Location Prediction using Twitter

Introduction:

Locations: cities, states, country are central to various activities and events like news, emergency events and many more. The idea of finding the location automatically has been an interesting area of research for the recent years. With big online social networks like Facebook, Instagram, twitter people share millions and millions of data each day. This provides researchers more real time data to analyze, conclude or predict anything from it.

Here, in this project we would be using one of the big social network Twitter to analyze the tweets and predict the location of the twitter user. It is important to find the location of a twitter user to respond to an emergency at a place, to understand people's daily routine and many more. Being motivated by these problems there has been many research works going on to predict the location of a user based on the public content available. We use a knowledge-based system to create parameters for predicting the location of the twitter user.

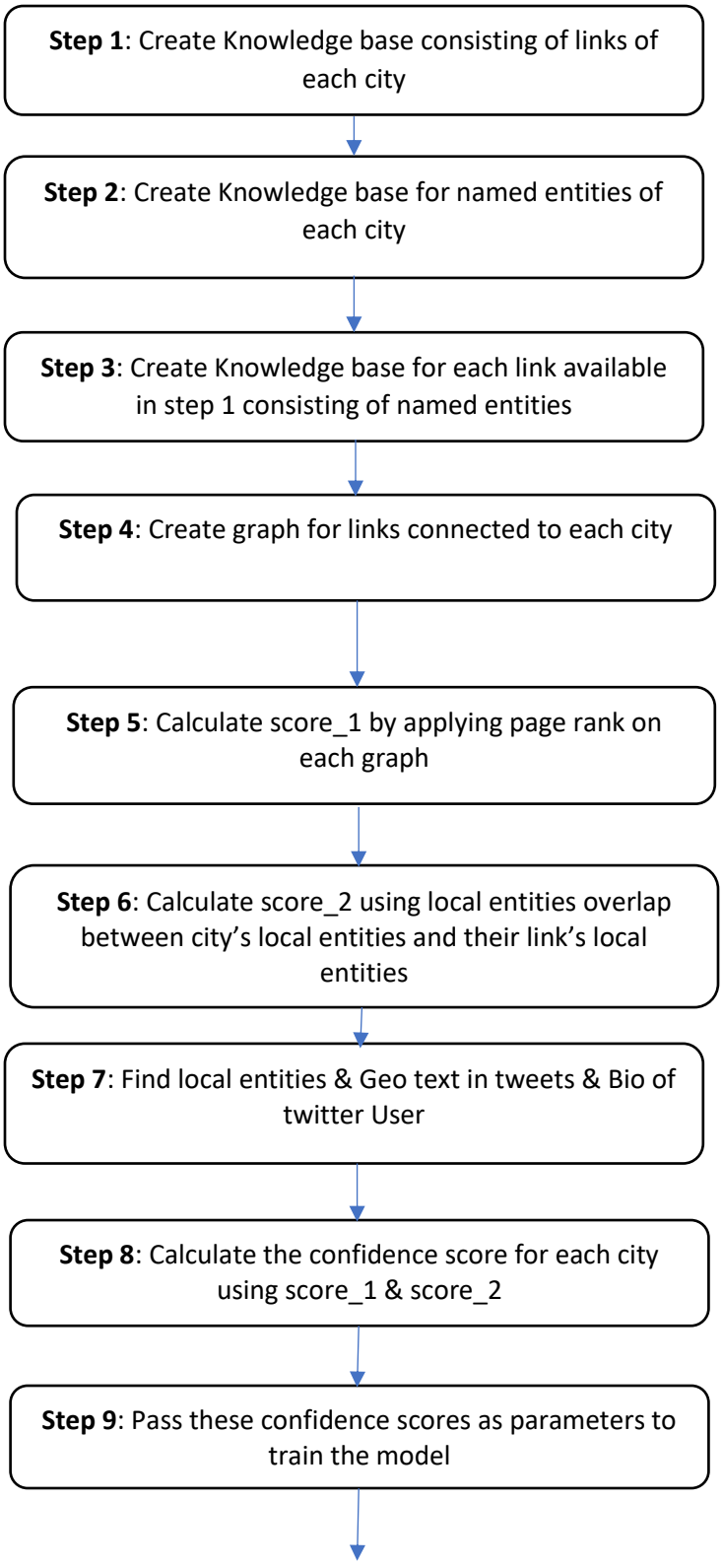
Prior Work:

People have explored various techniques to predict the location of a twitter user using various techniques. Some of them are

i. Analyzing the tweets of a user and comparing those tweets with the previously used words for that place.
ii. Analyzing the friends and followers of the twitter user to predict the location based on their tweets
iii. Predicting the location using (i) its reliance purely on tweet content, meaning no need for user IP information, private login information, or external knowledge bases; (ii) a classification component for automatically identifying words in tweets with a strong local geo-scope; and (iii) a lattice-based neighborhood smoothing model for refining a user's location estimate.

These are some of the techniques that were used previously in order to prdict the location of the twitter user.

Model:


Step 1: Create Knowledge base consisting of links of each city
Step 2: Create Knowledge base for named entities of each city
Step 3: Create Knowledge base for each link available in step 1 consisting of named entities
Step 4: Create graph for links connected to each city
Step 5: Calculate score_1 by applying page rank on each graph
Step 6: Calculate score_2 using local entities overlap between city's local entities and their link's local entities
Step 7: Find local entities & Geo text in tweets & Bio of twitter User
Step 8: Calculate the confidence score for each city using score_1 & score_2
Step 9: Pass these confidence scores as parameters to train the model

Step 10: Calculate the distance between the actual and predicted coordinates to measure accuracy

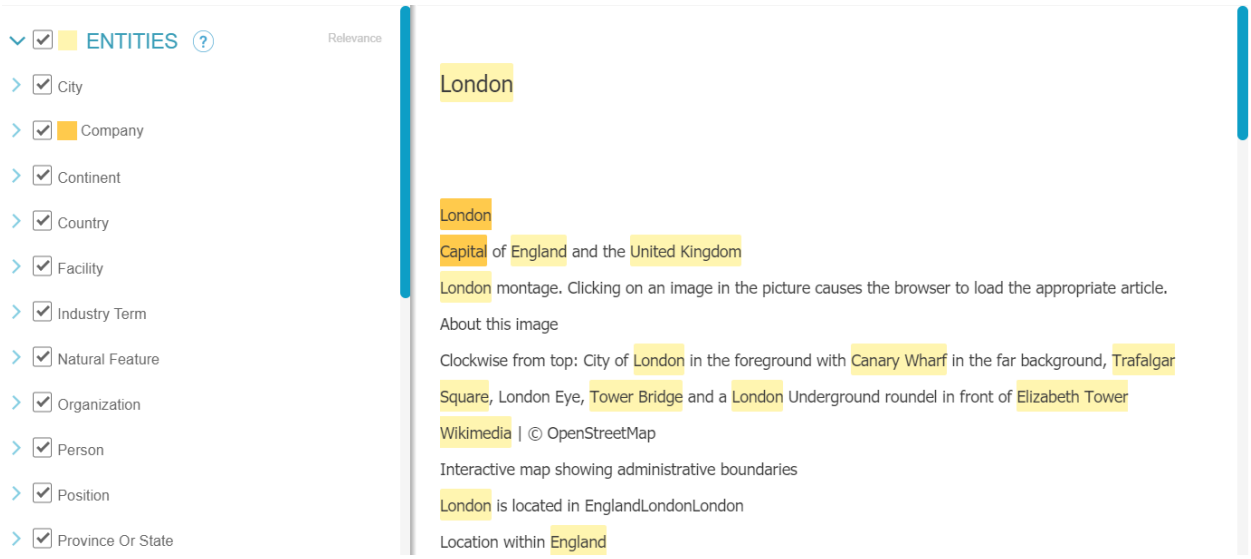
Creating Knowledge

The first step in building the model involves creating knowledge base for each city in the country. Knowledge base in this context is the combination of meaningful words that are most related to the city or in other words we can say that it's the collection of named entities that are related to the city. The process of creating a knowledge base involves the following processes:

1. Collecting the local entities of each city
2. Links going out from Wikipedia page of each city
3. Local entities present in each link of each city

Collecting the local entities of each city:

To collect the local entities of each city we make use of Wikipedia api to get the main page of each city and Spacy to extract the named entities. Here is an example of online api Openclasis that categorizes each word which are recognized as named entities.



The content of the Wikipedia page is parsed to extract the local entities of each city and is stored in the knowledge base. Image of named entities being added to each city given. Since we are interested in finding only the words that could help in predicting the location of the user, certain categories in named entities will be less importance in bolstering our cause. These categories such as 'QUANTITY', 'DATE', 'ORDINAL', 'CARDINAL', 'PERCENT', 'MONEY' are not considered to be part of knowledge base system. While updating the knowledge base with the named entities, there are possibilities that a word can be repeated more than once. We need to make sure that such kind of words is stored only once in the knowledge base system to reduce redundancy.

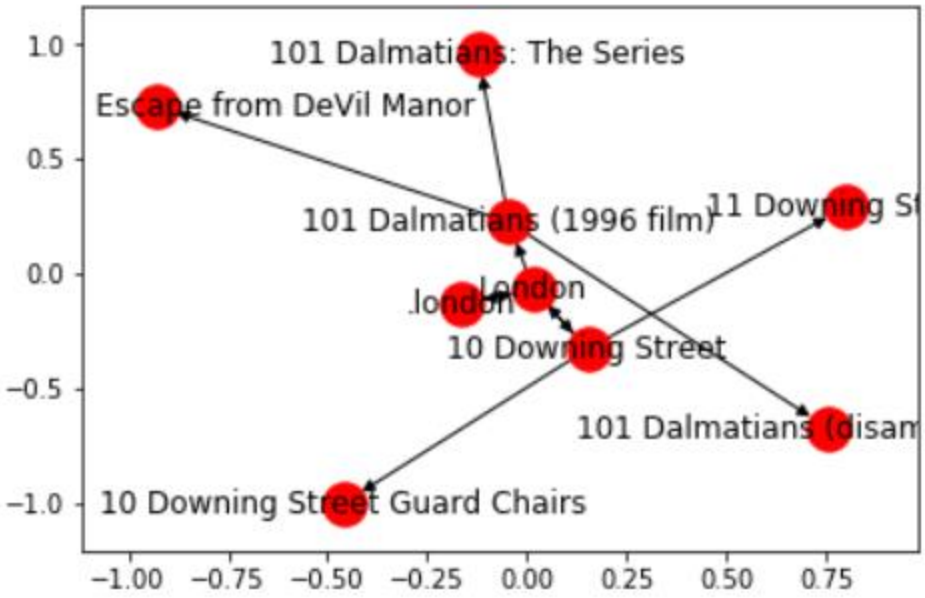
An example of knowledge base created for the city of London is shown below:

```
Out[4]: [('London',
          ['Docklands',
           'Napoleonic',
           'Baring',
           'London Southend Airport',
           'Alfred "refounded" London',
           'Canary Wharf',
           'Hanseatic',
           'James I in Westminster',
           "Hyde Park's",
           'the All England Club',
           'Jack',
           'Royal College of Music',
           'the Millennium Dome',
           'Dubai',
           'Highway',
           'Roman province',
           'Westminster',
           'The Small Faces',
```

Second, the Wikipedia page of each city is loaded, and the links that are directed from the Wikipedia page of the city are stored. Nodes are created for London and edges are created between the nodes and the city. On a second level of this same process, the links that are directed from each link's Wikipedia page are also taken into consideration and are checked. If the second page i.e) page of a link that is present in city's Wikipedia page has a link that is directed back to the city, then a backward edge is created between the link and the city which creates a cycle as shown below

Image of city name and corresponding links

```
Out[39]: [('London',
           ['.london',
            '101 Dalmatians (1996 film)',
            '10 Downing Street',
            '122 Leadenhall Street',
            '15 February 2003 anti-war protests',
            '1854 Broad Street cholera outbreak',
            '1896 Summer Olympics',
            '18th-century London',
            '1900 Summer Olympics',
            '1904 Summer Olympics',
            '1908 Summer Olympics',
            '1912 Summer Olympics',
            '1916 Summer Olympics',
            '1920 Summer Olympics',
            '1924 Summer Olympics',
            '1928 Summer Olympics',
            '1930 British Empire Games',
            '1932 Summer Olympics',
```



In this image there is a forward and backward link between London and 10 Downing street which reveals that a in Wikipedia page of 10 Downing street there is a link pointing back to London.

Calculating Scores:

i. Calculate scores using local entities:

This measure of calculating the scores using the local entities is implemented by comparing the local entities extracted for city's Wikipedia page and local entities in Wikipedia page of each link present in city's Wikipedia page. The scores are updated for each city by calculating the number of common words divided by the union of total number of local entities in city's Wikipedia page and number of local entities in each link's of that city. For example, refer the first line following image

```
('London', '101 Dalmatians (1996 film)', 5, 902, 172),
('London', '10 Downing Street', 20, 902, 359),
('Manchester', '100 King Street', 4, 624, 31),
('Manchester', '105.4 Century FM', 6, 624, 106),
('Manchester', '107 Piccadilly', 2, 624, 12),
('Bradford', '1911 FA Cup Final', 6, 700, 40),
('Bradford', '1946–47 Northern Rugby Football League season', 6, 700, 43),
('Bradford', '1948–49 Northern Rugby Football League season', 5, 700, 44)]
```

This image shows the number of common local entities between London and 101 Dalmatians(1996 film).

Total number of local entities (London) = 902

Total number of local entities (101 Dalmatians (1996 film) =172

Number of common local entities = 5

Overlapping score = Number of common local entities / (Total number of local entities (city) + Total number of local entities (city's link)

SO here, Overlapping score = 5/(902+172)

= 0.00465

The below image shows the calculated Overlapping score as 0.00465 between London and 101 Dalmations(1996 film)

```
[('London', '.london', 0.3657745336577453), ('London', '101 Dalmatians (1996 film)', 0.004655493482309125), ('London', '10 Downing Street', 0.01586042823156225), ('Manchester', '100 King Street', 0.0061068702290076335), ('Manchester', '105.4 Century FM', 0.00821917808219178), ('Manchester', '107 Piccadilly', 0.0031446540880503146), ('Bradford', '1911 FA Cup Final', 0.008108108108108109), ('Bradford', '1946–47 Northern Rugby Football League season', 0.008075370121130552), ('Bradford', '1948–49 Northern Rugby Football League season', 0.006720430107526882)]
```

ii. Page Rank

The other source of calculating source is through page rank. Page Rank reveals the importance of a node and here we have created a graph for the city and its links. Since a link is created back to the city, the page rank of that node will be more. The idea behind this is if the user tweet has a word that matches these nodes then the probability of that user being tweeted from that place will be more. So, we check the page rank of each node in the graph to calculate score.

The method **cal_score_overlap** will return the scores for each tweet across all cities by comparing the tweet and bio of the person with the knowledge base being created.

Results:

The aim is to find the location of the user from where he has posted the tweet. The tweet information and the bio information of the twitter user is considered to find the location. Local entities are extracted in the tweets and bio of a twitter user as shown.

```
{'Bradford': 0, 'London': 0, 'Manchester': 1.0}
723066126855344128
tweet deleted
non_duplicate_words_here ['Kellogg', 'Great Britain', 'Manchester', 'North West England']
finallist for ['Kellogg', 'Great Britain', 'Manchester', 'North West England', 'Manchester']
Kellogg
Great Britain
Manchester
North West England
Manchester
final {'Bradford': 2.029591600457315, 'London': 4.113184029204798, 'Manchester': 4.043160582880692}
{'Bradford': 2.029591600457315, 'London': 4.113184029204798, 'Manchester': 4.043160582880692}
723066093472899073
non_duplicate_words_here ['Award Winning International Dance Agency', 'Models', 'Exteme Talent']
finallist for ['Award Winning International Dance Agency', 'Models', 'Exteme Talent']
Award Winning International Dance Agency
Models
Exteme Talent
final {'Bradford': 0, 'London': 0, 'Manchester': 0}
{'Bradford': 0, 'London': 0, 'Manchester': 0}
```

At this stage we also extract the geotext indicated by the user in the tweets which will increase the confidence level of that user being in that place by a considerable margin. So, we extract the geotext from the tweet and bio of the twitter user.

As shown in the above image, the list of words containing the local entities and geo text is passed to the method **cal_score_overlap** which produces an output of dataframe that has scores calculated for each city for the lists of words being inputted.

	London	Manchester	Bradford
0	1.365775	0.000000	0.000000
1	0.000000	0.000000	1.000000
2	4.097324	3.044193	3.037700
4	0.000000	0.000000	0.000000
5	0.000000	1.000000	0.000000
6	4.113184	4.043161	2.029592
7	0.000000	0.000000	0.000000
8	2.772581	2.016438	2.000000
9	0.000000	0.000000	0.000000
11	0.000000	0.000000	0.000000
12	2.731549	2.034941	2.029592
13	2.731549	3.034941	2.029592
14	2.772581	2.016438	2.000000
15	2.731549	2.034941	2.029592

These values for each city are now the parameters for the algorithm Support Vector Regressor. Before passing in the input for the model, the dataset obtained is split into test and train and then the model is trained to predict the latitude and longitude. The distance between the original and predicted coordinates are calculated. The accuracy is measured by calculating the distance. If the distance between the coordinates are less than 100 miles, then it is assumed to be predicted correct and if the distance between the coordinates are more than 100 miles then it is assumed that the model has predicted the location wrongly. The accuracy obtained so far with this model is 8%.

Findings:

It is found from the dataset that for lists that have a greater number of local entities and geo text words the prediction is made correctly. So, one alternate solution to improve the accuracy would be increasing the local entities count to calculate the scores for each city. To increase the number of local entities, we need to analyze more tweets of the twitter user so that we can get more number of local entity words from the tweets.

```
Result: 391.77845521136317
Result: 754.2801025374813
Result: 655.7272923585195
Result: 693.8286120682221
Result: 655.7272923585195
Result: 627.0522143931057
Result: 627.0522143931057
Result: 802.0389080213397
Result: 634.7036309064075
Result: 301.81828852927197
Result: 9778.354140298889
Result: 655.7272923585195
Result: 28.706123176995437
Result: 9785.96600928011
Result: 696.7524129561677
Result: 659.7293693133045
Result: 663.7762495680026
Result: 9858.858834428056
Accuracy 8.695652173913043 %
```

Method **extractTweets** will pull 1000 tweets for each user and will create local entities for the twitter user. These words will be used to create the dataset for the model which will drastically improve the accuracy of the model. Here, due to time constraint, passing these lists to create dataset is not implemented.