Scopely Data Scientist Challenge

Introduction

This challenge is intended to test your ability of applying different statistical algorithms to building usable solutions. In particular, we hope that after you take this challenge we can assess your ability to:

- work with messy data
- analyze large datasets
- independently test algorithms with data sets
- create *usable* APIs. The ability to go end-to-end, from inception to delivery, is very important at our startup
- choose appropriate algorithms and statistical approaches depending on the type of data and business questions being asked

Like and Interest Data

The dataset that you will be working is about 1 GBs of *anonymized* Facebook Like and Interest data. The data is available at: http://bit.ly/scopely-likes-data The data is of the format: “UID”, “Like1”, “Like2”, etc. The data is noisy with lots of likes in foreign languages.

Part I: Analysis

Histogram

Build a histogram of the like data. That is, show a graph that displays likes on the X-axis and frequency count on the Y-axis. For this histogram, please pick an appropriate cutoff or aggregation for the long-tail.

Follow-up Questions

1. What was your approach for building this histogram?
2. How does your approach scale to 1 terabyte of like and interest data?

High-frequency Pairs

Build a histogram of high-frequency like pairs. We're trying to answer the question "when a user likes A, they also like B". Show the most frequently occurring like-pairs, e.g.

1. (pizza, cheese)
2. (kanye, jay-z)
3. (brazil, world cup)

Follow-up Questions

1. What was your approach for building this histogram?
2. How does your approach scale as the number of likes increases linearly?

Part II: Recommendation Systems

Recommended Likes

Build an API (or command line client) that recommends Likes based on an inputted set of Likes. For example if you submit (Kayne West, Jay-Z, Dr. Dre) it might recommend to you (Notorious B.I.G., Tupac, Kid Cudi, Eminem)

Expect that the API input is just a comma separated list in the query string, for example:

```
/?recommend=likes&likes=kanye west,jay-z,dr. dre
```

or, if using the command line:

> recommend-likes likes="kanye west,jay-z,dr.dre"

The API should return a list of likes in order of your metric for recommendation. The API should exclude any likes already liked by the user.

Follow-up Questions

1. Please describe your approach
2. How does your algorithm change as new data is brought into the system?
3. How real-time can your system be? For example, suppose the site gets popular in a new country and an influx of like data comes in that you hadn't seen, how would your system react? What are the limitations?

Recommended Users

Build an API that recommends users based on an inputted set of likes. For example, if you submit (Kanye West, Jay-Z, Dr. Dre) it might recommend to you:

- user Mr. Exact (Kanye West, Jay-Z, Dr. Dre)
- user Mrs. Close (Kanye West, Biggie, Tupac)
- user Ms. Kind-of-Close (Rihanna, Kid Cudi, Eminem)

Expect that the API input is just a comma-separated list in the query string, for example:

```
/?recommend=users&likes=kanye west,jay-z,dr. dre
/?recommend=users&users=<uid1>,<uid2>,<uid3>
```

The caller of the API may or may not be anonymous, you may or may not have seen their like information.

Follow-up Questions

1. Please describe your approach
2. How does your algorithm change as new users and likes are brought into the system?
3. How real-time can your system be? For example, suppose the site gets popular in a new country and an influx of like data comes in that you hadn't seen, how would your system react? What are the limitations?

Submission

Please submit the following:

1. The source code for your project (or link/dropbox to submitted solution)
2. Brief answers to the questions for each section