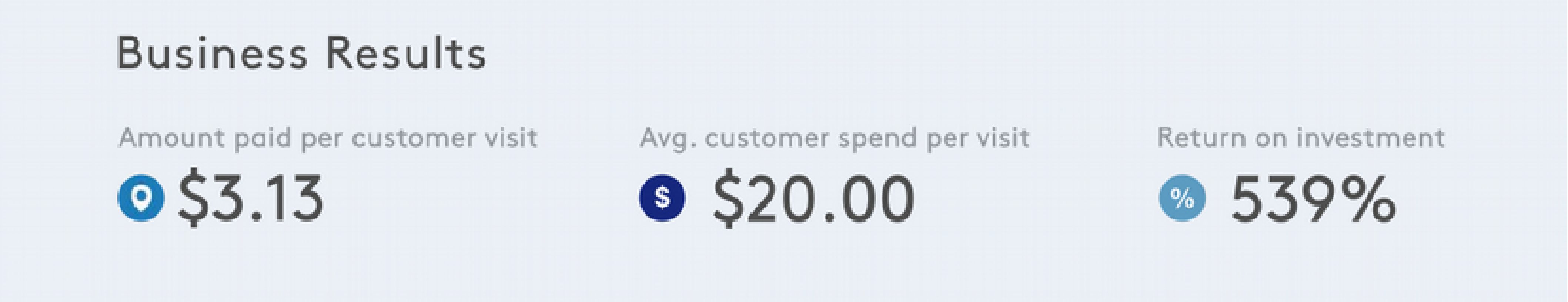
FourSquare

Location-based **So**cial Network

- Users “check-in” places
- Physical Coordinates based interaction
- Business model
  - Free for users – content providers
  - Premium for businesses
    - Analytics
    - Advertisements
    - Gamefication
- 50 million users
- 2 million businesses

Lawrence Chan & Hudson Tang, Manchester Pub



Business Results

Amount paid per customer visit

$3.13

Avg. customer spend per visit

$20.00

Return on investment

539%

FourSquare API

- User-related functionality
  - Search, Requests, Check-ins, Tips, …
- Venue-related functionality
  - Search, Explore, Categories
  - Statistics
  - Events

https://developer.foursquare.com/

oauth2 based access

- Create Developer Account
- Register your Application
- Acquire Access Token
  - Client ID
  - Client Secret

FourSquare + Python

Foursquare library

pip3 install foursquare

```
import foursquare


CLIENT_ID = "..."

CLIENT_SECRET = "..."

client =
foursquare.Foursquare(client_id=CLIENT_ID,
                      client_secret=CLIENT_SECRET)
```

Venue Look-up

```
# Check out DIAG
res = client.venues("4c31a1e26f1fef3b440dec3d")


res["venue"]["name"]
res["venue"]["hours"]
res["venue"]["popular"]
res["venue"]["location"]["lat"]
res["venue"]["location"]["lng"]
```

Explore Venues

```
search =
client.venues.explore(params={'ll':'41.89,12.50'})
for group in search['groups']:
    print(group["name"])
    for nv in group['items']:
        print(nv["venue"]["name"])

search = client.venues.explore(params={'near':'Rome,
Italy'})

search = client.venues.explore(params={'near':'Rome,
Italy','query':'transtevere'})
```

Explore vs Search

- Both allow to find venues
  - different methodology
- Explore is better for queries:
  - *what are some popular coffeeshops in this area?*
- Search is better for queries:
  - *where is the nearest Joe’s Coffee?*
  - *where am I right now?*

Next Venue

Returns venues that people often check in to after the current venue.

- Up to 5 venues are returned in each query
- Results are sorted by how many people have visited that venue after the current one

```
next_venues =
client.venues.nextvenues("4c31a1e26f1fef3b440dec3d")

for nv in next_venues['nextVenues']['items']:

    print(nv["name"])
```

To-do

- Explore Venues in Rome
- Form a Directed Graph:
  - Create 1 Node for each Venue
  - Assign 1 Edge from Node A to Node B if
    - Venue B appears on Next-Venue of Node A

How do you explore?

```
unexplored = set()
search =
client.venues.explore(params={'near':'Rome,
Italy'})
for item in search['venues']:
    unexplored.add(item["id"])

for node in unexplored:
    # lookup next venues
    unexplored.remove(node)
```

RuntimeError: Set changed size during iteration

Use a List Carefully!

```
while unexplored:
    key = unexplored.pop()
    # …
```

Beware of Rate Limits

Usage of the API is subject to rate limits.

- 5,000 userless requests per hour to venues/* endpoints
- 500 userless requests to most other endpoints groups per hour.

Split Graph setup + Search

Keeping Big Data in Memory

- As you are exploring the FourSquare DB data are stored on your python notebook
- What if python crashes?
- What if you want to switch off laptop?

Periodically Save Data

- NetworkX offers methods to save Data
- Use MongoDB
- Other ...

Search FourSquare

```
import networkx as nx
places = nx.DiGraph()

unexplored = set()
search = client.venues.search(params={'near':'Rome, Italy'})
for item in search['venues']:
    unexplored.add(item["id"])
    places.add_node(int(item["id"], 16),
                    id=item["id"],
                    name=item["name"],
                    lat=item["location"]["lat"],
                    long=item["location"]["lng"],
                    tipCount=item["stats"]["tipCount"],
                    checkinsCount=item["stats"]["checkinsCount"],
                    usersCount=item["stats"]["usersCount"])

    print("unexplored: ", len(unexplored), " - total venues: ",
len(places.nodes()), " - total links: ", len(places.edges()),
         "visiting: ", item["name"])
```

Explore FourSquare

```
import networkx as nx
places = nx.DiGraph()

unexplored = set()
explore = client.venues.explore(params={'near':'Rome, Italy'})
for venue in explore["groups"][0]["items"]:
    item = venue["venue"]
    unexplored.add(item["id"])
    places.add_node(int(item["id"], 16),
                    id=item["id"],
                    name=item["name"],
                    lat=item["location"]["lat"],
                    long=item["location"]["lng"],
                    tipCount=item["stats"]["tipCount"],
                    checkinsCount=item["stats"]["checkinsCount"],
                    usersCount=item["stats"]["usersCount"])

    print("unexplored: ", len(unexplored), " - total venues: ",
len(places.nodes()), " - total links: ", len(places.edges()),
         "visiting: ", item["name"])
```

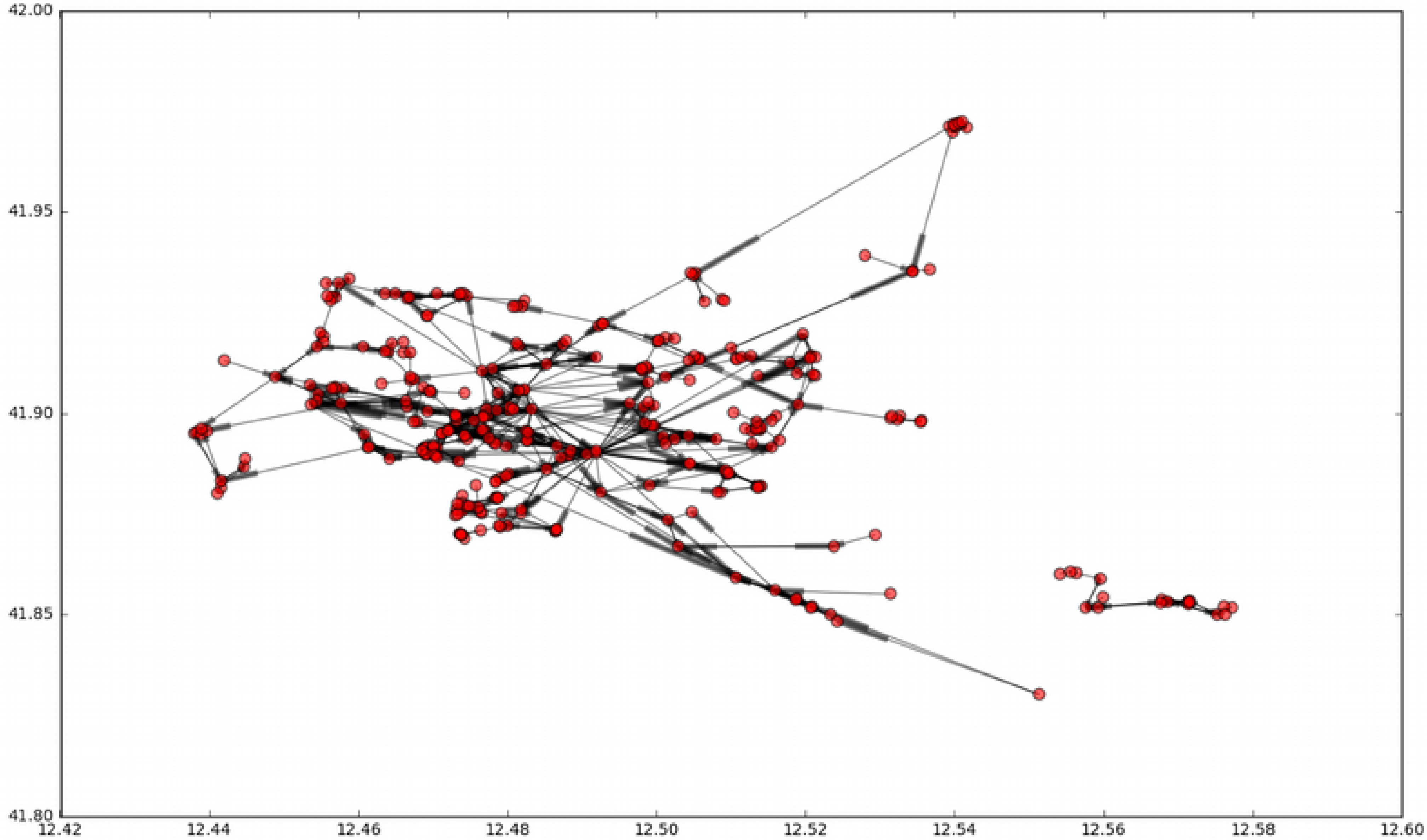
Lookup Next-Venues

```
while unexplored:
    key = unexplored.pop()
    next_venues = client.venues.nextvenues(key)
    keyInt = int(key, 16)
    node = places.node[keyInt]
    print("unexplored: ", len(unexplored), " - total venues: ", len(places.nodes()), " - total links: ",
len(places.edges()),
         "visiting: ", node["name"],
          "[new found: ", len(next_venues['nextVenues']['items']),
          "]")
    for item in next_venues['nextVenues']['items']:
        unexplored.add(item["id"])

        places.add_node(int(item["id"], 16),
                id=item["id"],
                name=item["name"],
                lat=item["location"]["lat"],
                long=item["location"]["lng"],
                tipCount=item["stats"]["tipCount"],
                checkinsCount=item["stats"]["checkinsCount"],
                usersCount=item["stats"]["usersCount"])

        places.add_edge(int(item["id"], 16), int(key, 16))

        print("\tunexplored: ", len(unexplored), " - total venues: ", len(places.nodes()), " - total links:
", len(places.edges()),
             "visiting: ", item["name"])
```


42.00
41.95
41.90
41.85
41.80
12.42
12.44
12.46
12.48
12.50
12.52
12.54
12.56
12.58
12.60

To-do

- Download 300 venues
  - Clean-up graph
- Visualize Venues + Relations
- Analyze Data
  - Identify node with highest centrality
  - Identify node with highest closeness centrality
  - Identify node with highest betweenness centrality
  - Identify node with highest pagerank