# DOCUMENTATION OF CRAWLER APP -

Rajasekhar Jetty

Click to follow the links:

# [INTRODUCTION](#_INTRODUCTION)

# [IMAGE](#_Image_Crawler_Application)CRAWLER [APPLICATION](#_Image_Crawler_Application)

# [ANALYZE METADATA](#_ANALYZE_METADATA)

# [FETCH SITE LOCATIONS](#_FETCH__SIGHT)

# [LOADING](#_LOAD_IMAGES) [IMAGES](#_LOAD_IMAGES)

# [GEOCODING LOCATIONS](#_PLOTTING_THE_LOCATIONS_1)

# [PROGRAMMING](#_Programming__Resources:_1) [RESOURCES](#_Programming__Resources:_1)

# [CHALLENGES](#_CHALLENGES_FACED)

# [SUMMARY](#_SUMMARY)

# INTRODUCTION

I started by collecting the relevant information from the web for summarizing the top 100 tourist spots in the world. After complete analysis of many travel websites and going through many top 50 lists .I finalized the top 100 tourist spots.

We have many websites that give relevant information about the tourist destinations but going through each and every website and collecting the exact the photos relevant to the sites is time consuming and error driven as we cannot tag the photo to the exact tourist site.

I feel that

**“Targeting a reliable source where the photo authenticity is confirmed by the user as well as the website can give us the most accurate photos with location tagging and metadata“**

Is better and smarter than

**“Crawling the entire World Wide Web and fetching the photos which cannot be assured of the location and its contents”**

I found <https://foursquare.com/> is quite appropriate to the problem in context. It has a huge repository of collection of photos that belong to different tourist sites added by the website and the users using this apps.

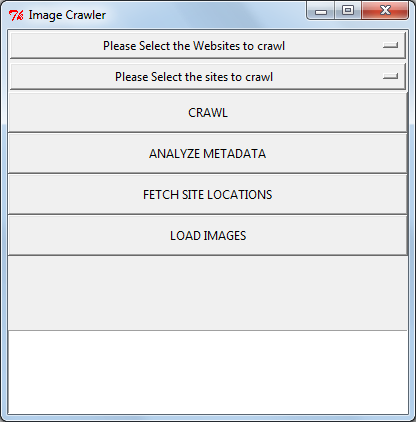
So, if I retrieve a tourist site photo from FOURSQUARE, I can point that photo belongs to particular location and what kind of sites it is such as Park, Beach, and Monument and tag it using Geocode.

# IMAGE CRAWLER APPLICATION

I built a desktop application that takes our tourist site list and can crawl the Foursquare website for the photos belong to the tourist site.

Following tasks are done by the app:

1. Crawling and generating the photo URLs for the given tourist site list.
2. Analyze the Metadata of the photos collected.
3. Downloading the photos based on the URLs generated.
4. Create a repository of all the site names and the cities collected from the crawler.
5. Plot the sites on the World Maps



After selecting the Website and the city, we have to click on the CRAWL button to initiate the crawling routine.(App is just meant for user interface, no validations done on Inputs)

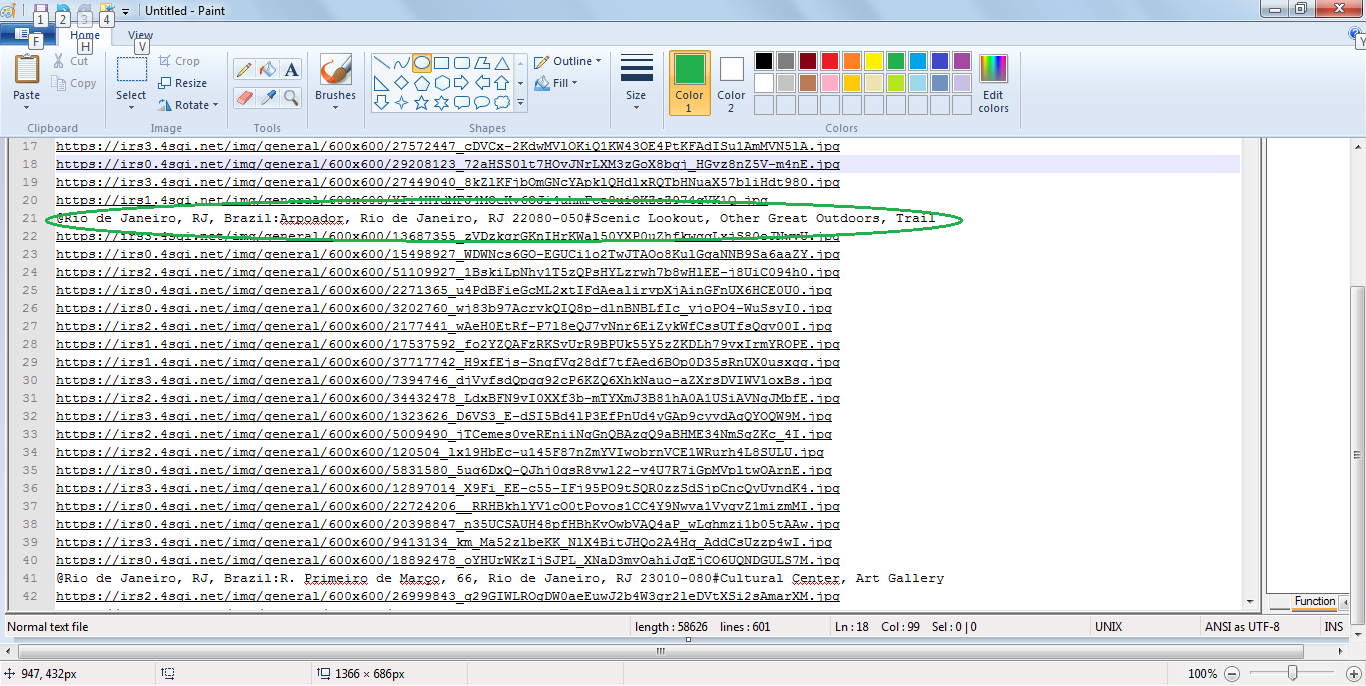
Crawling routine is composed of independently crawling threads.

If we select the “ALL” option in the city dropdown, a spider thread is initiated for each city in the dropdown and keeps collecting the photo links and writing this links to individual text files in the following location in such a way that all the photo URLS belonging to a particular sites are saved in a single file.

Once the crawling is completed, we can find all the text files with the city names indicating each textfile name.

@pic

The following structure is followed for the text files:



@CITYNAME : SITE ADDRESS\_1 #METADATA of the site

Photourl1

Photourl2

…..

…..

@CITYNAME : SITE ADDRESS\_2#METADATA of the site

Photourl1

Photourl2

…..

…..

So,this files are more like the temporary files which can store the links and provide them to the

Image Crawler on the timely basis designated by the user.

It is the user responsibility to clear this data when the image downloading process is completed.

# ANALYZE METADATA

On Clicking the Analyze Metadata button, application parses the metadata of the collected photos and produces a (key, Value) pair file which gives the composition of different categories such as Parks, Monuments, Beaches, Island, Landscape, Plaza etc.

Key indicates the Category in the collection

Value indicates the Count of the respective Category in the collection

(“Monument”,2) (“Trail”,1) (“Scenic Lookout”,1)

Eg: @Rio de Janeiro, RJ, Brazil:Arpoador, Rio de Janeiro, RJ 22080-050#**Scenic Lookout, Other Great Outdoors, Trail**

Metadata of a site is indicated in blue color above.

Rscript uses this Creating the word cloud using “R” script looks like:



Bigger text size indicates our photo collection mostly consists of the respective categories. So, in our photo collection most of the sites are “Beach” and second is “Historic site” and third is “Scenic Lookout”.

# FETCH SITE LOCATIONS

On clicking the Fetch Site Locations button, application gathers all the cities along with the addresses of the tourist spots available in the photo url repository and generates a repository site location data.

Sitelist.txt contains all this location data.

Eg: @**Rio de Janeiro, RJ, Brazil:Arpoador, Rio de Janeiro, RJ 22080-050**

@CITYNAME : SITE ADDRESS\_1

Here CITYNAME is **Rio de Janeiro, RJ, Brazil and** site address is **Arpoador, Rio de Janeiro, RJ 22080-050.**

# LOAD IMAGES

On clicking the LOAD IMAGES, **Image Downloader threads** are triggered and all the photos representing the links present in the repository are downloaded and systematically placed in the respective folders.

As it is a more time consuming (IO) process, it is delinked from the Crawler.

It is more like a batch processing routine where user triggers the image downloading,once the photo links are downloaded.

We will have a separated folder created for each city, which contains all the sites in that city .

Folder name will be “output” present on the C: Drive. It is taking around 30 minutes to download the

45,000 images from the foursquare website.

Service Denied errors are happening occasionally.

## 

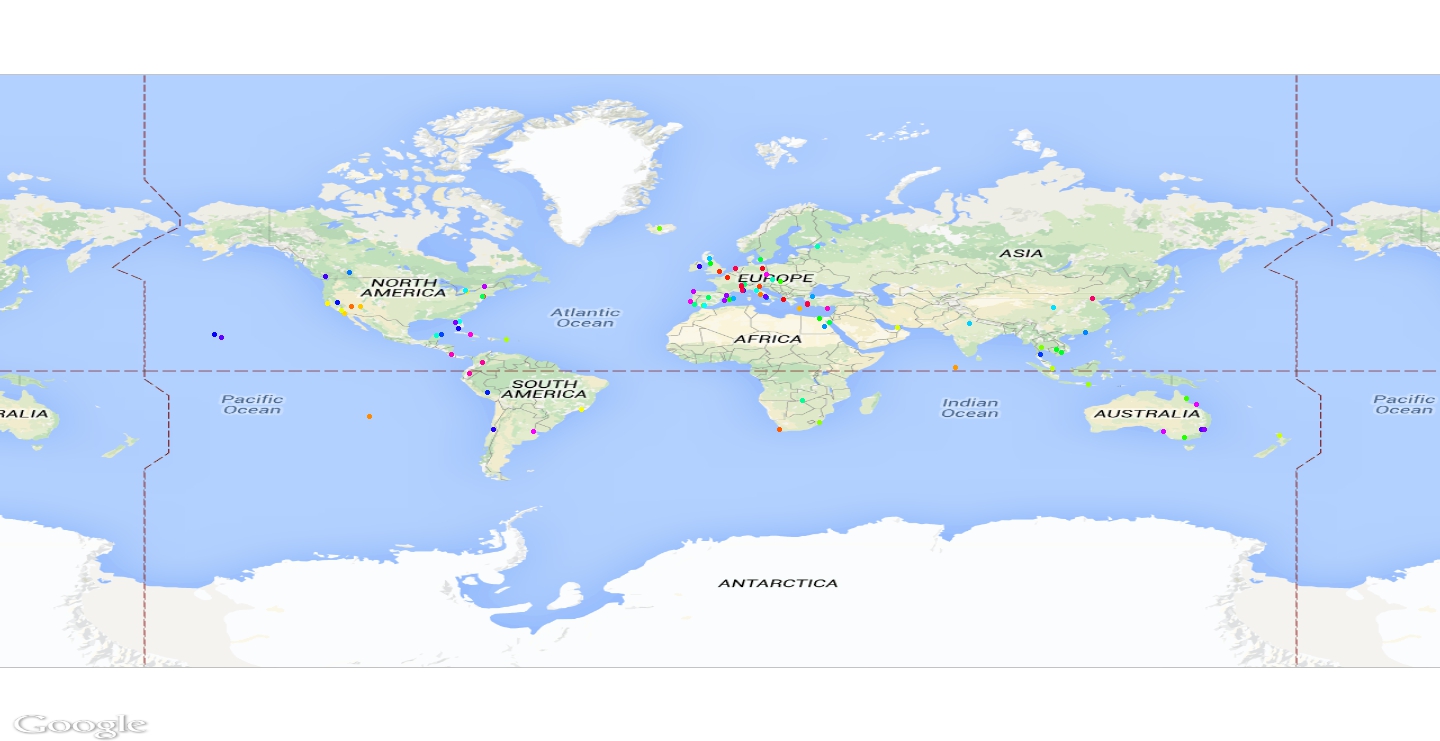
## PLOTTING THE LOCATIONS ON WORLDMAP USING GOOGLE GEOCODE

By using the “R” script which converts the raw address in to longitude and latitude coordinates, we can plot the sites on the world map using the RGooglemaps package.

Google geocode may not give the coordinates sometimes because of the improper string or ambiguity involved in the input.

The colored dots indicate out Top 100 sites locations on the World Map.

This shows the distribution of our Top 100 sites over the world.



# 

# Programming Resources:

**Application Development:**

Language: Python 2.7

Packages :

**BeautifulSoup** for WebCrawling.

**Tkinter** for User Interface.

**Threading** for running the crawlers.

**Data Analysis:**

Language: R 3.1

Packages:

**Rgooglemaps** for Geocoding on the worldmap.

**Wordcloud** for Metadata Analysis.

Execution: First create folders with names “output” and “photourls” in the C: Drive.

Make sure all the packages are installed for python.

Place all the python in one folder and execute the following command :

**python ImageCrawlerApp.py**

and the App starts running and the rest of the process is explained in my video. Please check this:

Final output of folders of images can be found in C:/output

All the metadata collected from the photo urls is used by Rscripts to generate good insites.

# CHALLENGES FACED

1. Getting rid of the “personal photos” tagged by the users to the sites.
2. Collection of metadata of the images to link the **“What they say” to “What they show”.**
3. ‘Unicode String ‘error, As the sites are from different countries, the language they use does not support ASCII encode.
4. Optimizing the image downloading threads.
5. Could not gather more than 70 images for some of the sites due to shortage of the coverage of the Foursquare.

## SUMMARY

Around **45,000** images collected successfully.

Image dataset size: **2.5 GigaBytes**

Please download this datasets at the urls mentioned in the mail communication.

# “Feeling Happy to have a virtual tour of world!! & getting the sneaky job done!! ”