Recommending tower location in a mobile network

BY: MOHAMED METWALLY

Introduction

- Background :
  - Mobile service demand is growing exponentially, the quality of service expected by the customers is getting higher, as the social media need is increasing
  - Planning and expanding the network is a tricky task to perform in order to satisfy that quality of service delivered to the customers
- Problem :
  - Selection of new proposed sites location and priotorizing them is a critical task. In order to get the maximum gain of the investment and enhance the network quality of service, to enrich the end user customer experience.
- Interest :
  - Radio network planners and budget proposal teams in the mobile network operators, will be interested in the selection criteria and priotorizing of the new sites location
- Area selected :
  - Zamalek island in Egypt, a top VIP customer area

Data acquiring

Data

Foursquare

Collected from mobile operator

Venues types in the area

Data volume traffic carried by network towers

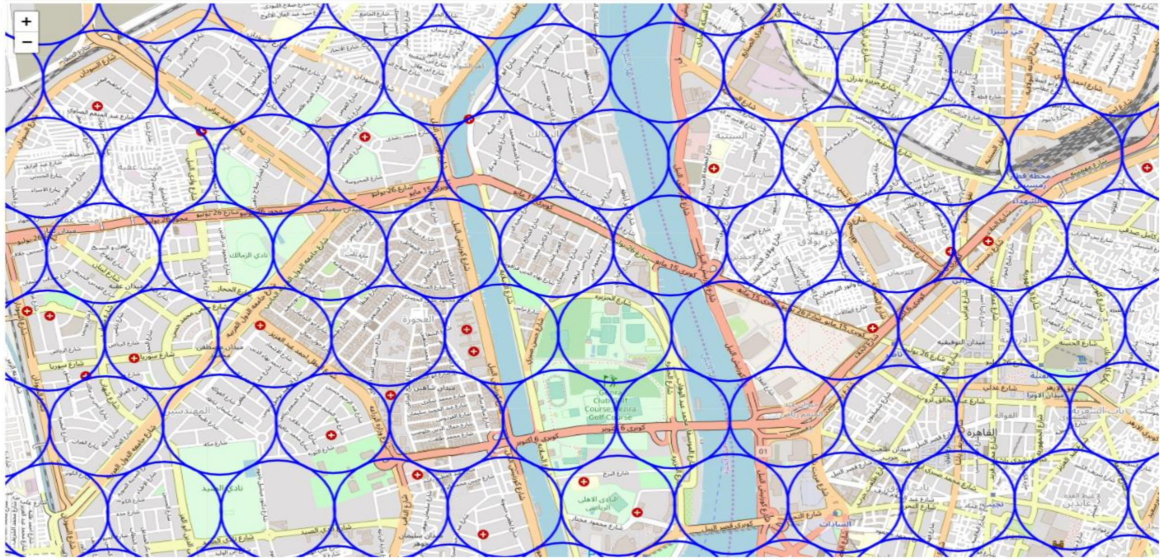
```
towerloc
```

	eNodeB	traffic	lat	lon
0	LCAIW20945	150	30.069542	31.218173
1	LCAIW20459	50	30.066012	31.221055
2	LCAIW20298	70	30.062500	31.217450
3	LCAIW20461	100	30.061819	31.223215
4	LCAIW20293	30	30.056056	31.224817
5	LCAIW20300	80	30.066537	31.230656

Analysis and data exploration

- A grid is created in order to split Zamalek into zones

```python
zamalek_center = (31.220228, 30.063513)
map_zamalek = folium.Map(location=zamalek_center, zoom_start=13)
folium.Marker(zamalek_center, popup='Alexanderplatz').add_to(map_zamalek)
for lat, lon in zip(latitudes, longitudes):
    #folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_color='blue', fill_opacity=1).add_to(map_berlin)
    folium.Circle([lat, lon], radius=300, color='blue', fill=False).add_to(map_zamalek)
    #folium.Marker([lat, lon]).add_to(map_berlin)
map_zamalek
```



Analysis and data exploration

- Our data from foursquare contains 100 venue, in Zamalek

```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axis=1)

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]

nearby_venues.head()
```

	name	categories	lat	lng
0	Zamalek Cinema	Multiplex	30.061760	31.218794
1	Zööba (زووبا)	Middle Eastern Restaurant	30.061248	31.219263
2	Maison 69	Boutique	30.063842	31.218536
3	Mandarine Koueider	Pastry Shop	30.062634	31.219732
4	Villa Baboushka	Boutique	30.062980	31.221455

```
print('{} venues were returned by Foursquare.'.format(nearby_venues.shape[0]))
```

```
100 venues were returned by Foursquare.
```

Analysis and data exploration

- The venues in Zamalek are categorized, a matrix is generated in order to check the venue type verses all the categories

```
# one hot encoding
onehot = pd.get_dummies(nearby_venues[['categories']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
onehot['Neighborhood'] = 'Zamalek'

# move neighborhood column to the first column
fixed_columns = [onehot.columns[-1]] + list(onehot.columns[:-1])
onehot = onehot[fixed_columns]

onehot.head()
```

	Neighborhood	American Restaurant	Art Gallery	Bakery	Bar	Bistro	Boat or Ferry	Bookstore	Boutique	Bubble Tea Shop	Burger Joint	Café	Coffee Shop	Cupcake Shop	Dessert Shop	Eastern European Restaurant	Food Stand	Gym	Gym / Fitness Center	Health & Beauty Service	Hotel	Hotel Bar	Ice Cream Shop	Indian Restaurant	Italian Restaurant	Japanese Restaurant	Jazz Club	Jewelry Store	Juice Bar
0	Zamalek	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	Zamalek	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Zamalek	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Zamalek	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Zamalek	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Analysis and data exploration

- The top 10 common venues in Zamalek are displayed to know the categories, multiplex, yoga studios and Japanese restaurant are the top venues in Zamalek area

[24]:
```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = onehot['Neighborhood']

for ind in np.arange(onehot.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(onehot.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

[24]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Zamalek	Multiplex	Yoga Studio	Japanese Restaurant	Indian Restaurant	Ice Cream Shop	Hotel Bar	Hotel	Health & Beauty Service	Gym / Fitness Center	Gym

Analysis and data exploration

- K-means clustering is used to cluster the venue data, 5 clusters are generated, showing the venue categories in each cluster

```
[25]: # set number of clusters
kclusters = 5

manhattan_grouped_clustering = onehot.drop('Neighborhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(manhattan_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]

[25]: array([0, 0, 0, 0, 0, 4, 0, 0, 0, 0], dtype=int32)

[26]: # add clustering labels
#neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

manhattan_merged = nearby_venues

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighborhood
manhattan_merged = onehot.join(neighborhoods_venues_sorted.set_index('Neighborhood'), on='Neighborhood')

manhattan_merged.head() # check the last columns!
```

[26]:

	Neighborhood	American Restaurant	Art Gallery	Bakery	Bar	Bistro	Boat or Ferry	Bookstore	Boutique	Bubble Tea Shop	Burger Joint	Café	Coffee Shop	Cupcake Shop	Dessert Shop	Eastern European Restaurant	Food Stand	Gym	Gym / Fitness Center	Health & Beauty Service	Hotel	Hotel Bar	Ice Cream Shop	Indian Restaurant	Italian Restaurant	Japanese Restaurant	Jazz Club	Jewelry Store	Juice Bar
0	Zamalek	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	Zamalek	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	Zamalek	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	Zamalek	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	Zamalek	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Analysis and data exploration

- Linking the venues from foursquare and the site locations by distance in order to get the nearest serving site to the venue

```
[27]: import pandas as pd
      from scipy.spatial.distance import cdist

[28]: def closest_point(point, points):
          """ Find closest point from a list of points. """
          return points[cdist([point], points).argmin()]

[29]:
      def match_value(df, col1, x, col2):
          """ Match value x from col1 row to value in col2. """
          return df[df[col1] == x][col2].values[0]

[30]:
      df1 = pd.DataFrame(towerloc)
      df2 = pd.DataFrame(nearby_venues)

[31]: df2

      •••

[32]:
      df1['point'] = [(x, y) for x,y in zip(df1['lat'], df1['lon'])]
      df2['point'] = [(x, y) for x,y in zip(df2['lat'], df2['lng'])]

[33]: df2['closest'] = [closest_point(x, list(df1['point'])) for x in df2['point']]

[34]: df2

      •••

[35]: df2['zone'] = [match_value(df1, 'point', x, 'eNodeB') for x in df2['closest']]

[36]: df2
```

[36]:

	name	categories	lat	lng	point	closest	zone
0	Zamalek Cinema	Multiplex	30.061760	31.218794	(30.061760222603002, 31.218793667827992)	(30.0625, 31.217449900000002)	LCAIW20298
1	Zööba (زووبا)	Middle Eastern Restaurant	30.061248	31.219263	(30.06124837014216, 31.219262645315787)	(30.0625, 31.217449900000002)	LCAIW20298
2	Maison 69	Boutique	30.063842	31.218536	(30.063842, 31.218536)	(30.0625, 31.217449900000002)	LCAIW20298
3	Mandarine Koueider	Pastry Shop	30.062634	31.219732	(30.06263394562906, 31.219732275388324)	(30.0625, 31.217449900000002)	LCAIW20298

Analysis and data exploration

- A pivot function is performed in order to get the count of venues serving each tower or site in the network

```
[37]: zamalekall=df2

[38]: pivot = zamalekall.pivot_

[39]: print(pivot)

                 categories  c
      zone
      LCAIW20293          19
      LCAIW20298          26
      LCAIW20300           8
      LCAIW20459          14
      LCAIW20461          23
      LCAIW20945          10
```

Results and conclusion

- Results :
  - The results shown, there are a site “LCAIW20298” that is serving a huge number of venues “26 venues”, those venues are representing 25% of the venues in Zamalek area, correlated that site is carrying average traffic, not the highest among the sites in Zamalek, which means those venues can have bad quality of service and need to have a new site.
  - The priority of those 26 venues could be applied from the 10^{th} category ranking performed earlier in Zamalek, in order to give them higher priority to have a new site.
- Discussion :
  - The analysis performed should be reviewed by the planner in the mobile operator in order to validate the analysis results, does it makes sense with respect to the company strategy
- Conclusion :
  - A recommender analysis is performed for a mobile operator in order to give insights about the currents sites analysis, traffic versus the served venues. And recommend which areas need to have a new site and priotorize them with respect to the VIP venues