Introduction

This data integration project is a preliminary analysis of walkability in the downtown Los Angeles area. The project integrates authoritative data from LA Metro, LA County, and the LA Sheriff's Department, with volunteered geographic data from Instagram and Foursquare. The data layers are assembled into a web map that is subsequently delivered via web application. The web application is composed of eight tabs that display eight map views of the study area, an overview and each LA Metro stop selected for the project.

I believe that this project will allow end users to explore the downtown Los Angeles area and the points of interest that are presented visually on the map. Users initially presented with an overview, as well as tabs that allow for the exploration of points around each rail stop. This application is valuable to users because they are able to explore the downtown area, and review points of interest from diverse sources. Users can then choose places to visit and plan their trip around the rail stops that service those places.

Data Collection and Assessment

My web application employs six data layers in order to perform the preliminary walkability analysis: OpenStreetMap, LA Metro Rail Stops, LA County LMS Data, LA County Sheriff's Data, Instagram Data, and Foursquare Data. I chose to use this mix because it includes both authoritative data and VGI data and affords me the opportunity to work with more diverse data sets.

OpenStreetMap

I chose to use OpenStreetMap as the base map for my project because of the potential for updates to be reflected immediately. Moreover, there are a number of locations marked with labels on the OpenStreetMap base layer. The presence of these location adds value for the end user.

The OpenStreetMap data appears to be correct, both spatially and temporally, for my study area. The points align with points and lines from authoritative data. Overall I am very pleased with the base map provided by OpenStreetMap. That said I should have been more critical in my analysis of the quality of the OpenStreetMap base layer.

LA County Metro Stops

The LA County Metro Stop data is the foundation of the project. It was used to define the study area, and the sub-areas that are represented in the final web application. The LA Metro data is authoritative, and consequently has a high level of accuracy; however, there were some inconsistencies in the data. The 7th Street Metro Station is referred to as Metro Center in one of the layers. It is also interesting to note that the Red and Purple Lines share a shapefile, while the Blue and Expo lines do not. I suspect that this is for historical reasons. Specifically, the Expo line is much newer than the Blue line, and the Blue line shapefile likely predate the construction of the Expo line. My final judgment is that the data is very accurate and useful for this project.

LA County LMS Data (Points of Interest)

The LA County LMS data provides an authoritative layer containing points of interest provided by Los Angeles County. I selected a subset of the data that fits with the general theme of the map, exploring points of interest on foot in downtown Los Angeles.

Though the data is authoritative and produced by LA County, there are a number of points that appear to be out of position. Often times, these are points that are on a street corner that end up in the center of the intersection. I did not do sufficient analysis to determine whether the error was in the OpenStreetMap base map, or the LA County data; however, the large degree of variation in the LA County data suggest that it contains slight positional errors. This data was nonetheless useful for the project because it allowed the user to get more information about the point if they so desired.

LA County Sheriff's Crime Data

The LA County Sheriff's crime data provides some sense of crime and safety in the area. It adds a second dimension to the application. It is meant to provide a sense of how safe a particular area is to the user. That said, it is not a complete picture, and the locations have been intentionally obscured. It is incomplete because data is released on a month by month basis, previous months did not appear to be available, and annual summaries are not timely. The metadata notes that the locations have been intentionally changed in order to protect the location, which makes the data that much harder to use.

The lack of completeness, and the location skewing, make this layer of moderate use to the end user. Perhaps the use of historical data to identify trends in a particular area would have been a better approach, as opposed to the current point data that I used.

Instagram Data

The Instagram data provides recent photos tagged with tags related to downtown Los Angeles. It provides the end user with some ideas about what other people do when they visit the area. It could potentially stimulate someone to visit a particular point, or take a particular trip.

The Instagram data is not complete, because it is queried from the API via tags, and I did not exhaustively review every tag associated with posts in my study area. Moreover, some of the tags returned data sets that were too large and would have violated the terms of my API key. Consequently, the data is limited.

The data is not spatially accurate, and points representing objects in my study area were found all over the globe. That said, a random selection of points queried within the study area did appear to be relevant. Finally, limiting the data to the study excluded those points, and reduced the data set further. The data is still useful for the project because it provides a thematic element that could be of interest to end users, providing insight into what others believe to be interesting in a given area.

Foursquare Data

I chose to use Foursquare data to identify businesses in my study area following a discussion with a friend who is a business owner. He informed me that the Foursquare data is likely better than alternatives such as Google Places and Yelp. Moreover, the Foursquare data was easier to acquire and work with than other sources such as ReferenceUSA because it is available via API.

The Foursquare data is neither complete nor spatially accurate. The data represents a good approximation of the location of the businesses; however, many of the businesses are not in exact locations. Moreover, some of the points are clumped together, further suggesting that they are not entirely accurate. That said, I am willing to accept the points as sufficiently useful for my project because they appear to be close enough to the place that they represent to be useful, and they provide links that allow users to obtain more information about the point.

Geospatial Data Integration Methods

Defining the study area

The first integration task was to define my study by selecting the rail stations that are in the downtown Los Angeles area. I used my own local knowledge to arbitrarily choose seven stops that service the greater downtown area. I chose the following stops: Chinatown, Union Station, Little Tokyo, Civic Center, Pershing Square, 7th Street Metro Center, and Pico. I decided on a buffer size once the stations were selected. I chose half a mile because it is accepted as a walkable distance.

With the stations and buffer size I was able to visualize the study area in ArcMap. The selected stations are from three distinct rail lines, and consequently are represented in three distinct layers. My first step was to select the stations that I desired by attribute and then create a layer from the selection. This process resulted in three layers that contained only the stations that I intended to use. Several of the stations are home to multiple lines, and I manually excluded the duplicates during the selection process.

The three layers containing the stations were then merged into a single layer via the merge tool. This required several modification to the attribute data, because several of the names were too large and caused the merge tool to generate errors. I was able to merge the three layers into a single layer once those changes were made.

The single layer was used as an input source for the buffer tool to generate the half mile buffer around the stations. I opted to dissolve the buffer so that I could generate one continuous buffer that represented the study area. The conclusion of this process resulted in layers for both the rail stations and the buffer around them. Each of the layers is saved as a shapefile that can be uploaded and used in my web application.

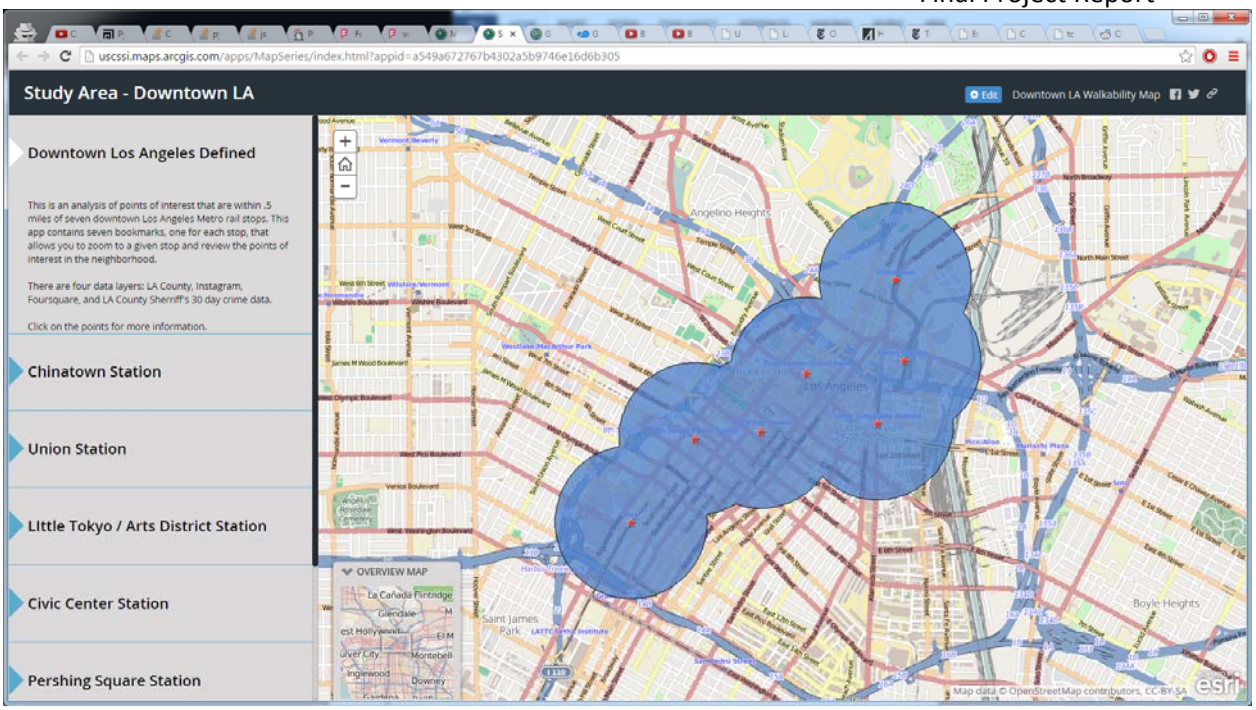

Figure 1 – Study Area and Rail Stations

LA County Points of Interest Data

The LA County LMS Data is delivered as a zipped geodatabase. I downloaded the database, opened it in ArcMap, and reviewed the attribute table. I noted several categories in the CAT1 and CAT2 field that would be of interest to potential users of my application, and I selected by attribute for those elements. I was able to exclude things like social services, health and mental health, and municipal services.

I created a layer from my attribute selection and used it as an input for a select by location to identify the points within my buffer area. I then exported that layer as a shapefile that I could then upload and use in my web application.

LA County Sheriff's Department

The Sheriff's department releases their data monthly as a delimited (CSV) file, which I downloaded. There are LAT and LONG fields in the file, so I was able to open the in ArcMap and plot the XY data in order to visualize the data. I had to select the CA State Plane FIPS V coordinate system when visualizing. I then selected those points that were within my buffer, created a layer from selection, and exported the layer as a shapefile that could be uploaded and used in my web application.

Social Media Data

I wrote several Python scripts to collect the social media data from Instagram and Foursquare and write it into CSV files.[1] The Instagram script used a number of popular tags for the downtown Los Angeles area because lat/long searches only returned 20 results. Searching via tags returned hundreds

[1] See appendices A and B for the Python script code

of paginated results that I could then iterate through and collect a larger, more useful set of Instagram points. I employed a recursive algorithm to iterate over the data set and retrieve all of the points for a given tag. I manually interrupted the process for several tags that returned very large data sets, several thousand records, so that I would not violate the terms of my Instagram API License Key.

The script generated several CSV files containing location and attribute data for photos submitted to Instagram by users. I opened that CSV in ArcMap and plotted the XY data for each CSV file. I then used the resulting layer as part of a select by location to limit the points to those that are within my study area. I did this for each of the CSV files generated by my script and then merged all of the layers into a single Instagram layer. The Instagram layer was then exported to a shapefile that could be uploaded and used in my web application.

The Foursqaure data was collected using a similar recursive algorithm; however, the selection method was proximity to each station. There are two endpoints, location and explore, with explore returning the most popular locations in the area in paginated results. I chose to use the Foursquare API explore endpoint because it would produce better results for my users.

I needed to determine the latitude and longitude of each station for inclusion in my queries. I then hard coded the latitude and longitude into my program, and ran it once for each station with a radius of 805 meters or approximately half a mile. Each execution produced a CSV file with latitude, longitude, and attribute data.

Each of the CSV files was opened in ArcMap, and the XY coordinates were plotted. The resulting layers were then merged into a single Foursquare layer. Finally the layer was exported as a shapefile that could be uploaded and integrated into my web application.

Creating the web application

Each of the shapefiles created in the previous steps was uploaded to ArcGIS Online and added to a new map. I initially created a single map with all of my data points on it. After working with several web application templates I created several copies of the map, which ultimately turned out to be unnecessary.

I tried several web application template once the map was created, and settled on the tabbed template that is reflected in my application. I decided to show the points of interest around each station, with each station represented by a tab. Moreover, I chose to put some basic data as well as Wikipedia links in the attribute pane for each tab. I chose to include an overview map and legend in the main configuration section of the application. Each tab has a custom map position and attribute set.

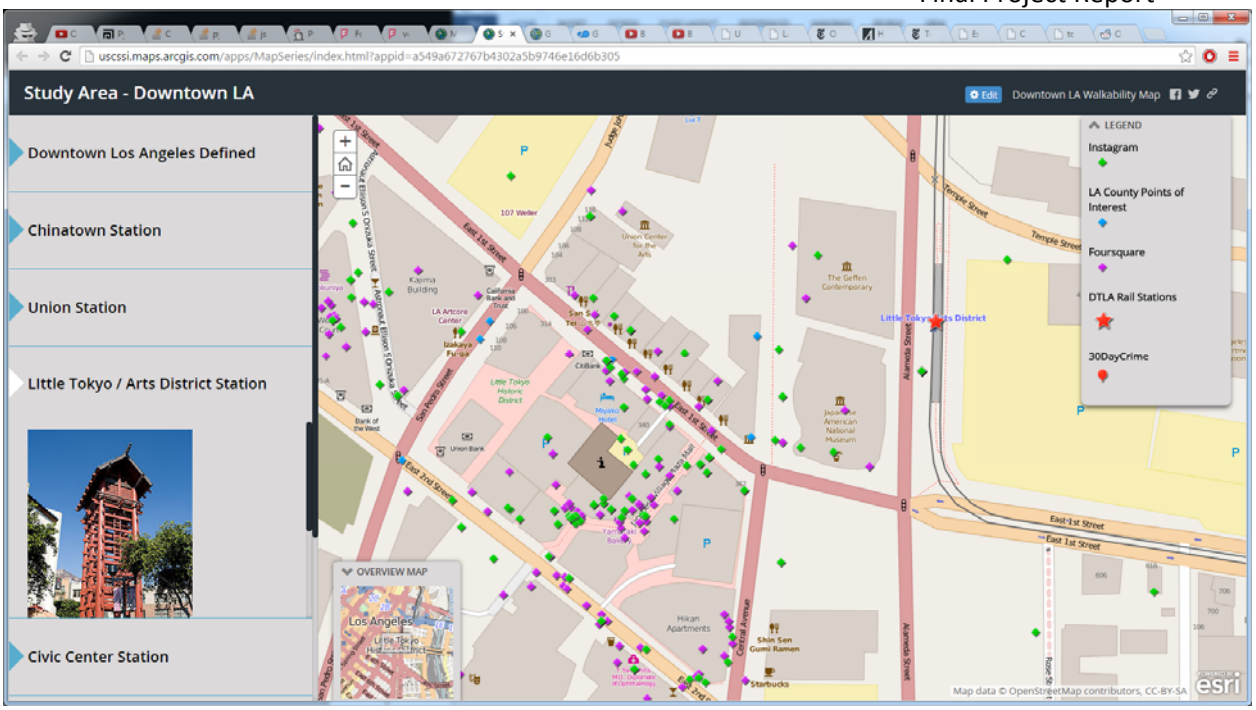

Figure 2 – Points of Interest View

Discussion of Geospatial Integration

Challenges

The first challenge that I encountered was determining how to define my study area, because it was not clear to me what exactly composed downtown Los Angeles. I decided to solve this problem by arbitrarily choosing the seven Metro Rail Stops that define downtown Los Angeles according to my local knowledge.

I ran into a second challenge once my study area was defined, slight variation in attribute data in the Metro Rail shapefiles. The same stop had different names in different shape files. I resolved this by eliminating duplicates from the manually selected stations.

My first attempt at a buffer around my rail stations failed because I was finding it difficult to merge the resulting buffers into a single entity. I looked at several tools and ultimately decided to merge the layers prior to creating the buffer.

I encountered errors when merging the three station layers that were caused by some of the attribute data apparently being too large. When I edited the attribute tables, and reduced the length of some of the names, the merge completed successfully. I did not devote any additional time in trying to solve or understand this.

I encountered several challenges with the social media data: working with paginated data, and handling missing data cleanly were major problems. I was able to employ a recursive method to work through the paginated data, calling the function with the next max_tag_id until the max_tag_id is no

longer defined (base case). I finally found a solution to missing data in the try/except structure within the Python language.

The recursive solution created another challenge when returned data sets were large. There are limits placed on the number of requests that can be made in a given time period, as well as total results for a given request. I solved the first problem by putting a short sleep timer prior to calling the function recursively. I manually interrupted the program in the case of larger data sets, which solved the second problem.

I found a number of tagged elements to be outside of my study area when using the Instagram API. In one case a photo tagged LittleTokyoDTLA had latitude and longitude coordinates that located it in Thailand. It was interesting finding a photo of artwork (the Rubik's Cube in front of JANAM) with coordinates in Thailand. My suspicion is that the image was uploaded to Instagram after returning to Thailand from a Los Angeles holiday.

Finally, I found that I was initially inconsistent in placing column headings in the CSV when I created them. This meant that some of the pop-ups in my web application did not have data. I solved this problem by making the column headings consistent and then reprocessing the data file. This did cost me time and effort, something I address in the next section.

Lessons Learned

The first, and biggest, lesson that I learned is that it is really difficult to wade through the large volume of data that is available. My initial plan was to use zip codes to select business data from the ReferenceUSA database available through the USC library. I found the process to be very cumbersome and difficult because of the way that the ReferenceUSA interface is designed. The system requires that the data be downloaded in small batches and limits the total data that can be retrieved.

Moreover, data from ReferenceUSA would have had to have been processed in a manner very similar to the social media data that I ended up using for the project. The availability of an API for both Instagram and Foursquare made the process of obtaining larger quantities of data much easier. Consequently, I believe that my preference going forward would be to automate data collection in order to avoid the manual search and download process that was required for ReferenceUSA.

In retrospect, it would have been nice to compare the Foursquare data, which I have already noted had accuracy problems, with the ReferenceUSA data, which claims to be authoritative. It would be very interesting to see how the various data sources compare to one another in terms of positional accuracy.

I would like to have compared the positional errors in the LA County LMS data and Foursquare data with an authoritative base map in addition to OpenStreetMap. It would also have been good to compare OpenStreetMap with an authoritative map directly. I was too ready to accept many of the data sources without sufficient critical analysis.

I should have included a layer that represented the rail lines on my final map. I included it initially, and debated about whether or not I should keep it. Ultimately, I decided not to keep the rail lines feature because I thought of the map as being more thematic. I think this was a mistake because the rail lines would add context and critical information that is required for trip planning.

Finally, I learned that column names and label need to be carefully prepared, especially when merging data. I created a lot of extra work for myself through my inattentiveness, only realizing the problem when I was well into the integration process. I was required to redo many of the earlier processes for several data sets, which cost resources.

Implications and Recommendations

I believe that there is a wealth of data available online with fairly permissive licenses and use restrictions. Moreover, the data is often available in GIS friendly formats, or via API, both of which greatly reduce the effort necessary when integrating data. That said, coordinate systems and projections do pose a very large potential source of error. Consequently, there are still some technical aspects that prevent this type of integration from being fully removed the realm of a specialist. Untrained users can go a long way in creating this type of application, but specialist knowledge is still required in some cases.

I have two major recommendations that came from this project: explore more online data sources and somehow catalog them, and authoritative sources should learn from sources like Instagram et. al. I was amazed by the volume of data that is available online, and continue to find sources even now, after the conclusion of the project. I believe that a major challenge is simply knowing where to find all of the data. Much of the data is not traditional GIS data, and is consequentially not labelled as spatial. Much of the data has a spatial component, but will not show up in a search for spatial or GIS data. The way that we search for data and the key words and phrases employed needs to be expanded.

Second, after reviewing several authoritative sources, ReferenceUSA being the source with which I spent the most time, I would suggest providing an API in addition to the web portal. Downloading large volumes of data via a web portal is cumbersome and untenable. I ultimately abandoned the data source because it did not suit my purpose. The implementation of a REST API that returns data in an easy to read format, such as JSON, would be a much better solution.

Appendix A – Foursquare API Script

```
#-------------------------------------------------------------------------------
# Name:       module3
# Purpose:
#
# Author:     CMH
#
# Created:    21/04/2015
# Copyright:  (c) CMH 2015
# Licence:    <your licence>
#-------------------------------------------------------------------------------

# Import all of the libraries
import httplib
import urllib
import json
import pprint

def insta( offset ):
   conn = httplib.HTTPSConnection("api.foursquare.com")

   headers = {"":""}
   parameters = "client_id=177e5e845cdf455380a79a33fba7edb4"
   #conn.request("GET", "/v1/tags/skidrokyo/media/recent", parameters, headers)

   #perhsing
   #lat = "34.0493159"
   #lng = "-118.251259"

   #7th metro
   #lat = "34.0486116"
   #lng = "-118.261131"

   #pico station
   #lat = "34.04366"
   #lng = "-118.264843"

   #civic center
```

```
#lat = "34.0534652"
#lng = "-118.250671"

#union station
#lat = "34.0550474"
#lng = "-118.2381289"

#chinatown
#lat = "34.0629137"
#lng = "-118.238279"

#little tokyo
#lat = "34.0507486"
#lng = "-118.237853"

conn.request("GET", "/v2/venues/explore?ll=" + lat + "," + lng +
"&client_id=NTQ5TQIWJSFKEZDPBJAZJ1Z2GZN3X1JN5IWNA4EHY54FNANA&client_secret=RUXCJO50EH
TST4WM0X4ZSO5HDGKIV42KNYCDIY1ZRYXIOPLL&v=20150421&radius=800&limit=50&offset=" + offset)

# Get the response and print the response information eg. 200 OK or 404 Not Found
response = conn.getresponse()
#print response.status, response.reason

# Get and print the actual data
data = response.read()

# Iteration 2 parse the json into a more useful data structure
parsed_json = json.loads(data)

# Load the pretty printer so that we can better see the structure of the data
pp = pprint.PrettyPrinter(indent=4)
#pp.pprint(parsed_json['response']['groups'][0]['items'])
#pp.pprint(parsed_json['response']['totalResults'])
#pp.pprint(parsed_json['data'])
#pp.pprint(data)

#'''
#json.dumps( parsed_json, sort_keys=True, indent=4, separators=(',', ': ') )
cont = 1
```

```
    items = parsed_json['response']['groups'][0]['items']
    for item in items:
        #theLine = ""
        try:
            print item['venue']['location']['lat'], ",",
            print item['venue']['location']['lng'], ",",
            print item['venue']['name'], ",",
            print item['tips'][0]['canonicalUrl'], ",",
            print ""
        except TypeError:
            pass
        except KeyError:
            pass
        #try:
            #print (item['location']['longitude'])
            #print (",")
        #except TypeError:
            #pass
    # Close the connection
    conn.close()
    try:
        #print int(offset) + 50
        if int(offset) + 50 < int(parsed_json['response']['totalResults']):
            insta( str( int(offset) + 50 ) );
    except TypeError:
            pass
    except KeyError:
        pass
    #'''
insta("0");
```

Appendix B – Instagram API Script

```
#-------------------------------------------------------------------------------
# Name:       module2
# Purpose:
#
# Author:      CMH
#
# Created:     17/04/2015
# Copyright:   (c) CMH 2015
# Licence:     <your licence>
#-------------------------------------------------------------------------------

# Import all of the libraries
import httplib
import urllib
import json
import pprint
import time

def insta( max_tag_id ):
    conn = httplib.HTTPSConnection("api.instagram.com")

    headers = {"":""}
    parameters = "client_id=177e5e845cdf455380a79a33fba7edb4"
    #conn.request("GET", "/v1/tags/skidrokyo/media/recent", parameters, headers)
    tag = "TrainLikeABeastLookLikeABeauty"
    conn.request("GET", "/v1/tags/" + tag +
"/media/recent?client_id=177e5e845cdf455380a79a33fba7edb4&max_tag_id=" + max_tag_id)
    #conn.request("GET",
"/v1/locations/search?lat=48.858844&lng=2.294351&distance=805&client_id=177e5e845cdf455380a79
a33fba7edb4&max_tag_id=" + max_tag_id)

    #f = open('skidrokyo.csv','w')

    # Get the response and print the response information eg. 200 OK or 404 Not Found
    response = conn.getresponse()
    #print response.status, response.reason

    # Get and print the actual data
```

```
    data = response.read()

    # Iteration 2 parse the json into a more useful data structure
    parsed_json = json.loads(data)

    # Load the pretty printer so that we can better see the structure of the data
    pp = pprint.PrettyPrinter(indent=4)
    #pp.pprint(parsed_json['pagination'])
    #pp.pprint(parsed_json['meta'])
    #pp.pprint(parsed_json['data'])
    #pp.pprint(data)
    #json.dumps( parsed_json, sort_keys=True, indent=4, separators=(',', ': ') )
    cont = 1
    items = parsed_json['data']
    for item in items:
        #theLine = ""
        try:
            print item['location']['latitude'], ",",
            #theLine = theLine + item['location']['latitude'] + ","
            print item['location']['longitude'], ",",
            #theLine = theLine + item['location']['longitude'] + ","
            print item['link'], ",",
            #theLine = theLine + item['link'] + ","
            print item['images']['standard_resolution']['url']
            #theLine = item['location']['latitude']
            #theLine = theLine + item['location']['latitude'] + "," #+ item['location']['longitude'] + "," +
item['link'] + "," + item['images']['standard_resolution']['url'] + "\n"
            #print theLine
            #print (",")
            #print "\n"
            #f.write(theLine)
        except TypeError:
            pass
        except KeyError:
            pass
        #try:
            #print (item['location']['longitude'])
            #print (",")
```

```
        #except TypeError:
            #pass
    # Close the connection
    #f.close()
    conn.close()
    try:
        #print parsed_json['pagination']['next_max_id']
        time.sleep(5)
        insta(parsed_json['pagination']['next_max_id']);
    except TypeError:
            pass
    except KeyError:
        pass

insta("0");
```