1. **AroundTheWorldLocal.py:** This is the local version of the crawler. It reads keywords from the input file (citiesInTheWorld.txt or countiesInTheUS.txt or countriesInTheWorld.txt), crawls Flickr API to collect Images, saves them on disk and write a json file as output (data\_City.json or data\_Country.json or data\_County.json) which contains all details of the image downloaded
2. **AroundTheWorldCluster.py:** This is the cluster version of the crawler. It reads keywords from the input file (citiesInTheWorld.txt or countiesInTheUS.txt or countriesInTheWorld.txt), crawls Flickr API to collect Images, saves them on disk and write a json file as output (data\_City.json or data\_Country.json or data\_County.json) which contains all details of the image downloaded
3. **citiesInTheWorld.txt:** This text file contains the list of all populated regions of the world. Each region is in a new line. This is used as input to the file *AroundTheWorldLocal.py*
4. **countiesInTheUS.txt:** This text file contains the list of all counties in the US. Each region is in a new line. This is used as input to the file *AroundTheWorldLocal.py*
5. **countriesInTheWorld.txt:** This text file contains the list of all the countries of the world. Each region is in a new line. This is used as input to the file *AroundTheWorldLocal.py*
6. **data\_City.json:** This json file is the output of the code AroundTheWorldLocal.py or AroundTheWorldCluster.py. It contains the list with details of all the images of **cities** downloaded from flickr. A sample of this file is:



1. **data\_Country.json:** This json file is the output of the code AroundTheWorldLocal.py or AroundTheWorldCluster.py. It contains the list with all the details of the images of countries downloaded from flickr.
2. **data\_County.json:** This json file is the output of the code AroundTheWorldLocal.py or AroundTheWorldCluster.py. It contains the list with all the details of the images of countries downloaded from flickr. A sample of this file is:



1. **extractDataMongo.py:** This python code generates all the bounding boxes of all quadrants at a zoom level. It also searches the Mongo database for the list of images bounding boxes overlap with the bounding box of each quadrant at that zoom level. Work on this code logic is still in progress. In the future this code will be capable of:

* calculating the Jaccard similarity of the images’ bounding box with respect to quadrants and integrate this value to the current formula that is calculating the score for an image.
* Finding the image with the top score for each quadrant at each zoom level
* Save the image with the naming convention: tile-z-x-y.png

1. **get\_location.py:** This python file reads names of locations from original data file (citiesInTheWorld.txt or countiesInTheUS.txt or countriesInTheWorld.txt) and then creates a place-coordinates dictionary. These coordinates are the north, south, east and west boundaries.
2. **add\_coord.py:** This python file adds the coordinates to the original data file in a new field called ‘box’
3. **data\_County\_new.json:** This is the new json after addition of the ‘box’ with north, south, east and west boundaries (as a result of running of ‘add\_coord.py’). A sample of this is:

