Assignment 2
CIS668 – NLP

Manjunath Panduranga
SUID: 637425699

ASSIGNMENT – 2

REGULAR EXPRESSIONS

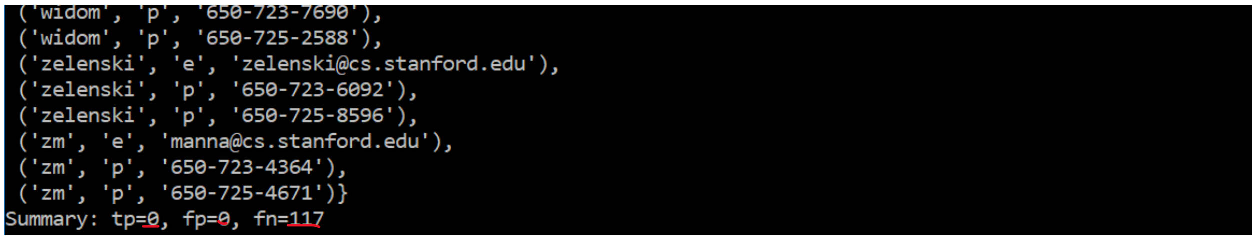
PART-1

After working through all the patterns, I am left with **0 False Positives** *and* **12 False negatives** *related to the emails.*
All the phone patterns are matched perfectly.
The below screenshot shows the un-matched and wrongly matched ones for the emails.

INTIALLY:

As we can see here, at the beginning we have:

TRUE POSITIVES	0
FALSE POSITIVES	0
FALSE NEGATIVES	117



```
 ('widom', 'p', '650-723-7690'),
 ('widom', 'p', '650-725-2588'),
 ('zelenski', 'e', 'zelenski@cs.stanford.edu'),
 ('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')}
Summary: tp=0, fp=0, fn=117
```

EMAIL PATTERNS:

1. epatterns.append('([A-Za-z.]+)\s*?@\s*?([A-Za-z.]+)\.edu')

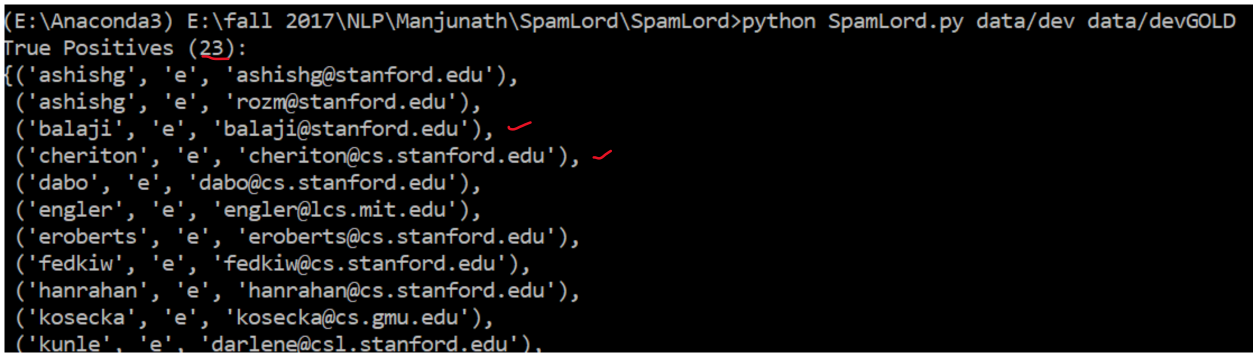
BEFORE:

INFO	EMAIL ADDRESS

'balaji', 'e', 'balaji@stanford.edu'	balaji@stanford.edu
'cheriton', 'e', 'cheriton@cs.stanford.edu'	cheriton@cs.stanford.edu

The email address provided above will definitely match with the pattern provided. This is because, there are emails provided which have space between the @ symbol and there are some more email address with dot in between the @ symbol

AFTER:



```
(E:\Anaconda3) E:\fall 2017\NLP\Manjunath\SpamLord\SpamLord>python SpamLord.py data/dev data/devGOLD
True Positives (23):
{('ashishg', 'e', 'ashishg@stanford.edu'),
 ('ashishg', 'e', 'rozm@stanford.edu'),
 ('balaji', 'e', 'balaji@stanford.edu'),
 ('cheriton', 'e', 'cheriton@cs.stanford.edu'),
 ('dabo', 'e', 'dabo@cs.stanford.edu'),
 ('engler', 'e', 'engler@lcs.mit.edu'),
 ('eroberts', 'e', 'eroberts@cs.stanford.edu'),
 ('fedkiw', 'e', 'fedkiw@cs.stanford.edu'),
 ('hanrahan', 'e', 'hanrahan@cs.stanford.edu'),
 ('kosecka', 'e', 'kosecka@cs.gmu.edu'),
 ('kunle', 'e', 'darlene@csl.stanford.edu'),
```

As we can see in the screenshot above the examples I provided has turned to true positive

And there are 23 true positives, therefore the total number of TP, FP, FN are:

TRUE POSITIVES	23
FALSE POSITIVES	0
FALSE NEGATIVES	94

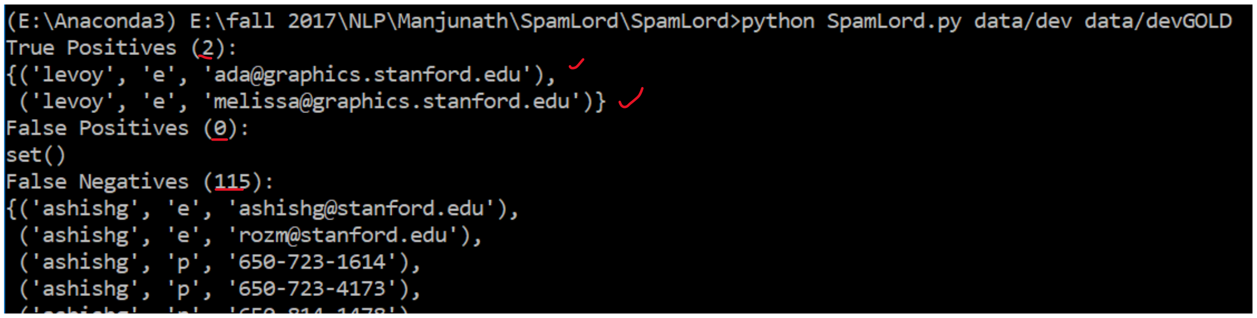
2. epatterns.append('([A-Za-z.]+)\s?&#x40;\s*([A-Za-z.]+)\.edu')

BEFORE:

INFO	EMAIL ADDRESS
'levoy', 'e', 'ada@graphics.stanford.edu'	ada&#x40;graphics.stanford.edu
'levoy', 'e', 'melissa@graphics.stanford.edu'	melissa&#x40;graphics.stanford.edu

As we can see in the email address show above, there are some email address provided with &#x40; instead of @symbol. Therefore by using the pattern shown above we can turn such email to TP

AFTER:



```
(E:\Anaconda3) E:\fall 2017\NLP\Manjunath\SpamLord\SpamLord>python SpamLord.py data/dev data/devGOLD
True Positives (2):
{('levoy', 'e', 'ada@graphics.stanford.edu'), ✓
 ('levoy', 'e', 'melissa@graphics.stanford.edu')} ✓
False Positives (0):
set()
False Negatives (115):
{('ashishg', 'e', 'ashishg@stanford.edu'),
 ('ashishg', 'e', 'rozm@stanford.edu'),
 ('ashishg', 'p', '650-723-1614'),
 ('ashishg', 'p', '650-723-4173'),
```

As we can see in the screenshot above the examples I provided has turned to true positive

And there are 2 true positives, therefore the total number of TP, FP, FN are:

TRUE POSITIVES	2
FALSE POSITIVES	0
FALSE NEGATIVES	115

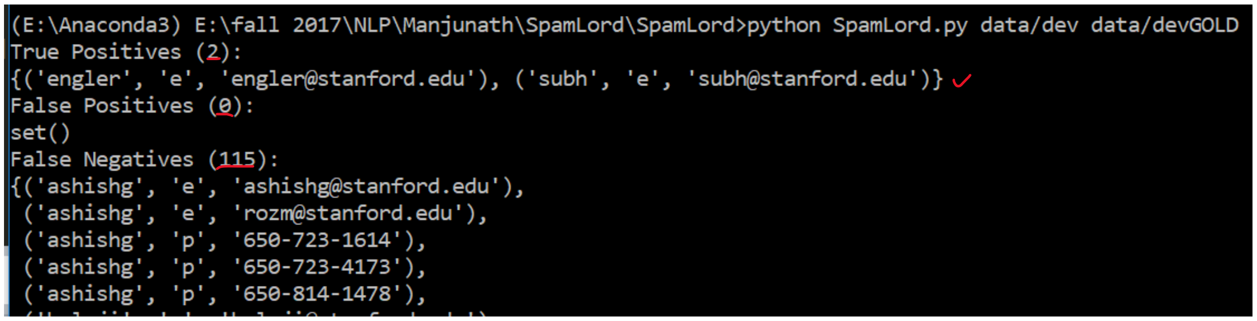
3. epatterns.append('([A-Za-z.]+)\s[A-Z]+\s([A-Za-z.]+)\s[A-Za-z]+\sedu')

BEFORE:

INFO	EMAIL ADDRESS
'subh', 'e', 'subh@stanford.edu'	subh AT stanford DOT edu
'engler', 'e', 'engler@stanford.edu'	engler WHERE stanford DOM edu

As seen from the above email pattern provided, there are AT and DOM instead of @ and .
So for this case, the pattern provided above have [A-Z] instead of @ and .

AFTER:



```
(E:\Anaconda3) E:\fall 2017\NLP\Manjunath\SpamLord\SpamLord>python SpamLord.py data/dev data/devGOLD
True Positives (2):
{('engler', 'e', 'engler@stanford.edu'), ('subh', 'e', 'subh@stanford.edu')}
False Positives (0):
set()
False Negatives (115):
{('ashishg', 'e', 'ashishg@stanford.edu'),
 ('ashishg', 'e', 'rozm@stanford.edu'),
 ('ashishg', 'p', '650-723-1614'),
 ('ashishg', 'p', '650-723-4173'),
 ('ashishg', 'p', '650-814-1478'),
```

As we can see in the screenshot above the examples I provided has turned to true positive

And there are 2 true positives, therefore the total number of TP, FP, FN are:

TRUE POSITIVES	2
FALSE POSITIVES	0
FALSE NEGATIVES	115

---

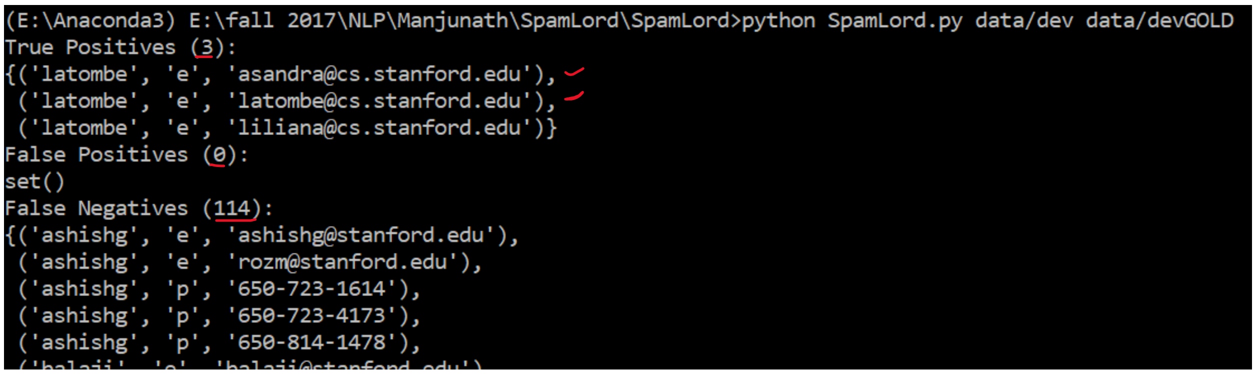
4. epatterns.append('([A-Za-z.]+)\s*<del>\@\s*([a-z.]+)\.edu')

   BEFORE:

INFO	EMAIL ADDRESS
'latombe', 'e', 'latombe@cs.stanford.edu'	latombe<del>@cs.stanford.edu
'latombe', 'e', 'asandra@cs.stanford.edu'	asandra<del>@cs.stanford.edu

As shown from the above example, the <del> which is a javascript function is included in between name and @ symbol. So for this case the above pattern is suitable which has <del> function included

AFTER:



```
(E:\Anaconda3) E:\fall 2017\NLP\Manjunath\SpamLord\SpamLord>python SpamLord.py data/dev data/devGOLD
True Positives (3):
{('latombe', 'e', 'asandra@cs.stanford.edu'),
 ('latombe', 'e', 'latombe@cs.stanford.edu'),
 ('latombe', 'e', 'liliana@cs.stanford.edu')}
False Positives (0):
set()
False Negatives (114):
{('ashishg', 'e', 'ashishg@stanford.edu'),
 ('ashishg', 'e', 'rozm@stanford.edu'),
 ('ashishg', 'p', '650-723-1614'),
 ('ashishg', 'p', '650-723-4173'),
 ('ashishg', 'p', '650-814-1478'),
```

As we can see in the screenshot above the examples I provided has turned to true positive

And there are 3 true positives, therefore the total number of TP, FP, FN are:

TRUE POSITIVES	3
FALSE POSITIVES	0
FALSE NEGATIVES	114

---

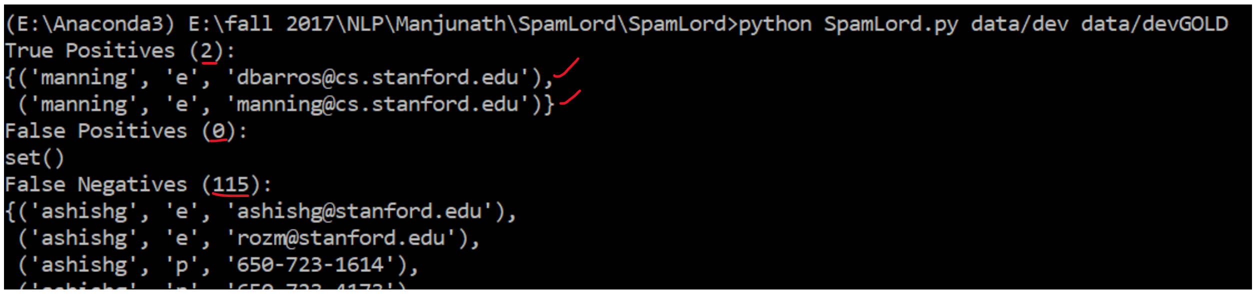
5. epatterns.append('([A-Za-z.]+)\s*<at symbol>\s*([a-z.]+)\.edu')

BEFORE:

INFO	EMAIL ADDRESS
'manning', 'e', 'dbarros@cs.stanford.edu'	manning <at symbol> cs.stanford.edu
'manning', 'e', 'manning@cs.stanford.edu'	dbarros <at symbol> cs.stanford.edu

As seen there is space and <at symbol> used instead of @. The <at symbol> does not carry any meaning. But replace the @. Therefore we can use <at symbol> in our pattern.

AFTER:


```
(E:\Anaconda3) E:\fall 2017\NLP\Manjunath\SpamLord\SpamLord>python SpamLord.py data/dev data/devGOLD
True Positives (2):
{('manning', 'e', 'dbarros@cs.stanford.edu'),
 ('manning', 'e', 'manning@cs.stanford.edu')}
False Positives (0):
set()
False Negatives (115):
{('ashishg', 'e', 'ashishg@stanford.edu'),
 ('ashishg', 'e', 'rozm@stanford.edu'),
 ('ashishg', 'p', '650-723-1614'),
```

As we can see in the screenshot above the examples I provided has turned to true positive

And there are 2 true positivies, therefore the total number of TP, FP, FN are:

TRUE POSITIVES	2
FALSE POSITIVES	0
FALSE NEGATIVES	115

Manjunath Panduranga
SUID: 637425699

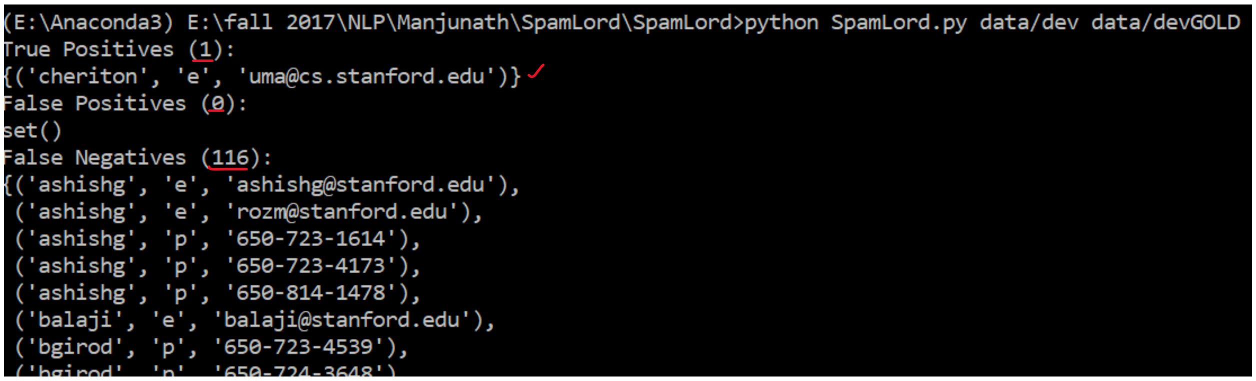
6. epatterns.append('([A-Za-z.]+)@([A-Za-z.]+)\.EDU')

BEFORE:

INFO	EMAIL ADDRESS
'cheriton', 'e', 'uma@cs.stanford.edu'	*uma at cs.Stanford.EDU*

Some email address can be capital as well. For example, the .EDU above is in caps. So in such cases this pattern is very useful, which recognizes the capital case

AFTER:



```
(E:\Anaconda3) E:\fall 2017\NLP\Manjunath\SpamLord\SpamLord>python SpamLord.py data/dev data/devGOLD
True Positives (1):
{('cheriton', 'e', 'uma@cs.stanford.edu')}
False Positives (0):
set()
False Negatives (116):
{('ashishg', 'e', 'ashishg@stanford.edu'),
 ('ashishg', 'e', 'rozm@stanford.edu'),
 ('ashishg', 'p', '650-723-1614'),
 ('ashishg', 'p', '650-723-4173'),
 ('ashishg', 'p', '650-814-1478'),
 ('balaji', 'e', 'balaji@stanford.edu'),
 ('bgirod', 'p', '650-723-4539'),
 ('bgirod', 'p', '650-724-3648'),
```

As we can see in the screenshot above the examples I provided has turned to true positive

And there are 1 true positivies, therefore the total number of TP, FP, FN are:

TRUE POSITIVES	1
FALSE POSITIVES	0
FALSE NEGATIVES	116

Manjunath Panduranga
SUID: 637425699

PHONE PATTERNS:

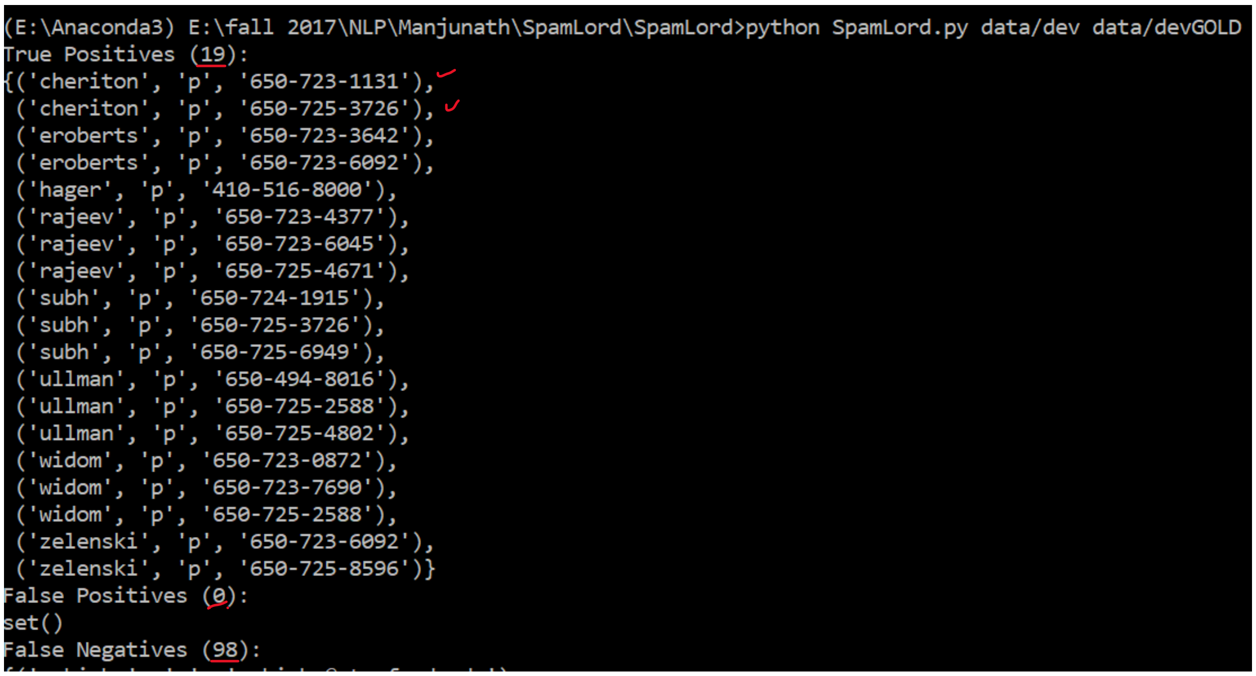
1. ppatterns.append('(\d{3})-(\d{3})-(\d{4})')

BEFORE:

INFO	PHONE NUM
'cheriton', 'p', '650-723-1131'	650-723-1131
'cheriton', 'p', '650-725-3726'	650-725-3726

As seen from the above example, the phone numbers are provided with '-' between every three numbers, starting from beginning. So in such case, the pattern is included with '-'.

AFTER:



```
(E:\Anaconda3) E:\fall 2017\NLP\Manjunath\SpamLord\SpamLord>python SpamLord.py data/dev data/devGOLD
True Positives (19):
{('cheriton', 'p', '650-723-1131'),
 ('cheriton', 'p', '650-725-3726'),
 ('eroberts', 'p', '650-723-3642'),
 ('eroberts', 'p', '650-723-6092'),
 ('hager', 'p', '410-516-8000'),
 ('rajeev', 'p', '650-723-4377'),
 ('rajeev', 'p', '650-723-6045'),
 ('rajeev', 'p', '650-725-4671'),
 ('subh', 'p', '650-724-1915'),
 ('subh', 'p', '650-725-3726'),
 ('subh', 'p', '650-725-6949'),
 ('ullman', 'p', '650-494-8016'),
 ('ullman', 'p', '650-725-2588'),
 ('ullman', 'p', '650-725-4802'),
 ('widom', 'p', '650-723-0872'),
 ('widom', 'p', '650-723-7690'),
 ('widom', 'p', '650-725-2588'),
 ('zelenski', 'p', '650-723-6092'),
 ('zelenski', 'p', '650-725-8596')}
False Positives (0):
set()
False Negatives (98):
```

As we can see in the screenshot above the examples I provided has turned to true positive

And there are 19 true positivies, therefore the total number of TP, FP, FN are:

TRUE POSITIVES	19
FALSE POSITIVES	0
FALSE NEGATIVES	98

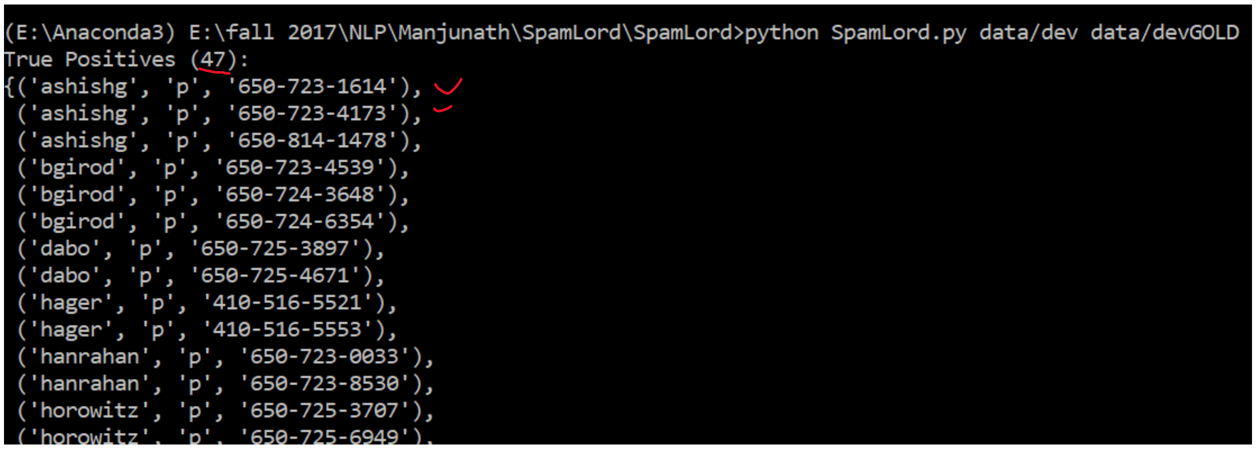
2. ppatterns.append('\s?\((\d{3})\)\s?(\d{3})-(\d{4})')

BEFORE:

INFO	PHONE NUM
('ashishg', 'p', '650-723-1614')	(650)723-1614
('ashishg', 'p', '650-723-4173')	(650)723-4173

As seen from the above example, the phone numbers are provided with braces for the first three numbers since it is area code, and '-' is provided between next three numbers, starting from beginning. So in such case, the pattern is included with braces and '-'.

AFTER:



```
(E:\Anaconda3) E:\fall 2017\NLP\Manjunath\SpamLord\SpamLord>python SpamLord.py data/dev data/devGOLD
True Positives (47):
{('ashishg', 'p', '650-723-1614'),
 ('ashishg', 'p', '650-723-4173'),
 ('ashishg', 'p', '650-814-1478'),
 ('bgirod', 'p', '650-723-4539'),
 ('bgirod', 'p', '650-724-3648'),
 ('bgirod', 'p', '650-724-6354'),
 ('dabo', 'p', '650-725-3897'),
 ('dabo', 'p', '650-725-4671'),
 ('hager', 'p', '410-516-5521'),
 ('hager', 'p', '410-516-5553'),
 ('hanrahan', 'p', '650-723-0033'),
 ('hanrahan', 'p', '650-723-8530'),
 ('horowitz', 'p', '650-725-3707'),
 ('horowitz', 'p', '650-725-6949'),
```

As we can see in the screenshot above the examples I provided has turned to true positive

And there are 47 true positivies, therefore the total number of TP, FP, FN are:

TRUE POSITIVES	47
FALSE POSITIVES	0
FALSE NEGATIVES	70

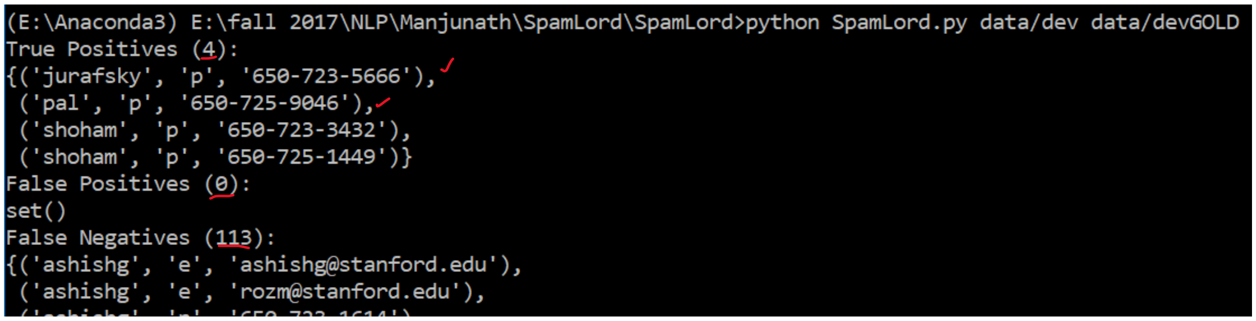
3. ppatterns.append('\s?\+1\s?(\d{3})\s?-?(\d{3})\s?-?(\d{4})')

BEFORE:

INFO	PHONE NUM
'jurafsky', 'p', '650-723-5666'	650 723 5666
'pal', 'p', '650-725-9046'	650 725 9046

AFTER:

As seen from the above example, the phone numbers are provided with spaces between every three numbers, starting from beginning. So in such case, the pattern is included with spaces.



```
(E:\Anaconda3) E:\fall 2017\NLP\Manjunath\SpamLord\SpamLord>python SpamLord.py data/dev data/devGOLD
True Positives (4):
{('jurafsky', 'p', '650-723-5666'),✓
 ('pal', 'p', '650-725-9046'),✓
 ('shoham', 'p', '650-723-3432'),
 ('shoham', 'p', '650-725-1449')}
False Positives (0):
set()
False Negatives (113):
{('ashishg', 'e', 'ashishg@stanford.edu'),
 ('ashishg', 'e', 'rozm@stanford.edu'),
```

As we can see in the screenshot above the examples I provided has turned to true positive

And there are 4 true positives, therefore the total number of TP, FP, FN are:

TRUE POSITIVES	4
FALSE POSITIVES	0
FALSE NEGATIVES	113

4. ppatterns.append('\s?\[(\d{3})\]\s?(\d{3})-(\d{4})')

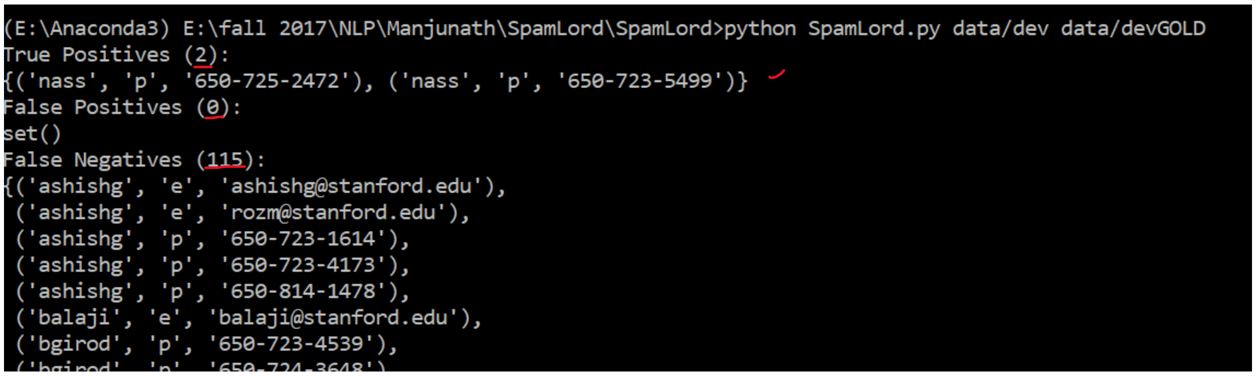
BEFORE:

INFO	PHONE NUM
‘nass', 'p', '650-723-5499'	[650] 723-5499

'nass', 'p', '650-725-2472'	[650] 725-2472

As seen from the above example, the phone numbers are provided with square braces for the first three numbers since it is area code, and '-' is provided between next three numbers, starting from beginning. So in such case, the pattern is included with square braces and '-'.

AFTER:



```
(E:\Anaconda3) E:\fall 2017\NLP\Manjunath\SpamLord\SpamLord>python SpamLord.py data/dev data/devGOLD
True Positives (2):
{('nass', 'p', '650-725-2472'), ('nass', 'p', '650-723-5499')}
False Positives (0):
set()
False Negatives (115):
{('ashishg', 'e', 'ashishg@stanford.edu'),
 ('ashishg', 'e', 'rozm@stanford.edu'),
 ('ashishg', 'p', '650-723-1614'),
 ('ashishg', 'p', '650-723-4173'),
 ('ashishg', 'p', '650-814-1478'),
 ('balaji', 'e', 'balaji@stanford.edu'),
 ('bgirod', 'p', '650-723-4539'),
 ('bgirod', 'p', '650-724-3648'),
```

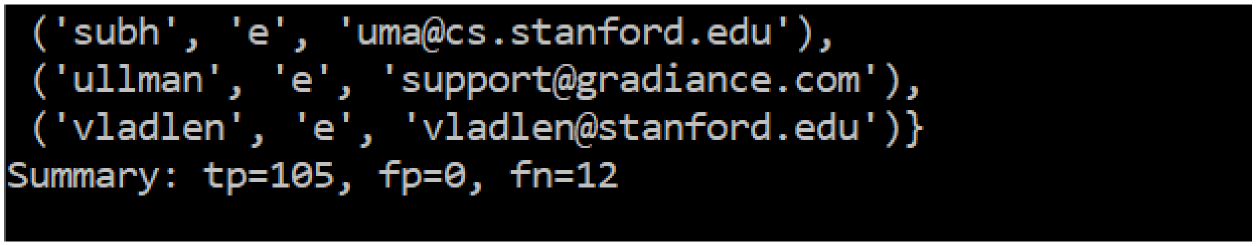
As we can see in the screenshot above the examples I provided has turned to true positive

And there are 2 true positives, therefore the total number of TP, FP, FN are:

TRUE POSITIVES	2
FALSE POSITIVES	0
FALSE NEGATIVES	115

TOTAL:

TP = 105, FP = 0, FN = 12



```
 ('subh', 'e', 'uma@cs.stanford.edu'),
 ('ullman', 'e', 'support@gradiance.com'),
 ('vladlen', 'e', 'vladlen@stanford.edu')}
Summary: tp=105, fp=0, fn=12
```

PART-2

OPTION-2

a) False Negatives: (total 12)

1. ***Obscured email:*** d-l-w-h-@-s-t-a-n-f-o-r-d-.-e-d-u
   Matching email file: dlwh', 'e', 'dlwh@stanford.edu'
   This email is not matched due to the presence of '–' between each letter.
   We need to use string replace function to match this email correctly with the email in the file.

2. ***Obscured email:*** hager at cs dot jhu dot edu
   Matching email file: 'hager', 'e', 'hager@cs.jhu.edu'

   This email is not matched because there is

- no @symbol,
- there are spaces between each word,
- dot is replace by the word dot.

We need three parenthesis to identify or match this email.

This email can be matched if the pattern is too specific to this email address.

3. ***Obscured email:*** jks at robotics;stanford;edu
   Matching email file: 'jks', 'e', 'jks@robotics.stanford.edu'
   This is not matched due to:
   - presence of ';' between robotics and Stanford.
   - spaces between each words.
   Though we can simplify the spaces between each word, We need three parentheses to match this one or we have to use string replace function to convert it and match with the file.

4. ***Obscured email:*** obfuscate('stanford.edu','jurafsky')
   Matching email file: 'jurafsky', 'e', 'jurafsky@stanford.edu'

This is very difficult pattern to match as the email id is written in

- reverse manner and
- without any usage of '@' or 'at' or 'AT' symbols.

5. ***Obscured email:*** lam at cs.stanford.edu
   Matching email file: 'lam', 'e', 'lam@cs.stanford.edu'
   This is not matched due to:
- the spaces in between name and at
- the @ symbol is replaced by at
- though we can solve this pattern, it would result in many false positive patterns. This is again harmful.

6. ***Obscured email:*** ouster (followed by &ldquo;@cs.stanford.edu&rdquo;)
   Matching email file: 'ouster', 'e', 'ouster@cs.stanford.edu'
   The above email is considered to be obscured email address and I was not able to provide a generalized pattern to this obscured email because of
- Braces present in between
- Followed by is used
- &ldquo , &rdquo is present
- Present of Semicolon

   However, we can simplify this by being too specific

7. ***Obscured email:*** teresa.lynn (followed by "@stanford.edu")
   Matching email file: 'ouster', 'e', 'teresa.lynn@stanford.edu'
   The above email is considered to be obscured email address and I was not able to provide a generalized pattern to this obscured email because,
- Braces present in between
- Followed by is used
- Present of Semicolon and ""

   However, we can simplifiy this by being too specific

8. ***Obscured email:*** pal at cs stanford edu
   Matching email file: 'pal@cs.stanford.edu'
   This is not matched due to the
- presence of space " " between cs and
- Stanford instead of a '.'.

   We need three parentheses to match this one or we have to use string replace function to convert it and match with the file.

9. ***Obscured email:*** support at gradiance dt com
   Matching email file: 'ullman', 'e', 'support@gradiance.com'
   This is not matched due to the
- presence of 'com' address (not .edu type) and
- also 'dt' inplace of '.' .

  We need to compose a different for loop using String replace function with .com type to have this matched perfectly.

10. ***Obscured email:*** serafim at cs dot stanford dot edu
    Matching email file: 'serafim', 'e', 'serafim@cs.stanford.edu'
    The above email is considered to be obscured email address and I was not able to provide a generalized pattern to this obscured email because :
- @ and . is replaced by at and dot
- Spaces provided in between them

  We need three parenthesis to identify or match this email.

11. ***Obscured email:*** uma at cs dot stanford dot edu
    Matching email file: 'subh', 'e', 'uma@cs.stanford.edu'
    The above email is considered to be obscured email address and I was not able to provide a generalized pattern to this obscured email.
- @ and . is replaced by at and dot
- Spaces provided in between them

  We need three parenthesis to identify or match this email.

12. ***Obscured email:*** vladlen at <!-- die!--> stanford <!-- spam pigs!--> dot <!-- die!--> edu
    Matching email file: 'vladlen', 'e', 'vladlen@stanford.edu'
    The above email is considered to be obscured email address and I was not able to provide a generalized pattern to this obscured email.
    However, I got a pattern to simplify this email:

```
epatterns.append('([A-Za-z]+)\sat\s\W+\w+\W+([a-z]+)\s[\W\w]+dot[\W\w]+edu')
```

but this pattern will only be specific for this obscured email

Note: I did not find any False positives in Part 1 (FP = 0)

---

b) Obscured email addresses or phone numbers from web

1. RANDOM ARRAY

Split your email address into multiple parts and create an array in JavaScript out of these parts. Next join these parts in the correct order and use the .innerHTML property to add the email address to the web page

```
<span id="email"></span>

<script>
  var parts = ["john", "abc", "com", "&#46;", "&#64;"];
  var email = parts[0] + parts[4] + parts[1] + parts[3] + parts[2];
  document.getElementById("email").innerHTML=email;
</script>
```

2. TURN OFF 'DISPLAY'

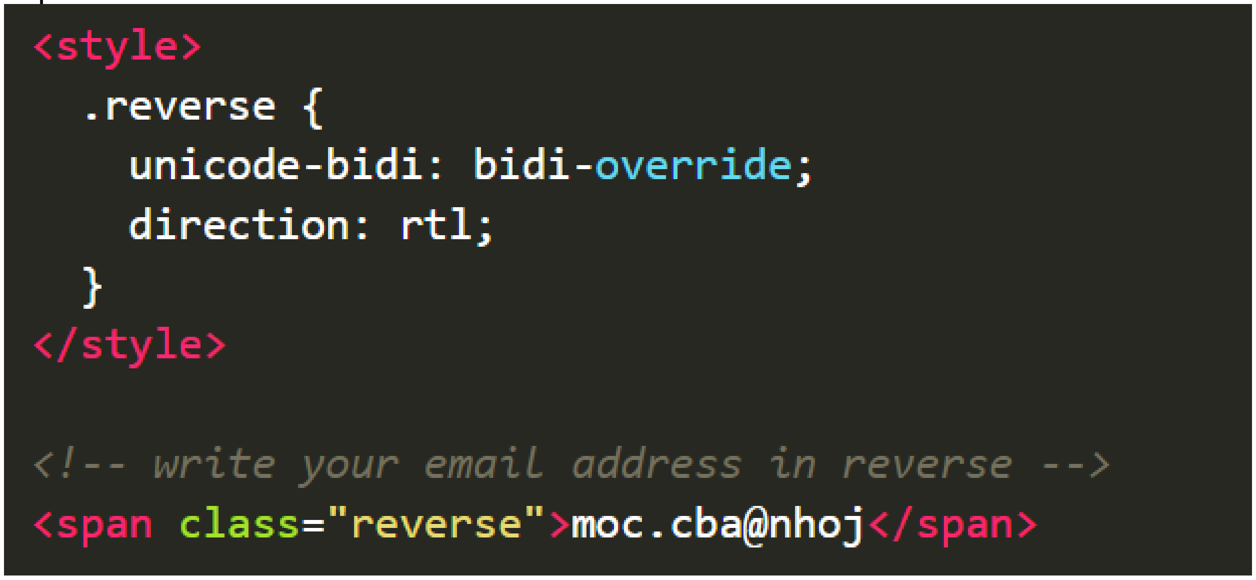
You can add extra characters to your email address to confuse the spam bots and then use the CSS 'display' property to render your actual email address on the screen while hiding all the extra bits

```
<style>
  #dummy {
    display: none;
  }
</style>

<!-- You can add any number of z tags but they'll stay hidden -->
john<span id="dummy">REMOVE</span>@abc<span id="dummy">REMOVE</span>.com
```

3. REVERSE THE DIRECTION

You can write your email address in reverse (john@abc.com as moc.cba@nhoj) and then use the unicode-bidi and direction CSS properties to instruct the browser to display the text in reverse (or correct) direction. The text is selectable but the address would copied in reverse direction.



```
<style>
  .reverse {
    unicode-bidi: bidi-override;
    direction: rtl;
  }
</style>

<!-- write your email address in reverse -->
<span class="reverse">moc.cba@nhoj</span>
```