

Image 5.3 Completely inclusive and partially inclusive boxes to a particular region

To achieve this work, we have used an algorithm which groups regions by comparing coordinates regions. This algorithm is shown below,

Algorithm 4. Grouping of regions

1. start
2. load coordinate information files
3. iterate through the list from first element to last
    a. iterate through the list from first element to last
        (X1' >= X1 AND Y1' >= Y1) AND (X2' <= X2 AND Y2' >= Y2)
        i. AND (X3' <= X3 AND Y3' <= Y3) AND (X4' >= X4 AND Y4' <= Y4) AND I != J
            1. Add that box
                (X1' + 10 >= X1 AND Y1' + 10 >= Y1) AND (X2' – 10 <= X2 AND Y2' + 10 >= Y2)
        i. AND (X3' – 10 <= X3 AND Y3' – 10 <= Y3) AND (X4' + 10 >= X4 AND Y4' – 10 < = Y4) and I != J
            a. Add that box
4. Store final list into file system

Algorithm 4 Explanation:
First load .json file using *json.load()* method. Now for every boxes compare coordinate of other boxes with this one using the complete inclusive and partially inclusive conditions. Final list is stored in local file system using *json.dump()* method.

Now, we have grouped regions with there coordinate information in a file. This information is used to map region to its captions. An algorithm is shown here to achieve this task,

Algorithm 5. Mapping of caption

1. Start
2. Load grouped region file and region caption file

3. Compare caption index with region index
    a. If caption index matches with region index
        i. Store captions
    b. Else discard the caption
4. Store grouped captions into another file.
5. Repeat 2 to 4 for every regions
6. End

Algorithm 5. Explanation:
Load region information file using *json.load()* method and pass path to the file as parameter. In this case captions are dictionary of list of dictionary (in python). For every caption index, match caption index with region index. And finally store that information.

This grouped captions are converted to sentences. Sentence conversion algorithm is shown below,

Algorithm 6. Sentence Making
1. Start
2. Load caption files
    a. Take first caption
    b. Add another caption with a ". "
3. Store sentence into file.
4. Repeat 2 to 3 for all caption files
5. End

Algorithm 6. Explanation:
Load caption files from local system. Captions are a python dictionary. We take one caption from this dictionary and join with another caption with ". " separation. Finally, we store that sentences into new files.

The final step is summarization. Here, we have taken grouped sentences and search for most common noun within those sentences. Then using target verbs and articles, we have joined those sentences.

Algorithm 7. Summarization
1. Start
2. Load sentence list
3. Take first grouped sentences
    a. Tokenize sentence using *nltk* to find *noun, verbs and articles*
    b. Find most common noun, verb and article
    c. Join noun, verbs and articles to make meaningful sentences
4. Store that sentence into a list.
5. End

Algorithm 7 Explanation:
Load sentence file. From this sentence file take the first group sentences. This group is tokenize using *natural language processing toolkit*. We find most common noun, verb and article from this tokenize sentences. Then join those to make meaningful summarize sentence.

Next section explains the details of this algorithm and sample code with input and output is given.

6. Implementation Details

First we have taken a sample video as a raw input for our approach. Videos that are generally available, are of 30 frames per second or some of them are of 60 frames per second. We have noticed that change is object is not very prominent from subsequent frames so we need to skip some of the frames. Initially we have taken a video which is 5 minutes long and 30 frames per second. After extracting frames from this video we get around 9000 frames. However, changes of object in those frames are very tough to detect. Next we decided to extract frames from a short length video around 20 seconds long. Incidentally that also became useless as it produces 600 frames. Therefore, finally we have solved this problem by extracting one frames per second which gives us significantly less number of frames and the change of object is also very prominent. Here is the program code to for film extraction,

Code(python):
```
import cv2
import math
# input path to video
```

```
videoFile = str(input("Enter input to video:"))
# output path for frames
framesFolder = str(input("Enter output path for frames:"))
cap = cv2.VideoCapture(videoFile)
# find frame rate of video
frameRate = cap.get(5)
# iterate through the loop till video is open
while cap.isOpened() :
#     current frame number
    frameId = cap.get(1)
# take one frame
    ret, frame = cap.read()
    if (ret!= True) :
        break
# frameId is divisible by frameRate
    if int(frameId) % math.floor(frameRate) == 0 :
# create new frame file
        filename = framesFolder + "/image_" + str(int(frameId)) + ".jpg"
# write frame file to local system
        cv2.imwrite(filename, frame)
# release video file
cap.release()
# print completion flag
print "Done!"
```

Here we skipped frames using '*if int(frameId) % math.floor(frameRate) == 0' statement.*

From an intermediate file of densecap model we extracted the coordinates and captioning information of regions. We have used the following python program to achieve this work,

Code(python):

```
import json
import os
i = 0
# take file input from local system
for file1 in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/
citmore/citmoreresults/') :
    print(file1)
# Load one file
    s = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
citmoreresults/{}'.format(file1)))
# find values for the key 'results' from dictionary s
    for imginfo in s['results'] :
# store the file in local system
        json.dump(imginfo, open('/Users/arin/Desktop/final_yr_project/
frame_extract/citmore/imginfodir/imginfo{}.json'.format(i), 'w'))
        i = i + 1
```

All frame extraction information is stored inside one file after dense captioning. Above code separates this information into separate files.

For every region we have coordinate information of one vertex along with this height and width. From this information we have calculated all four coordinate of a region. We have used below python code to find this information,

Code(python):

```
import os
i = 0
# load image information directory
for file in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
imginfodir/') :
# load image information file
    s = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
imginfodir/{}'.format(file)))
    l = []
# take the list from dictionary s for the key 'boxes'
    for crd in s['boxes'] :
```

```
        x1, y1 = crd[0], crd[1]
        h, w = crd[2], crd[3]
#         print('x1:{} y1:{} h:{} w:{}'.format(x1, y1, h, w))
# for top right corner
        x2, y2 = x1 + h, y1
# for bottom right corner
        x3, y3 = x1 + h, y1 + w
# for left bottom corner
        x4, y4 = x1, y1 + w
        t = (x1, y1, x2, y2, x3, x3, x4, y4)
#         print('x1:{} y1:{} x2:{} y2:{} x3:{} y3:{} x4:{} y4:{}
\n'.format(x1,y1,x2,y2,x3,y3,x4,y4))
# append this information into a new list
        l.append(t)
# store this list into a file and save it into local system
    json.dump(l, open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
imgcrd/imgcrdinfo{}.json'.format(i), 'w'))
    print(l)
    print('----------------------')
    i = i + 1
    l = []
```

After running this code, we have coordinate information of vertices of each region. This information is used in next step for grouping regions.

Output:

```
(448.91363525391, 227.10025024414, 559.01666259766, 227.10025024414,
559.01666259766, 559.01666259766, 448.91363525391, 364.63320922852),
(297.74209594727, 70.234451293945, 339.472137451176, 70.234451293945,
339.472137451176, 339.472137451176, 297.74209594727, 235.529373168945),
(5.6920547485352, 66.218231201172, 167.2067947387652, 66.218231201172,
167.2067947387652, 167.2067947387652, 5.6920547485352, 200.411987304692)
```

Grouping of regions is our next task. Here we have considered partially inclusive and completely inclusive boxes. For completely inclusive boxes we just have to compare the coordinates of one box with another one. However, for partially inclusive boxes, we have to manipulate the coordinates. Formula for this comparison is already mentioned in methodology of implementation section. Python code is given here,

Code(python):

```
import os
import json
k = 0
# load image coordinate directory
for file in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
imgcrd/'):
# load image coordinate file
    l = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
imgcrd/{}'.format(file)))
    inclv = {}
    i = 0
# iterate through the list
    for crd in l[:]:
        j = 0
        temp = {}
        for test in l[:]:
#             for complete inclusive boxes
            if (test[0]>=crd[0] and test[1]>=crd[1]) and (test[2]<=crd[2] and
test[3]>=crd[3]) and (test[4]<=crd[4] and test[5]<=crd[5]) and (test[6]>=crd[6] and
test[7]<=crd[7]) and i != j :
                temp[j] = test
#             for partly inclusive boxes
            elif (test[0] + 10 >= crd[0] and test[1] + 10 >= crd[1]) and (test[2] -
10 <= crd[2] and test[3] + 10 >= crd[3]) and (test[4] - 10 <= crd[4] and test[5] -
10 <= crd[5]) and (test[6] + 10 >= crd[6] and test[7] - 10 <= crd[7]) and i != j:
                temp[j] = test
            j = j + 1
        if len(temp.keys()) == 0:
            inclv[i] = {}
```

```
        else :
            inclv[i] = temp
        i = i + 1
 # store file into disk
    json.dump(inclv, open('/Users/arin/Desktop/final_yr_project/frame_extract/
citmore/grouping/boxgrouping{}.json'.format(k), 'w'))
    k = k + 1
```

Output:

```
{0: {}, 1: {}, 2: {17: [21.303546905518, 56.571365356445, 104.890872955323,
56.571365356445, 104.890872955323, 104.890872955323, 21.303546905518,
115.705902099609]}, 3: {}, 4: {6: [717.78485107422, 220.7709197998,
720.48358154297, 220.7709197998, 720.48358154297, 720.48358154297, 717.78485107422,
292.44253540038596]}
```

From the output of this above code we can notice that for region no. 2, region no. 17 is inclusive to it. Again for region no. 4, region no. 6 is also inclusive.

The grouped regions are mapped by using captions from densecap model. We match index of regions with index of captions for this mapping to perform. Below python code and output explain this work.

Code(python):

```
import os
import json
x = 0
y = 1
g = 0
i = 0
cpt = {}
# load grouping information directory
for file in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
grouping/'):
# load grouping files
l = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
grouping/{}'.format(file)))
    if i == 2:
        i = 0
        y = y + 1
# load captioning file output from densecap
    r = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
citmoreresults/results{}.json'.format(y)))
# take list from the dictionary with the key name 'captions'
    c = r['results'][i]['captions']
    for key in l:
        cptlist = []
        for k in l[key].keys():
            t = {k:c[int(k)]}
            cptlist.append(t)
        cpt[str((key, c[int(key)]))] = cptlist
    i = i + 1

# store information into a new file
json.dump(cpt, open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
grptocpt/cpt{}.json'.format(g), 'w'))
    g = g + 1
```

Output:

```
"('0', 'a man on a motorcycle')": [], "('1', 'a light pole')": [], "('2', 'a sign
on the building')": [{'17': 'a blue and white sign'}], "('3', 'a man on a
motorcycle')": [], "('4', 'white van parked on the street')": [{'6': 'white van
parked on the street'}], "('5', 'a man walking down the street')": [], "('6',
'white van parked on the street')": [],
```

After we get grouped-captions, we have to find sentences from using grouped captions. Below python code is used to make sentences,

Code(python):

```
import os
```

```
import json
textsent = []
j = 0
x = 0
i = 0
while x <= 9:
# load groupcaption folder
    data = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/
citmore/grptocpt/cpt{}.json'.format(x)))
    keylist = list(data.keys())
    i = 0
# iterate this loop for the length of keylist
    while i < len(keylist):
# take one grouped caption
        cap = list(data[keylist[i]])
#         print('################cpt{}.json'.format(x))
#         print(cap, len(cap))

        if len(cap) > 1:
            capdict = dict()
# take index value pair from cap list
            for num, items in enumerate(cap):
                for key in items.keys():
                    capdict[num] = items[key]
            text = set([capdict[key] for key in capdict.keys()])
#             print(text)
#             print('--------------')
            mystring = ""
            for sen in list(text):
                mystring = mystring + sen + ". "
            textsent.insert(j, str(mystring))
            j = j + 1
#         else:
#             print("not added")

        i = i + 1
    x = x + 1
# print (list(set(textsent))[1], len(set(textsent)),type(textsent),
type(textsent[0]))
for item in list(set(textsent)) :
    print(item)
    print('----------')
# store textsent list after eliminating duplicate
json.dump(list(set(textsent)),open('/Users/arin/Desktop/final_yr_project/
frame_extract/citmore/citmoreresults/textsent.json','w'))
```

Output:

```
a man wearing a black shirt. man wearing a black shirt.
----------
a building with a yellow roof. a white car in the background. a white building with
a red roof. a sign on the building. a white sign on the side of the road. a light
pole in the background. a sign on a building.
----------
blue and white bus. a man walking on the sidewalk. window on a bus. the bus has a
white stripe. the bus has a black stripe. a bus. a bus on the side of the road.
windows on the bus. windows on the side of the bus. the bus is black. the bus is
blue. the bus has a yellow stripe. a building in the background. a blue and white
building. the bus is blue and white.
```

The output of the above code convert grouped captions into sentences. This sentences are summarized to get the final summarized sentence.

The Summarization code is written below,

Code (python):

```
from collections import Counter
import nltk
import re
```

```
import json

def commonprefix(m):
    if not m:
        return ''
s1 = min(m)
s2 = max(m)
for i, c in enumerate(s1):
        if c != s2[i]:
            return s1[:i]
    return s1

def makesent(text):
    sents = list(filter(lambda x: x!= '', [line.strip() for line in
text.split('.')]))
tags, target_sents_NN, target_sents_VBG, req_lines = list(), list(), list(), list()
    dummylines, objects = list(), list()
    lookup, target = dict(), dict()
    # use nlkt to tokenize sentences
    for line in sents:
        parse = nltk.word_tokenize(line)
        tags.append(nltk.pos_tag(parse))
    counts = Counter([item for line in tags for item in line])

    targetVBG = None
    targetNN = None
# find most common noun
    for item in counts.most_common():
        if item[0][1] == 'NN':
            targetNN = item[0][0]
            break
    # find most common article
    for item in counts.most_common() :
        if item[0][1] == 'DT':
            targetDT = item[0][0]
            break

    for key, value in counts.items():
        lookup.setdefault(key[1],[]).append(key[0])

    for line in tags:
        for word in line:
            if word[0] == targetNN:
                target_sents_NN.append(line)
#find most common verbs
    for item in Counter([item for line in target_sents_NN for item in
line]).most_common():
        if item[0][1] == 'VBG':
            targetVBG = str(item[0][0])
            break

    if targetVBG is None:
         return;
    elif targetNN is None:
         return;
    else :
        for line in target_sents_NN:
            if (targetNN, 'NN') in line and (targetVBG, 'VBG') in line:
                target_sents_VBG.append(line)

# join noun, verbs and articles
    if 'VBZ' not in lookup:
         pass
    else:
        VBZ_list = lookup['VBZ']
target_prefix = [' '.join([targetNN, targetVBG])]
        target_prefix.append(' '.join([targetDT, targetNN, targetVBG]))
for vbz in VBZ_list:
            target_prefix.append(' '.join([targetDT, targetNN, vbz, targetVBG]))
```

```
        for line in sents:
            for tp in target_prefix:
                if line.startswith(tp):
                    req_lines.append(line)

        for line in req_lines:
            dummylines.append(line.replace(targetDT + ' ', ''))

        prefix = commonprefix(dummylines)

        for line in dummylines:
            line = re.sub(prefix, '',line)
            if 'and ' in line:
                diff = list()
                diff = line.split(' and ')
                for d in diff:
                    objects.append(d)
                    break
            else:
                objects.append(line)

        obj = list(set(objects))
         obj.sort()
        obj.sort(key=len)
         req_objects = []
         req_objects = list(filter(lambda x: [x for i in obj if x in i and x != i]
== [], obj))
         if len(req_objects) > 0:
            final_text = ' '.join([targetDT, prefix]) + ', '.join(req_objects[:-1])
+ ' and ' + req_objects[-1]
            mysummary.append(final_text.capitalize())


if __name__ == '__main__':
    mysummary = []
# load sentences file
    mytext = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/
citmore/citmoreresults/textsent.json'))
for item in mytext:
        makesent(item)
# print summarize text
    print(str(mysummary))
```

Output:

```
'A man wearing tie, black suit and white shirt'
```

7. Result

We have used 3 different real world video and tested this system to identify the human event. Through the final result depends on the target result of every step through which we have reached to the final step, however the summarization step or the final step has significant effect on the final output of the total project. Below, we have shown 3 videos and final summarized text as output from the system.

Video 1:

Output:
A man wearing black coat, white shirt and a tie.

Video 2:

Output:
A man wearing hat, black shirt and black helmet. A man wearing blue shirt, black shirt, white shirt and black helmet. A man wearing black shirt, black shoes and green shirt.

Video 3:

Output:
A railing, black road and black car.

8. Limitations

This project has two main limitations. First one we have faced at the time of dense captioning. DenseCap in a newly build tools and the accuracy of this tool depends on how well it is trained as we have said in the *implementation details* section of the report. After detailed diagnosis, we have identified that machine configuration is an important area of concern. Initially, we were using a machine with an INTEL i5 processor with 4gb of RAM. Processing single image is taking a bit long time when the image size increases to 100kb, the machine is not showing any output / it is not responding. Sometimes terminal shows out of memory error directly. Then we have come to know that a machine with i5/i7 processor and 8gb+ RAM would be more reliable for running this model. This machine can process a bit large image without any critical issue. We can also run more than one image smoothly. The accuracy level of this model can be improved by training this model with large dataset which can be downloaded from the VISUAL GENOME WEBSITE and region description. Some dataset that we have identified for training dense-cap model, are of size 9 to 12 GB. Huge dataset like the above one also demands premium machine with high hardware configuration.

Second limitation we have faced at the final step of this project. Every region gives a chunk of sentences, generally the number of sentences ranges from 10 to 15. Therefore, before the final step we have a huge collection of sentences. Now we need a summarization algorithm to summarize this huge collection of sentences. Designing of a summarization algorithm is itself an open research which takes significant time and effort. We have to design a simple summarization algorithm which is not perfect for every kind of input. This input specific algorithm also effects our final result of summarization.

9. Conclusion

Starting from the frame extraction step to the finishing step i.e. summarization step, we have faced lot of difficulty to achieve our short term goal or targeted result of each steps. To over come our hurdles, we take help from various algorithm and mathematical formulas. Though the final result is not accurate but we have gradually identified the limitations and the area of betterment of this project.

Better design of summarization algorithm and correct information extractions from the video may reasonably increase the performance of this system. Project like this one, identification of event can be improved with the help of machine learning tools. If we can train a model using video, then we might get better and accurate results. Identification of the story behind a video can be the extended part of this project. Critical store of a video can help us to solve real world problem. If a machine can identify the fact of a video, then it can decide what to do with respect to the fact. This type of intelligence can be used in CCTV cameras to identify critical situation and automation of emergency steps to minimize the effect of accidents like fire, child drowning into pools and many more. Considering all areas, finally we can say project and development on machine learning and computer vision has a very good impact on mankind.

Appendix

Video Sample 1

Random video as input

Extraction of frames from video

Step 1: frame extraction

Code:

```
import cv2
import math
# input path to video
videoFile = str(input("Enter input to video:"))
# output path for frames
framesFolder = str(input("Enter output path for frames:"))
cap = cv2.VideoCapture(videoFile)
# find frame rate of video
frameRate = cap.get(5)
while cap.isOpened() :
#     current frame number
    frameId = cap.get(1)
    ret, frame = cap.read()
    if (ret != True) :
        break
    if int(frameId) % math.floor(frameRate) == 0 :
        filename = framesFolder + "/image_" + str(int(frameId)) + ".jpg"
#         cv2.imshow(filename, frame)
        cv2.imwrite(filename, frame)
cap.release()
print "Done!"
```

Step 2: fetching captioning information and coordinate information form results to seperate .json file

Code:

```
import json
import os
i = 0
for file1 in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/
citmore/citmoreresults/') :
    print(file1)
    s = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
citmoreresults/{}'.format(file1)))
#     print(s)
    for imginfo in s['results'] :
        json.dump(imginfo, open('/Users/arin/Desktop/final_yr_project/
frame_extract/citmore/imginfodir/imginfo{}.json'.format(i), 'w'))
        i = i + 1
```

Step 3: calculating (x1, y1) to (x4, y4) from x, y, h, w information

Code:

```
import os
i = 0
for file in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
imginfodir/') :
    s = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
imginfodir/{}'.format(file)))
    l = []
    for crd in s['boxes'] :
        x1, y1 = crd[0], crd[1]
        h, w = crd[2], crd[3]
#         print('x1:{} y1:{} h:{} w:{}'.format(x1, y1, h, w))
        x2, y2 = x1 + h, y1
        x3, y3 = x1 + h, y1 + w
        x4, y4 = x1, y1 + w
        t = (x1, y1, x2, y2, x3, x3, x4, y4)
#         print('x1:{} y1:{} x2:{} y2:{} x3:{} y3:{} x4:{} y4:{}
\n'.format(x1,y1,x2,y2,x3,y3,x4,y4))
        l.append(t)
    json.dump(l, open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
imgcrd/imgcrdinfo{}.json'.format(i), 'w'))
    print(l)
    print('-----------------------')
    i = i + 1
    l = []
```

Output:

```
(448.91363525391, 227.10025024414, 559.01666259766, 227.10025024414,
559.01666259766, 559.01666259766, 448.91363525391, 364.63320922852),
(297.74209594727, 70.234451293945, 339.472137451176, 70.234451293945,
339.472137451176, 339.472137451176, 297.74209594727, 235.529373168945),
(5.6920547485352, 66.218231201172, 167.2067947387652, 66.218231201172,
167.2067947387652, 167.2067947387652, 5.6920547485352, 200.411987304692),
(625.05224609375, 244.04396057129, 701.566772460938, 244.04396057129,
701.566772460938, 701.566772460938, 625.05224609375, 356.79931640625),
(695.56787109375, 226.72953796387, 720.484008789062, 226.72953796387,
720.484008789062, 720.484008789062, 695.56787109375, 285.914825439456),
(164.02105712891, 244.55447387695, 222.870239257816, 244.55447387695,
222.870239257816, 222.870239257816, 164.02105712891, 322.21725463866903),
(717.78485107422, 220.7709197998, 720.48358154297, 220.7709197998, 720.48358154297,
720.48358154297, 717.78485107422, 292.44253540038596), (-14.858428955078,
32.760116577148, 633.414154052732, 32.760116577148, 633.414154052732,
633.414154052732, -14.858428955078, 393.246688842768), (2.6283340454102,
232.95195007324, 163.3941421508802, 232.95195007324, 163.3941421508802,
163.3941421508802, 2.6283340454102, 396.17407226562), (666.83520507812,
233.66798400879, 711.857543945308, 233.66798400879, 711.857543945308,
711.857543945308, 666.83520507812, 286.060363769532), (318.61865234375,
20.845993041992, 719.10754394531, 20.845993041992, 719.10754394531,
719.10754394531, 318.61865234375, 275.391601562502), (684.61572265625,
250.57122802734, 719.949462890625, 250.57122802734, 719.949462890625,
719.949462890625, 684.61572265625, 319.262451171871), (615.12701416016,
-5.2597198486328, 659.354797363285, -5.2597198486328, 659.354797363285,
659.354797363285, 615.12701416016, 292.04039000146472), (369.5666809082,
297.72595214844, 559.09118652343, 297.72595214844, 559.09118652343,
559.09118652343, 369.5666809082, 391.540832519534), (403.12246704102,
226.34155273438, 720.02661132813, 226.34155273438, 720.02661132813,
720.02661132813, 403.12246704102, 390.00964355469), (714.27038574219,
246.67366027832, 720.44177246094, 246.67366027832, 720.44177246094,
720.44177246094, 714.27038574219, 315.564636230468), (138.25224304199,
228.68853759766, 195.42448425292798, 228.68853759766, 195.42448425292798,
195.42448425292798, 138.25224304199, 333.96728515625), (21.303546905518,
56.571365356445, 104.890872955323, 56.571365356445, 104.890872955323,
104.890872955323, 21.303546905518, 115.705902099609), (500.72549438477,
136.01525878906, 546.308593750004, 136.01525878906, 546.308593750004,
546.308593750004, 500.72549438477, 236.63259887695), (449.24472045898,
29.345222473145, 522.478820800777, 29.345222473145, 522.478820800777,
522.478820800777, 449.24472045898, 237.65393829345498), (576.91687011719,
169.72654724121, 650.333251953128, 169.72654724121, 650.333251953128,
650.333251953128, 576.91687011719, 292.08203125), (313.64096069336,
209.80198669434, 552.86010742188, 209.80198669434, 552.86010742188,
552.86010742188, 313.64096069336, 276.876098632817), (187.64819335938,
106.2624206543, 247.930480957036, 106.2624206543, 247.930480957036,
247.930480957036, 187.64819335938, 222.76492309571), (246.06643676758,
209.87301635742, 380.62753295899, 209.87301635742, 380.62753295899,
380.62753295899, 246.06643676758, 262.208526611326), (140.36820983887,
6.3952255249023, 409.97532653809003, 6.3952255249023, 409.97532653809003,
409.97532653809003, 140.36820983887, 246.5432052612323), (455.35916137695,
213.94621276855, 509.173187255856, 213.94621276855, 509.173187255856,
509.173187255856, 455.35916137695, 307.42098999023), (443.76116943359,
6.2064895629883, 723.30401611328, 6.2064895629883, 723.30401611328,
723.30401611328, 443.76116943359, 136.8271560668983), (606.26556396484,
211.52726745605, 714.88153076172, 211.52726745605, 714.88153076172,
714.88153076172, 606.26556396484, 285.481506347652), (280.40124511719,
183.46530151367, 339.884704589846, 183.46530151367, 339.884704589846,
339.884704589846, 280.40124511719, 244.961181640623), (2.4079933166504,
264.29257202148, 100.8393745422364, 264.29257202148, 100.8393745422364,
100.8393745422364, 2.4079933166504, 338.495819091792), (578.9599609375,
122.87017822266, 659.507934570312, 122.87017822266, 659.507934570312,
659.507934570312, 578.9599609375, 200.291351318363), (604.52215576172,
243.52014160156, 657.386169433595, 243.52014160156, 657.386169433595,
657.386169433595, 604.52215576172, 324.066772460935), (676.70654296875,
210.58619689941, 719.981201171875, 210.58619689941, 719.981201171875,
719.981201171875, 676.70654296875, 247.63407897948798), (702.24212646484,
```

```
165.4997253418, 719.21282958984, 165.4997253418, 719.21282958984, 719.21282958984,
702.24212646484, 209.745300292972), (426.3434753418, 126.49272155762,
719.77490234375, 126.49272155762, 719.77490234375, 719.77490234375, 426.3434753418,
205.516891479495), (16.335754394531, 144.59031677246, 152.686828613281,
144.59031677246, 152.686828613281, 152.686828613281, 16.335754394531,
244.91062927246), (38.155921936035, 36.280319213867, 251.40705108642499,
36.280319213867, 251.40705108642499, 251.40705108642499, 38.155921936035,
343.126449584957), (98.519454956055, 92.517318725586, 187.58076477050798,
92.517318725586, 187.58076477050798, 187.58076477050798, 98.519454956055,
210.814071655276), (419.), (210.89366149902, 288.48266601562, 268.2483215332,
288.48266601562, 268.2483215332, 268.2483215332, 210.89366149902, 374.07495117187)]
```

Step 4: grouping boxes togather for a particular object, save it in a separate file into /Users/arin/Desktop/final_yr_project/frame_extract/citmore/grouping/{}

```
import os
import json
k = 0
for file in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
imgcrd/'):
    l = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
imgcrd/{}'.format(file)))
    inclv = {}
    i = 0
    for crd in l[:]:
        j = 0
        temp = {}
        for test in l[:]:
#             for complete inclusive boxes
            if (test[0]>=crd[0] and test[1]>=crd[1]) and (test[2]<=crd[2] and
test[3]>=crd[3]) and (test[4]<=crd[4] and test[5]<=crd[5]) and (test[6]>=crd[6] and
test[7]<=crd[7]) and i != j :
                temp[j] = test
#             for partly inclusive boxes
            elif (test[0] + 10 >= crd[0] and test[1] + 10 >= crd[1]) and (test[2] -
10 <= crd[2] and test[3] + 10 >= crd[3]) and (test[4] - 10 <= crd[4] and test[5] -
10 <= crd[5]) and (test[6] + 10 >= crd[6] and test[7] - 10 <= crd[7]) and i != j:
                temp[j] = test
            j = j + 1
        if len(temp.keys()) == 0:
            inclv[i] = {}
        else :
            inclv[i] = temp
        i = i + 1
    print(inclv)
    print('------------------')
    json.dump(inclv, open('/Users/arin/Desktop/final_yr_project/frame_extract/
citmore/grouping/boxgrouping{}.json'.format(k), 'w'))
    k = k + 1
```

Output:

```
0: {}, 1: {}, 2: {17: [21.303546905518, 56.571365356445, 104.890872955323,
56.571365356445, 104.890872955323, 104.890872955323, 21.303546905518,
115.705902099609]}, 3: {}, 4: {6: [717.78485107422, 220.7709197998,
720.48358154297, 220.7709197998, 720.48358154297, 720.48358154297, 717.78485107422,
292.44253540038596]}, 5: {}, 6: {}, 7: {0: [448.91363525391, 227.10025024414,
559.01666259766, 227.10025024414, 559.01666259766, 559.01666259766,
448.91363525391, 364.63320922852], 1: [297.74209594727, 70.234451293945,
339.472137451176, 70.234451293945, 339.472137451176, 339.472137451176,
297.74209594727, 235.529373168945], 2: [5.6920547485352, 66.218231201172,
167.2067947387652, 66.218231201172, 167.2067947387652, 167.2067947387652,
5.6920547485352, 200.411987304692], 5: [164.02105712891, 244.55447387695,
222.870239257816, 244.55447387695, 222.870239257816, 222.870239257816,
164.02105712891, 322.21725463866903], 8: [2.6283340454102, 232.95195007324,
163.3941421508802, 232.95195007324, 163.3941421508802, 163.3941421508802,
2.6283340454102, 396.17407226562], 13: [369.5666809082, 297.72595214844,
559.09118652343, 297.72595214844, 559.09118652343, 559.09118652343, 369.5666809082,
```

391.540832519534], 16: [138.25224304199, 228.68853759766, 195.42448425292798,
228.68853759766, 195.42448425292798, 195.42448425292798, 138.25224304199,
333.96728515625], 17: [21.303546905518, 56.571365356445, 104.890872955323,
56.571365356445, 104.890872955323, 104.890872955323, 21.303546905518,
115.705902099609], 18: [500.72549438477, 136.01525878906, 546.308593750004,
136.01525878906, 546.308593750004, 546.308593750004, 500.72549438477,
236.63259887695], 19: [449.24472045898, 29.345222473145, 522.478820800777,
29.345222473145, 522.478820800777, 522.478820800777, 449.24472045898,
237.65393829345498], 21: [313.64096069336, 209.80198669434, 552.86010742188,
209.80198669434, 552.86010742188, 552.86010742188, 313.64096069336,
276.876098632817], 22: [187.64819335938, 106.2624206543, 247.930480957036,
106.2624206543, 247.930480957036, 247.930480957036, 187.64819335938,
222.76492309571], 23: [246.06643676758, 209.87301635742, 380.62753295899,
209.87301635742, 380.62753295899, 380.62753295899, 246.06643676758,
262.208526611326], 25: [455.35916137695, 213.94621276855, 509.173187255856,
213.94621276855, 509.173187255856, 509.173187255856, 455.35916137695,
307.42098999023], 28: [280.40124511719, 183.46530151367, 339.884704589846,
183.46530151367, 339.884704589846, 339.884704589846, 280.40124511719,
244.961181640623], 29: [2.4079933166504, 264.29257202148, 100.8393745422364,
264.29257202148, 100.8393745422364, 100.8393745422364, 2.4079933166504,
338.495819091792], 35: [16.335754394531, 144.59031677246, 152.686828613281,
144.59031677246, 152.686828613281, 152.686828613281, 16.335754394531,
244.91062927246], 36: [38.155921936035, 36.280319213867, 251.40705108642499,
36.280319213867, 251.40705108642499, 251.40705108642499, 38.155921936035,
343.126449584957], 37: [98.519454956055, 92.517318725586, 187.58076477050798,
92.517318725586, 187.58076477050798, 187.58076477050798, 98.519454956055,
210.814071655276], 38: [419.74407958984, 143.49041748047, 453.890441894528,
143.49041748047, 453.890441894528, 453.890441894528, 419.74407958984,
231.249053955079], 39: [315.37423706055, 128.2751159668, 360.896759033206,
128.2751159668, 360.896759033206, 360.896759033206, 315.37423706055,
243.91586303711], 40: [47.750457763672, 242.79281616211, 144.04913330078102,
242.79281616211, 144.04913330078102, 144.04913330078102, 47.750457763672,
315.52041625976597], 41: [313.82730102539, 225.14889526367, 384.653594970702,
225.14889526367, 384.653594970702, 384.653594970702, 313.82730102539,
280.74118041992], 42: [244.40829467773, 168.56246948242, 586.08801269531,
168.56246948242, 586.08801269531, 586.08801269531, 244.40829467773,
243.841705322264], 43: [138.36849975586, 25.266616821289, 364.90130615234,
25.266616821289, 364.90130615234, 364.90130615234, 138.36849975586,
107.517593383789], 44: [582.10528564453, 236.33387756348, 629.986267089842,
236.33387756348, 629.986267089842, 629.986267089842, 582.10528564453,
309.978027343753], 45: [477.48782348633, 199.77072143555, 547.036437988283,
199.77072143555, 547.036437988283, 547.036437988283, 477.48782348633,
264.70822143555], 46: [547.17181396484, 252.59045410156, 585.198791503902,
252.59045410156, 585.198791503902, 585.198791503902, 547.17181396484,
306.542724609372], 47: [151.67250061035, 218.28964233398, 260.15637207031,
218.28964233398, 260.15637207031, 260.15637207031, 151.67250061035,
275.860748291011], 49: [4.5660209655762, 195.84759521484, 128.0969657897962,
195.84759521484, 128.0969657897962, 128.0969657897962, 4.5660209655762,
283.297821044918], 50: [340.53628540039, 241.07583618164, 525.47937011719,
241.07583618164, 525.47937011719, 525.47937011719, 340.53628540039,
330.94857788085903], 52: [1.3072509765625, 28.064804077148, 87.6423873901365,
28.064804077148, 87.6423873901365, 87.6423873901365, 1.3072509765625,
334.708297729488], 53: [248.92810058594, 198.79270935059, 297.261169433596,
198.79270935059, 297.261169433596, 297.261169433596, 248.92810058594,
238.849655151371], 56: [547.38635253906, 222.29359436035, 610.59924316406,
222.29359436035, 610.59924316406, 610.59924316406, 547.38635253906,
278.938537597655], 57: [475.4440612793, 76.413902282715, 559.572143554691,
76.413902282715, 559.572143554691, 559.572143554691, 475.4440612793,
298.437736511235], 58: [130.93870544434, 218.03198242188, 526.66960144043,
218.03198242188, 526.66960144043, 526.66960144043, 130.93870544434,
391.48846435547], 59: [503.77868652344, 273.62924194336, 578.630493164065,
273.62924194336, 578.630493164065, 578.630493164065, 503.77868652344,
326.43295288086], 60: [582.90399169922, 188.00549316406, 623.303039550782,
188.00549316406, 623.303039550782, 623.303039550782, 582.90399169922,
254.254302978513], 61: [419.65939331055, 200.7008972168, 495.65847778320597,
200.7008972168, 495.65847778320597, 495.65847778320597, 419.65939331055,
254.497711181644], 62: [183.51025390625, 43.508563995361, 239.343658447266,
43.508563995361, 239.343658447266, 239.343658447266, 183.51025390625,
143.464191436767], 63: [334.73049926758, 163.0982208252, 379.602996826174,

163.0982208252, 379.602996826174, 379.602996826174, 334.73049926758,
253.908950805669], 64: [400.96356201172, 57.64924621582, 477.49725341797,
57.64924621582, 477.49725341797, 477.49725341797, 400.96356201172,
234.72206115723], 66: [214.16561889648, 210.44905090332, 264.311706542964,
210.44905090332, 264.311706542964, 264.311706542964, 214.16561889648,
246.757797241211], 67: [318.84030151367, 29.146408081055, 390.92092895507596,
29.146408081055, 390.92092895507596, 390.92092895507596, 318.84030151367,
232.769790649415], 68: [41.53050994873, 306.18377685547, 113.644752502441,
306.18377685547, 113.644752502441, 113.644752502441, 41.53050994873,
394.228820800782], 70: [2.6925563812256, 29.929988861084, 62.5712413787846,
29.929988861084, 62.5712413787846, 62.5712413787846, 2.6925563812256,
155.71277999878401], 73: [332.05429077148, 125.77360534668, 542.04833984375,
125.77360534668, 542.04833984375, 542.04833984375, 332.05429077148,
216.04521179199202], 74: [191.70806884766, 181.00103759766, 405.56219482422,
181.00103759766, 405.56219482422, 405.56219482422, 191.70806884766,
290.8942565918], 75: [451.10684204102, 166.30229187012, 517.969543457036,
166.30229187012, 517.969543457036, 517.969543457036, 451.10684204102,
232.837539672854], 76: [8.8318710327148, 117.25373077393, 79.9512939453128,
117.25373077393, 79.9512939453128, 79.9512939453128, 8.8318710327148,
213.830596923832], 78: [253.19209289551, 74.088455200195, 366.03472900391,
74.088455200195, 366.03472900391, 366.03472900391, 253.19209289551,
144.088058471679], 79: [516.75286865234, 180.68209838867, 573.64300537109,
180.68209838867, 573.64300537109, 573.64300537109, 516.75286865234,
273.010253906248], 81: [55.881805419922, 65.820022583008, 191.84633,
199.77072143555, 547.036437988283, 199.77072143555, 547.036437988283,
547.036437988283, 477.48782348633, 264.70822143555], 48: [677.43475341797,
139.41126708984, 273.16369628906, 674.33441162109, 273.16369628906,
674.33441162109, 674.33441162109, 539.11126708984, 342.706726074216], 96:
[709.4581909179 719.21282958984, 719.21282958984, 702.24212646484,
209.745300292972], 48: [677.43475341797, 139.40336608887, 715.128356933595,
139.40336608887, 715.12835693303808594, 105.12034606934, 666.317382812502,
105.12034606934, 666.317382812502, 666, 225.14889526367, 384.653594970702,
384.653594970702, 313.82730102539, 280.74118041992], 53: [248.92810058594,
198.79270935059, 297.261169433596, 198.79270935059, 297.261169433596,
297.261169433596, 248.92810058594, 238.849655151371], 66: [214.16561889648,
210.44905090332, 264.311706542964, 210.44905090332, 264.311706542964,
264.311706542964, 214.16561889648, 246.757797241211], 82: [221.22888183594,
173.12124633789, 330.91369628906, 173.12124633789, 330.91369628906,
330.91369628906, 221.22888183594, 225.623474121093], 83: [355.21563720703,
205.3994140625, 404.810546874999, 205.3994140625, 404.810546874999,
404.810546874999, 355.21563720703, 279.686218261719], 103: [207.92926025391,
222.80574035645, 319.87951660157, 222.80574035645, 319.87951660157,
319.87951660157, 207.92926025391, 283.727874755864]}, 75: {}, 76: {}, 77: {}, 78:
{}, 79: {}, 80: {33: [702.24212646484, 165.4997253418, 719.21282958984,
165.4997253418, 719.21282958984, 719.21282958984, 702.24212646484,
209.745300292972], 48: [677.43475341797, 139.40336608887, 715.128356933595,
139.40336608887, 715.128356933595, 715.128356933595, 677.43475341797,
204.40071105957298], 71: [713.14001464844, 192.87799072266, 720.1953125000025,
192.87799072266, 720.1953125000025, 720.1953125000025, 713.14001464844,
235.765136718754], 72: [697.63092041016, 191.3943939209, 719.45416259766,
191.3943939209, 719.45416259766, 719.45416259766, 697.63092041016,
236.58299255371202]}, 81: {}, 82: {}, 83: {}, 84: {}, 85: {}, 86: {}, 87: {}, 88:
{}, 89: {}, 90: {}, 91: {}, 92: {112: [212.05619812012, 254.17930603027,
250.02067565918202, 254.17930603027, 250.02067565918202, 250.02067565918202,
212.05619812012, 317.208953857418]}, 93: {}, 94: {}, 95: {}, 96: {}, 97: {}, 98:
{}, 99: {}, 100: {53: [248.92810058594, 198.79270935059, 297.261169433596,
198.79270935059, 297.261169433596, 297.261169433596, 248.92810058594,
238.849655151371], 82: [221.22888183594, 173.12124633789, 330.91369628906,
173.12124633789, 330.91369628906, 330.91369628906, 221.22888183594,
225.623474121093], 110: [159.19024658203, 189.19012451172, 270.17025756836,
189.19012451172, 270.17025756836, 270.17025756836, 159.19024658203,
231.125762939454]}, 101: {}, 102: {}, 103: {}, 104: {}, 105: {111:
[536.18005371094, 4.4121675491333, 635.304443359378, 4.4121675491333,
635.304443359378, 635.304443359378, 536.18005371094, 31.9583520889283], 115:
[476.48675537109, 4.5265064239502, 577.12384033203, 4.5265064239502,
577.12384033203, 577.12384033203, 476.48675537109, 28.39670753479021]}, 106: {},
107: {}, 108: {}, 109: {}, 110: {}, 111: {}, 112: {}, 113: {}, 114: {}, 115: {},
116: {}}

Step 5: grouping of captions complete inclusive and partly inclusive updated

```
import os
import json
x = 0
y = 1
g = 0
i = 0
cpt = {}
for file in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
grouping/'):
    l = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
grouping/{}'.format(file)))
    if i == 2:
        i = 0
        y = y + 1
    r = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/citmore/
citmoreresults/results{}.json'.format(y)))
    c = r['results'][i]['captions']
    for key in l:
        cptlist = []
        for k in l[key].keys():
            t = {k:c[int(k)]}
            cptlist.append(t)
        cpt[str((key, c[int(key)]))] = cptlist
    i = i + 1
    print(cpt)
    print('----------------')
    json.dump(cpt, open('/Users/arin/Desktop/final_yr_project/frame_extract/
citmore/grptocpt/cpt{}.json'.format(g), 'w'))
    g = g + 1
```

Output:

{"('0', 'a man on a motorcycle')": [], "('1', 'a light pole')": [], "('2', 'a sign on the building')": [{'17': 'a blue and white sign'}], "('3', 'a man on a motorcycle')": [], "('4', 'white van parked on the street')": [{'6': 'white van parked on the street'}], "('5', 'a man walking down the street')": [], "('6', 'white van parked on the street')": [], "('7', 'a street scene')": [{'0': 'a man on a motorcycle'}, {'1': 'a light pole'}, {'2': 'a sign on the building'}, {'5': 'a man walking down the street'}, {'8': 'man with a black shirt'}, {'13': 'a motorcycle on the road'}, {'16': 'a man walking on the sidewalk'}, {'17': 'a blue and white sign'}, {'18': 'a street light'}, {'19': 'green trees in the background'}, {'21': 'a fence in the background'}, {'22': 'a tree in the distance'}, {'23': 'people walking on the sidewalk'}, {'25': 'a person wearing a helmet'}, {'28': 'a white sign on the sidewalk'}, {'29': 'man wearing a hat'}, {'35': 'a sign on the wall'}, {'36': 'a sign on the side of the road'}, {'37': 'a sign on the building'}, {'38': 'a light pole'}, {'39': 'a light pole'}, {'40': 'man with black hair'}, {'41': 'a blue and white sign'}, {'42': 'trees in the background'}, {'43': 'power lines above the street'}, {'44': 'a blue and white sign'}, {'45': 'a man wearing a hat'}, {'46': 'a white car'}, {'47': 'a man in a black shirt'}, {'49': 'a man with a black hat'}, {'50': 'a man on a bike'}, {'52': 'a large white sign'}, {'53': 'a white car in the background'}, {'56': 'a man in a red shirt'}, {'57': 'a light pole'}, {'58': 'a road with a lot of people'}, {'59': 'man wearing black helmet'}, {'60': 'a yellow and black sign'}, {'61': 'a green and white sign'}, {'62': 'a street light'}, {'63': 'a light pole in the background'}, {'64': 'a tree in the distance'}, {'66': 'a car in the background'}, {'67': 'a light pole'}, {'68': 'the elephant has a large ear'}, {'70': 'a blue and white sign'}, {'73': 'a street light'}, {'74': 'a white car in the background'}, {'75': 'a tall green tree'}, {'76': 'a sign on the building'}, {'78': 'a light pole'}, {'79': 'a red and white sign'}, {'81': 'a sign on the pole'}, {'82': 'a green tree in the distance'}, {'83': 'a fence in the background'}, {'84': 'white line on the road'}, {'86': 'a white sign on the sidewalk'}, {'87': 'a light pole'}, {'89': 'a light pole in the background'}, {'90': 'a fence in the background'}, {'92': 'a man walking on the sidewalk'}, {'93': 'a tall tree in the distance'}, {'94': 'the wheel of a motorcycle'}, {'95': 'a man in a red shirt'}, {'97': 'a yellow and white building'}, {'98': 'a man wearing a black shirt'}, {'99': 'a tall street light'}, {'100': 'a green tree in the distance'}, {'101': 'a motorcycle on the road'}, {'102': 'a fence in the background'}, {'103': 'a white line on the road'}, {'108': 'a light pole in the background'}, {'110': 'a green tree in the background'},

{'112': 'a white line on the road'}, {'114': 'a white sign on the pole'}, {'116': 'white line on the road'}], "('8', 'man with a black shirt')": [{'29': 'man wearing a hat'}, {'40': 'man with black hair'}, {'68': 'the elephant has a large ear'}], "('9', 'a white van on the street')": [{'4': 'white van parked on the street'}], "('10', 'tall trees on the sidewalk')": [{'18': 'a street light'}, {'19': 'green trees in the background'}, {'21': 'a fence in the background'}, {'30': 'yellow top of building'}, {'32': 'blue and white bus'}, {'33': 'a sign on the building'}, {'34': 'trees in the background'}, {'38': 'a light pole'}, {'39': 'a light pole'}, {'41': 'a blue and white sign'}, {'45': 'a man wearing a hat'}, {'48': 'a white building with a red roof'}, {'51': 'a tall light pole'}, {'55': 'a building with a lot of windows'}, {'56': 'a man in a red shirt'}, {'60': 'a yellow and black sign'}, {'61': 'a green and white sign'}, {'63': 'a light pole in the background'}, {'64': 'a tree in the distance'}, {'67': 'a light pole'}, {'71': 'blue and white sign on building'}, {'72': 'blue and white sign on the side of the road'}, {'73': 'a street light'}, {'75': 'a tall green tree'}, {'77': 'trees in the background'}, {'79': 'a red and white sign'}, {'80': 'a street light'}, {'83': 'a fence in the background'}, {'85': 'a yellow pole'}, {'89': 'a light pole in the background'}, {'90': 'a fence in the background'}, {'93': 'a tall tree in the distance'}, {'97': 'a yellow and white building'}, {'108': 'a light pole in the background'}], "('11', 'a white van parked on the street')": [{'15': 'a white van parked on the street'}], "('12', 'a tall light pole')": [{'85': 'a yellow pole'}], "('13', 'a motorcycle on the road')": [{'94': 'the wheel of a motorcycle'}], "('14', 'two people on a motorcycle')": [{'0': 'a man on a motorcycle'}, {'3': 'a man on a motorcycle'}, {'4': 'white van parked on the street'}, {'6': 'white van parked on the street'}, {'9': 'a white van on the street'}, {'11': 'a white van parked on the street'}, {'15': 'a white van parked on the street'}, {'31': 'a man in a blue shirt'}, {'44': 'a blue and white sign'}, {'46': 'a white car'}, {'56': 'a man in a red shirt'}, {'59': 'man wearing black helmet'}, {'65': 'a man on a motorcycle'}, {'96': 'a white car on the road'}, {'101': 'a motorcycle on the road'}], "('15', 'a white van parked on the street')": []ck sign')": [], "('61', 'a green and white sign')": [], "('62', 'a street light')": [], "('63', 'a light pole in the background')": [], "('64', 'a tree in the distance')": [{'38': 'a light pole'}, {'108': 'a light pole in the background'}], "('65', 'a man on a motorcycle')": [], "('66', 'a car in the background')": [], "('67', 'a light pole')": [], "('68', 'the elephant has a large ear')": [], "('69', 'a cloudy sky')": [{'104': 'a tree in the background'}, {'106': 'white clouds in blue sky'}], "('70', 'a blue and white sign')": [], "('71', 'blue and white sign on building')": [], "('72', 'blue and white sign on the side of the road')": [{'71': 'blue and white sign on building'}], "('73', 'a street light')": [], "('74', 'a white car in the background')": [{'23': 'people walking on the sidewalk'}, {'28': 'a white sign on the sidewalk'}, {'41': 'a blue and white sign'}, {'53': 'a white car in the background'}, {'66': 'a car in the background'}, {'82': 'a green tree in the distance'}, {'83': 'a fence in the background'}, {'103': 'a white line on the road'}], "('75', 'a tall green tree')": [], "('76', 'a sign on the building')": [], "('77', 'trees in the background')": [], "('78', 'a light pole')": [], "('79', 'a red and white sign')": [], "('80', 'a street light')": [{'33': 'a sign on the building'}, {'48': 'a white building with a red roof'}, {'71': 'blue and white sign on building'}, {'72': 'blue and white sign on the side of the road'}], "('81', 'a sign on the pole')": [], "('82', 'a green tree in the distance')": [], "('83', 'a fence in the background')": [], "('84', 'white line on the road')": [], "('85', 'a yellow pole')": [], "('86', 'a white sign on the sidewalk')": [], "('87', 'a light pole')": [], "('88', 'a tall pole in the background')": [], "('89', 'a light pole in the background')": [], "('90', 'a fence in the background')": [], "('91', 'a telephone pole')": [], "('92', 'a man walking on the sidewalk')": [{'112': 'a white line on the road'}], "('93', 'a tall tree in the distance')": [], "('94', 'the wheel of a motorcycle')": [], "('95', 'a man in a red shirt')": [], "('96', 'a white car on the road')": [], "('97', 'a yellow and white building')": [], "('98', 'a man wearing a black shirt')": [], "('99', 'a tall street light')": [], "('100', 'a green tree in the distance')": [{'53': 'a white car in the background'}, {'82': 'a green tree in the distance'}, {'110': 'a green tree in the background'}], "('101', 'a motorcycle on the road')": [], "('102', 'a fence in the background')": [], "('103', 'a white line on the road')": [], "('104', 'a tree in the background')": [], "('105', 'power lines in the sky')": [{'111': 'power lines in the sky'}, {'115': 'white clouds in blue sky'}], "('106', 'white clouds in blue sky')": [], "('107', 'a tree in the distance')": [], "('108', 'a light pole in the background')": [], "('109', 'a tree trunk')": [], "('110', 'a green tree in the background')": [], "('111', 'power lines in the sky')": [], "('112', 'a white line on the road')": [], "('113', 'power lines in the sky')": [], "('114', 'a white sign on the pole')": [], "('115', 'white clouds in blue sky')": [], "('116', 'white line on the road')": []}

Step 6: create sentences

```
import os
import json
textsent = []
j = 0
x = 0
i = 0
while x <= 9:
    data = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/
citmore/grptocpt/cpt{}.json'.format(x)))
    keylist = list(data.keys())
    i = 0
    while i < len(keylist):
        cap = list(data[keylist[i]])
#         print('###############cpt{}.json'.format(x))
#         print(cap, len(cap))

        if len(cap) > 1:
            capdict = dict()
            for num, items in enumerate(cap):
                for key in items.keys():
                    capdict[num] = items[key]
            text = set([capdict[key] for key in capdict.keys()])
#             print(text)
#             print('---------------')
            mystring = ""
            for sen in list(text):
                mystring = mystring + sen + ". "
            textsent.insert(j, str(mystring))
            j = j + 1
#         else:
#             print("not added")

        i = i + 1
    x = x + 1
# print (list(set(textsent))[1], len(set(textsent)),type(textsent),
type(textsent[0]))
for item in list(set(textsent)) :
    print(item)
    print('----------')
json.dump(list(set(textsent)),open('/Users/arin/Desktop/final_yr_project/
frame_extract/citmore/citmoreresults/textsent.json','w'))
```

Output:

```
a man wearing a black shirt. man wearing a black shirt.
----------
a building with a yellow roof. a white car in the background. a white building with
a red roof. a sign on the building. a white sign on the side of the road. a light
pole in the background. a sign on a building.
----------
blue and white bus. a man walking on the sidewalk. window on a bus. the bus has a
white stripe. the bus has a black stripe. a bus. a bus on the side of the road.
windows on the bus. windows on the side of the bus. the bus is black. the bus is
blue. the bus has a yellow stripe. a building in the background. a blue and white
building. the bus is blue and white.
----------
a light on a pole. a tree in the distance. a light pole. a green tree in the
distance. a tall pole in the distance. a light pole in the background. a blue and
white sign. a tall light pole. a blue and white building.
----------
a green tree in the background. the person is wearing a black shirt. a hand holding
a cell phone. man riding bicycle. a red and white building. a light pole on the
sidewalk. the tire is black. a tennis court. the top of the bus. a tennis player. a
man wearing a hat. a man on a bus. a sign on the side of the building. a tall
building. statue of a man on a building. people walking on the sidewalk. the man is
wearing black pants. the shirt is pink. bus on the street. a man in a red shirt. a
large white building. a sign on a building. a sign on a pole. man wearing a blue
```

shirt. a bike on the sidewalk. a man walking on the sidewalk. man wearing a red shirt. a white sign on the sidewalk. man in yellow shirt. a sign on the wall. a white sign on a building. the bike is black. people standing on the sidewalk. a man in a blue shirt. a window on a bus. a sign on the building. windows on the bus. the road is grey. the ground is white. a silver knife. a white wall. blue and white sign on pole. a man in a black shirt. a man wearing a black hat. a man wearing a blue shirt. yellow and black bus. a man on a bike. a tall white building. woman wearing a green shirt. a green and white sign. white snow on the ground.
---
a fence on the side of the road. a blue and white sign. a fence in the background.
---
a green tree in the background. a green tree in the distance. a tall green tree.
---
a building with a lot of windows. a tree in the distance. a yellow pole. a tall green tree. a light pole. a white building with a red roof. a red and white sign. a tall tree in the distance. blue and white sign on building. a man wearing a hat. a man in a red shirt. a blue and white sign. a street light. blue and white bus. yellow top of building. a yellow and black sign. green trees in the background. a sign on the building. a fence in the background. trees in the background. a light pole in the background. a yellow and white building. a tall light pole. blue and white sign on the side of the road. a green and white sign.
---
a light pole in the background. a telephone pole. a tall green tree. power lines on the pole.
---
a tree in the distance. a white bus. white clouds in blue sky. a tall green tree. a light pole. a white building with a red roof. a car in the street. a tall tree in the distance. a blue and white building. green leaves on trees. power lines in the sky. a light on a pole. a building with a yellow roof. a white car in the background. a green tree in the distance. a sign on a building. a blue and white sign. a tree in the background. a sign on the building. a white sign on the side of the road. a street light on a pole. trees in the background. a tree with green leaves. a white building in the background. a tall pole in the distance. a white building with a green roof. a yellow street light. a light pole in the background. a tall light pole.
---
blue and white sign on building. a man wearing a hat. a yellow and white building. a yellow and black sign. a sign on the building. a man in a red shirt. a red and white sign. blue and white sign on the side of the road. blue and white bus.
---
a bus window. the roof of the train.
---
yellow and black sign on the train. yellow and black sign. window on a bus.
---
a green tree in the background. window on the bus. trees in the background. a yellow and black bus. the bus has a windshield. the number <UNK> on the front of the bus. the windshield of a bus. yellow and black license plate.
---
a bus window. windows on the bus. window on a bus. a window on a train.
---
windows on the building.
---
white line on the road. a white car on the road. white lines on the road.
---
a fence on the side of the road. a fence in the background. a tree in the distance. a tall green tree. a white plastic container. a light pole. a green tree in the distance. a light post. a light pole in the background. a white car on the road. a blue and white sign.
---
a white door on the bus. a bus stop sign. window on a bus. a man standing in the bus.
---
lights on the front of the bus. license plate on the truck. the bike is black. license plate on the front of the bus. a wheel on a truck. front wheel of a truck.
---
window on the building. windows on the building. window on a building. a white building in the background. a tree in the background. a street light on a pole.
---

window on the bus. a blue and white bus. blue and white sign. windows on the bus. a bus on the road. the bus is blue. the windshield of a bus. blue and white bus.
---
a bus on the road. the bus is white. the bus has a window. windows on the bus.
---
a street light. a man wearing a hat.
---
a tree in the background. a street light on a pole.
---
a tall tree in the distance. a light pole.
---
a sign on the side of the building. a white building with a red roof. yellow and black sign. a sign on the sidewalk.
---
the person is wearing a black shirt. man wearing a red shirt. a hand holding a cell phone. the bike is black. a man in a blue shirt. the shirt is pink. the tire is black. man wearing a blue shirt.
---
bus number <UNK>. the bus is number.
---
white van parked on the street. blue and white bus. a white van on the street.
---
windows on the bus.
---
a tree in the distance. a white sign on the sidewalk. a man wearing a black shirt. a man with a black shirt. a blue sign with white lettering. a sign on the building. a tall white building. a street light. street light on pole. a street sign on a the side of the road. a tree in the
. a person walking on the sidewalk. a white car on the road. the road is grey.
---
white clouds in blue sky. a tree in the background.
---
a red and white traffic light. a white sign on the sidewalk. a sign on a pole. a man in a red shirt.
---
a sign on the sidewalk. a white house in the background. a car on the road. people walking on the sidewalk. a white car in the background. a building in the distance. a white car. white car parked on the street. a sign on the building. a car in the street. a person walking on the sidewalk. a red and white sign. a person walking on the street. a sign on a pole. the road is grey. a white sign on a pole. a street light on a pole.
---
man wearing black shoes. green and white sign. a man walking on the sidewalk. man wearing black shirt. a man wearing a black shirt. a man wearing a blue shirt. a white and blue flag. man wearing a black shirt. the man is wearing jeans. a red and white sign. a blue and white building. a green and white sign.
---
a building with a white roof. the truck is white. yellow and black sign. a wheel on a truck. a sign on the side of the building. a blue and white car. a green and white sign. a bus. a yellow bus. a street light. the bus is black. a white car on the road. a car parked on the street. a tall light pole. a blue car. a blue truck. blue and white bus.
---
license plate on the car. a fence on the side of the road. a green tree in the background. a tree in the distance. a sign on the pole. license plate on the truck. white clouds in blue sky. a tall green tree. a light pole. the bus has a number on it. a telephone pole. a white boat in the background. a sign on the bus. power lines in the sky. the bus is red. a man wearing a helmet. a light on a pole. the water is calm. the road is wet. window on the bus. the front wheel of a car. the bus has a windshield. a green tree in the distance. the bus is yellow. a blue and white sign. street light on pole. a motorcycle parked on the street. a large green tree. a person in the background. a white line on the ground. people standing on the sidewalk. a tree in the background. a sign on the building. a bus on the road. a light pole in the background. a motorcycle on the road. yellow and black license plate. a fence in the background. trees in the background. the bike has a black tire. a black and white bike. a man in a black shirt. green leaves on tree. a white and blue sign. a man on a motorcycle. a white car on the road. a tall light pole. a green and white sign.

```
a yellow and blue bus. a man walking on the sidewalk. the bus has a white stripe.
the windshield on the bus. the bus is green. a yellow bus. the window on the train.
windows on the bus. a white building with a red roof. the bus is blue.
----------
a window on a car. the side mirror on the car.
----------
a white car in the street. top of the building. a car on the road. a yellow and
white sign. a blue and white car. a yellow and black sign. yellow bus on road. a
yellow bus. a yellow street sign. a building in the background. yellow and black
bus. a yellow car on the road. a white sign on the side of the road. a car in the
street. a white car on the road. a black tire on a car. front tire of a car.
----------
a red and white sign. a light pole.
----------
a tall white building. a window on a building.
```

Step 7: final summarization

```
from collections import Counter
import nltk
import re
import json

def commonprefix(m):
    if not m:
        return ''
        s1 = min(m)
#     print s1
        s2 = max(m)
#     print s2
    for i, c in enumerate(s1):
        if c != s2[i]:
            return s1[:i]
    return s1

def makesent(text):
    sents = list(filter(lambda x: x!= '', [line.strip() for line in
text.split('.')]))
tags, target_sents_NN, target_sents_VBG, req_lines = list(), list(), list(), list()
    dummylines, objects = list(), list()
    lookup, target = dict(), dict()

    for line in sents:
        parse = nltk.word_tokenize(line)
        tags.append(nltk.pos_tag(parse))
    counts = Counter([item for line in tags for item in line])

    targetVBG = None
    targetNN = None
    for item in counts.most_common():
        if item[0][1] == 'NN':
            targetNN = item[0][0]
            break

    for item in counts.most_common() :
        if item[0][1] == 'DT':
            targetDT = item[0][0]
            break

    for key, value in counts.items():
        lookup.setdefault(key[1],[]).append(key[0])

    for line in tags:
        for word in line:
            if word[0] == targetNN:
                target_sents_NN.append(line)

    for item in Counter([item for line in target_sents_NN for item in
line]).most_common():
```

```
        if item[0][1] == 'VBG':
            targetVBG = str(item[0][0])
            break

    if targetVBG is None:
         return;
    elif targetNN is None:
         return;
    else :
        for line in target_sents_NN:
            if (targetNN, 'NN') in line and (targetVBG, 'VBG') in line:
                target_sents_VBG.append(line)

    if 'VBZ' not in lookup:
#         print("cannot display")
        pass
    else:
        VBZ_list = lookup['VBZ']
         target_prefix = [' '.join([targetNN, targetVBG])]
        target_prefix.append(' '.join([targetDT, targetNN, targetVBG]))
         for vbz in VBZ_list:
            target_prefix.append(' '.join([targetDT, targetNN, vbz, targetVBG]))

        for line in sents:
            for tp in target_prefix:
                if line.startswith(tp):
                    req_lines.append(line)

        for line in req_lines:
            dummylines.append(line.replace(targetDT + ' ', ''))

        prefix = commonprefix(dummylines)

        for line in dummylines:
            line = re.sub(prefix, '',line)
            if 'and ' in line:
                diff = list()
                diff = line.split(' and ')
                for d in diff:
                    objects.append(d)
                    break
            else:
                objects.append(line)
#             print objects

        obj = list(set(objects))
#         print obj
        obj.sort()
        obj.sort(key=len)
         req_objects = []
         req_objects = list(filter(lambda x: [x for i in obj if x in i and x != i]
== [], obj))
#         print(len(req_objects))
        if len(req_objects) > 0:
            final_text = ' '.join([targetDT, prefix]) + ', '.join(req_objects[:-1])
+ ' and ' + req_objects[-1]
            mysummary.append(final_text.capitalize())


if __name__ == '__main__':
    mysummary = []
    mytext = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/
citmore/citmoreresults/textsent.json'))
#     print(mytext[0])
    for item in mytext:
        makesent(item)
    print(str(mysummary))
```

Output:

```
'A man wearing black hat, red shirt and blue shirt', 'A man wearing hat and ', 'A
man wearing black shirt and ', 'A man wearing blue shirt, ack shirt and ack shoes',
'A man wearing white shirt and ', 'A man wearing hat, black shirt and black
helmet', 'A man wearing blue shirt, black shirt, white shirt and black helmet', 'A
man wearing black shirt, black shoes and green shirt'
```

Video sample 2

Step 1. Frame extraction

Code:

```
import cv2
import math
# input path to video
videoFile = str(input("Enter input to video:"))
# output path for frames
framesFolder = str(input("Enter output path for frames:"))
cap = cv2.VideoCapture(videoFile)
# find frame rate of video
frameRate = cap.get(5)
while cap.isOpened() :
#     current frame number
    frameId = cap.get(1)
    ret, frame = cap.read()
    if (ret != True) :
        break
    if int(frameId) % math.floor(frameRate) == 0 :
        filename = framesFolder + "/image_" + str(int(frameId)) + ".jpg"
#         cv2.imshow(filename, frame)
        cv2.imwrite(filename, frame)
cap.release()
print "Done!"
```

Ouput:

```
Enter input to video:"/Users/arin/Desktop/final_yr_project/frame_extract/
mrbeantrim.mp4"
Enter output path for frames:"/Users/arin/Desktop/final_yr_project/frame_extract/
frames"
Done!
```

Step 2. fetching captioning information and coordinate information form results to seperate .json file

Code:

```
import json
s = json.load(open('results.json'))
i = 0
for imginfo in s['results'] :
    json.dump(imginfo, open('/Users/arin/Desktop/final_yr_project/notebook/
imginfodir/imginfo{}.json'.format(i), 'w'))
    i = i + 1
```

Step 3. calculating (x1, y1) to (x4, y4) from x, y, h, w information

Code:

```
import os
i = 0
for file in os.listdir('/Users/arin/Desktop/final_yr_project/notebook/imginfodir/')
:
    s = json.load(open('/Users/arin/Desktop/final_yr_project/notebook/imginfodir/
{}'.format(file)))
    l = []
    for crd in s['boxes'] :
        x1, y1 = crd[0], crd[1]
        h, w = crd[2], crd[3]
#         print('x1:{} y1:{} h:{} w:{}'.format(x1, y1, h, w))
        x2, y2 = x1 + h, y1
        x3, y3 = x1 + h, y1 + w
        x4, y4 = x1, y1 + w
        t = (x1, y1, x2, y2, x3, x3, x4, y4)
#         print('x1:{} y1:{} x2:{} y2:{} x3:{} y3:{} x4:{} y4:{}
\n'.format(x1,y1,x2,y2,x3,y3,x4,y4))
        l.append(t)
    json.dump(l, open('/Users/arin/Desktop/final_yr_project/notebook/imgcrd/
imgcrdinfo{}.json'.format(i), 'w'))
    i = i + 1
    l = []
```

Output:

[(448.91363525391, 227.10025024414, 559.01666259766, 227.10025024414,
559.01666259766, 559.01666259766, 448.91363525391, 364.63320922852),
(297.74209594727, 70.234451293945, 339.472137451176, 70.234451293945,
339.472137451176, 339.472137451176, 297.74209594727, 235.529373168945),
(5.6920547485352, 66.218231201172, 167.2067947387652, 66.218231201172,
167.2067947387652, 167.2067947387652, 5.6920547485352, 200.411987304692),
(625.05224609375, 244.04396057129, 701.566772460938, 244.04396057129,
701.566772460938, 701.566772460938, 625.05224609375, 356.79931640625),
(695.56787109375, 226.72953796387, 720.484008789062, 226.72953796387,
720.484008789062, 720.484008789062, 695.56787109375, 285.914825439456),
(164.02105712891, 244.55447387695, 222.870239257816, 244.55447387695,
222.870239257816, 222.870239257816, 164.02105712891, 322.21725463866903),
(717.78485107422, 220.7709197998, 720.48358154297, 220.7709197998, 720.48358154297,
720.48358154297, 717.78485107422, 292.44253540038596), (-14.858428955078,
32.760116577148, 633.414154052732, 32.760116577148, 633.414154052732,
633.414154052732, -14.858428955078, 393.246688842768), (2.6283340454102,
232.95195007324, 163.3941421508802, 232.95195007324, 163.3941421508802,
163.3941421508802, 2.6283340454102, 396.17407226562), (666.83520507812,
233.66798400879, 711.857543945308, 233.66798400879, 711.857543945308,
711.857543945308, 666.83520507812, 286.060363769532), (318.61865234375,
20.845993041992, 719.10754394531, 20.845993041992, 719.10754394531,
719.10754394531, 318.61865234375, 275.391601562502), (684.61572265625,
250.57122802734, 719.949462890625, 250.57122802734, 719.949462890625,
719.949462890625, 684.61572265625, 319.262451171871), (615.12701416016,
-5.2597198486328, 659.354797363285, -5.2597198486328, 659.354797363285,
659.354797363285, 615.12701416016, 292.0403900146472), (369.5666809082,
297.72595214844, 559.09118652343, 297.72595214844, 559.09118652343,
559.09118652343, 369.5666809082, 391.540832519534), (403.12246704102,
226.34155273438, 720.02661132813, 226.34155273438, 720.02661132813,
720.02661132813, 403.12246704102, 390.00964355469), (714.27038574219,
246.67366027832, 720.44177246094, 246.67366027832, 720.44177246094,
720.44177246094, 714.27038574219, 315.564636230468), (138.25224304199,
228.68853759766, 195.42448425292798, 228.68853759766, 195.42448425292798,
195.42448425292798, 138.25224304199, 333.96728515625), (21.303546905518,
56.571365356445, 104.890872955323, 56.571365356445, 104.890872955323,
104.890872955323, 21.303546905518, 115.705902099609), (500.72549438477,
136.01525878906, 546.308593750004, 136.01525878906, 546.308593750004,
546.308593750004, 500.72549438477, 236.63259887695), (449.24472045898,
29.345222473145, 522.478820800777, 29.345222473145, 522.478820800777,
522.478820800777, 449.24472045898, 237.65393829345498), (576.91687011719,
169.72654724121, 650.333251953128, 169.72654724121, 650.333251953128,
650.333251953128, 576.91687011719, 292.08203125), (313.64096069336,
209.80198669434, 552.86010742188, 209.80198669434, 552.86010742188,
552.86010742188, 313.64096069336, 276.876098632817), (187.64819335938,
106.2624206543, 247.930480957036, 106.2624206543, 247.930480957036,
247.930480957036, 187.64819335938, 222.76492309571), (246.06643676758,
209.87301635742, 380.62753295899, 209.87301635742, 380.62753295899,
380.62753295899, 246.06643676758, 262.208526611326), (140.36820983887,
6.3952255249023, 409.97532653809003, 6.3952255249023, 409.97532653809003,
409.97532653809003, 140.36820983887, 246.5432052612323), (455.35916137695,
213.94621276855, 509.173187255856, 213.94621276855, 509.173187255856,
509.173187255856, 455.35916137695, 307.42098999023), (443.76116943359,
6.2064895629883, 723.30401611328, 6.2064895629883, 723.30401611328,
723.30401611328, 443.76116943359, 136.8271560668983), (606.26556396484,
211.52726745605, 714.88153076172, 211.52726745605, 714.88153076172,
714.88153076172, 606.26556396484, 285.481506347652), (280.40124511719,
183.46530151367, 339.884704589846, 183.46530151367, 339.884704589846,
339.884704589846, 280.40124511719, 244.961181640623), (2.4079933166504,
264.29257202148, 100.8393745422364, 264.29257202148, 100.8393745422364,
100.8393745422364, 2.4079933166504, 338.495819091792), (578.9599609375,
122.87017822266, 659.507934570312, 122.87017822266, 659.507934570312,
659.507934570312, 578.9599609375, 200.291351318363), (604.52215576172,
243.52014160156, 657.386169433595, 243.52014160156, 657.386169433595,
657.386169433595, 604.52215576172, 324.066772460935), (676.70654296875,
210.58619689941, 719.981201171875, 210.58619689941, 719.981201171875,
719.981201171875, 676.70654296875, 247.63407897948798), (702.24212646484,
165.4997253418, 719.21282958984, 165.4997253418, 719.21282958984, 719.21282958984,

```
702.24212646484, 209.745300292972), (426.3434753418, 126.49272155762,
719.77490234375, 126.49272155762, 719.77490234375, 719.77490234375, 426.3434753418,
205.516891479495), (16.335754394531, 144.59031677246, 152.686828613281,
144.59031677246, 152.686828613281, 152.686828613281, 16.335754394531,
244.91062927246), (38.155921936035, 36.280319213867, 251.40705108642499,
36.280319213867, 251.40705108642499, 251.40705108642499, 38.155921936035,
343.126449584957), (98.519454956055, 92.517318725586, 187.58076477050798,
92.517318725586, 187.58076477050798, 187.58076477050798, 98.519454956055,
210.814071655276), (419.74407958984, 143.49041748047, 453.890441894528,
143.49041748047, 453.890441894528, 453.890441894528, 419.74407958984,
231.249053955079), (315.37423706055, 128.2751159668, 360.896759033206,
128.2751159668, 360.896759033206, 360.896759033206, 315.37423706055,
243.91586303711), (47.750457763672, 242.79281616211, 144.04913330078102,
242.79281616211, 144.04913330078102, 144.04913330078102, 47.750457763672,
315.52041625976597), (313.82730102539, 225.14889526367, 384.653594970702,
225.14889526367, 384.653594970702, 384.653594970702, 313.82730102539,
280.74118041992), (244.40829467773, 168.56246948242, 586.08801269531,
168.56246948242, 586.08801269531, 586.08801269531, 244.40829467773,
243.841705322264), (138.36849975586, 25.266616821289, 364.90130615234,
25.266616821289, 364.90130615234, 364.90130615234, 138.36849975586,
107.517593383789), (582.10528564453, 236.33387756348, 629.986267089842,
236.33387756348, 629.986267089842, 629.986267089842, 582.10528564453,
309.978027343753), (477.48782348633, 199.77072143555, 547.036437988283,
199.77072143555, 547.036437988283, 547.036437988283, 477.48782348633,
264.70822143555), (547.17181396484, 252.59045410156, 585.198791503902,
252.59045410156, 585.198791503902, 585.198791503902, 547.17181396484,
306.542724609372), (151.67250061035, 218.28964233398, 260.15637207031,
218.28964233398, 285.50561523438, 200.47770690918202, 200.47770690918202,
121.57148742676, 391.11065673829), (161.65270996094, 73.747894287109,
220.253570556643, 73.747894287109, 220.253570556643, 220.253570556643,
161.65270996094, 198.881286621089), (97.159790039062, 155.44609069824,
381.328460693362, 155.44609069824, 381.328460693362, 381.328460693362,
97.159790039062, 231.314834594724), (506.50384521484, 298.97610473633,
607.25885009765, 298.97610473633, 607.25885009765, 607.25885009765,
506.50384521484, 358.63333129883), (245.94021606445, 241.95083618164,
429.32064819336, 241.95083618164, 429.32064819336, 429.32064819336,
245.94021606445, 299.749542236328), (207.92926025391, 222.80574035645,
319.87951660157, 222.80574035645, 319.87951660157, 319.87951660157,
207.92926025391, 283.727874755864), (4.7006530761719, 2.183032989502,
163.9552001953119, 2.183032989502, 163.9552001953119, 163.9552001953119,
4.7006530761719, 36.754730224609006), (443.55953979492, 6.0226764678955,
653.3388671875, 6.0226764678955, 653.3388671875, 653.3388671875, 443.55953979492,
52.0274372100835), (-4.1593017578125, 2.6282892227173, 393.4732971191375,
2.6282892227173, 393.4732971191375, 393.4732971191375, -4.1593017578125,
21.0774240493773), (1.8392753601074, 12.850992202759, 117.97542953490739,
12.850992202759, 117.97542953490739, 117.97542953490739, 1.8392753601074,
69.722970962525), (402.17111206055, 180.97819519043, 436.67306518555,
180.97819519043, 436.67306518555, 436.67306518555, 402.17111206055,
243.99540710449202), (91.798271179199, -0.47585678100586, 131.978393554687,
-0.47585678100586, 131.978393554687, 131.978393554687, 91.798271179199,
84.43066787719714), (159.19024658203, 189.19012451172, 270.17025756836,
189.19012451172, 270.17025756836, 270.17025756836, 159.19024658203,
231.125762939454), (536.18005371094, 4.4121675491333, 635.304443359378,
4.4121675491333, 635.304443359378, 635.304443359378, 536.18005371094,
31.9583520889283), (212.05619812012, 254.17930603027, 250.02067565918202,
254.17930603027, 250.02067565918202, 250.02067565918202, 212.05619812012,
317.208953857418), (318.21746826172, 9.6084480285645, 560.44976806641,
9.6084480285645, 560.44976806641, 560.44976806641, 318.21746826172,
87.3016853332525), (97.648818969727, 45.838184356689, 165.63221740722702,
45.838184356689, 165.63221740722702, 165.63221740722702, 97.648818969727,
99.365310668945), (476.48675537109, 4.5265064239502, 577.12384033203,
4.5265064239502, 577.12384033203, 577.12384033203, 476.48675537109,
28.3967075347902), (210.89366149902, 288.48266601562, 268.2483215332,
288.48266601562, 268.2483215332, 268.2483215332, 210.89366149902, 374.07495117187)]
```

Step 4. grouping boxes togather for a particular object, save it in a separate file into /Users/arin/Desktop/final_yr_project/notebook/grouping/{}

Code:

```
import os
import json
k = 0
for file in os.listdir('/Users/arin/Desktop/final_yr_project/notebook/imgcrd/'):
    l = json.load(open('/Users/arin/Desktop/final_yr_project/notebook/imgcrd/
{}'.format(file)))
    inclv = {}
    i = 0
    for crd in l[:]:
        j = 0
        temp = {}
        for test in l[:]:
#             for complete inclusive boxes
            if (test[0]>=crd[0] and test[1]>=crd[1]) and (test[2]<=crd[2] and
test[3]>=crd[3]) and (test[4]<=crd[4] and test[5]<=crd[5]) and (test[6]>=crd[6] and
test[7]<=crd[7]) and i != j :
                temp[j] = test
#             for partly inclusive boxes
            elif (test[0] + 10 >= crd[0] and test[1] + 10 >= crd[1]) and (test[2] -
10 <= crd[2] and test[3] + 10 >= crd[3]) and (test[4] - 10 <= crd[4] and test[5] -
10 <= crd[5]) and (test[6] + 10 >= crd[6] and test[7] - 10 <= crd[7]) and i != j:
                temp[j] = test
            j = j + 1
        if len(temp.keys()) == 0:
            inclv[i] = {}
        else :
            inclv[i] = temp
        i = i + 1
    json.dump(inclv, open('/Users/arin/Desktop/final_yr_project/notebook/grouping/
boxgrouping{}.json'.format(k), 'w'))
    k = k + 1
```

Output:

{0: {4: [590.55200195312, 256.69003295898, 719.93139648437, 256.69003295898,
719.93139648437, 719.93139648437, 590.55200195312, 490.44717407226], 7:
[310.45626831055, 220.71647644043, 491.73110961914, 220.71647644043,
491.73110961914, 491.73110961914, 310.45626831055, 484.93077087402], 16:
[631.71209716797, 86.943885803223, 717.710266113282, 86.943885803223,
717.710266113282, 717.710266113282, 631.71209716797, 215.357521057133], 17:
[325.03350830078, 207.64100646973, 415.65411376953, 207.64100646973,
415.65411376953, 415.65411376953, 325.03350830078, 369.90954589844], 20:
[488.5341796875, 143.77476501465, 552.314086914062, 143.77476501465,
552.314086914062, 552.314086914062, 488.5341796875, 259.27752685547], 23:
[206.21730041504, 344.61114501953, 293.60379028320403, 344.61114501953,
293.60379028320403, 293.60379028320403, 206.21730041504, 470.52893066406], 25:
[665.58850097656, 439.34286499023, 718.261718749998, 439.34286499023,
718.261718749998, 718.261718749998, 665.58850097656, 495.835601806636], 27:
[408.51895141602, 144.58410644531, 587.14135742188, 144.58410644531,
587.14135742188, 587.14135742188, 408.51895141602, 482.86572265625], 29:
[334.40814208984, 154.27684020996, 457.8989868164, 154.27684020996, 457.8989868164,
457.8989868164, 334.40814208984, 252.86753845214798], 30: [606.17272949219,
213.14845275879, 718.26330566407, 213.14845275879, 718.26330566407,
718.26330566407, 606.17272949219, 298.648376464845], 34: [187.10610961914,
242.33053588867, 301.30639648437, 242.33053588867, 301.30639648437,
301.30639648437, 187.10610961914, 364.36752319336], 43: [265.5866394043,
351.02569580078, 352.296966552738, 351.02569580078, 352.296966552738,
352.296966552738, 265.5866394043, 474.95440673828], 44: [667.93212890625,
357.3928527832, 718.613037109375, 357.3928527832, 718.613037109375,
718.613037109375, 667.93212890625, 474.47500610351], 52: [61.788509368896,
282.74325561523, 185.996791839596, 282.74325561523, 185.996791839596,
185.996791839596, 61.788509368896, 423.06460571289], 54: [576.0849609375,
117.81317138672, 679.61645507812, 117.81317138672, 679.61645507812,
679.61645507812, 576.0849609375, 285.06652832031], 55: [151.53538513184,
299.05126953125, 246.215866088871, 299.05126953125, 246.215866088871,
246.215866088871, 151.53538513184, 440.47204589844], 56: [422.79193115234,
223.96914672852, 522.013061523434, 223.96914672852, 522.013061523434,
522.013061523434, 422.79193115234, 402.61959838868], 58: [521.33416748047,
276.77243041992, 608.70184326172, 276.77243041992, 608.70184326172,
608.70184326172, 521.33416748047, 388.75637817383], 59: [403.48120117188,

385.99124145508, 534.52685546876, 385.99124145508, 534.52685546876,
534.52685546876, 403.48120117188, 480.505218505860097], 62: [113.31601715088,
122.01174163818, 241.72794342040999, 122.01174163818, 241.72794342040999,
241.72794342040999, 113.31601715088, 344.45326995849], 63: [205.79716491699,
122.70941925049, 330.39239501953, 122.70941925049, 330.39239501953,
330.39239501953, 205.79716491699, 224.5696105957], 64: [471.52819824219,
286.93789672852, 563.20593261719, 286.93789672852, 563.20593261719,
563.20593261719, 471.52819824219, 459.92343139649], 65: [541.30224609375,
105.86389923096, 620.880493164062, 105.86389923096, 620.880493164062,
620.880493164062, 541.30224609375, 267.69626617432], 66: [518.67724609375,
191.81205749512, 661.52014160156, 191.81205749512, 661.52014160156,
661.52014160156, 518.67724609375, 433.44805908203], 70: [372.79565429688,
120.68418884277, 518.07873535157, 120.68418884277, 518.07873535157,
518.07873535157, 372.79565429688, 338.40979003906], 71: [241.240234375,
180.68826293945, 329.563232421875, 180.68826293945, 329.563232421875,
329.563232421875, 241.240234375, 346.31741333007], 75: [250.09945678711,
301.47869873047, 374.57028198242, 301.47869873047, 374.57028198242,
374.57028198242, 250.09945678711, 417.07733154297], 76: [79.572158813477,
221.86695861816, 206.13667297363702, 221.86695861816, 206.13667297363702,
206.13667297363702, 79.572158813477, 300.248718261715], 78: [281.31719970703,
198.49938964844, 394.89959716797, 198.49938964844, 394.89959716797,
394.89959716797, 281.31719970703, 283.40798950195597]}}

Step 5. grouping of captions complete inclusive and partly inclusive updated

Code:

```
import os
import json
x = 0
y = 1
g = 0
i = 0
cpt = {}
for file in os.listdir('/Users/arin/Desktop/final_yr_project/notebook/grouping/'):
    l = json.load(open('/Users/arin/Desktop/final_yr_project/notebook/grouping/
{}'.format(file)))
    if i == 2:
        i = 0
        y = y + 1
    r = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/results/
mrbeantrimresults/results{}.json'.format(y)))
    c = r['results'][i]['captions']
    for key in l:
        cptlist = []
        for k in l[key].keys():
            t = {k:c[int(k)]}
            cptlist.append(t)
        cpt[str((key, c[int(key)]))] = cptlist
    i = i + 1
    print(cpt)
    print('----------------')
    json.dump(cpt, open('/Users/arin/Desktop/final_yr_project/notebook/grptocpt/
cpt{}.json'.format(g), 'w'))
    g = g + 1
```

Output:

{"('0', 'a man in a suit')": [{'4': 'a man wearing a black jacket'}, {'7': 'man wearing a suit'}, {'16': 'man wearing a white shirt'}, {'17': 'the man is wearing a white shirt'}, {'20': 'a poster of a man'}, {'23': 'wine glass with wine'}, {'25': 'a white plastic bag'}, {'27': 'a brown wooden door'}, {'29': 'man has a beard'}, {'30': 'black jacket on man'}, {'34': 'a white shirt on a man'}, {'43': 'a glass of wine'}, {'44': 'a white handle on a chair'}, {'52': 'a white chair'}, {'54': 'man wearing black shirt'}, {'55': 'a white chair'}, {'56': 'the man is wearing a black jacket'}, {'58': 'a white box on the wall'}, {'59': 'a man wearing a watch'}, {'62': 'a brown chair'}, {'63': 'the man has a black hair'}, {'64': 'the jacket is black'}, {'65': 'a white door'}, {'66': 'a man in a blue shirt'}, {'70': 'a man wearing a tie'}, {'71': 'the shirt is black'}, {'75': 'a white napkin on the table'}, {'76': 'a chair in the background'}, {'78': 'the man is wearing a white shirt'}], "('1', 'television on the wall')": [{'38': 'a lamp on the wall'}], "('2',

'a man wearing a suit')": [{'10': 'black hat on mans head'}, {'16': 'man wearing a white shirt'}, {'30': 'black jacket on man'}], "('3', 'man with short hair')": [], "('4', 'a man wearing a black jacket')": [{'25': 'a white plastic bag'}, {'44': 'a white handle on a chair'}], "('5', 'people sitting at table')": [{'18': 'a chair with a pillow'}, {'23': 'wine glass with wine'}, {'33': 'a small white plate'}, {'34': 'a white shirt on a man'}, {'39': 'a glass of wine'}, {'49': 'a white plate on a bus'}, {'50': 'the handle of the suitcase'}, {'52': 'a white chair'}, {'55': 'a white chair'}, {'60': 'a brown chair'}, {'61': 'a white plastic bag'}, {'62': 'a brown chair'}, {'76': 'a chair in the background'}], "('6', 'man wearing a blue shirt')": [{'3': 'man with short hair'}, {'29': 'man has a beard'}, {'71': 'the shirt is black'}, {'78': 'the man is wearing a white shirt'}], "('7', 'man wearing a suit')": [], "('8', 'a plate of food')": [{'9': 'a white plate on a table'}, {'13': 'a silver spoon on a plate'}, {'22': 'a black and silver pot'}, {'37': 'a small black and white plate'}, {'39': 'a glass of wine'}, {'67': 'a white plate on a table'}], "('9', 'a white plate on a table')": [], "('10', 'black hat on mans head')": [], "('11', 'the glass is empty')": [{'23': 'wine glass with wine'}, {'43': 'a glass of wine'}], "('12', 'flowers in a vase')": [], "('13', 'a silver spoon on a plate')": [], "('14', 'a silver metal faucet')": [], "('15', 'a white box')": [], "('16', 'man wearing a white shirt')": [], "('17', 'the man is wearing a white shirt')": [], "('18', 'a chair with a pillow')": [{'50': 'the handle of the suitcase'}, {'60': 'a brown chair'}, {'76': 'a chair in the background'}], "('19', 'a brown table cloth')": [{'22': 'a black and silver pot'}, {'35': 'a silver sink'}, {'67': 'a white plate on a table'}], "('20', 'a poster of a man')": [], "('21', 'a large wooden door')": [{'1': 'television on the wall'}, {'10': 'black hat on mans head'}, {'16': 'man wearing a white shirt'}, {'32': 'man wearing glasses'}, {'38': 'a lamp on the wall'}, {'42': 'a brown wooden frame'}, {'51': 'a man wearing a hat'}, {'57': 'the hair of a man'}, {'63': 'the man has a black hair'}, {'68': 'a window with a white frame'}, {'69': 'a window on the wall'}, {'73': 'a white box on the table'}, {'77': 'a person wearing a black shirt'}, {'79': 'a white laptop'}], "('22', 'a black and silver pot')": [], "('23', 'wine glass with wine')": [], "('24', 'a silver and black glass')": [{'14': 'a silver metal faucet'}, {'35': 'a silver sink'}, {'49': 'a white plate on a bus'}], "('25', 'a white plastic bag')": [], "('26', 'a wooden shelf')": [{'20': 'a poster of a man'}, {'69': 'a window on the wall'}], "('27', 'a brown wooden door')": [{'20': 'a poster of a man'}, {'56': 'the man is wearing a black jacket'}, {'59': 'a man wearing a watch'}, {'64': 'the jacket is black'}], "('28', 'a man wearing a hat')": [], "('29', 'man has a beard')": [], "('30', 'black jacket on man')": [], "('31', 'a small white sink')": [], "('32', 'man wearing glasses')": [], "('33', 'a small white plate')": [], "('34', 'a white shirt on a man')": [], "('35', 'a silver sink')": [], "('36', 'a chair in the room')": [{'33': 'a small white plate'}, {'49': 'a white plate on a bus'}], "('37', 'a small black and white plate')": [], "('38', 'a lamp on the wall')": [], "('39', 'a glass of wine')": [], "('40', 'a black hat on the head')": [], "('41', 'a white plate on a table')": [{'47': 'a white plastic bag'}], "('42', 'a brown wooden frame')": [{'38': 'a lamp on the wall'}], "('43', 'a glass of wine')": [], "('44', 'a white handle on a chair')": [], "('45', 'a wooden table')": [], "('46', 'a man wearing a black shirt')": [{'63': 'the man has a black hair'}], "('47', 'a white plastic bag')": [], "('48', 'a wooden table')": [{'28': 'a man wearing a hat'}, {'40': 'a black hat on the head'}, {'45': 'a wooden table'}, {'53': 'a wooden door'}], "('49', 'a white plate on a bus')": [], "('50', 'the handle of the suitcase')": [], "('51', 'a man wearing a hat')": [{'79': 'a white laptop'}], "('52', 'a white chair')": [], "('53', 'a wooden door')": [{'40': 'a black hat on the head'}], "('54', 'man wearing black shirt')": [], "('55', 'a white chair')": [], "('56', 'the man is wearing a black jacket')": [], "('57', 'the hair of a man')": [], "('58', 'a white box on the wall')": [], "('59', 'a man wearing a watch')": [], "('60', 'a brown chair')": [], "('61', 'a white plastic bag')": [], "('62', 'a brown chair')": [], "('63', 'the man has a black hair')": [], "('64', 'the jacket is black')": [], "('65', 'a white door')": [], "('66', 'a man in a blue shirt')": [{'58': 'a white box on the wall'}], "('67', 'a white plate on a table')": [], "('68', 'a window with a white frame')": [], "('69', 'a window on the wall')": [], "('70', 'a man wearing a tie')": [], "('71', 'the shirt is black')": [], "('72', 'a light brown wooden door')": [], "('73', 'a white box on the table')": [], "('74', 'a black chair')": [], "('75', 'a white napkin on the table')": [], "('76', 'a chair in the background')": [], "('77', 'a person wearing a black shirt')": [], "('78', 'the man is wearing a white shirt')": [], "('79', 'a white laptop')": []}

Step 6. Sentence making

Code:

```
import os
import json
textsent = []
j = 0
x = 0
i = 0
while x <= 5:
    data = json.load(open('/Users/arin/Desktop/final_yr_project/notebook/grptocpt/
cpt{}.json'.format(x)))
    keylist = list(data.keys())
    i = 0
    while i < len(keylist):
        cap = list(data[keylist[i]])
#         print('################cpt{}.json'.format(x))
#         print(cap, len(cap))

        if len(cap) > 1:
            capdict = dict()
            for num, items in enumerate(cap):
                for key in items.keys():
                    capdict[num] = items[key]
            text = set([capdict[key] for key in capdict.keys()])
#             print(text)
#             print('--------------')
            mystring = ""
            for sen in list(text):
                mystring = mystring + sen + ". "
            textsent.insert(j, str(mystring))
            j = j + 1
#         else:
#             print("not added")

        i = i + 1
    x = x + 1
# print (textsent)
for item in list(set(textsent)) :
    print(item)
    print('----------')
json.dump(textsent,open('/Users/arin/Desktop/final_yr_project/frame_extract/
results/mrbeantrimresults/textsent.json','w'))
```

Output:

```
a white plate on a table. a silver sink. a black and silver pot.
----------
a black and white laptop. a white shirt on a man. a silver spoon on a plate. a
flower in a vase. a brown leather bag. a brown bag. man wearing a white shirt. a
black and silver knife. the jacket is brown. the suit is black. the man is wearing
a white shirt. a man wearing a blue shirt. a small black and white plate. a white
plate. white refrigerator door. the man is wearing a tie. a silver counter top. a
silver fork on a plate. a man wearing a tie. man has brown hair. a silver faucet. a
glass of wine.
----------
a man wearing a black shirt. man wearing a black tie. a white plastic bag. a black
dress shirt.
----------
the man has short hair. a plant in a pot. a plant in the background. a brown wooden
door. the hair is black. a dark brown hair.
----------
a window on the wall. a poster of a man.
----------
a green plant in the background. a small white table. a wooden shelf. television on
the wall. a black hair on a head. a black lamp.
----------
a light on the wall. a plant in a pot. a black hair on a head. a plant in the
background.
----------
a light on the wall. a plant in a pot. a black hat on the head. a plant in the
background.
```

---
a small book on the wall. a person sitting in the chair. a white shirt on a man. a piece of flowers. a plant in a pot. a man in a black shirt. the man is wearing a white shirt. a man with dark hair. a light in the background. a man with a hat on. a person wearing a shirt. a door in the background. man has brown hair. the man has black hair. a person wearing a hat. a small white tv. a black hair on a mans head.
---
black hat on mans head. black jacket on man. man wearing a white shirt.
---
the man is wearing a white shirt. man wearing a white shirt.
---
man wearing a black suit. a black jacket on a man. a man wearing a black shirt.
---
the man is wearing a black hat. the man is wearing glasses.
---
a man wearing a tie. the man is smiling.
---
a tv on the stand. a piece of flowers. a plant in a pot. a small lamp on the wall. a small white tv.
---
a black shirt on a man. man wearing a suit. a light brown wooden frame. a man wearing a black shirt. a white chair. man wearing a black shirt. a black piece of a person. a man wearing glasses. man wearing a black suit. a lamp on the table. a pair of blue jeans. a large black and white chair. a man with a beard. a white box. the man is wearing a black jacket. a tv on the stand. a wooden shelf. the man is wearing a black suit. man wearing glasses. the man has glasses. a poster of a store.
---
and white plate. a plate of food. a plate on a table. a white plate.
---
a silver sink. a silver and black door.
a white door on the side of the bus. white refrigerator door. a small glass of wine. a silver sink. a white and black bottle. a silver metal bar. a large glass of wine. a white box on the counter. a silver metal faucet.
---
a silver metal plate. a silver sink. a silver spoon on a plate. a silver metal sink. a plate of food. a silver metal bar. a table cloth. a white plate. a silver metal faucet.
---
a tv on the stand. a brown chair. a brown wooden wall. a white and black umbrella. a black piece of a person. a green plant in the background. a window in the room. a large black and white chair. a plant in a pot. a small wooden table. a man with short hair. a wooden shelf. a wooden door. a white wooden chair. a man with a beard. a white chair. a poster of a store. the man is smiling.
---
a brown chair. a white door on the side of the bus. a white shirt with a collar. a small glass of wine. a flower in a vase. a brown leather bag. a white chair. a silver metal sink. a large glass of wine. a small black and white plate. a white door. a silver metal plate. white refrigerator door. a black metal chair. a small white table. a white and black bottle. a man wearing a suit. a man wearing a tie. a silver sink. a glass of water. a silver metal bar. a white and black glass. a white box on the counter. a silver metal faucet.
---
a black chair. a man wearing a black shirt. a chair in the room. man wearing a suit. man wearing a black suit. a man wearing glasses. a white plastic bag. a small white table. a white pillow. a brown table cloth. a white sign on the wall. man with glasses on face. a wooden shelf. a black suit. a white napkin. a white plate on a table. a white chair. a shelf of a book. a white plate.
---
a man wearing glasses. man wearing a black suit. a white pillow. man with glasses on face. a man wearing a black shirt.
---
a silver faucet. a flower pot on the table. man wearing a white shirt. a white plastic bag. a silver sink. the man is wearing a black jacket. a small bowl on the table. a glass of wine. a silver spoon. the man is wearing a white shirt. a man wearing a white shirt. a silver metal tray. a clear glass of water. a man wearing a black shirt. a white plate on a table. a small glass table. a white plate.

```
the handle of the suitcase. wine glass with wine. a brown chair. a white plate on a bus. a chair with a pillow. a white shirt on a man. a small white plate. a white plastic bag. a glass of wine. a white chair. a chair in the background.
----------
a silver counter top. a black and silver knife.
----------
a brown chair. a white and black umbrella. a white plate on a bus. a window in the room. a small glass of wine. a white table cloth. a white wooden chair. a white chair. a black and white chair.
----------
the hair is black. the man has short hair.
----------
a plate of food. a fork on a plate. a pair of blue jeans.
----------
a white plastic bag.
----------
a small bowl on the table. a silver spoon. a silver metal tray. a white plate on a table. a white plate.
----------
a wooden table. a white pillow on the couch. a white metal chair.
----------
the man has short hair. a flat screen tv. a white shirt with a collar. a man in a suit. a white box on the wall. a flower in a vase. a tree in a room. a brown leather bag. a brown wooden door. a window in a store. a man wearing a shirt. a man wearing a blue shirt. the hair is black. a white door. the man is smiling. a black and white umbrella. a small white table. a person holding a fork. a man wearing a suit. a plant in the background. a plate of food. a man wearing a tie. the man is wearing a black jacket. the man has dark hair. a plant in a pot. a green and white tv. a dark brown hair.
----------
a white door on the side of the bus. a silver sink. a silver metal tray.
----------
a white refrigerator. a small white glass. a white plate on a bus. a small glass of wine. a silver sink. a black and white plate. a white table cloth. a person holding a glass. a silver metal faucet.
----------
a person wearing a white shirt. a light on the wall. a brown wooden table. a wall behind the cat. a wooden table. man wearing a white shirt. a white plastic bag. a man wearing a hat. the man is wearing a white shirt. a man with short hair. the hair is black. a woman wearing a white shirt. a shelf of a book. television on the wall. a green plant in the background. a black and white umbrella. a small white table. a plant in the background. a brown wooden chair. the man is wearing a black jacket. a black hair on a head. a white wooden door. a black handle on the side of the car. black hair on the head. a man holding a cup. a plant in a pot. a plant in a window. a wooden shelf. a man wearing a white shirt. a light brown wall. a black lamp.
----------
a silver faucet. a white plate on the table. a silver bowl on a table. a white plate on a table. a silver knife.
----------
a small book on the wall. the man has black hair. a black hair on a mans head.
----------
a white wall behind the cat. a glass of wine. the handle of the bus.
----------
a silver counter top. a white plate. a black and silver knife.
----------
a silver spoon on a plate. a glass of wine. a black and silver pot. a white plate on a table. a small black and white plate.
----------
```

Step 7. Final summarization

Code:

```
from collections import Counter
import nltk
import re

def commonprefix(m):
    if not m:
```

```
        return ''
#     testing
#     for line in m:
#         print line, len(line)
    s1 = min(m)
    print s1
    s2 = max(m)
    print s2
    for i, c in enumerate(s1):
        if c != s2[i]:
            return s1[:i]
    return s1

def makesent(text):
    sents = filter(lambda x: x!= '', [line.strip() for line in text.split('.')])
    print sents
      tags, target_sents_NN, target_sents_VBG, req_lines = list(), list(), list(),
list()
    dummylines, objects = list(), list()
    lookup, target = dict(), dict()

    for line in sents:
        parse = nltk.word_tokenize(line)
        tags.append(nltk.pos_tag(parse))
    counts = Counter([item for line in tags for item in line])
#     print tags

    targetVBG = None
    targetNN = None
    for item in counts.most_common():
        if item[0][1] == 'NN':
            targetNN = item[0][0]
            break

    for item in counts.most_common() :
        if item[0][1] == 'DT':
            targetDT = item[0][0]
            break

    for key, value in counts.items():
        lookup.setdefault(key[1],[]).append(key[0])

    for line in tags:
        for word in line:
            if word[0] == targetNN:
                target_sents_NN.append(line)
                print line
                print '------------------'
#     targetVBG = str(0)
#     targetNN = str(0)

          for item in Counter([item for line in target_sents_NN for item in
line]).most_common():
        if item[0][1] == 'VBG':
            targetVBG = str(item[0][0])
            break

    if targetVBG is None:
        print "no targetVBG"
        return;
    elif targetNN is None:
        print "no targetNN"
        return;
    else :
        for line in target_sents_NN:
            if (targetNN, 'NN') in line and (targetVBG, 'VBG') in line:
                target_sents_VBG.append(line)

    if 'VBZ' not in lookup:
```

```
        print "cannot display"
    else:
        VBZ_list = lookup['VBZ']
        print VBZ_list
#     if len(VBZ_list) == 0:
#         print "Cannot display"
#         return
#     else:
        target_prefix = [' '.join([targetNN, targetVBG])]
        target_prefix.append(' '.join([targetDT, targetNN, targetVBG]))
        print target_prefix
        for vbz in VBZ_list:
            target_prefix.append(' '.join([targetDT, targetNN, vbz, targetVBG]))
            print target_prefix

        for line in sents:
            for tp in target_prefix:
                if line.startswith(tp):
                    req_lines.append(line)
                    print req_lines

        for line in req_lines:
            dummylines.append(line.replace(targetDT + ' ', ''))
        print dummylines

        prefix = commonprefix(dummylines)

        for line in dummylines:
            line = re.sub(prefix, '',line)
            if 'and ' in line:
                diff = list()
                diff = line.split(' and ')
                for d in diff:
                    objects.append(d)
                    break
            else:
                objects.append(line)
            print objects

        obj = list(set(objects))
        print obj
        obj.sort()
        obj.sort(key=len)
        print obj

        req_objects = filter(lambda x: [x for i in obj if x in i and x != i] == [],
obj)
        final_text = ' '.join([targetDT, prefix]) + ', '.join(req_objects[:-1]) + '
and ' + req_objects[-1]

        print final_text.capitalize()


if __name__ == '__main__':
#     text = raw_input("Enter your text:")
#     print "Your text:" + text
    text = "the man has glasses. a man wearing a white shirt. a white door. a pair
of blue jeans. the shirt is black. a white shirt on a man. man wearing a suit and
tie. a man in a white shirt. a glass of wine. man has white shirt. a wooden table.
a white box. a white box on the table. a man in a black shirt. a man wearing a tie.
the mans hair is black. a wooden door. man wearing a black suit. a picture of a
woman. a person standing in the background. a white napkin in the background. black
hat on mans head. man wearing a white shirt. a chair with a white seat. a pair of
brown pants. the man is wearing a tie."
    makesent(text)
```

Output:

```
['the man has glasses', 'a man wearing a white shirt', 'a white door', 'a pair of
blue jeans', 'the shirt is black', 'a white shirt on a man', 'man wearing a suit
and tie', 'a man in a white shirt', 'a glass of wine', 'man has white shirt', 'a
wooden table', 'a white box', 'a white box on the table', 'a man in a black shirt',
'a man wearing a tie', 'the mans hair is black', 'a wooden door', 'man wearing a
black suit', 'a picture of a woman', 'a person standing in the background', 'a
white napkin in the background', 'black hat on mans head', 'man wearing a white
shirt', 'a chair with a white seat', 'a pair of brown pants', 'the man is wearing a
tie']
[('the', 'DT'), ('man', 'NN'), ('has', 'VBZ'), ('glasses', 'NNS')]
------------------
[('a', 'DT'), ('man', 'NN'), ('wearing', 'VBG'), ('a', 'DT'), ('white', 'JJ'),
('shirt', 'NN')]
------------------
[('a', 'DT'), ('white', 'JJ'), ('shirt', 'NN'), ('on', 'IN'), ('a', 'DT'), ('man',
'NN')]
------------------
[('man', 'NN'), ('wearing', 'VBG'), ('a', 'DT'), ('suit', 'NN'), ('and', 'CC'),
('tie', 'NN')]
------------------
[('a', 'DT'), ('man', 'NN'), ('in', 'IN'), ('a', 'DT'), ('white', 'JJ'), ('shirt',
'NN')]
------------------
[('man', 'NN'), ('has', 'VBZ'), ('white', 'JJ'), ('shirt', 'NN')]
------------------
[('a', 'DT'), ('man', 'NN'), ('in', 'IN'), ('a', 'DT'), ('black', 'JJ'), ('shirt',
'NN')]
------------------
[('a', 'DT'), ('man', 'NN'), ('wearing', 'VBG'), ('a', 'DT'), ('tie', 'NN')]
------------------
[('man', 'NN'), ('wearing', 'VBG'), ('a', 'DT'), ('black', 'JJ'), ('suit', 'NN')]
------------------
[('man', 'NN'), ('wearing', 'VBG'), ('a', 'DT'), ('white', 'JJ'), ('shirt', 'NN')]
------------------
[('the', 'DT'), ('man', 'NN'), ('is', 'VBZ'), ('wearing', 'VBG'), ('a', 'DT'),
('tie', 'NN')]
------------------
['has', 'is']
['man wearing', 'a man wearing']
['man wearing', 'a man wearing', 'a man has wearing']
['man wearing', 'a man wearing', 'a man has wearing', 'a man is wearing']
['a man wearing a white shirt']
['a man wearing a white shirt', 'man wearing a suit and tie']
['a man wearing a white shirt', 'man wearing a suit and tie', 'a man wearing a
tie']
['a man wearing a white shirt', 'man wearing a suit and tie', 'a man wearing a
tie', 'man wearing a black suit']
['a man wearing a white shirt', 'man wearing a suit and tie', 'a man wearing a
tie', 'man wearing a black suit', 'man wearing a white shirt']
['man wearing white shirt', 'man wearing suit and tie', 'man wearing tie', 'man
wearing black suit', 'man wearing white shirt']
man wearing black suit
man wearing white shirt
['white shirt']
['white shirt', 'suit']
['white shirt', 'suit', 'tie']
['white shirt', 'suit', 'tie', 'black suit']
['white shirt', 'suit', 'tie', 'black suit', 'white shirt']
['white shirt', 'tie', 'black suit', 'suit']
['tie', 'suit', 'black suit', 'white shirt']
A man wearing tie, black suit and white shirt
```

Video sample 3

Random video as input

Extracted frames from video

Step 1. Frame extraction

Code:

```
import cv2
import math
# input path to video
videoFile = str(input("Enter input to video:"))
# output path for frames
framesFolder = str(input("Enter output path for frames:"))
cap = cv2.VideoCapture(videoFile)
# find frame rate of video
frameRate = cap.get(5)
while cap.isOpened() :
```

```
#     current frame number
    frameId = cap.get(1)
    ret, frame = cap.read()
    if (ret != True) :
        break
    if int(frameId) % math.floor(frameRate) == 0 :
        filename = framesFolder + "/image_" + str(int(frameId)) + ".jpg"
#         cv2.imshow(filename, frame)
        cv2.imwrite(filename, frame)
cap.release()
print "Done!
```

Step 2: fetching captioning information and coordinate information form results to seperate .json file

Code:

```
import json
import os
i = 0
for file1 in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/
college/collegeresults/') :
    print(file1)
    s = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/college/
collegeresults/{}'.format(file1)))
    for imginfo in s['results'] :
        json.dump(imginfo, open('/Users/arin/Desktop/final_yr_project/
frame_extract/college/imginfodir/imginfo{}.json'.format(i), 'w'))
        i = i + 1
```

Output:

```
results1.json
results2.json
results3.json
results4.json
```

Step 3. calculating (x1, y1) to (x4, y4) from x, y, h, w information

Code:

```
import os
i = 0
for file in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/college/
imginfodir/') :
    s = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/college/
imginfodir/{}'.format(file)))
    l = []
    for crd in s['boxes'] :
        x1, y1 = crd[0], crd[1]
        h, w = crd[2], crd[3]
#         print('x1:{} y1:{} h:{} w:{}'.format(x1, y1, h, w))
        x2, y2 = x1 + h, y1
        x3, y3 = x1 + h, y1 + w
        x4, y4 = x1, y1 + w
        t = (x1, y1, x2, y2, x3, x3, x4, y4)
#         print('x1:{} y1:{} x2:{} y2:{} x3:{} y3:{} x4:{} y4:{}
\n'.format(x1,y1,x2,y2,x3,y3,x4,y4))
        l.append(t)
    print(l)
    print('--------------------')
    json.dump(l, open('/Users/arin/Desktop/final_yr_project/frame_extract/college/
imgcrd/imgcrdinfo{}.json'.format(i), 'w'))
    i = i + 1
    l = []
```

Output:

```
[(5.760871887207, 178.60296630859, 205.246620178227, 178.60296630859,
205.246620178227, 205.246620178227, 5.760871887207, 362.41131591797),
(346.00012207031, 4.3760375976562, 719.48156738281, 4.3760375976562,
719.48156738281, 719.48156738281, 346.00012207031, 411.0943603515662),
(-4.3327331542969, 70.911102294922, 503.3035888671831, 70.911102294922,
503.3035888671831, 503.3035888671831, -4.3327331542969, 403.708099365232),
```

(-1.4464416503906, 4.5795745849609, 115.0092239379894, 4.5795745849609,
115.0092239379894, 115.0092239379894, -1.4464416503906, 186.7059936523409),
(433.82736206055, 140.18608093262, 574.65814208985, 140.18608093262,
574.65814208985, 574.65814208985, 433.82736206055, 364.31036376953),
(181.77850341797, -0.83271789550781, 400.98071289063, -0.83271789550781,
400.98071289063, 400.98071289063, 181.77850341797, 316.3087615966822),
(156.02764892578, 113.96541595459, 281.96649169922, 113.96541595459,
281.96649169922, 281.96649169922, 156.02764892578, 368.45183563232),
(173.58073425293, -1.0945091247559, 614.14653015137, -1.0945091247559,
614.14653015137, 614.14653015137, 173.58073425293, 99.0305366516141),
(180.32514953613, 264.04708862305, 266.308441162107, 264.04708862305,
266.308441162107, 266.308441162107, 180.32514953613, 375.30105590821),
(-12.255401611328, 267.50091552734, 373.467773437502, 267.50091552734,
373.467773437502, 373.467773437502, -12.255401611328, 396.98889160156),
(65.384727478027, 46.052536010742, 220.649894714357, 46.052536010742,
220.649894714357, 220.649894714357, 65.384727478027, 184.518585205082),
(51.84899520874, 275.00497436523, 127.55344772338799, 275.00497436523,
127.55344772338799, 127.55344772338799, 51.84899520874, 344.33584594726096),
(604.71514892578, 134.31509399414, 715.62811279297, 134.31509399414,
715.62811279297, 715.62811279297, 604.71514892578, 409.79641723633),
(218.35494995117, 146.62098693848, 352.40115356445, 146.62098693848,
352.40115356445, 352.40115356445, 218.35494995117, 300.6381225586),
(76.925300598145, 191.04260253906, 213.52323150634498, 191.04260253906,
213.52323150634498, 213.52323150634498, 76.925300598145, 263.730072021482),
(25.428121566772, 73.731018066406, 82.083662033081, 73.731018066406,
82.083662033081, 82.083662033081, 25.428121566772, 172.6044769287 11),
(208.13703918457, 201.59913635254, 270.625946044922, 201.59913635254,
270.625946044922, 270.625946044922, 208.13703918457, 267.632568359376),
(310.447265625, 130.52584838867, 358.68896484375, 130.52584838867, 358.68896484375,
358.68896484375, 310.447265625, 257.08215332031), (-4.2806396484375,
5.5872993469238, 405.8452758789025, 5.5872993469238, 405.8452758789025,
405.8452758789025, -4.2806396484375, 61.0547828674318), (-5.4667053222656,
148.54866027832, 283.7705993652344, 148.54866027832, 283.7705993652344,
283.7705993652344, -5.4667053222656, 244.18931579589798), (137.48077392578,
223.3932800293, 212.371856689452, 223.3932800293, 212.371856689452,
212.371856689452, 137.48077392578, 330.74148559571), (401.23931884766,
7.1558227539062, 654.34075927735, 7.1558227539062, 654.34075927735,
654.34075927735, 401.23931884766, 185.4366607666062), (646.75958251953,
106.69625854492, 705.837097167968, 106.69625854492, 705.837097167968,
705.837097167968, 646.75958251953, 240.37161254883), (286.61437988281,
191.42770385742, 390.296875, 191.42770385742, 390.296875, 390.296875,
286.61437988281, 392.19766235351), (349.21697998047, 114.35859680176, 629.78515625,
114.35859680176, 629.78515625, 629.78515625, 349.21697998047, 226.37480163574),
(204.63290405273, 247.12237548828, 342.04788208007, 247.12237548828,
342.04788208007, 342.04788208007, 204.63290405273, 318.271057128905),
(477.59033203125, 146.5467376709, 567.189575195312, 146.5467376709,
567.189575195312, 567.189575195312, 477.59033203125, 235.793960571291),
(485.85934448242, -1.01975631713 87, 621.4287109375, -1.01975631713 87,
621.4287109375, 621.4287109375, 485.85934448242, 106.2776832580613),
(680.99932861328, 124.71248626709, 717.225280761718, 124.71248626709,
717.225280761718, 717.225280761718, 680.99932861328, 194.826873779297),
(448.34063720703, 349.81948852539, 722.04046630859, 349.81948852539,
722.04046630859, 722.04046630859, 448.34063720703, 401.107025146484),
(340.86090087891, 136.69076538086, 380.316223144535, 136.69076538086,
380.316223144535, 380.316223144535, 340.86090087891, 233.996490478516),
(3.2427024841309, 185.34788513184, 75.6408195495609, 185.34788513184,
75.6408195495609, 75.6408195495609, 3.2427024841309, 237.346237182621),
(282.64437866211, 129.55368041992, 408.39828491211, 129.55368041992,
408.39828491211, 408.39828491211, 282.64437866211, 204.598266601561),
(663.669921875, 220.15365600586, 715.994018554688, 220.15365600586,
715.994018554688, 715.994018554688, 663.669921875, 288.56204223632903),
(203.7483215332, 290.38537597656, 295.76889038085596, 290.38537597656,
295.76889038085596, 295.76889038085596, 203.7483215332, 343.330932617185),
(184.57604980469, 2.455753326416, 311.57202148438, 2.455753326416, 311.57202148438,
311.57202148438, 184.57604980469, 101.8352394104), (2.0612678527832,
20.907974243164, 55.947315216064204, 20.907974243164, 55.947315216064204,
55.947315216064204, 2.0612678527832, 133.987396240234), (105.84544372559,
84.358238220215, 200.22706604004298, 84.358238220215, 200.22706604004298,
200.22706604004298, 105.84544372559, 135.761779785156), (274.07223510742,

```
203.80160522461, 329.116119384764, 203.80160522461, 329.116119384764,
329.116119384764, 274.07223510742, 303.72451782226597), (93.964538574219,
127.11717987061, 129.965606689453, 127.11717987061,.144775390623, 252.02328491211,
571.144775390623, 571.144775390623, 504.34713745117, 368.13687133789),
(2.1936416625977, 347.8811340332, 67.8072586059567, 347.8811340332,
67.8072586059567, 67.8072586059567, 2.1936416625977, 391.848602294919)]
```

Step 4. grouping boxes togather for a particular object, save it in a separate file into /Users/arin/Desktop/ final_yr_project/frame_extract/college/grouping/{}

Code:

```
import os
import json
k = 0
for file in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/college/
imgcrd/'):
    l = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/college/
imgcrd/{}'.format(file)))
    inclv = {}
    i = 0
    for crd in l[:]:
        j = 0
        temp = {}
        for test in l[:]:
#             for complete inclusive boxes
                if (test[0]>=crd[0] and test[1]>=crd[1]) and (test[2]<=crd[2] and
test[3]>=crd[3]) and (test[4]<=crd[4] and test[5]<=crd[5]) and (test[6]>=crd[6] and
test[7]<=crd[7]) and i != j :
                  temp[j] = test
#             for partly inclusive boxes
            elif (test[0] + 10 >= crd[0] and test[1] + 10 >= crd[1]) and (test[2] -
10 <= crd[2] and test[3] + 10 >= crd[3]) and (test[4] - 10 <= crd[4] and test[5] -
10 <= crd[5]) and (test[6] + 10 >= crd[6] and test[7] - 10 <= crd[7]) and i != j:
                temp[j] = test
            j = j + 1
        if len(temp.keys()) == 0:
            inclv[i] = {}
        else :
            inclv[i] = temp
        i = i + 1
    print(inclv)
    print('-----------------')
     json.dump(inclv, open('/Users/arin/Desktop/final_yr_project/frame_extract/
college/grouping/boxgrouping{}.json'.format(k), 'w'))
    k = k + 1
```

Ouput:

```
{0: {11: [51.84899520874, 275.00497436523, 127.55344772338799, 275.00497436523,
127.55344772338799, 127.55344772338799, 51.84899520874, 344.33584594726096], 14:
[76.925300598145, 191.04260253906, 213.52323150634498, 191.04260253906,
213.52323150634498, 213.52323150634498, 76.925300598145, 263.730072021482], 20:
[137.48077392578, 223.3932800293, 212.371856689452, 223.3932800293,
212.371856689452, 212.371856689452, 137.48077392578, 330.74148559571], 31:
[3.2427024841309, 185.34788513184, 75.6408195495609, 185.34788513184,
75.6408195495609, 75.6408195495609, 3.2427024841309, 237.346237182621], 40:
[21.330047607422, 209.02836608887, 109.85432434082, 209.02836608887,
109.85432434082, 109.85432434082, 21.330047607422, 324.88952636719], 54:
[88.472877502441, 215.01881408691, 142.793441772461, 215.01881408691,
142.793441772461, 142.793441772461, 88.472877502441, 294.281219482418], 57:
[91.116119384766, 260.08520507812, 179.279998779297, 260.08520507812,
179.279998779297, 179.279998779297, 91.116119384766, 345.401550292964], 59:
[39.538791656494, 197.42950439453, 149.942569732664, 197.42950439453,
149.942569732664, 149.942569732664, 39.538791656494, 243.285827636718], 77:
[0.4094295501709, 257.30526733398, 51.8016414642329, 257.30526733398,
51.8016414642329, 51.8016414642329, 0.4094295501709, 347.96218872069903], 80:
[2.0235261917114, 198.48083496094, 32.722767829895396, 198.48083496094,
32.722767829895396, 32.722767829895396, 2.0235261917114, 295.030426025393], 83:
[98.583602905273, 174.07369995117, 168.777114868164, 174.07369995117,
```

168.777114868164, 168.777114868164, 98.583602905273, 222.795471191404]}, 1: {4: [433.82736206055, 140.18608093262, 574.65814208985, 140.18608093262, 574.65814208985, 574.65814208985, 433.82736206055, 364.31036376953], 12: [604.71514892578, 134.31509399414, 715.62811279297, 134.31509399414, 715.62811279297, 715.62811279297, 604.71514892578, 409.79641723633], 21: [401.23931884766, 7.1558227539062, 654.34075927735, 7.1558227539062, 654.34075927735, 654.34075927735, 401.23931884766, 185.4366607666062], 22: [646.75958251953, 106.69625854492, 705.837097167968, 106.69625854492, 705.837097167968, 705.837097167968, 646.75958251953, 240.37161254883], 24: [349.21697998047, 114.35859680176, 629.78515625, 114.35859680176, 629.78515625, 629.78515625, 349.21697998047, 226.37480163574], 26: [477.59033203125, 146.5467376709, 567.189575195312, 146.5467376709, 567.189575195312, 567.189575195312, 477.59033203125, 235.793960571291], 27: [485.85934448242, -1.0197563171387, 621.4287109375, -1.0197563171387, 621.4287109375, 621.4287109375, 485.85934448242, 106.2776832580613], 28: [680.99932861328, 124.71248626709, 717.225280761718, 124.71248626709, 717.225280761718, 717.225280761718, 680.99932861328, 194.826873779297], 29: [448.34063720703, 349.81948852539, 722.040466308591, 349.81948852539, 722.040466308591, 722.040466308591, 448.34063720703, 401.107025146484], 30: [340.86090087891, 136.69076538086, 380.316223144535, 136.69076538086, 380.316223144535, 380.316223144535, 340.86090087891, 233.996490478516], 33: [663.669921875, 220.15365600586, 715.994018554688, 220.15365600586, 715.994018554688, 715.994018554688, 663.669921875, 288.562042236329031], 41: [685.29968261719, 49.203311920166, 718.156372070315, 49.203311920166, 718.156372070315, 718.156372070315, 685.29968261719, 159.489795684816], 48: [542.16784667969, 54.987197875977, 664.89428710938, 54.987197875977, 664.89428710938, 664.89428710938, 542.16784667969, 367.228286743167], 51: [685.09216308594, 233.41772460938, 717.875244140628, 233.41772460938, 717.875244140628, 717.875244140628, 685.09216308594, 368.32220458985], 52: [633.55755615234, 220.72299194336, 684.702819824215, 220.72299194336, 684.702819824215, 684.702819824215, 633.55755615234, 380.94039916992], 64: [621.93902587891, 60.212985992432, 715.15679931641, 60.212985992432, 715.15679931641, 715.15679931641, 621.93902587891, 169.318378448482], 67: [650.03588867188, 353.01422119141, 718.833618164068, 353.01422119141, 718.833618164068, 718.833618164068, 650.03588867188, 402.060974121098], 69: [512.96081542969, 183.83453369141, 563.992797851565, 183.83453369141, 563.992797851565, 563.992797851565, 512.96081542969, 271.245635986332], 71: [394.00061035156, 145.63208007812, 500.50396728515, 145.63208007812, 500.50396728515, 500.50396728515, 394.00061035156, 283.845062255859], 74: [687.69171142578, 159.34019470215, 718.031311035155, 159.34019470215, 718.031311035155, 718.031311035155, 687.69171142578, 277.25933837891], 75: [341.34823608398, 387.19903564453, 544.697082519531, 387.19903564453, 544.697082519531, 544.697082519531, 341.34823608398, 403.616333007811], 76: [347.10791015625, 184.33923339844, 402.271667480469, 184.33923339844, 402.271667480469, 402.271667480469, 347.10791015625, 276.489105224612], 81: [371.59710693359, 185.93217468262, 452.247436523434, 185.93217468262, 452.247436523434, 452.247436523434, 371.59710693359, 324.17645263672], 86: [553.01025390625, 115.20367431641, 625.592407226562, 115.20367431641, 625.592407226562, 625.592407226562, 553.01025390625, 193.609619140629], 87: [585.16198730469, 61.770477294922, 676.045043945315, 61.770477294922, 676.045043945315, 676.045043945315, 585.16198730469, 126.325149536133], 88: [607.08245849609, 239.96371459961, 707.16986083984, 239.96371459961, 707.16986083984, 707.16986083984, 607.08245849609, 303.425567626954], 96: [449.50152587891, 121.13920593262, 585.0307006836, 121.13920593262, 585.0307006836, 585.0307006836, 449.50152587891, 188.34831237793202], 102: [526.54119873047, 111.1600112915, 606.448791503908, 111.1600112915, 606.448791503908, 606.448791503908, 526.54119873047, 272.56339263916], 103: [565.59020996094, 1.0208740234375, 642.20202636719, 1.0208740234375, 642.20202636719, 642.20202636719, 565.59020996094, 205.0808105468775], 105: [599.79534912109, 76.991355895996, 656.011047363278, 76.991355895996, 656.011047363278, 656.011047363278, 599.79534912109, 219.94942474365598], 106: [447.22500610352, 193.98551940918, 512.35440063477, 193.98551940918, 512.35440063477, 512.35440063477, 447.22500610352, 317.68020629883], 108: [515.64617919922, 319.14685058594, 592.643371582032, 319.14685058594, 592.643371582032, 592.643371582032, 515.64617919922, 392.885803222659], 110: [559.35052490234, 314.83517456055, 628.787658691402, 314.83517456055, 628.787658691402, 628.787658691402, 559.35052490234, 385.247528076175], 111: [689.30627441406, 28.026664733887, 718.714599609372, 28.026664733887, 718.714599609372, 718.714599609372, 689.30627441406, 92.806190490723], 112: [416.43389892578, 118.42413330078, 501.149597167968,

118.42413330078, 501.149597167968, 501.149597167968, 416.43389892578, 191.149444580077], 113: [467.1291809082, 294.23764038086, 547.002197265622, 294.23764038086, 547.002197265622, 547.002197265622, 467.1291809082, 391.27273559570403], 114: [487.921875, 274.52001953125, 674.01953125, 274.52001953125, 674.01953125, 674.01953125, 487.921875, 394.69891357422], 115: [559.17156982422, 191.69918823242, 620.355407714845, 191.69918823242, 620.355407714845, 620.355407714845, 559.17156982422, 334.34451293945], 116: [546.55718994141, 261.76538085938, 604.893493652348, 261.76538085938, 604.893493652348, 604.893493652348, 546.55718994141, 371.12091064454], 117: [504.34713745117, 252.02328491211, 571.144775390623, 252.02328491211, 571.144775390623, 571.144775390623, 504.34713745117, 368.13687133789]}, 2: {0: [5.760871887207, 178.60296630859, 205.246620178227, 178.60296630859, 205.246620178227, 205.246620178227, 5.760871887207, 362.41131591797], 6: [156.02764892578, 113., 184, 403.616333007811]}, 56: {}, 57: {}, 58: {}, 59: {}, 60: {}, 61: {}, 62: {}, 63: {}, 64: {}, 65: {}, 66: {}, 67: {}, 68: {}, 69: {}, 70: {97: [1.727840423584, 2.7788591384888, 76.290004730225, 2.7788591384888, 76.290004730225, 76.290004730225, 1.727840423584, 11.2132940292359]}, 71: {}, 72: {}, 73: {84: [330.38565063477, 1.7220611572266, 467.97378540039, 1.7220611572266, 467.97378540039, 467.97378540039, 330.38565063477, 92.0827178955076], 112: [416.43389892578, 118.42413330078, 501.149597167968, 118.42413330078, 501.149597167968, 501.149597167968, 416.43389892578, 191.149444580077]}, 74: {}, 75: {}, 76: {}, 77: {}, 78: {}, 79: {104: [8.7613410949707, 384.8298034668, 100.44926834106471, 384.8298034668, 100.44926834106471, 100.44926834106471, 8.7613410949707, 402.510650634769]}, 80: {}, 81: {}, 82: {}, 83: {}, 84: {}, 85: {93: [154.43676757812, 63.892105102539, 217.400787353511, 63.892105102539, 217.400787353511, 217.400787353511, 154.43676757812, 119.476806640625], 100: [198.70899963379, 78.647109985352, 240.964706420899, 78.647109985352, 240.964706420899, 240.964706420899, 198.70899963379, 121.702865600586]}, 86: {}, 87: {}, 88: {}, 89: {}, 90: {}, 91: {}, 92: {}, 93: {}, 94: {}, 95: {}, 96: {}, 97: {}, 98: {}, 99: {101: [0.47516250610352, 5.3240346908569, 88.23997879028352, 5.3240346908569, 88.23997879028352, 88.23997879028352, 0.47516250610352, 24.567236900329902], 107: [3.1521282196045, 8.5281276702881, 41.1567821502685, 8.5281276702881, 41.1567821502685, 41.1567821502685, 3.1521282196045, 57.26284599304211]}, 100: {}, 101: {97: [1.727840423584, 2.7788591384888, 76.290004730225, 2.7788591384888, 76.290004730225, 76.290004730225, 1.727840423584, 11.2132940292359]}, 102: {}, 103: {}, 104: {}, 105: {}, 106: {}, 107: {}, 108: {}, 109: {}, 110: {}, 111: {}, 112: {}, 113: {}, 114: {108: [515.64617919922, 319.14685058594, 592.643371582032, 319.14685058594, 592.643371582032, 592.643371582032, 515.64617919922, 392.885803222659], 110: [559.35052490234, 314.83517456055, 628.787658691402, 314.83517456055, 628.787658691402, 628.787658691402, 559.35052490234, 385.247528076175]}, 115: {}, 116: {}, 117: {}, 118: {}}

Step 5. grouping of captions complete inclusive and partly inclusive updated

Code:

```
import os
import json
x = 0
y = 1
g = 0
i = 0
cpt = {}
for file in os.listdir('/Users/arin/Desktop/final_yr_project/frame_extract/college/
grouping/'):
    l = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/college/
grouping/{}'.format(file)))
    if i == 2:
        i = 0
        y = y + 1
    r = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/college/
collegeresults/results{}.json'.format(y)))
    c = r['results'][i]['captions']
    for key in l:
        cptlist = []
        for k in l[key].keys():
            t = {k:c[int(k)]}
            cptlist.append(t)
```

```
        cpt[str((key, c[int(key)]))] = cptlist
    i = i + 1
    print(cpt)
    print('----------------------')
    json.dump(cpt, open('/Users/arin/Desktop/final_yr_project/frame_extract/
college/grptocpt/cpt{}.json'.format(g), 'w'))
    g = g + 1
```

Output:

{"('0', 'a black car parked on the street')": [{'11': 'license plate on the car'}, {'14': 'the car is blue'}, {'20': 'a silver car'}, {'31': 'a black and white sign'}, {'40': 'the car is black'}, {'54': 'the car is black'}, {'57': 'the front of a car'}, {'59': 'the car is blue'}, {'77': 'a car on the road'}, {'80': 'a black car'}, {'83': 'the car is blue'}], "('1', 'a white wall')": [{'4': 'a white door'}, {'12': 'white metal pole'}, {'21': 'green leaves on tree'}, {'22': 'a street light on a pole'}, {'24': 'a building in the background'}, {'26': 'a white and black door'}, {'27': 'green leaves on tree'}, {'28': 'a light on a pole'}, {'29': 'a patch of grass'}, {'30': 'a tree in the background'}, {'33': 'a black door on the wall'}, {'41': 'a light on the wall'}, {'48': 'a white wall'}, {'51': 'a black pole'}, {'52': 'white door on the building'}, {'64': 'a white door on the building'}, {'67': 'a white building'}, {'69': 'a white door on the door'}, {'71': 'a door on the side of a train'}, {'74': 'a black metal pole'}, {'75': 'the sidewalk is made of bricks'}, {'76': 'green grass growing on the side of the road'}, {'81': 'a green and white train'}, {'86': 'a green plant on the wall'}, {'87': 'a pole in the background'}, {'88': 'a white door on the wall'}, {'96': 'a brown wooden door'}, {'102': 'a white wall'}, {'103': 'a tree in the background'}, {'105': 'white pole in the background'}, {'106': 'white paint on the door'}, {'108': 'part of a floor'}, {'110': 'part of a floor'}, {'111': 'a black and white sign'}, {'112': 'a green and white house'}, {'113': 'part of a floor'}, {'114': 'part of a white wall'}, {'115': 'white paint on wall'}, {'116': 'part of a white wall'}, {'117': 'part of a wall'}], "('2', 'a car parked in a parking lot')": [{'0': 'a black car parked on the street'}, {'6': 'a small green bush'}, {'8': 'green plant growing on the ground'}, {'9': 'a paved road'}, {'11': 'license plate on the car'}, {'13': 'a small tree'}, {'14': 'the car is blue'}, {'15': 'green tree in front of building'}, {'16': 'plant in front of a window'}, {'17': 'a red and white sign'}, {'19': 'a white car parked on the street'}, {'20': 'a silver car'}, {'23': 'a sidewalk with a lot of tracks'}, {'25': 'leaves on the ground'}, {'30': 'a tree in the background'}, {'31': 'a black and white sign'}, {'32': 'a person standing on the sidewalk'}, {'34': 'green leaves on the ground'}, {'37': 'window on a building'}, {'38': 'a person standing on a sidewalk'}, {'39': 'a tree on the sidewalk'}, {'40': 'the car is black'}, {'42': 'a small tree on the sidewalk'}, {'43': 'a tree in the background'}, {'44': 'a white building with a red roof'}, {'45': 'green leaves on the tree'}, {'46': 'a train track'}, {'47': 'a patch of dirt'}, {'50': 'a white car on the road'}, {'53': 'a house in the background'}, {'54': 'the car is black'}, {'56': 'dirt on the ground'}, {'57': 'the front of a car'}, {'59': 'the car is blue'}, {'60': 'the car is black'}, {'61': 'a small door on the building'}, {'62': 'green tree on the side of the road'}, {'63': 'a green tree in front of a building'}, {'65': 'a tree on the sidewalk'}, {'66': 'a car parked on the street'}, {'68': 'a window on a building'}, {'71': 'a door on the side of a train'}, {'72': 'a shadow on the ground'}, {'76': 'green grass growing on the side of the road'}, {'77': 'a car on the road'}, {'78': 'a green bush'}, {'79': 'the ground is made of concrete'}, {'80': 'a black car'}, {'81': 'a green and white train'}, {'82': 'green leaves on the tree'}, {'83': 'the car is blue'}, {'89': 'a window on a building'}, {'90': 'white car parked on the street'}, {'91': 'a green tree'}, {'92': 'window on a building'}, {'93': 'a window on a building'}, {'94': 'green leaves on the tree'}, {'95': 'a tree in the background'}, {'98': 'a green tree in front of the building'}, {'100': 'window on the building'}, {'104': 'part of a floor'}, {'106': 'white paint on the door'}, {'109': 'a green bush'}, {'112': 'a green and white house'}, {'118': 'white line on the road'}], "('3', 'a tree in the background')": [{'15': 'green tree in front of building'}, {'36': 'a tree in the distance'}, {'53': 'a house in the background'}, {'61': 'a small door on the building'}, {'63': 'a green tree in front of a building'}, {'68': 'a window on a building'}, {'97': 'power lines in the sky'}, {'99': 'power lines in the sky'}, {'101': 'power lines in the sky'}, {'107': 'a tree in the distance'}], "('4', 'a white door')": [{'26': 'a white and black door'}, {'69': 'a white door on the door'}, {'106': 'white paint on the door'}, {'117': 'part of a wall'}], "('5', 'a tree with green leaves')": [{'13': 'a small tree'}, {'16': 'plant in front of a window'}, {'17': 'a red and white sign'}, {'25': 'leaves on the ground'}, {'30': 'a tree in the background'}, {'32': 'a person standing on the sidewalk'}, {'35': 'palm
```
```

tree in the background'}, {'38': 'a person standing on a sidewalk'}, {'42': 'a small tree on the sidewalk'}, {'45': 'green leaves on the tree'}, {'65': 'a tree on the sidewalk'}, {'72': 'a shadow on the ground'}, {'76': 'green grass growing on the side of the road'}, {'78': 'a green bush'}, {'82': 'green leaves on the tree'}, {'91': 'a green tree'}, {'92': 'window on a building'}, {'94': 'green leaves on the tree'}, {'95': 'a tree in the background'}, {'98': 'a green tree in front of the building'}, {'100': 'window on the building'}, {'109': 'a green bush'}], "('6', 'a small green bush')": [{'8': 'green plant growing on the ground'}, {'16': 'plant in front of a window'}, {'45': 'green leaves on the tree'}, {'78': 'a green bush'}, {'82': 'green leaves on the tree'}, {'92': 'window on a building'}, {'94': 'green leaves on the tree'}, {'95': 'a tree in the background'}, {'98': 'a green tree in front of the building'}, {'109': 'a green bush'}], "('7', 'green leaves on tree')": [{'27': 'green leaves on tree'}, {'35': 'palm tree in the background'}, { and white train')": [], "('82', 'green leaves on the tree')": [], "('83', 'the car is blue')": [], "('84', 'green leaves on tree')": [], "('85', 'a house in the background')": [{'93': 'a window on a building'}, {'100': 'window on the building'}], "('86', 'a green plant on the wall')": [], "('87', 'a pole in the background')": [], "('88', 'a white door on the wall')": [], "('89', 'a window on a building')": [], "('90', 'white car parked on the street')": [], "('91', 'a green tree')": [], "('92', 'window on a building')": [], "('93', 'a window on a building')": [], "('94', 'green leaves on the tree')": [], "('95', 'a tree in the background')": [], "('96', 'a brown wooden door')": [], "('97', 'power lines in the sky')": [], "('98', 'a green tree in front of the building')": [], "('99', 'power lines in the sky')": [{'101': 'power lines in the sky'}, {'107': 'a tree in the distance'}], "('100', 'window on the building')": [], "('101', 'power lines in the sky')": [{'97': 'power lines in the sky'}], "('102', 'a white wall')": [], "('103', 'a tree in the background')": [], "('104', 'part of a floor')": [], "('105', 'white pole in the background')": [], "('106', 'white paint on the door')": [], "('107', 'a tree in the distance')": [], "('108', 'part of a floor')": [], "('109', 'a green bush')": [], "('110', 'part of a floor')": [], "('111', 'a black and white sign')": [], "('112', 'a green and white house')": [], "('113', 'part of a floor')": [], "('114', 'part of a white wall')": [{'108': 'part of a floor'}, {'110': 'part of a floor'}], "('115', 'white paint on wall')": [], "('116', 'part of a white wall')": [], "('117', 'part of a wall')": [], "('118', 'white line on the road')": []}

Step 6. Sentence making

Code:

```
import os
import json
textsent = []
j = 0
x = 0
i = 0
while x <= 7:
    data = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/
college/grptocpt/cpt{}.json'.format(x)))
    keylist = list(data.keys())
    i = 0
    while i < len(keylist):
        cap = list(data[keylist[i]])
#         print('################cpt{}.json'.format(x))
#         print(cap, len(cap))

        if len(cap) > 1:
            capdict = dict()
            for num, items in enumerate(cap):
                for key in items.keys():
                    capdict[num] = items[key]
            text = set([capdict[key] for key in capdict.keys()])
#             print(text)
#             print('--------------')
            mystring = ""
            for sen in list(text):
                mystring = mystring + sen + ". "
            textsent.insert(j, str(mystring))
            j = j + 1
#         else:
#             print("not added")
```

```
        i = i + 1
    x = x + 1
    print(set(textsent))
    print('----------------')
# print (set(textsent))
json.dump(list(set(textsent)),open('/Users/arin/Desktop/final_yr_project/
frame_extract/college/collegeresults/textsent.json','w'))
```

Output:

```
['a white building with a red roof. window on a building. a house with a roof. a
window on a building. green tree on the side of the road. ', 'a green tree in front
of a building. power lines in the sky. a tree in the distance. a window on a
building. a small door on the building. a house in the background. green tree in
front of building. ', 'a white and black door. a white door on the door. white
paint on the door. part of a wall. ', 'a green plant on the wall. a brown wooden
door. a white and black door. a tree in the background. a green and white house. ',
'part of a floor. ', 'the front of a car. the ground is made of concrete. part of a
floor. license plate on the car. green leaves on the ground. green plant growing on
the ground. dirt on the ground. the car is black. white line on the road. ', 'power
lines in the sky. a tree in the distance. ', 'a green plant on the wall. a brown
wooden door. green leaves on tree. a green and white house. ', 'green leaves on
tree. a green and white house. ', 'a green plant on the wall. white paint on wall.
part of a white wall. white pole in the background. ', 'a small tree on the
sidewalk. green leaves on the tree. a green bush. a person standing on a sidewalk.
', 'a silver car. a black car. the car is blue. the front of a car. license plate
on the car. a black and white sign. a car on the road. the car is black. ', 'window
on a building. a small tree on the sidewalk. a green tree in front of the building.
a green tree in front of a building. leaves on the ground. a black and white sign.
green leaves on the ground. a car parked on the street. a car on the road. a small
green bush. a green bush. green tree in front of building. the ground is made of
concrete. a white building with a red roof. a house in the background. a train
track. a shadow on the ground. a black car. white line on the road. white paint on
the door. a white car on the road. green tree on the side of the road. a tree in
the background. a window on a building. white car parked on the street. a person
standing on a sidewalk. a black car parked on the street. a green tree. the front
of a car. a patch of dirt. a paved road. window on the building. a red and white
sign. plant in front of a window. a small door on the building. the car is black. a
person standing on the sidewalk. license plate on the car. a white car parked on
the street. a green and white train. green leaves on the tree. a silver car. a
green and white house. the car is blue. a tree on the sidewalk. part of a floor. a
sidewalk with a lot of tracks. green plant growing on the ground. a small tree.
dirt on the ground. a door on the side of a train. green grass growing on the side
of the road. ', 'a white door on the wall. white door on the building. a light on a
pole. a black pole. a black door on the wall. a black metal pole. a white building.
', 'window on the building. a window on a building. ', 'a white door on the wall. a
patch of grass. a brown wooden door. green leaves on tree. part of a white wall. a
black and white sign. a black door on the wall. white paint on wall. a white
building. white pole in the background. part of a wall. a white door on the
building. a door on the side of a train. a light on a pole. a light on the wall. a
pole in the background. a tree in the background. a white door. a green and white
house. a street light on a pole. white metal pole. a white wall. a green plant on
the wall. white paint on the door. a green and white train. white door on the
building. a building in the background. part of a floor. a black pole. a white door
on the door. a black metal pole. green grass growing on the side of the road. a
white and black door. the sidewalk is made of bricks. ', 'green leaves on the tree.
window on a building. a tree in the background. a green tree in front of the
building. a green bush. green plant growing on the ground. plant in front of a
window. ', 'green leaves on tree. palm tree in the background. ', 'green leaves on
the tree. window on a building. a tree in the background. a person standing on a
sidewalk. a green tree in front of the building. a green tree. a tree on the
sidewalk. a shadow on the ground. leaves on the ground. window on the building.
green grass growing on the side of the road. a green bush. a red and white sign. a
small tree. plant in front of a window. a small tree on the sidewalk. a person
standing on the sidewalk. palm tree in the background. ', 'part of a white wall.
the sidewalk is made of bricks. part of a floor. a shadow on the ground. part of a
wall. ', 'green grass growing on the side of the road. a tree in the background. ',
'green leaves on the tree. white car parked on the street. a green tree in front of
```

```
the building. the car is blue. a black and white sign. a green bush. a car parked
on the street. a house in the background. a white car on the road. ']
```

Step 7. Summarization

```
from collections import Counter
import nltk
import re
import json
def commonprefix(m):
    if not m:
        return ''
s1 = min(m)
#     print s1
    s2 = max(m)
#     print s2
    for i, c in enumerate(s1):
        if c != s2[i]:
            return s1[:i]
    return s1

def makesent(text):
    sents = list(filter(lambda x: x!= '', [line.strip() for line in
text.split('.')]))
#     print (sents)
#     print ('-------------')
    tags, target_sents_NN, target_sents_VBG, req_lines = list(), list(), list(),
list()
    dummylines, objects = list(), list()
    lookup, target = dict(), dict()

    for line in sents:
        parse = nltk.word_tokenize(line)
        tags.append(nltk.pos_tag(parse))
    counts = Counter([item for line in tags for item in line])
#     print tags

    targetVBG = None
    targetNN = None
    for item in counts.most_common():
        if item[0][1] == 'NN':
            targetNN = item[0][0]
            break

    for item in counts.most_common() :
        if item[0][1] == 'DT':
            targetDT = item[0][0]
            break

    for key, value in counts.items():
        lookup.setdefault(key[1],[]).append(key[0])

    for line in tags:
        for word in line:
            if word[0] == targetNN:
                target_sents_NN.append(line)
#                 print line
#                 print '-------------------'
#     targetVBG = str(0)
#     targetNN = str(0)

    for item in Counter([item for line in target_sents_NN for item in
line]).most_common():
        if item[0][1] == 'VBG':
            targetVBG = str(item[0][0])
            break

    if targetVBG is None:
#         print("no targetVBG")
```

```
        return;
    elif targetNN is None:
#         print("no targetNN")
        return;
    else :
        for line in target_sents_NN:
            if (targetNN, 'NN') in line and (targetVBG, 'VBG') in line:
                target_sents_VBG.append(line)

    if 'VBZ' not in lookup:
#         print("cannot display")
        pass
    else:
        VBZ_list = lookup['VBZ']
#         print VBZ_list
#     if len(VBZ_list) == 0:
#         print "Cannot display"
#         return
#     else:
        target_prefix = [' '.join([targetNN, targetVBG])]
        target_prefix.append(' '.join([targetDT, targetNN, targetVBG]))
#         print target_prefix
        for vbz in VBZ_list:
            target_prefix.append(' '.join([targetDT, targetNN, vbz, targetVBG]))
#             print target_prefix

        for line in sents:
            for tp in target_prefix:
                if line.startswith(tp):
                    req_lines.append(line)
#                     print req_lines

        for line in req_lines:
            dummylines.append(line.replace(targetDT + ' ', ''))
#         print dummylines

        prefix = commonprefix(dummylines)

        for line in dummylines:
            line = re.sub(prefix, '',line)
            if 'and ' in line:
                diff = list()
                diff = line.split(' and ')
                for d in diff:
                    objects.append(d)
                    break
            else:
                objects.append(line)
#             print objects

        obj = list(set(objects))
#         print obj
        obj.sort()
        obj.sort(key=len)
#         print(obj)
#         print('-----------')
        req_objects = []
#         req_objects.append(str(filter(lambda x: [x for i in obj if x in i and x !
= i] == [], obj)))
        req_objects = list(filter(lambda x: [x for i in obj if x in i and x != i]
== [], obj))
#         print(len(req_objects))
        if len(req_objects) > 0:
            final_text = ' '.join([targetDT, prefix]) + ', '.join(req_objects[:-1])
+ ' and ' + req_objects[-1]
            mysummary.append(final_text.capitalize())


if __name__ == '__main__':
```

```
    mysummary = []
    mytext = json.load(open('/Users/arin/Desktop/final_yr_project/frame_extract/
college/collegeresults/textsent.json'))
#     print(mytext[0])
    for item in mytext:
        makesent(item)
    print(str(mysummary))
#     text = raw_input("Enter your text:")
#     print "Your text:" + text
#     text = "the man has glasses. a man wearing a white shirt. a white door. a
pair of blue jeans. the shirt is black. a white shirt on a man. man wearing a suit
and tie. a man in a white shirt. a glass of wine. man has white shirt. a wooden
table. a white box. a white box on the table. a man in a black shirt. a man wearing
a tie. the mans hair is black. a wooden door. man wearing a black suit. a picture
of a woman. a person standing in the background. a white napkin in the background.
black hat on mans head. man wearing a white shirt. a chair with a white seat. a
pair of brown pants. the man is wearing a tie."
#     makesent(text)
```

Output:

```
'A metal railing on the building, black road and black car’
```