TRIVENI ASHOK NAIK
SUID: 406461644

NLP HOMEWORK 2

1. (Required) Write more regular expressions that correct false positives or fit false negatives. Do as best as you can using the epatterns and ppatterns lists in the program. List the regular expressions that you write, examples of email or phone numbers that match each pattern, and a short English explanation of what expressions the pattern matches. For each expression, give the results (TP, FP and FN) before and after you added the expression. And give the text that matches, not only the results! When you are done with as many as you can do, give the output of the program.

Solution :

Before writing the Regular Expression when I just ran the SpamLord.py program, I got False Positive(FP) = 0, True Positive (TP) = 1 and False Negative(FN) = 116. I did the following tasks to reduce the number of False negatives.

Email Addresses:

1. The Regular Expression that matched here are the one that are the most

- These are some of the mails that match. But there are other mails also that match this expression.Here are some examples.
- ashishg @ stanford.edu , balaji@stanford.edu , uma@cs.stanford.EDU , dabo @ cs.stanford.edu , eroberts@cs.stanford.edu , patrick.young@stanford.edu.
- The Regular Expression for this is given as
  epatterns.append('([A-Za-z.]+)\s+[Aa][Tt]\s+([A-Za-z.]+\s*[Dd]?[Oo]?[Tt.;]?\s?[a-zA-Z]+\s*)[Dd]?[Oo]?[Tt.;]?\s?edu')
  epatterns.append('([A-Za-z.\-]+)\s*@\s*([A-Za-z.\-]+)\.[\-Ee]+[\-Dd]+[Uu]+')
- In these Regular Expression, I am trying to match all the kinds of patterns @, DOT/dot , Aa/Tt in the expression and also if it has semicolon and hyphen in the expression.
- * is written here which takes 0 more times the previous character. There is also ? which is 0 or more times previous character(optional). + is used to match the one or more preceding characters.
- Results : False Positive(FP) = 9, True Positive (TP) = 25 and False Negative(FN) = 92

2. The expression that I chose here is from the **Lantobe** file.

- The Email Address is asandra<del>@cs.stanford.edu, latombe<del>@cs.stanford.edu, liliana<del>@cs.stanford.edu
- The Regular Expression for this is given as
  epatterns.append('([A-Za-z]+)<del>@([A-Za-z.]+)\.edu')

TRIVENI ASHOK NAIK
SUID: 406461644

- There is ? which is 0 or more times previous character(optional). + is used to match the one or more preceding characters. The HTML tags are written as it is to capture it.
- This Regular Expression matches the Email Address. After Processing this the results are given below.
- Results : False Positive(FP) = 9, True Positive (TP) = 28 and False Negative(FN) = 89.

3. The expression that I chose here is from the **Manning** file.

- The Email Address is manning <at symbol> cs.stanford.edu, dbarros <at symbol> cs.stanford.edu
- The Regular Expression for this is given as **epatterns.append('([A-Za-z]+)\s<[A-Za-z]+\s[A-Za-z]+>\s([A-Za-z.]+)\.edu')**
- There is ? which is 0 or more times previous character(optional). + is used to match the one or more preceding characters. The HTML tags are written as it is to capture it.
- This Regular Expression matches the Email Address. After Processing this the results are given below.
- Results : False Positive(FP) = 9, True Positive (TP) = 30 and False Negative(FN) = 87.

4. The expression that I chose here is from the **levoy** file.

- The Email Address is ada@graphics.stanford.edu melissa@graphics.stanford.edu
- The Regular Expression for this is given as **epatterns.append('([A-Za-z]+)&#[A-Za-z][0-9]+;([A-Za-z.]+)\.edu')**
- There is ? which is 0 or more times previous character(optional). + is used to match the one or more preceding characters. The HTML tags are written as it is to capture it.
- This Regular Expression matches the Email Address. After Processing this the results are given below.
- Results : False Positive(FP) = 9, True Positive (TP) = 32 and False Negative(FN) = 85.

5. The expression that I chose here is from the **engler** file.

The Email Address is engler WHERE stanford DOM edu

- The Regular Expression for this is given as **epatterns.append('([A-Za-z]+)\sWHERE\s([A-Za-z]+)\sDOM\sedu')**
- Here to capture the HTML Tags as it is, the portion of the regex that needs to be matched is written as it is.
- This Regular Expression matches the Email Address. After Processing this the results are given below.
- Results : False Positive(FP) = 9, True Positive (TP) = 33 and False Negative(FN) = 84.

6. The expression that I chose here is from the **ouster** file.

- The Email Address is manning ouster (followed by “@cs.stanford.edu”) and teresa.lynn (followed by "@stanford.edu")

- The Regular Expression for this is given as
  epatterns.append('([A-Za-z.]+)\s\(followed by "@([A-Za-z.]+)\.edu')
  epatterns.append('([A-z]+)\s+\(followed by\s?(?:“)?@([A-z.-]+)+\.edu')
- There is ? which is 0 or more times previous character(optional). + is used to match the one or more preceding characters. The HTML tags are written as it is to capture it.
- This Regular Expression matches the Email Address. After Processing this the results are given below.
- Results : False Positive(FP) = 9, True Positive (TP) = 35 and False Negative(FN) = 82.

7. The expression that I chose here is from the **Vladlen** file.

- The Email Address is vladlen at <!-- die!--> stanford <!-- spam pigs!--> dot <!-- die!--> edu
- The Regular Expression for this is given as
  epatterns.append('([A-Za-z]+)\sat\s<!-- die!-->\s([A-Za-z]+)\s<!-- spam pigs!-->\sdot\s<!-- die!-->\s[Ee][Dd][Uu]')
- There is ? which is 0 or more times previous character(optional). + is used to match the one or more preceding characters. The HTML tags are written as it is to capture it.
- This Regular Expression matches the Email Address. After Processing this the results are given below.
- Results : False Positive(FP) = 9, True Positive (TP) = 36 and False Negative(FN) = 81.

8. The expression that I chose here is from the **Subh** file.
   - The Email Address is subh AT stanford DOT edu
- The Regular Expression for this is given as
  epatterns.append('([A-Za-z]+)\s[Aa][Tt]\s([A-Za-z]+)\sDOT\sedu')
- There is ? which is 0 or more times previous character(optional). + is used to match the one or more preceding characters. The HTML tags are written as it is to capture it.
- This Regular Expression matches the Email Address. After Processing this the results are given below.
- Results : False Positive(FP) = 9, True Positive (TP) = 37 and False Negative(FN) = 80.

The phone numbers:

1. The Phone numbers from the bgirod, dabo, Hanrahan, Kosecka , Kunle , Lam , Latombe , Levoy , Manning , Nick , Ok , Rinard , Serafim , Thm , Zm files have a particular structure.

- All have phone numbers in the form of (xxx) xxx-xxxx
- Some Phone numbers are (650) 724-6354 , (650) 723-4539 , (650) 724-3648
- The Regular Expression for this is given as
  ppatterns.append ('[\(](\d{3})[\)]+\s(\d{3})[\-]+(\d{4})')
- This Regular Expression matches the Phone Numbers. After Processing this the results are given below.

- Results : False Positive(FP) = 9, True Positive (TP) = 76 and False Negative(FN) = 41.

2 The Phone numbers from the Cheriton, Eroberts, Hager, Rajeev, Subh, Ulman , Widom , Zelenski files have a particular structure.

- All have phone numbers in the form of xxx- xxx-xxxx
- Some Phone numbers are 650-723-1131, 650-725-3726, 650-724-3648
- The Regular Expression for this is given as
  ppatterns.append('(\d{3})-(\d{3})-(\d{4})')
- This Regular Expression matches the Phone Numbers. After Processing this the results are given below.
- Results : False Positive(FP) = 9, True Positive (TP) = 95 and False Negative(FN) = 22.

3. The Phone numbers from the Ashish, Horowitz, Jurafsky, Pal, Shoham, Tim, files have a particular structure.

- All have phone numbers in the form of (xxx)xxx-xxxx
- Some Phone numbers are (650)723-1614, (650)723-4173, (650)814-1478
- The Regular Expression for this is given as
  ppatterns.append('[\(](\d{3})[\)]+(\d{3})[\-]+(\d{4})')
- This Regular Expression matches the Phone Numbers. After Processing this the results are given below.
- Results : False Positive(FP) = 9, True Positive (TP) = 107 and False Negative(FN) = 10.

4. The Phone numbers from the Nass file has a particular structure.

- All have phone numbers in the form of [xxx]xxx-xxxx
- Some Phone numbers are [650] 723-5499 , [650] 725-2472
- The Regular Expression for this is given as
  ppatterns.append('[\[](\d{3})[\]]+(\d{3})[\-]+(\d{4})')
- This Regular Expression matches the Phone Numbers. After Processing this the results are given below.
- Results : False Positive(FP) = 9, True Positive (TP) = 109 and False Negative(FN) =8.

3. (Option) Python programming: Continue working on the regular expressions to match more examples by having other lists of patterns. For example, you may want to have patterns that will match three parts of the email addresses, or you may want to make a list of patterns for email addresses that end in .com. For each list, you will need to add a part to process_filename that matches that list and puts its parts into a standard format answer.

TRIVENI ASHOK NAIK
SUID: 406461644

Write a section in the report that describes your approach and gives examples of the extended regular expressions or processing that match some of the emails. If you add code to the program, add comments to the program that you submit.

The expression that I chose here is from the **Ulman** file.

- The Email Address is support at gradiance dt com
- I have initialized a new function called as the tpatterns so that it could capture all the ".com " . This was found in the Ulman file.
- The Regular Expression for this is given as
  tpatterns = []
- **tpatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+)\sdt\scom')**
- This Regular Expression matches the Email Address. After Processing this the results are given below.
- Results : False Positive(FP) = 9, True Positive (TP) = 110 and False Negative(FN) = 7

- I added this extra portion of the code in the SpamLord.py file. I added this part of the code to the process_filename function.
- This was added so that it matched all the patterns which end with .com

```
for tpat in tpatterns:
    # each epat has 2 sets of parentheses so each match will have 2 items in a list
    matches = re.findall(tpat,line)
    for m in matches:
        # string formatting operator % takes elements of list m
        #   and inserts them in place of each %s in the result string
        email = '%s@%s.com' % m
        res.append((name,'e',email))
```

- The expression that I chose here is from the **Pal , jks, Hager** files.
- The Email Address is pal at cs stanford edu, jks at robotics;stanford;edu and for Hager hager at cs dot jhu dot edu , serafim at cs dot stanford dot edu and uma at cs dot stanford dot edu
- I have initialized a new function called as the lpatterns so that it could capture all the three part of the email address.
- This Regular Expression matches the Email Address. After Processing this the results are given below.
- The Regular Expression for this is given as

lpatterns = []
 # for pal
lpatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+)\s([A-Za-z]+)\sedu')

TRIVENI ASHOK NAIK
SUID: 406461644

#jks robotics
lpatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+);([A-Za-z]+);edu')
#for hager
lpatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+)\sdot\s([A-Za-z]+)\sdot\sedu')

- Results : False Positive(FP) = 9, True Positive (TP) = 115 and False Negative(FN) = 2

- This part of the code was added to match all the 3 parts of the email addresses.
- It was added to the process_filename function.
- To match all the 3 parts of the email address - %s@%s.%s.edu

```
for lpat in lpatterns:
        # each epat has 2 sets of parentheses so each match will have 2 items in a list
        matches = re.findall(lpat,line)
        for m in matches:
           # string formatting operator % takes elements of list m
           #   and inserts them in place of each %s in the result string
           email = '%s@%s.%s.edu' % m
           res.append((name,'e',email))
```

Screenshot:

```
 ('zm', 'e', 'manna@cs.stanford.edu'),
 ('zm', 'p', '650-723-4364'),
 ('zm', 'p', '650-725-4671')}
False Positives (9):
{('dlwh', 'e', 'd-l-w-h-@-s-t-a-n-f-o-r-d-.edu'),
 ('hager', 'e', 'hager@cs dot jhu .edu'),
 ('jks', 'e', 'jks@robotics;stanford.edu'),
 ('jure', 'e', 'server@cs.stanford.edu'),
 ('pal', 'e', 'pal@cs stanford .edu'),
 ('plotkin', 'e', 'server@infolab.stanford.edu'),
 ('serafim', 'e', 'serafim@cs dot stanford .edu'),
 ('subh', 'e', 'subh@stanford dot .edu'),
 ('subh', 'e', 'uma@cs dot stanford .edu')}
False Negatives (2):
{('dlwh', 'e', 'dlwh@stanford.edu'), ('jurafsky', 'e', 'jurafsky@stanford.edu')}
Summary: tp=115, fp=9, fn=2
Trivenis-MacBook-Pro:SpamLord triveninaik$
```

The final Result:

```
 ('zelenski', 'p', '650-723-6092'),
 ('zelenski', 'p', '650-725-8596'),
 ('zm', 'e', 'manna@cs.stanford.edu'),
 ('zm', 'p', '650-723-4364'),
 ('zm', 'p', '650-725-4671')}
False Positives (2):
{('jure', 'e', 'server@cs.stanford.edu'),
 ('plotkin', 'e', 'server@infolab.stanford.edu')}
False Negatives (2):
{('dlwh', 'e', 'dlwh@stanford.edu'), ('jurafsky', 'e', 'jurafsky@stanford.edu')}
Summary: tp=115, fp=2, fn=2
Trivenis-MacBook-Pro:SpamLord triveninaik$
```

TRIVENI ASHOK NAIK
SUID: 406461644

- Results : False Positive(FP) = 2, True Positive (TP) = 115 and False Negative(FN) = 2

Result: There are 2 false positives and 2 False Negatives in the final result. For False Positives, the files are jure and plotkin. It still appears there because it requires more than 3 parts of the email addresses which is not being done in the code.

In the case of False Negatives, we have dlwh and as well as jurafsky . Dlwh has lot of hyphens which becomes extremely difficult for the SpamLord to detect and Process.
In case of Jurafsky, the email address has to be reversed which becomes difficult to process for the SpamLord.