Homework 2: SpamLord!
Siri Chandana Sambatur (829955672)

Initially, regular expressions for mobile patterns are appended to the patterns list to detect the phone numbers. Handling mobile numbers is easier than email patterns because they are very few combinations of how they could exist in the html pages. The following is the list of steps taken to detect all the phone numbers in the files-

1) Initially, two regular expressions were used to detect phone numbers

Regular Expressions- ‘ (\d{3})-(\d{3})-(\d{4}) ’ and ‘ (\d{3})\s(\d{3})-(\d{4}) ’

Patterns matched- 650-723-1131, 650 725-372

Summary- It successfully matched 21 patterns (tp=21, fp=0, fn=96)

2) Then, some phone numbers were embedded in brackets like [] or () and also, the spaces and - present between them was optional. So the regular expressions given above were combined and modified to the following form.

Regular Expression- ‘ [\(|\[]?(\d{3})[\)|\]]?\s?[-]?(\d{3})\s?[-]?(\d{4}) ’

Patterns matched- 650-723-1131, 650 725-372, (678)345 3456, [890] 234 2345

Summary- This pattern detected all the phone numbers in the files. However, this pattern increased the false positives count. (tp=72, fp=7, fn=45)

3) So, to decrease the false positive count the pattern was modified as follows. The meta character was included and ‘$’ to indicate the end of line character.

Regular Expression- ‘ [\(|\[]?(\d{3})[\)|\]]?\s?[-]?(\d{3})\s?[-]?(\d{4})\D+$ ’

Patterns matched- 650-723-1131, 650 725-372, (678)345 3456, [890] 234 2345

Summary- eliminated all the false positives, but all the phone numbers were not detected.
(tp=66, fp=0, fn=51)

4) To detect all the patterns additional regular expression was added to detect punctuations that are to be present other than end of line character.

Regular Expression- ‘ [\(|\[]?(\d{3})[\)|\]]?\s?[-]?(\d{3})\s?[-]?(\d{4})\D+[\s|,|\<|}|^;] ’

Patterns matched- 650-723-1131 <span>, 650 725-372 ;

Summary- (tp=72, fp=0, fn=45)

5) Also, to eliminate the false positives the if condition is added to process_file() where if there is ‘id’ in the line means that the number detected is a part of the html tags. This eliminated 10 false positives. This problem was caused in ‘latombe’ and ‘nass’ file.

Finally the regular expressions used are-

1) [\(|\[]?(\d{3})[\)|\]]?\s?[-]?(\d{3})\s?[-]?(\d{4})\D+$
2) [\(|\[]?(\d{3})[\)|\]]?\s?[-]?(\d{3})\s?[-]?(\d{4})\D+[\s|,|\<|}|^;]

And the changes made to the process-file() are as follows-

1) Added an if condition to check is 'id' is present in the line.
2) Replace the following characters { (,) , [,] , \s} with blank characters.

After matching all the phone numbers, regular expressions were created to match the email patterns in the files. The following is the way in which the regular expressions were added to match the email patterns-

1) Initially, added the following regular expressions to match simple patterns.

Regular Expression- '([A-Za-z.]+)@([A-Za-z.]+)\.edu' and '([A-Za-z.]+)\s@\s([A-Za-z.]+)\.edu'
Patterns matched- abc@xyz.edu, abc @xyz.edu
Summary- This matched most of the emails (tp=94, fp=0, fn=23)

2) Then, when missing email patterns were checked it could be seen that '@' is written in various forms like 'at', 'AT' or 'where'. Also, it was found that '.' character is written as 'dot', 'dt', 'DOT'. So, included these variety of possibilities in the regular expression.

Regular Expression- '([A-Za-z0-9._-]+)\s*(?:@| at | AT|WHERE)\s*([A-Za-z0-9._-]+)\s?(?:.| dot | DOT | dt | DOM)edu'
Patterns matched- abc@xyz.edu, abc @xyz.edu, abd at xyc.edu, adc @ xyz dt edu
Summary- tp=72, fp=0, fn=45

3) On further analysis it was found that '@' is also represented with '&#x40;' and using the html '<at symbol> tag. So, the following are the changes in the regular expression. Also, it is seen that in some places 'edu' occurs in capitals like 'EDU'

Regular Expression- '([A-Za-z0-9._-]+)\s*(\(f.*y.*)?(?:@| at | AT |&#x40;|WHERE)\s*([A-Za-z0-9._-]+)\s?(?:.| dot | DOT | dt | DOM)[edu|EDU]'
Patterns matched- abc@xyz.edu, abc @xyz.edu, abd at xyc.edu, adc @ xyz dt edu, abc<at symbol>xyz.EDU
Summary- It found some false positive patterns (tp=107, fp=4, fn=10).

4) Then, in some files like 'latombe', del tag has been used to strike out the domain of the email to identify this the following regular expression was added.

Regular Expression- ([A-Za-z.]+)\s*\<at symbol\>\s*([A-Za-z.]+)(?:.| dot | DOT | dt | DOM)(edu|EDU|)
Matched patterns- abc@xyz.edu, abc @xyz.edu, abd at xyc.edu, adc @ xyz dt edu, abc<at symbol>xyz.EDU, abc<del>@xyz.com

Summary- False positives still exist. (tp=108, fp=4, fn=7)

5) To eliminate the false positives an if condition is added in the process_file() and also to identify the email in 'dlwh' file and in the 'ouster' file a few emails were written using 'followed by' string, so '(\(f.*y.*)?' and '-?e-?d-?u' has been added to the above regular expression. Now the modified regular expression is-

Regular Expression- '([A-Za-z0-9._-]+)\s*(\(f.*y.*)?(?:@| at | AT |&#x40;|WHERE)\s*([A-Za-z0-9._-]+)\s?(?:.| dot | DOT | dt | DOM)(-?e-?d-?u|EDU|edu)'

Patterns Matched- abc@xyz.edu, abc @xyz.edu, abd at xyc.edu, adc @ xyz dt edu, abc<at symbol>xyz.EDU, abc<del>@xyz.edu, a-c-b-@-x-y-z-e-d-u, ouster (followed by &ldquo;@cs.stanford.edu&rdquo;)

Summary- Few emails still remain unmatched. (tp=110, fp=0, fn=7)

Finally, the regular expressions used to detect email patterns is-

1) '([A-Za-z0-9._-]+)\s*(\(f.*y.*)?(?:@| at | AT |&#x40;|WHERE)\s*([A-Za-z0-9._-]+)\s?(?:.| dot | DOT | dt | DOM)(-?e-?d-?u|EDU|edu)'
2) '([A-Za-z.]+)\s?\<del\>@([A-Za-z.]+)(?:.| dot | DOT | dt | DOM)(edu|EDU|)'
3) '([A-Za-z.]+)\s*\<at symbol\>\s*([A-Za-z.]+)(?:.| dot | DOT | dt | DOM)(edu|EDU)'

The following is the list of changes made to the process_file()-

1) Modified the match patterns so that it could be of the form 'name' and 'domain'
2) Replaced the characters {-,\<del\>,..} with spaces

The following is a list of complex email patterns that couldn't be detected and reasons behind why the couldn't be detected using our regular expressions-

1) When the email id is in the form of plain words, when trying to detect these time of emails, the fp rate is increasing drastically

The following patterns couldn't be detected with the regular expressions

- serafim at cs dot stanford dot edu
- subh AT stanford DOT edu
- hager at cs dot jhu dot edu
- pal at cs stanford edu,
- jks at robotics;stanford;edu

2) Few emails have been generated by obfuscate() in javascript like the jurafsky
3) The email id which was embedded in an image tag, that is the email id is present in the image. In vladlen file- email id is embedded in image

Option 3- Modified the process_file() function to match email ids that end in “.com” as well. For this the regular expressions used are as follows-

1) ‘([A-Za-z0-9._-]+)\s*(\(f.*y.*)?(?:@| at | AT |&#x40;|WHERE)\s*([A-Za-z0-9._-]+)\s?(?:.| dot | DOT | dt | DOM)(-?e-?d-?u|EDU|edu|com|COM)’
2) '([A-Za-z.]+)\s?\<del\>@([A-Za-z.]+)(?:.| dot | DOT | dt | DOM)(edu|EDU|com|COM)’
3) ‘([A-Za-z.]+)\s*\<at symbol\>\s*([A-Za-z.]+)(?:.| dot | DOT | dt | DOM)(edu|EDU|com|COM)’

And the changes made to the process_file() can be seen in the ‘spamlord.py’ file. The pattern is now extracted from the regular expression as ‘name’ ,’email’ , and ‘domain’. This is a three sub categories of the email id as ‘name@email.domain'. So, the regular expressions are modified to extract these three and then stored in the email variable in process_file(). So, this could match various patterns like - ‘abc<del>@xyz.com’ , ‘abc @ xyz.com', ‘abc at xyz .. COM’. Also, we could match other email patterns ending ‘.in’ etc. by adding them to our regular expression and the process_file() identify them.

Conclusion-

All of the phone number patterns have been successfully detected with regular expressions, while a few email patterns could not be detected. The precision of finding the regular expressions is 100% and the recall is 94.01%.