interview-problems

Table of Contents

1 The Spam Problem

1.1 The Problem

We here at Initrode do a great deal of "customer outreach" and "digital marketing" and "unsolicited emails with the goal of selling people unregulated herbal supplements". Of course respect for the customer and the avoidance of the appearance of spam is of the utmost importance to us. We are good internet citizens! It's not like we're trying to get people to buy Cialis. What we do in our health supplement business is totally different from that - Cialis is FDA-approved.

When we send out our totally reputable emails, it's important that we minimize the chance that we get excessive spam complaints. At the same time, if we're too conservative, we'll end up not sending an email to someone who gives their money to a Nigerian prince instead of us!

1.2 The Rules

We need you to write a system that will process our email batches and decide whether or not to send each email based on the following rules:

1. Must send a maximum of one email per address.
2. Must never send an email with a spam score of more than 0.3.
3. The running mean of spam ratings must never get above 0.05. (In other words, as each email is processed the mean spam score of all the emails that have sent in the batch needs to be recalculated and can't ever be more than 0.05.)
4. The mean spam score of the most recent 100 emails sent can't go above 0.1.

1.3 The Data

Your input is a collection of email records, each of which will have an `:email-address` key and a `:spam-score` key. We're not very good at math or counting, so we're not exactly sure how many emails there will be. Last time I think they said there were 5 million. Or maybe it was 10 million? Gotta remember to ask Randy about that later

Here's one example email record:

```
{:email-address "john123@dired.com"
 :spam-score 0.12}
```

Here's a handy dandy spec for the email records:

```
(require '[clojure.spec :as s]
         '[clojure.spec.gen :as gen])

(def email-domains
  #{"indeediot.com"
    "monstrous.com"
    "linkedarkpattern.com"
    "dired.com"
    "lice.com"
    "careershiller.com"
    "glassbore.com"})

(def email-regex
  #"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,63}$")

(s/def ::email-address
  (s/with-gen
    (s/and string? #(re-matches email-regex %))
    #(->>
      (gen/tuple (gen/such-that not-empty (gen/string-alphanumeric))
                 (s/gen email-domains))
      (gen/fmap (fn [[addr domain]] (str addr "@" domain))))))

(s/def ::spam-score
  (s/double-in :min 0 :max 1))

(s/def ::email-record
  (s/keys :req-un [::email-address ::spam-score]))
```

You can use any pattern, approach, argument and return signature, and composition method you'd like to do this.

It would be great if you did this problem in a git repository that you sent in.

1.4 Even more stuff

Some ideas for totally optional bonus credit things that you could do if you find yourself either especially bored or especially excited by this project:

1. So, rule #3 has you dropping emails based on the running mean. But it's possible that you're just getting a spike of high spam score emails that temporarily push the mean over the limit, but if some of them were sent later the mean wouldn't go over the limit. Instead of using a running mean, make it so that the **final** mean doesn't exceed 0.05.
2. Make your code super parallel.
3. Build an interface that shows what the email processor code is doing - for instance current values for the mean spam rating, group counts in the current window, number of emails accepted, number of emails rejected and the reason, maybe some visualizations. Have fun with it!
4. Actually send emails, sell some herbal supplements, give me the money.

Author: mg

Created: 2017-12-22 Fri 15:25

Emacs 25.2.2 (Org mode 8.2.10)

Validate