PA1 Spamlord: Instructions

Help Center

Homework 1: Spamlord [starter code]

Harvesting emails and phone numbers

Due: Tuesday March 20 11:59pm Pacific Daylight Time

Here's your chance to be a SpamLord! Yes, you too can build regexes to help spread evil throughout the galaxy!

More specifically, your goal in this assignment is to write regular expressions that extract phone numbers and regular expressions that extract email addresses.

To start with a trivial example, given

```
jurafsky@stanford.edu
```

your job is to return

```
jurafsky@stanford.edu
```

But you also need to deal with more complex examples created by people trying to shield their addresses, such as the following types of examples that I'm sure you've all seen:

```
jurafsky(at)cs.stanford.edu
jurafsky at csli dot stanford dot edu
```

You should even handle examples that look like the following (as it appears on your screen; we've used metachars on this page to make it display properly):

```
 <script type="text/javascript">obfuscate('stanford.edu','jurafsky')</script>
```

For all of the above you should return the corresponding email address

```
jurafsky@stanford.edu OR jurafsky@cs.stanford.edu OR jurafsky@csli.stanford.edu
```

as appropriate.
Similarly, for phone numbers, you need to handle examples like the following:

```
TEL +1-650-723-0293
Phone:  (650) 723-0293
Tel (+1): 650-723-0293
<a href="contact.html">TEL</a> +1&thinsp;650&thinsp;723&thinsp;0293
```

all of which should return the following canonical form:

```
650-723-0293
```

(you can assume all phone numbers we give you will be inside North America).
In order to make it easier for you to do this and other homeworks, we will be giving you some data to test your code on, what is technically called a *development test set*. This is a document with some emails and phone numbers, together with the correct answers, so that you can make sure your code extracts all and only the right information.

You will be graded on how well your regular expressions find emails and phone numbers in a different *test set* that we have. Because you don't know exactly what trickery goes on in this test set, you should be creative in looking at the web and thinking of different types of ways of encoding emails and phone numbers, and not just rely on the methods we've listed here, or are listed in the homework.

You won't have to deal with: really difficult examples like images of any kind, or examples that require parsing names into first/last like:

```
 "first name"@cs.stanford.edu
```

or difficult examples like our friend Jim Martin, whose email was listed only as:

```
To send me email, try the simplest address that makes sense.
```

Where can I find the starter code?

As for all assignments in this class, you may use either Java, python. We have provided starter code written in both languages and the relevant configuration files for compiling and running your java program with `ant`, `make` and Eclipse (the java directory is a valid eclipse project). The details for each of these methods are described in the README file. Here is what we give you (the actual link is at the top of the page) :

```
hw1/
  data/
    dev/ # The development set
      aiken
      ashishg
      ...
    devGold # What we think are the right answers (If you see something we do
n't, let us know!)
  java/
    SpamLord.java # java starter code
    Submit.java # java submission code
    build.xml # an ant build file
    Makefile # a make makefile
    README # a file which explains how to compile, develop and submit
    Base64.java # a dependency for the submission code
    json-simple-1.1.1.jar # a dependency for the submission code
  python/
    SpamLord.py # python starter code
    submit.py # python submission code
```

By default, if you execute:

```
$ cd java
$ mkdir classes
$ javac -d classes -target 1.6 SpamLord.java
$ java -cp "classes" SpamLord ../data/dev ../data/devGOLD
```

or

```
$cd python
$python SpamLord.py ../data/dev/ ../data/devGOLD
```

It will run your code on the files contained in data/dev/ and compare the results of a simple regular expression against the correct results. The results will look something like this:

```
True Positives (4)      ###############################
balaji  e       balaji@stanford.edu
nass    e       nass@stanford.edu
shoham  e       shoham@stanford.edu
thm     e       pkrokel@stanford.edu
False Positives (1)     ###############################
psyoung e       young@stanford.edu
False Negatives (110)   ###############################
ashishg e       ashishg@stanford.edu
ashishg e       rozm@stanford.edu
ashishg p       650-723-1614
...
```

The true positive section displays e-mails and phone numbers which the starter code correctly matches, the false positive section displays e-mails which the starter code regular expressions match but which are not correct, and the false negative section displays e-mails and phone numbers which the starter code did not match, but which do exist in the html files. Your goal, then, is to reduce the number of false positives and negatives to zero.

UPDATE:
We've been informed of some new real e-mails which were not included in the original copy of the devGOLD file (pa1-spamlord-v1.tar.gz). The new e-mails are:

```
ouster  e       ouster@cs.stanford.edu
engler  e       engler@stanford.edu
```

They have been added to the new devGOLD file (pa1-spamlord-v2.tar.gz). The grading policy for this will be as generous as possible: if you find the new e-mail and submit it as a guess, we will not count it as a false positive, but if you don't find the new e-mail (because you did the assignment with the pa1-spamlord-v1 version) we won't penalize you for the false negative.

Where should I write my Java code?

The function

```
public List processFile(String fileName, BufferedReader input) {
```

has been flagged for you with the universal "TODO" marker. This function takes in a file and a file name and returns a list of Contact objects which it found in that file. The Contact object is just a simple class which encapsulates the filename, the type ("e" for e-mail, or "p" for phone), and the actual value of the e-mail or phone number. Here is its declaration and data members:

```
class Contact implements Comparable{
    String fileName;
    String type;
    String value;
    ...
```

Your job is to build a system of regular expressions which output as many correct Contact objects as possible.

Where should I write my Python code?

The function

```
def process_file(name, f):
```

has been flagged for you with the universal "TODO" marker. This function takes in a file object (or any iterable of strings) and returns a list of tuples representing e-mails or phone numbers found in that file. Specifically the tuple has the format:

```
(filename, type, value)
```

where type is either 'e', for e-mail, or 'p' for phone number, and value is just the actual e-mail or phone number.

What format should the my phone numbers and e-mail have?

The canonical forms we expect are:

```
user@example.com
650-555-1234
```

The case of the e-mails you find should not matter because the starter code and the grading code will lowercase your matched e-mails before comparing them against the gold set. (Note that this is only true for a revised version of the starter code released 1-11-12 at 11:59. Prior this revision the starter code did not lowercase e-mails, though it will not effect the grading.)

How do I turn in my work?

For each language, we have provided a submission script/program which will run your program and capture the e-mails and phones numbers that it finds. This data will then be sent to our servers where it will be stored and graded once the assignment is due. For each language, the submission program will programmatically call your script and then prompt you for your username and a password. The password can be found at the top of the Assignments page on this website. It will also ask you about which parts of the assignment you wish to submit. For this assignment your system will be evaluated in two parts: on an unseen test set, and on the development set, which is just the contents of the `dev` directory. Here are the basic shell commands which you will need to execute for Java:

```
$ cd java
$ mkdir classes
$ javac -cp "json-simple-1.1.1.jar:." -d classes *.java
$ java -cp "json-simple-1.1.1.jar:classes" Submit
```

and for Python

```
$ cd python
$ python submit.py
```

In order to properly encode your submissions, we use the `json-simple` library for Java and the `json` module for Python. Unfortunately this complicates the submissions process because you need to have the json-simple-1.1.1.jar on the class path for compilation and submission in java and you need to be running a version of python with the built-in `json` module (2.6 and later). It is also important that these scripts be executed from either the java or python directories so that they can easily import the code you've written and so that the relative path "../data/dev" correctly points to the development data (it is hard-coded into the submission scripts). Also, make sure that the files in "../data/dev" are the original version we distributed when you actually submit. If your program finds any new contact info it will be counted as a false positive by the grader. If you want to try you own test cases, just make a copy of the assignment.

UPDATE: Some people have been having trouble with submission errors due to weird characters like quotes winding up in their solutions. This break previous versions of the assignment, but we've created a new version (pa1-spamlord-v5) which should be more robust to these sorts of problems. So, if you encountered an "Error Processing Submission" message when you went to look at your grade, try downloading v4 of the assignment and copying you SpamLord.java file into the fresh directory.