Homework #1: Spamlord

Deadline: 11:59pm Saturday 8 March 2014

Please present a report with all your answers, explanations, and sample images or plots. Submit also a soft copy of the source code and binaries used to generate these results. Please note that copying of any results or source code will result in ZERO credit for the whole homework.

Acknowledgment: This homework is adapted from Chris Manning and Dan Jurafsky's Coursera NLP class from 2012.

Here's your chance to be a SpamLord! Yes, you too can build regexes (regular expressions) 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 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 will be using Python. We have provided starter code for you. 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):

```
CMP462 HW01 Data/
  data/
    dev/ # The development set
      aiken
      ashishg
      ...
    devGold # What we think are the right answers
  python/
    SpamLord.py # python starter code
```

By default, if you execute:

```
$cd python
```

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

or from inside a Spyder, press F5 and run the SpamLord.py script.

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.

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.

Requirements

You are required to fill in the function `process_file` and all required functions/code to detect ALL emails and phone numbers in the dev set. In other words, you are required to obtain ZERO false positives and negatives. Please submit your code and report in one zip file, named CMP462.HW01.bench#.firstname.lastname.zip. For example, if your name is Mohamed Aly and your bench number is 26, your file should be named CMP462.HW01.26.Mohamed.Aly.zip

Grading

3 pts: working code
2 pts: report and submission file name
5 pts: zero false positives/negatives
-1 pt: for any 10% loss in FP or FN