CSCE-638, Programming Assignment #1 SpamLord

Rohan Chaudhury

UIN: 432001358

Solution:

I have used **Python 3.7.2** to code the solution. The code file name is SpamLord.py. In the same directory where the provided code and data are, we can run the code with the following command:

python SpamLord.py <data_dir> <gold_file>

where,

data_dir is the directory containing the files from which we are going to extract the emails and phone numbers

and,

gold_file contains the list of actual emails and phone numbers to be extracted

Example usage: **python SpamLord.py data_dev/dev/ data_dev/devGOLD**

I have modified the function

def process_file(name, f):

and added **regex patterns** in order to code the solution.

Final output on the given dataset:

Summary:

tp=57, fp=0, fn=0

```
PS D:\Texas A&M University\Natural Language Processing\PA1\final_submission> python SpamLord.py data_dev/dev/ data_dev/devGOLD
True Positives (57):
{('Ahmed', 'e', 'tanzir@tamu.edu'),
 ('Ahmed', 'p', '979-845-4908'),
 ('Amato', 'e', 'amato@tamu.edu'),
 ('Amato', 'p', '979-458-0722'),
 ('Amato', 'p', '979-862-2275'),
 ('Andersen', 'e', 'flemminglandersen@tamu.edu'),
 ('Andersen', 'p', '979-845-3510'),
 ('Bettati', 'e', 'bettati@cs.tamu.edu'),
 ('Bettati', 'p', '979-845-5469'),
 ('Chai', 'e', 'jchai@cs.tamu.edu'),
 ('Chai', 'p', '979-845-3510'),
 ('Chaspari', 'e', 'chaspari@usc.edu'),
 ('Chaspari', 'p', '213-740-3477'),
 ('Choe', 'e', 'choe@tamu.edu'),
 ('Choe', 'p', '979-845-5466'),
 ('DaSilva', 'e', 'dilma@cse.tamu.edu'),
 ('Daugherity', 'e', 'daugher@neo.tamu.edu'),
 ('Daugherity', 'p', '979-845-1308'),
 ('Davis', 'e', 'davis@tamu.edu'),
 ('Davis', 'p', '979-845-4094'),
 ('Furuta', 'e', 'furuta@cs.tamu.edu'),
 ('Furuta', 'p', '979-845-3839'),
 ('Gooch', 'e', 'gooch@cse.tamu.edu'),
 ('Gooch', 'p', '979-845-5534'),
 ('Gu', 'e', 'ccs17tutorials@gmail.com'),
 ('Gu', 'e', 'guofei@cse.tamu.edu'),
 ('Gu', 'p', '979-845-2475'),
 ('Gutierrez-Osuna', 'e', 'rgutier@cse.tamu.edu'),
 ('Gutierrez-Osuna', 'p', '979-845-2942'),
 ('Hammond', 'e', 'hammond@tamu.edu'),
 ('Hammond', 'p', '979-353-0899'),
 ('Hu', 'e', 'hu@cse.tamu.edu'),
 ('Hu', 'e', 'xiahu@tamu.edu'),
 ('Hu', 'p', '979-845-8873'),
 ('Ioerger', 'e', 'ioerger@cs.tamu.edu'),
 ('Ioerger', 'p', '979-845-0161'),
 ('JHuang', 'e', 'jeff@cse.tamu.edu'),
 ('JHuang', 'e', 'jeffhuang@tamu.edu'),
 ('JHuang', 'p', '979-458-0722'),
 ('JHuang', 'p', '979-845-5485'),
 ('Jafari', 'e', 'jjackson@tamus.edu'),
 ('Jafari', 'e', 'lmcdow@tamu.edu'),
 ('Jafari', 'e', 'rjafari@tamu.edu'),
 ('Jafari', 'p', '979-458-9808'),
 ('Jafari', 'p', '979-862-4413'),
 ('Jafari', 'p', '979-862-8098'),
 ('Jimenez', 'e', 'djimenez@cs.tamu.edu'),
 ('Jimenez', 'p', '979-845-2434'),
 ('Juan', 'e', 'garay@cse.tamu.edu'),
 ('Juan', 'p', '979-845-4359'),
 ('Kim', 'e', 'ejkim@cs.tamu.edu'),
 ('Kim', 'p', '979-845-3660'),
 ('Klappenecker', 'e', 'klappi@cse.tamu.edu'),
 ('Klappenecker', 'p', '979-458-0608'),
 ('Lee', 'e', 'hlee@cse.tamu.edu'),
 ('Lee', 'p', '979-845-2490'),
 ('deWitte', 'e', 'paula.dewitte@tamu.edu')}
False Positives (0):
set()
False Negatives (0):
set()
Summary: tp=57, fp=0, fn=0
```

So, I have handled all the 57 true positive cases in the dataset (including the special cases which were safe to ignore as mentioned in Canvas).

Final Score:

True Positive = 57

False Positive = 0

False Negative = 0

The special cases handled are as follows:

1. The email addresses that are generated by Javascript functions. Cases: Amato (email), Jhuang (email)

   I have written regex patterns to handle them

2. The phone numbers, fax numbers, or emails that are written in multiple lines

   I am concatenating 2 subsequent lines in the files and then running my regex patterns on them.

3. HTML tags present in the middle of the email address.

   The regex patterns are handling these cases also

ANALYSIS:

Regex Patterns and Cases Handled by them:

A. Normal Email regex Patterns:

1. '(?P<name>[A-Za-z0-9_.-]+)\s*@\s*(?P<domain>[A-Za-z_. -]+(edu|com|org|net|gov))',
2. '(?P<name>[A-Za-z0-9_.-]+)\s*@\s*(?P<domain>[A-Za-z_. -]+\.br)\W',
3. '(?P<name>[A-Za-z0-9_.-]+)\s*%40\s*(?P<domain>[A-Za-z_.-]+\.(edu|com|org|net|gov))',
4. '(?P<name>[A-Za-z0-9_.-]+)\s*%40\s*(?P<domain>[A-Za-z_.-]+\.br)\W',
5. '(?P<name>[A-Za-z0-9_.-]+)\s+\<*\(*[aA][tT]\>*\)*\s+(?P<domain>[A-Za-z_. ;<>()-]+ (edu|org|gov))',
6. '(?P<name>[A-Za-z0-9_.-]+)\s+\<*\(*[aA][tT]\>*\)*\s+(?P<domain>[A-Za-z_. ;<>()-]+ com)',
7. r'(?P<name>[A-Za-z0-9_.-]+)\s+\<*\(*[aA][tT]\>*\)*\s+(?P<domain>[A-Za-z_. ;<>()-]+ net)',

8. r'(?P<name>[A-Za-z0-9_.-]+)\s+\<*\(*[aA][tT]\>*\)*\s+(?P<domain>[A-Za-z_. ;<>()-]+ br)',
9. '(?P<name>[A-Za-z0-9_.-]+)[<(_]+at symbol[>)_]+(?P<domain>[A-Za-z_. ;<>()-]+(?:edu))\W',
10. '(?P<name>[A-Za-z0-9_.-]+)[<(_]+[aA][tT]*[>)_]+(?P<domain>[A-Za-z_. ;<>()-]+(?:edu))\W',
11. '(?P<name>[A-Za-z0-9_.-]+)<[/span()]*>\s+[aA][tT]\s+<[A-Za-z0-9()=]*>(?P<domain>[A-Za-z_. ;<>()-]+edu)',
12. '(?P<name>[A-Za-z0-9_.-]+)\s+\[please add [aA][tT] sign here\]\s+(?P<domain>[\[\]A-Za-z_. ;<>()-]+edu)',
13. '(?P<name>[A-Za-z0-9_.-]+)\s+[aA][tT]\s+(?P<domain>[A-Za-z_.-;]+\.(edu|com|org|net|gov))',
14. '(?P<name>[A-Za-z0-9_.-]+)\s+[aA][tT]\s+(?P<domain>[A-Za-z_.-;]+\.br)\W',
15. '(?P<name>[A-Za-z0-9_.-]+)\s*\[[aA][tT]\]\s*(?P<domain>[A-Za-z_.-;]+\.(edu|com|org|net|gov))',
16. '(?P<name>[A-Za-z0-9_.-]+)\s*\[[aA][tT]\]\s*(?P<domain>[A-Za-z_.-;]+\.br)\W',
17. '(?P<name>[A-Za-z0-9_.-]+)\s*\([aA][tT]\)\s*(?P<domain>[A-Za-z_.-;]+\.(edu|com|org|net|gov))',
18. '(?P<name>[A-Za-z0-9_.-]+)\s*\([aA][tT]\)\s*(?P<domain>[A-Za-z_.-;]+\.br)\W',
19. '(?P<name>[A-Za-z0-9_.-]+)\s+[\[\]A-Za-z]+\s+[aA][tT]\s+[\[\]A-Za-z]+\s+(?P<domain>cse[\[\]A-Za-z_.; <>()-]+edu)',
20. '(?P<name>[A-Za-z0-9_.-]+)\s+[\[\]A-Za-z]+\s+[aA][tT]\s+[\[\]A-Za-z]+\s+(?P<domain>tamu[\[\]A-Za-z_.; <>()-]+edu)',
21. r'write_mail\(\"(?P<domain>[A-Za-z_.-]+)\",\"(?P<name>[A-Za-z0-9_.-]+)\"\)',
22. r'mail_link\(\"(?P<name>[A-Za-z_.-]+)\",[ ]*\"(?P<domain>[A-Za-z0-9_.-]+)\"',
23. r'obfuscate\(\'(?P<domain>[A-Za-z_.-]+)\',[ ]*\'(?P<name>[A-Za-z0-9_.-]+)\'\)']

These patterns handle the extraction of the following examples of emails patterns (all the upper case character cases of the below examples are also handled):

1. roh@gmail.com (all the cases are also handled for net/gov/edu/org/br domains)
2. roh.rc@gmail.com
3. roh-rc@gmail.com
4. roh@cs.tamu.edu
5. roh@abc.dc.com
6. write_mail('domain','name')
7. mail_link('name', 'domain', …
8. obsfucate('domain','name')
9. roh at gmail dot com
10. roh [at] gmail <dot> com
11. roh (at) gmail.com
12. roh <at> gmail [dot] com
13. roh <at symbol> gmail dot com
14. roh [please add at sign here] gmail dot com
15. roh_at_gmail_dot_com
16. roh-at-gmail-dot-com

and all possible combinations of the above,

The regex patterns divide the matches into 2 groups with the names: 'name' and 'domain'. The groups are process in the process_file function and then concatenated as: match.group('name')@match.group('domain').

B. Special email regex patterns:

1. '[A-Z0-9]+[<(_]+(dot)+[<(_]+[A-Z0-9]+[<(_]*dot[<(_]*[A-Z0-9]+[<(_]+[aA][tT]*[>)_]+[A-Z0-9]*[<(_]*(dot)*[<(_]*[A-Z0-9]+[<(_]*dot[<(_]*(edu|com)',
2. '[A-Z0-9]+[<(_]*dot[<(_]*[A-Z0-9]+[<(_]+[aA][tT]*[>)_]+[A-Z0-9]*[<(_]*(dot)*[<(_]*[A-Z0-9]+[<(_]*dot[<(_]*(edu|com)'

These 2 regex patterns handle the following special cases:

1. roh dot cha dot rc at cs dot tamu dot edu (final ans: roh.cha.rc@cs.tamu.edu)
2. roh dot cha dot rc at cs dot edu (final ans: roh.cha.rc@cs.edu)
3. roh dot rc at cs dot tamu dot edu (final ans: roh.rc@cs.tamu.edu)
4. roh dot rc at cs dot edu (final ans: roh.rc@cs.edu)

The patterns are written in such a way so that the entire email is extracted and not a part of it. For example:

For the case of roh dot cha dot rc at cs dot tamu dot edu, it was highly possible to get cha.rc@cs.tamu.edu, rc@cs.tamu.edu in addition to roh.cha.rc@cs.tamu.edu, but those cases are also handled in the code (using flags in the process file function) thereby always giving the complete email address and not the subsets.

C. Phone Number regex pattern

1. '\D+[+1\.]*[(]*([\d]{3})[)-/.]+([\d]{3})[-/.]+([\d]{4})\D+'

This regex pattern handles the extraction of the following kind of number patterns:

1. XXX-XXX-XXXX
2. (XXX)-XXX-XXXX
3. (XXX) XXX XXXX
4. (XXX) (XXX) XXXX
5. XXX-XXX XXXX
6. XXX XXX XXXX
7. (XXX)-(XXX) XXXX
8. (XXX)-(XXX)-(XXXX)

And all possible combinations of the above,

Code also considers the case that fax numbers are also similar in nature and doesn't consider them in the final output. The code does so by employing the use of a flag which denotes whether the keywords ('fax') or ('ph|p:|tel|office|voice') had come before it. If the keyword 'fax' had arrived before the present matched pattern, it does not consider that pattern. Otherwise if the keywords ('ph|p:|tel|office|voice') had arrived before the present matched pattern, it considers that pattern in the final output. Finally, all the patterns are formatted in the following pattern:

XXX-XXX-XXXX

Limitations of the code:

1. I am not handling images of email addresses or phone numbers which are embedded in the websites

2. I am not handling email addresses which require parsing names into first/last like:

   "first name"@cse.tamu.edu

3. I am handling only a limited number of HTML tags that present in the middle of the email address

I have searched several other personal websites to see what other patterns might be possible and included these cases in the code.