COMS 4995-02 Social Networks Spring 2016

Data Challenge 1: Finding your way in a Geo-Social Network

We have learned in class that structures play a very special role in social networks; they do not systematically arranged all connections that are formed, but they significantly influence which ones are more likely to hold statistically. The goal of this first contest is to give you a chance to leverage properties of a real social network to build an algorithm solving a graph exploration problem with minimum cost.

Let us take an example: If you live in a place for a long time, you are likely to have lots of friends who also live in same place. This fact is helpful when we want to predict how a social graph develops in the vicinity of a given individual. Research [1] [2] has shown that geographic distance does affect social structure. But does this inform the design of new algorithms? Or, to put it differently, are you able to find a model for this geo-social correlation and use it to find ways to explore social graph effectively?
Here your goal will be to find a feasible path to a destination using a minimum of local queries, a problem related to Milgram routing.

Motivating scenario

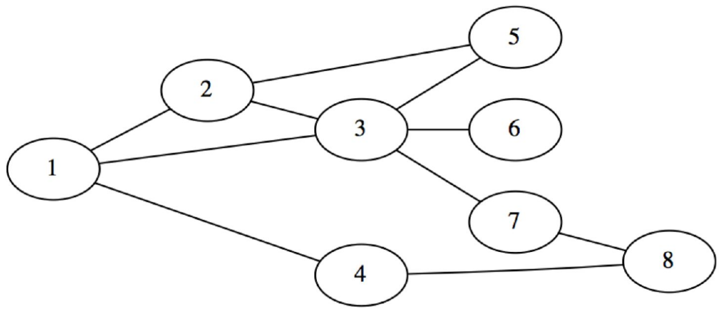
You, an active social network user, want to get connected to a target person through the network. Imagine this person is the CTO of a new start-up you heard about, and you have a particular idea to pitch to her. Most likely, a direct email from you (an unknown unsolicited stranger) would be ignored, even if you insist, so your goal is to find a chain of acquaintance through people who know each other, to convince each member of this chain and in the end reach that person.

At an abstract level, finding this chain appears relatively simple:

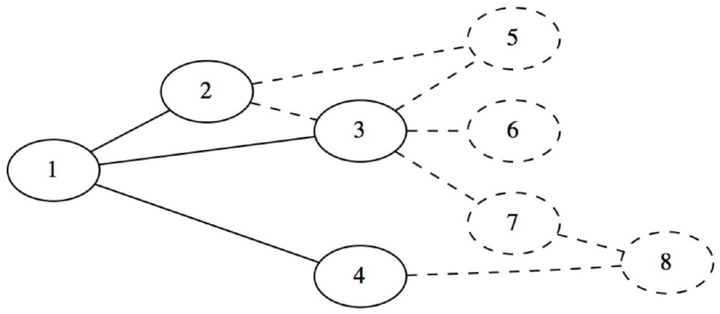
1. Begin with your friends. List them as known persons.
2. Pick the one who you think is the closest to the target from all known person, spend time convincing this person, asking him or her to disclose to you her or his list of friends. Then add this person's friends to your known persons list.
3. Repeat 2 until the target person is in the known persons list.

So, basically what you will do is that you choose one of your friends that you think can get you closer to the target, and then check his/her friend list. If you find someone in the list that you think is closer to the target, you go to check that person's friend list. You do this repeatedly until you find a path to the target person. Remember each query takes some resource (time to contact the person, convince him or her, perhaps even by providing some reward in case the project is successful), so you would like to minimize those.

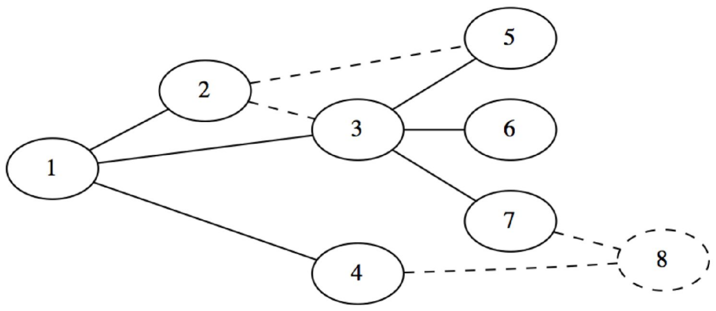
Let us illustrate this on an example. The following social graph contains you labeled as 1 and the target person in this case with label 8.



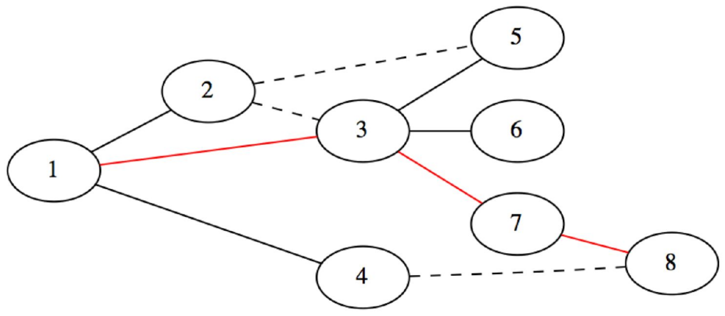
At the beginning, 2, 3 and 4 are your friends. You don't know the rest of the network.



You think 3 or 3's friend is more likely to know 8. So you go check 3's friend list and found 5,6 and 7



Now you think 7 or 7's friend is more likely to know 8, so you go check 7's friend list, and, bingo, you find 8. The path is 1->3->7->8



As a result of this exploration, you can get connected to 8 with the help from 3 and 7. Only two intermediaries were necessary.

Deciding whom to explore is the key aspect of this process. For the previous example, if you choose 4 instead of 3 or 7, you will get a shorter path 1->4->8.

This example seems puzzling as no information in advance differentiate 3 from 4 as one being *closer* to the target. This is where the most important aspect of this data challenge comes into play: you do have **additional information**. In this case, we assume that all users are members of a popular social network where you can publicly "check-ins" to venues (yes, what a strange idea to do that, but let's assume it for the sake of this challenge, and that everyone accepted this service's EULA). As an example, using the data you immediately found that 4 and 8 both checked-in at a coffee shop almost at the same time, while 3 and 8 seems to be far away from each other. Knowing this, you will think that 4 is closer to 8 than 3, an important hint to find a single intermediary to connect to the CTO.

Task Description and format

In this challenge[training phase], you will be given:

- Two numbers, standing for source ID and target ID.
- A file that contains lines of [time latitude longitude nodeID] tuples, representing check-in data.
- A RESTful API that responses to friend list query. You will request a user's friend list through API.
- *A file contains edges between nodes (undirected graph). You can use this training data to develop your algorithms. Notice that you won't be given it (the full graph) during the actual data challenge.

There will be two main tasks (see organization of the contest below):

1. Find a path from the source user to the target user in the social graph with the minimum *number of queries* made to the server.
2. Find a path from the source user to the target user in the social graph with the shortest geographic movement. Here, we take as a convention that user's position is defined as his/her **last** check-in location. If a user has no check-in, he's forbidden for this task.

There are two data sets: a training data set and a testing data set. Both of them contain a social graph and corresponding check-in data, and the two sets have same distribution. The latter one will be used in a 24-hour competition, in which you will run your code to solve specific test cases.

The API is now running with the social graph of the training set, and you're given corresponding check-in data. To help you tune your model, the social graph of the training set is also given to you. However in the competition you will have only the check-in data of the testing set. The social graph will **NOT** be given and requesting from the API will be the only way to learn the structure of it.

Serverless RESTful API

***Important: this RESTful API is built with Amazon AWS, which has an account throttling; please be benign, do NOT write an infinite loop calling this API. TAs and your fellow students will appreciate it.**

***When you see "{"message":"Missing Authentication Token"}", do not freak out, that happens merely because you enter an invalid URL or query parameters, except for case [1] (see the end of this session).**

The server accepts HTTPS request and returns JSON strings. The available operations are,

- [GET]Start a test case
  Address:
  https://6io70nu9pi.execute-api.us-east-1.amazonaws.com/smallworld/start/testcaseID
  Example Query:
  https://6io70nu9pi.execute-api.us-east-1.amazonaws.com/smallworld/start/1
  Example Return:
  ```
  {
    "reach testcase number" : "1",
    "source node" : 4853,
    "target node" : 39456,
    "wish you" : "best luck"
  }
  ```
  The example test case is finding a route from 4853 to 39456.
  Right now, you are only supplied with **1** test case. Since the whole training dataset is provided, you can construct your own test cases.

- [GET]Query a user's friend list
  Address:
  https://6io70nu9pi.execute-api.us-east-1.amazonaws.com/smallworld/neighbors?node=nodeID&uni=yourUNI&testcaseID=testcaseID]
  Example Query:
  https://6io70nu9pi.execute-api.us-east-1.amazonaws.com/smallworld/neighbors?testcaseID=1&node=4853&uni=yh2901
  Example Return:
  ```
  {
    "neighbors" : [{"414":[ 28, 0.36]}, {"1643":[ 108, 0.05]}, {"2937":[ 100,
  0.07]}, {"3353":[ 2, 1.00]}, {"3775":[ 84, 0.07]}, {"4298":[ 35, 0.23]},
  {"4348":[ 82, 0.11]}, {"4451":[ 10, 0.24]}, {"4511":[ 23, 0.17]}, {"4771":[ 20,
  0.43]}, {"4862":[ 26, 0.12]},...,{"54611":[ 18, 0.18]}, {"55970":[ 2, 1.00]},
  {"56753":[ 13, 0.19]}, {"57364":[ 109, 0.07]}]

  }
  ```

  Where neighbors contains <nodeID: degree, local_clustering_coefficient > pairs. Degrees of friends and local clustering coefficient are given to you in

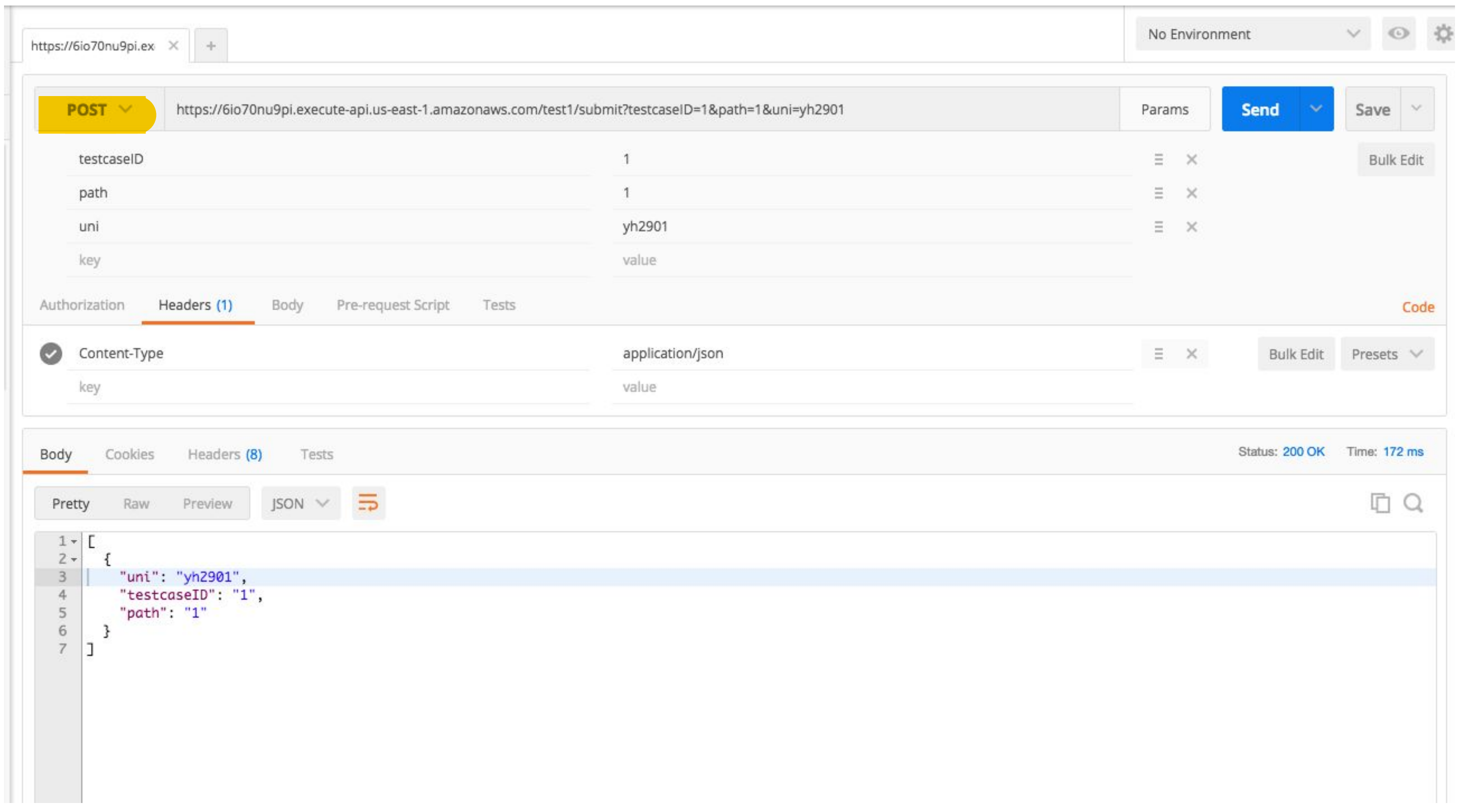
the hope to help you make better decision.
We will count how many queries you made. The number of queries for each test is limited to 2000 queries. Once you reached this limit, the API will return {"error":"Step exceeds limit"}.
If you see {"error message" : "your uni is not authorized for this operation!"}, then you either made a typo on your uni, or your uni is not known to your TAs, contact them immediately!

- [POST] Submit a path (read **[1]** at the end of this session before you proceed!)
  Address:
  https://6io70nu9pi.execute-api.us-east-1.amazonaws.com/smallworld/submit?testcaseID=testcaseID&path=path&uni=yourUNI
  Example Query:
  https://6io70nu9pi.execute-api.us-east-1.amazonaws.com/smallworld/submit?testcaseID=1&path=1,2,3,4&uni=yh2901 **[1]**
  Example Return: {
  "hey": "you just submitted the following",
  "uni": "yh2901",
  "testcaseID": "1",
  "path": "1,2,3,4"
  }
  you are only allowed to submit **1 path to 1 test case once**; an error message will be returned if do it more than once.

- [GET]Check your submission
  Address:
  https://6io70nu9pi.execute-api.us-east-1.amazonaws.com/smallworld/check/yourUNI/testcaseID
  Example Query:
  https://6io70nu9pi.execute-api.us-east-1.amazonaws.com/smallworld/check/yh2901/1
  Example Return: {
  "you are" : "yh2901",
  "your are at testcase" : 1,
  "your entered path " : 1,2,3,4,
  "total number of query on this test case" : 1
  }

[1] for unclear reasons, if you do POST (submit a path) using the given URL through a browser or curl will return "{"message":"Missing Authentication Token"}" . The alternatives are

1. **Postman Chrome Extension**



```
POST https://6io70nu9pi.execute-api.us-east-1.amazonaws.com/test1/submit?testcaseID=1&path=1&uni=yh2901
```

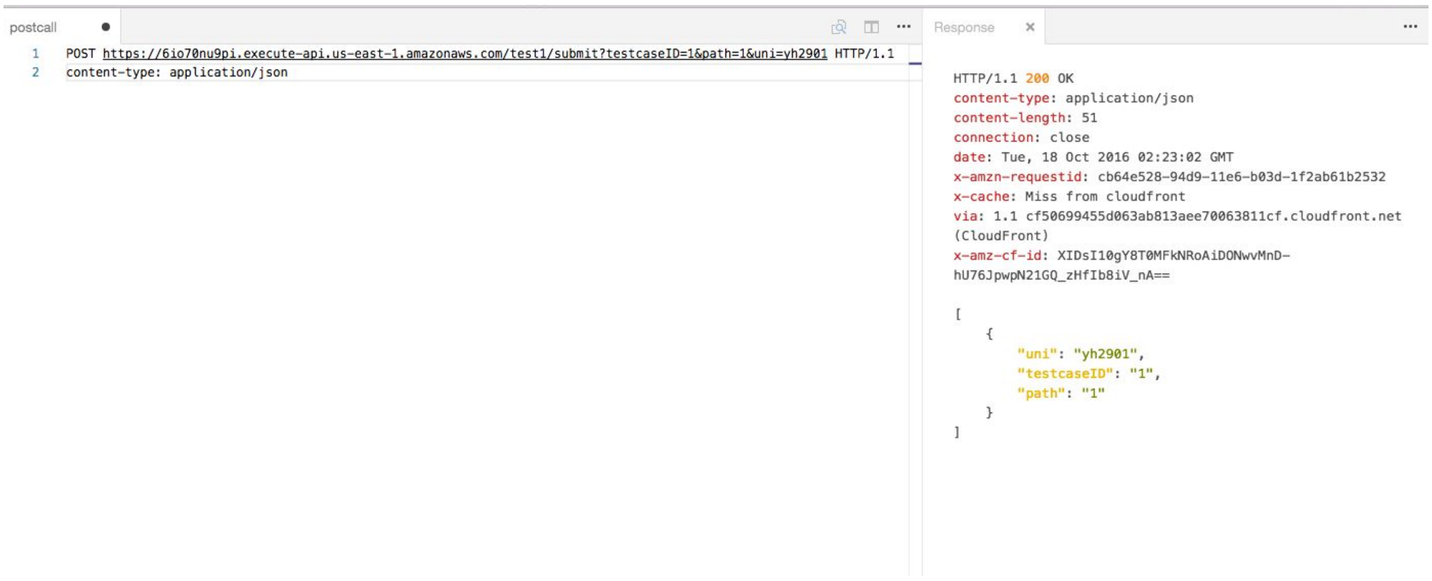
key	value
testcaseID	1
path	1
uni	yh2901

key	value
Content-Type	application/json

Status: 200 OK Time: 172 ms

```
[
  {
    "uni": "yh2901",
    "testcaseID": "1",
    "path": "1"
  }
]
```

2. RestClient with vscode



```
POST https://6io70nu9pi.execute-api.us-east-1.amazonaws.com/test1/submit?testcaseID=1&path=1&uni=yh2901 HTTP/1.1
content-type: application/json
```

```
HTTP/1.1 200 OK
content-type: application/json
content-length: 51
connection: close
date: Tue, 18 Oct 2016 02:23:02 GMT
x-amzn-requestid: cb64e528-94d9-11e6-b03d-1f2ab61b2532
x-cache: Miss from cloudfront
via: 1.1 cf50699455d063ab813aee70063811cf.cloudfront.net
(CloudFront)
x-amz-cf-id: XIDsI10gY8T0MFkNRoAiDONwvMnD-
hU76JpwpN21GQ_zHfIb8iV_nA==

[
    {
        "uni": "yh2901",
        "testcaseID": "1",
        "path": "1"
    }
]
```

If you don’t know how to request and parse json objects through RESTful API, check out following libraries. Note, you don't have to learn them in depth; simply reading some examples and enabling yourself to call a RESTful API and consume the response.

- C/C++: libcurl or Asio HTTP Client + rapidjson or PropertyTree JSON
- Java: Apache HttpClient + json-simple or org.json
- Python: requests
- Matlab: JSON Parser (For HTTP requests use Matlab built-in function urlread)
- R: RCurl + RJSONIO

Organization of the contest and grading

The grading of this challenge is organized in three phases:

- Phase I: Provide a correct answer to the challenge (i.e. write something that works at least for the simplest test case, and be able to commit some correct results in the competition)

- Phase II: Results in the competition for task 1: # of queries made if a path is found within query limit. Your score will be computed with respect to your classmates' # of queries.

- Phase III: Results in the competition for task 2: Geo-length of committed path (cost) if a valid path is found within query limit. Your score will be computed with respect to your classmates' # of queries. The smaller cost the better (while preserving accuracy). We will release a more detailed document regarding grading as the challenge approaches.

In terms of dates,

- We will answer **all** questions from now until **October 30th**. After which we will **not** answer **any** question related to phase I (formatting, installation, data access etc.). So plan either to be early or, if you like to be last minute, you'll be on your own. Questions and clarifications specific to Phase II and Phase III and how to optimize your code will still be answered, but remember that TAs and office hours become very busy close to the end of the competition.
- We will have a Q&A session during class on **Nov 3rd**, where we will discuss questions you might have regarding the challenge. Think of it as extra office hours.
- The challenge will occur on **Nov 9-10th**. The competition of data challenge 1 will be held online from 11/9/2016 11:55pm EDT to 11/10/2014 11:55pm EDT. We will provide a set of test cases on which you will run your algorithm and return the paths. After you finished all test cases, upload your source code and a README file explaining your solution to Courseworks. Students who get best results will be invited to briefly present their solutions in class.
- All submissions must be finalized by Nov 10th, **11:55pm**.

References

1. Scellato, S., Noulas, A., & Lambiotte, R. (2011). Socio-spatial Properties of Online Location-based Social Networks. Proceedings of the International Conference Weblogs and Social Media (ICWSM).

2. Cho, E., Myers, S. A., & Leskovec, J. (2011). Friendship and mobility: user movement in location-based social networks (pp. 1082–1090). Presented at the KDD '11: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM Request Permissions. http://doi.org/10.1145/2020408.2020579