# SpamLord! Using Regular Expressions to find email addresses and phone numbers

**Part 1-**
Write more regular expressions that correct false positives or fit false negatives. 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. Give the text that matches, i.e. the obfuscated example, not only the results! When you are done with as many as you can do, give the output of the program.

I started with **ppatterns list**-

1.) **ppatterns.append('(\d{3})-(\d{3})-(\d{4})')**
This pattern matches phone no. divided in 3 parts and separated by '-' where first part has 3 digits, second part also has 3 digits and third part has 4 digits.

With this pattern I was able to match **19** phone no such as:

| **True Positives (19):** | **Obfuscated version:** |
|---|---|
| {('cheriton', 'p', '650-723-1131'), | '650-723-1131' |
| ('cheriton', 'p', '650-725-3726'), | '650-725-3726' |
| ('eroberts', 'p', '650-723-3642'), | '650-723-3642' |
| ('eroberts', 'p', '650-723-6092'), | '650-723-6092' |
| ('hager', 'p', '410-516-8000'), | '410-516-8000' |
| ('rajeev', 'p', '650-723-4377'), | '650-723-4377' |
| ('rajeev', 'p', '650-723-6045'), | '650-723-6045' |
| ('rajeev', 'p', '650-725-4671'), | '650-725-4671' |
| ('subh', 'p', '650-724-1915'), | '650-724-1915' |
| ('subh', 'p', '650-725-3726'), | '650-725-3726' |
| ('subh', 'p', '650-725-6949'), | '650-725-6949' |
| ('ullman', 'p', '650-494-8016'), | '650-494-8016' |
| ('ullman', 'p', '650-725-2588'), | '650-725-2588' |
| ('ullman', 'p', '650-725-4802'), | '650-725-4802' |
| ('widom', 'p', '650-723-0872'), | '650-723-0872' |
| ('widom', 'p', '650-723-7690'), | '650-723-7690' |
| ('widom', 'p', '650-725-2588'), | '650-725-2588' |
| ('zelenski', 'p', '650-723-6092'), | '650-723-6092' |
| ('zelenski', 'p', '650-725-8596')} | '650-725-8596' |

2.) **ppatterns.append('(\d{3})[- ](\d{3})[- ](\d{4})')**
This pattern matches phone no. divided in 3 parts and separated by '- ' or by whitespace where first part has 3 digits, second part also has 3 digits and third part has 4 digits.

With this pattern I was able to match **23** phone no such as:

| **True Positives (23):** | **Obfuscated version:** |
|---|---|
| {('cheriton', 'p', '650-723-1131'), | *'650-723-1131'* |
| ('cheriton', 'p', '650-725-3726'), | *'650-725-3726'* |

| | |
|---|---|
| ('eroberts', 'p', '650-723-3642'), | *'650-723-3642'* |
| ('eroberts', 'p', '650-723-6092'), | *'650-723-6092'* |
| ('hager', 'p', '410-516-8000'), | *'410-516-8000'* |
| ('jurafsky', 'p', '650-723-5666'), | ‘650 723 5666' |
| ('pal', 'p', '650-725-9046'), | ‘650 725 9046’ |
| ('rajeev', 'p', '650-723-4377'), | *'650-723-4377'* |
| ('rajeev', 'p', '650-723-6045'), | *'650-723-6045'* |
| ('rajeev', 'p', '650-725-4671'), | *'650-725-4671'* |
| ('shoham', 'p', '650-723-3432'), | **‘650 723-3432’** |
| ('shoham', 'p', '650-725-1449'), | **‘650 725-1449’** |
| ('subh', 'p', '650-724-1915'), | *'650-724-1915'* |
| ('subh', 'p', '650-725-3726'), | *'650-725-3726'* |
| ('subh', 'p', '650-725-6949'), | *'650-725-6949'* |
| ('ullman', 'p', '650-494-8016'), | *'650-494-8016'* |
| ('ullman', 'p', '650-725-2588'), | *'650-725-2588'* |
| ('ullman', 'p', '650-725-4802'), | *'650-725-4802'* |
| ('widom', 'p', '650-723-0872'), | *'650-723-0872'* |
| ('widom', 'p', '650-723-7690'), | *'650-723-7690'* |
| ('widom', 'p', '650-725-2588'), | *'650-725-2588'* |
| ('zelenski', 'p', '650-723-6092'), | *'650-723-6092'* |
| ('zelenski', 'p', '650-725-8596')} | *'650-725-8596’* |

Note- Clearly this pattern matches all phone no in previous list and some additional no as well.

Since use of parenthesis and square bracket in area code part of phone no, is very common in North America, I tried these three patterns-

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

This pattern matches phone no. divided in 3 parts where first two part may be separated by whitespace (not mandatory) and second-third part are separated by ‘-‘ Also, the first part in this pattern must be in parenthesis().

With this pattern I was able to match **47** phone no such as:

| **True Positives (47):** | **Obfuscated version:** |
|---|---|
| {('ashishg', 'p', '650-723-1614'), | **'(650)723-1614'** |
| ('ashishg', 'p', '650-723-4173'), | **'(650)723-4173'** |
| ('ashishg', 'p', '650-814-1478'), | **'(650)814-1478'** |
| ('bgirod', 'p', '650-723-4539'), | '(650) 723-4539' |
| ('bgirod', 'p', '650-724-3648'), | '(650) 724-3648' |
| ('bgirod', 'p', '650-724-6354'), | '(650) 724-6354' |
| ('dabo', 'p', '650-725-3897'), | '(650) 725-3897' |
| ('dabo', 'p', '650-725-4671'), | '(650) 725-4671' |
| ('hager', 'p', '410-516-5521'), | '(410) 516-5521' |

| | |
|---|---|
| ('hager', 'p', '410-516-5553'), | '(410) 516-5553' |
| ('hanrahan', 'p', '650-723-0033'), | '(650) 723-0033' |
| ('hanrahan', 'p', '650-723-8530'), | '(650) 723-8530' |
| ('horowitz', 'p', '650-725-3707'), | **'(650)725-3707'** |
| ('horowitz', 'p', '650-725-6949'), | **'(650)725-6949'** |
| ('kosecka', 'p', '703-993-1710'), | '(703) 993-1710' |
| ('kosecka', 'p', '703-993-1876'), | '(703) 993-1876' |
| ('kunle', 'p', '650-723-1430'), | '(650) 723-1430' |
| ('kunle', 'p', '650-725-3713'), | '(650) 725-3713' |
| ('kunle', 'p', '650-725-6949'), | '(650) 725-6949' |
| ('lam', 'p', '650-725-3714'), | '(650) 725-3714' |
| ('lam', 'p', '650-725-6949'), | '(650) 725-6949' |
| ('latombe', 'p', '650-721-6625'), | '(650) 721-6625' |
| ('latombe', 'p', '650-723-0350'), | '(650) 723-0350' |
| ('latombe', 'p', '650-723-4137'), | '(650) 723-4137' |
| ('latombe', 'p', '650-725-1449'), | '(650) 725-1449’ |
| ('levoy', 'p', '650-723-0033'), | '(650) 723-0033' |
| ('levoy', 'p', '650-724-6865'), | '(650) 724-6865’ |
| ('levoy', 'p', '650-725-3724'), | '(650) 725-3724' |
| ('levoy', 'p', '650-725-4089'), | '(650) 725-4089' |
| ('manning', 'p', '650-723-7683'), | '(650) 723-7683' |
| ('manning', 'p', '650-725-1449'), | '(650) 725-1449' |
| ('manning', 'p', '650-725-3358'), | '(650) 725-3358' |
| ('nick', 'p', '650-725-4727'), | '(650) 725-4727' |
| ('ok', 'p', '650-723-9753'), | '(650) 723-9753' |
| ('ok', 'p', '650-725-1449'), | '(650) 725-1449' |
| ('rinard', 'p', '617-253-1221'), | '(617) 253-1221' |
| ('rinard', 'p', '617-258-6922'), | '(617) 258-6922' |
| ('serafim', 'p', '650-723-3334'), | '(650) 723-3334' |
| ('serafim', 'p', '650-725-1449'), | '(650) 725-1449' |
| ('thm', 'p', '650-725-3383'), | '(650) 725-3383' |
| ('thm', 'p', '650-725-3636'), | '(650) 725-3636' |
| ('thm', 'p', '650-725-3938'), | '(650) 725-3938' |
| ('tim', 'p', '650-724-9147'), | '(650)724-9147' |
| ('tim', 'p', '650-725-2340'), | **'(650)725-2340'** |
| ('tim', 'p', '650-725-4671'), | **'(650)725-4671'** |
| ('zm', 'p', '650-723-4364'), | '(650) 723-4364' |
| ('zm', 'p', '650-725-4671')} | '(650) 725-4671' |

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

This pattern matches phone no. divided in 3 parts where second-third part are separated by ‘-‘ Also, the first part in this pattern must be in parenthesis().

With this pattern I was able to match **8** phone no such as:

| **True Positives (8):** | **Obfuscated version:** |
|---|---|
| {('ashishg', 'p', '650-723-1614'), | '(650)723-1614' |
| ('ashishg', 'p', '650-723-4173'), | '(650)723-4173' |
| ('ashishg', 'p', '650-814-1478'), | '(650)814-1478' |
| ('horowitz', 'p', '650-725-3707'), | '(650)725-3707' |
| ('horowitz', 'p', '650-725-6949'), | '(650)725-6949' |
| ('tim', 'p', '650-724-9147'), | ‘(650)724-9147 |
| ('tim', 'p', '650-725-2340'), | '(650)725-2340' |
| ('tim', 'p', '650-725-4671')} | '(650)725-4671' |

**Note- Clearly all phone no found by this pattern are already present in 3rd pattern**

5.) **ppatterns.append('\[(\d{3})\][ ]*(\d{3})-(\d{4})')**
This pattern matches phone no. divided in 3 parts where first two part may be separated by whitespaces (not mandatory) second-third part are separated by ‘-‘ Also, the first part in this pattern must be in square bracket[].

With this pattern I was able to match **2** phone no such as:

| **True Positives (2):** | **Obfuscated version:** |
|---|---|
| {('nass', 'p', '650-723-5499'), | ‘[650] 723-5499’ |
| ('nass', 'p', '650-725-2472')} | ‘[650] 725-2472’ |

**All phone no present in devGOLD list are found by combination of these three patterns:**

| | |
|---|---|
| ppatterns.append('(\d{3})[- ](\d{3})[- ](\d{4})') | -matches 23 phone no. |
| ppatterns.append('\((\d{3})\)[ ]*(\d{3})-(\d{4})') | -matches 47 phone no. |
| ppatterns.append('\[(\d{3})\][ ]*(\d{3})-(\d{4})') | -matches remaining 2 phone no. |
| **Total** | **- 72 phone no.** |

**Output-**
**True Positives (72):**

```
{('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'),
 ('cheriton', 'p', '650-723-1131'),
 ('cheriton', 'p', '650-725-3726'),
 ('dabo', 'p', '650-725-3897'),
 ('dabo', 'p', '650-725-4671'),
 ('eroberts', 'p', '650-723-3642'),
 ('eroberts', 'p', '650-723-6092'),
 ('hager', 'p', '410-516-5521'),
 ('hager', 'p', '410-516-5553'),
 ('hager', 'p', '410-516-8000'),
 ('hanrahan', 'p', '650-723-0033'),
 ('hanrahan', 'p', '650-723-8530'),
 ('horowitz', 'p', '650-725-3707'),
 ('horowitz', 'p', '650-725-6949'),
 ('jurafsky', 'p', '650-723-5666'),
 ('kosecka', 'p', '703-993-1710'),
 ('kosecka', 'p', '703-993-1876'),
 ('kunle', 'p', '650-723-1430'),
 ('kunle', 'p', '650-725-3713'),
 ('kunle', 'p', '650-725-6949'),
 ('lam', 'p', '650-725-3714'),
 ('lam', 'p', '650-725-6949'),
 ('latombe', 'p', '650-721-6625'),
 ('latombe', 'p', '650-723-0350'),
 ('latombe', 'p', '650-723-4137'),
 ('latombe', 'p', '650-725-1449'),
 ('levoy', 'p', '650-723-0033'),
 ('levoy', 'p', '650-724-6865'),
 ('levoy', 'p', '650-725-3724'),
 ('levoy', 'p', '650-725-4089'),
 ('manning', 'p', '650-723-7683'),
 ('manning', 'p', '650-725-1449'),
 ('manning', 'p', '650-725-3358'),
 ('nass', 'p', '650-723-5499'),
 ('nass', 'p', '650-725-2472'),
 ('nick', 'p', '650-725-4727'),
 ('ok', 'p', '650-723-9753'),
 ('ok', 'p', '650-725-1449'),
 ('pal', 'p', '650-725-9046'),
 ('rajeev', 'p', '650-723-4377'),
 ('rajeev', 'p', '650-723-6045'),
 ('rajeev', 'p', '650-725-4671'),
 ('rinard', 'p', '617-253-1221'),
 ('rinard', 'p', '617-258-6922'),
 ('serafim', 'p', '650-723-3334'),
 ('serafim', 'p', '650-725-1449'),
 ('shoham', 'p', '650-723-3432'),
 ('shoham', 'p', '650-725-1449'),
 ('subh', 'p', '650-724-1915'),
 ('subh', 'p', '650-725-3726'),
 ('subh', 'p', '650-725-6949'),
 ('thm', 'p', '650-725-3383'),
 ('thm', 'p', '650-725-3636'),
  ('thm', 'p', '650-725-3938'),
  ('tim', 'p', '650-724-9147'),
  ('tim', 'p', '650-725-2340'),
  ('tim', 'p', '650-725-4671'),
  ('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'),
  ('zm', 'p', '650-723-4364'),
  ('zm', 'p', '650-725-4671')}
```

**False Positives (0):**

```
set()
```

**False Negatives (45):**

```
{('ashishg', 'e', 'ashishg@stanford.edu'),
 ('ashishg', 'e', 'rozm@stanford.edu'),
 ('balaji', 'e', 'balaji@stanford.edu'),
 ('cheriton', 'e', 'cheriton@cs.stanford.edu'),
 ('cheriton', 'e', 'uma@cs.stanford.edu'),
 ('dabo', 'e', 'dabo@cs.stanford.edu'),
 ('dlwh', 'e', 'dlwh@stanford.edu'),
 ('engler', 'e', 'engler@lcs.mit.edu'),
 ('engler', 'e', 'engler@stanford.edu'),
 ('eroberts', 'e', 'eroberts@cs.stanford.edu'),
 ('fedkiw', 'e', 'fedkiw@cs.stanford.edu'),
 ('hager', 'e', 'hager@cs.jhu.edu'),
 ('hanrahan', 'e', 'hanrahan@cs.stanford.edu'),
 ('jks', 'e', 'jks@robotics.stanford.edu'),
 ('jurafsky', 'e', 'jurafsky@stanford.edu'),
 ('kosecka', 'e', 'kosecka@cs.gmu.edu'),
 ('kunle', 'e', 'darlene@csl.stanford.edu'),
 ('kunle', 'e', 'kunle@ogun.stanford.edu'),
 ('lam', 'e', 'lam@cs.stanford.edu'),
 ('latombe', 'e', 'asandra@cs.stanford.edu'),
 ('latombe', 'e', 'latombe@cs.stanford.edu'),
 ('latombe', 'e', 'liliana@cs.stanford.edu'),
 ('levoy', 'e', 'ada@graphics.stanford.edu'),
 ('levoy', 'e', 'melissa@graphics.stanford.edu'),
 ('manning', 'e', 'dbarros@cs.stanford.edu'),
 ('manning', 'e', 'manning@cs.stanford.edu'),
 ('nass', 'e', 'nass@stanford.edu'),
('nick', 'e', 'nick.parlante@cs.stanford.edu'),
('ouster', 'e', 'ouster@cs.stanford.edu'),
('ouster', 'e', 'teresa.lynn@stanford.edu'),
('pal', 'e', 'pal@cs.stanford.edu'),
('psyoung', 'e', 'patrick.young@stanford.edu'),
('rinard', 'e', 'rinard@lcs.mit.edu'),
('serafim', 'e', 'serafim@cs.stanford.edu'),
('shoham', 'e', 'shoham@stanford.edu'),
('subh', 'e', 'subh@stanford.edu'),
('subh', 'e', 'uma@cs.stanford.edu'),
('thm', 'e', 'pkrokel@stanford.edu'),
('ullman', 'e', 'support@gradiance.com'),
('ullman', 'e', 'ullman@cs.stanford.edu'),
('vladlen', 'e', 'vladlen@stanford.edu'),
('widom', 'e', 'siroker@cs.stanford.edu'),
('widom', 'e', 'widom@cs.stanford.edu'),
('zelenski', 'e', 'zelenski@cs.stanford.edu'),
('zm', 'e', 'manna@cs.stanford.edu')}
```
**Summary: tp=72, fp=0, fn=45**

**epattern list-**
**I have divided these patterns into two categories-**
**One which I could guess based on general knowledge (without documentation)**

1.) **epatterns.append('([A-Za-z0-9._]+)\s*@\s*([A-Za-z0-9._]+)\.edu')**
This pattern matches email divided in 2 parts where these two part are separated by '@' and whitespaces (before and after '@' although not mandatory) Also email ends with '.edu'string. Each part can have alpha-numeric characters with special characters like '. ' and ' _'

With this pattern I was able to match **23** emails such as:

| **True Positives (23):** | **Obfuscated version:** |
|---|---|
| {('ashishg', 'e', 'ashishg@stanford.edu'), | **'ashishg @ stanford.edu'** |
| ('ashishg', 'e', 'rozm@stanford.edu'), | **'rozm @ stanford.edu'** |
| ('balaji', 'e', 'balaji@stanford.edu'), | 'balaji@stanford.edu' |
| ('cheriton','e', 'cheriton@cs.stanford.edu'), | 'cheriton@cs.stanford.edu' |
| ('dabo', 'e', 'dabo@cs.stanford.edu'), | **'dabo @ cs.stanford.edu'** |
| ('engler', 'e', 'engler@lcs.mit.edu'), | 'engler@lcs.mit.edu' |
| ('eroberts','e','eroberts@cs.stanford.edu'), | 'eroberts@cs.stanford.edu' |
| ('fedkiw', 'e', 'fedkiw@cs.stanford.edu'), | 'fedkiw@cs.stanford.edu' |
| ('hanrahan','e','hanrahan@cs.stanford.edu'), | 'hanrahan@cs.stanford.edu' |
| ('kosecka', 'e', 'kosecka@cs.gmu.edu'), | 'kosecka@cs.gmu.edu' |
| ('kunle', 'e', 'darlene@csl.stanford.edu'), | 'darlene@csl.stanford.edu' |
| ('kunle', 'e', 'kunle@ogun.stanford.edu'), | 'kunle@ogun.stanford.edu' |
| ('nass', 'e', 'nass@stanford.edu'), | 'nass@stanford.edu' |
| ('nick', 'e', 'nick.parlante@cs.stanford.edu'), | 'nick.parlante@cs.stanford.edu' |
| ('psyoung','e','patrick.young@stanford.edu'), | 'patrick.young@stanford.edu' |

| | |
|---|---|
| ('rinard', 'e', 'rinard@lcs.mit.edu'), | 'rinard@lcs.mit.edu' |
| ('shoham', 'e', 'shoham@stanford.edu'), | 'shoham@stanford.edu' |
| ('thm', 'e', 'pkrokel@stanford.edu'), | 'pkrokel@Stanford.edu' |
| ('ullman', 'e', 'ullman@cs.stanford.edu'), | **'ullman @ cs.stanford.edu'** |
| ('widom', 'e', 'siroker@cs.stanford.edu'), | 'siroker@cs.stanford.edu' |
| ('widom', 'e', 'widom@cs.stanford.edu'), | 'widom@cs.stanford.edu' |
| ('zelenski', 'e', 'zelenski@cs.stanford.edu'), | 'zelenski@cs.stanford.edu' |
| ('zm', 'e', 'manna@cs.stanford.edu')} | 'manna@cs.stanford.edu' |

If I think like an anti-spammer, I can change only common parts of email such as '@' '.' 'com' to some other not easily guessable forms while at the same time not modifying domain and userid part of email.
**@ can be changed to - at, AT, at the rate, whitespaces, where, WHERE**
**. can be changed to dot, DOT, dt , DT, whitespaces, domain, dm, DM, DOM, dom**
**edu can be changed to edu,(edu), (EDU)**

Since I am not allowed to use more than two parenthesis so I cannoit use a pattern like-
**epatterns.append('([A-Za-z0-9._]+)\s*(**at|AT|WHERE|@+at\s*the\s*rate\s*)\s*([A-Za-z0-9._]+)\s*(dom|DOM|.|dot|DOT|dt|DT|dm|domain|DM)\s*(edu|EDU|\(edu\)|.)**')**

**So I tried different permutation and combinations of each-@ . edu and I have reported some of them (which gave at least one true positive).**

**2.) epatterns.append('([A-Za-z0-9._]+)\s*at\s*([A-Za-z0-9._]+)\.EDU')**
This pattern matches email divided in 2 parts where these two part are separated by 'at' and whitespaces(before and after 'at' although not mandatory) Also email ends with '.EDU'string. Each part can have alpha-numeric characters with special characters like '. ' and ' _'

With this pattern I was able to match emails such as:

| **True Positives :** | **Obfuscated version:** |
|---|---|
| {('cheriton', 'e', 'uma@cs.stanford.edu')} | 'cheriton at cs.stanford.edu' |

**3.) epatterns.append('([A-Za-z0-9._]+)\s*AT\s*([A-Za-z0-9._]+)\s*DOT\s*edu')**
This pattern matches email divided in 2 parts where these two part are separated by 'AT' and whitespaces (before and after 'AT' although not mandatory) Also email ends with 'edu'string. Also this pattern will have mandatory 'DOT' string before 'edu' and may be separated by whitespaces(not mandatory). Each part can have alpha-numeric characters with special characters like '. ' and ' _'

With this pattern I was able to match emails such as:

| **True Positives :** | **Obfuscated version:** |
|---|---|
| {('subh', 'e', 'subh@stanford.edu')} | 'subh AT stanford DOT edu' |

4.) **epatterns.append('([A-Za-z0-9._-]+)\s*WHERE\s*([A-Za-z0-9._]+)\s*DOM\s*edu')**
This pattern matches email divided in 2 parts where these two parts are separated by 'WHERE' and whitespaces (before and after 'WHERE' although not mandatory) Also email ends with 'edu'string. Also this pattern will have mandatory 'DOT' string before 'edu' and may be separated by whitespaces (not mandatory). Each part can have alpha-numeric characters with special characters like '. ' and ' _'

With this pattern I was able to match emails such as:

| **True Positives :** | **Obfuscated version:** |
|---|---|
| {('engler', 'e', 'engler@stanford.edu') | 'engler WHERE stanford DOM edu' |
| ('cheriton', 'e', 'cheriton@cs.stanford.edu')} | 'cheriton at cs.stanford.edu' |

This pattern also generated two false positives:
{('jure', 'e', 'server@cs.stanford.edu'),
('plotkin', 'e', 'server@infolab.stanford.edu')}

**On examining the obfuscated version of these, I found that server was like a stop word in this document which typicalls states url in words**
**<address>Apache/2.2.4 (Fedora) Server at cs.stanford.edu Port 80</address>**
**<address>Apache/2.0.54 (Fedora) Server at infolab.stanford.edu Port 80</address>**

**I have provided a solution (using python programming) in part 2 for this problem by ignoring email in process_file function that matches with a pattern but have userid part as server**

**Note-** Before coming to above coclusion, I tried following patterns-
([A-Za-z0-9._]+)\s*at\s*([A-Za-z0-9._]+)\.edu
This generated-
True Positives (2):
{('cheriton', 'e', 'cheriton@cs.stanford.edu'),
('lam', 'e', 'lam@cs.stanford.edu')}

False Positives (7):
{('aiken', 'e', 's@urn.stanford.edu'),
('fedkiw', 'e', 'www.m@h.ucla.edu'),
('jure', 'e', 'server@cs.stanford.edu'),
('lam', 'e', 'cleansl@e.stanford.edu'),
('nass', 'e', 'communic@ion.stanford.edu'),
('pal', 'e', 'd@a.cs.washington.edu'),
('plotkin', 'e', 'server@infolab.stanford.edu')}

In order to reduce false negatives, I changed the minimum word length to 2 as email will have more than one character in userid. This reduced the false positives to 5
False Positives (5):

{('fedkiw', 'e', 'www.m@h.ucla.edu'),
('jure', 'e', 'server@cs.stanford.edu'),
('lam', 'e', 'cleansl@e.stanford.edu'),
('nass', 'e', 'communic@ion.stanford.edu'),
('plotkin', 'e', 'server@infolab.stanford.edu')}

On further investigation I found that ‘at’ must have minimum one whitespace on both of its side else words like math, S**at**urn, communic**at**ion, d**at**a will also be matched.
Then I came to the current version of this pattern.

5.) **epatterns.append('([A-Za-z0-9._]+)\s*at\s*([A-Za-z0-9._]+)\s*dt\s*com')**
This pattern matches email divided in 2 parts where these two parts are separated by ‘at’ and whitespaces (before and after ‘at’ although not mandatory) Also email ends with ‘com’string. Also this pattern will have mandatory ‘dt’ string before ‘com’ and may be separated by whitespaces (not mandatory). Each part can have alpha-numeric characters with special characters like ‘. ‘ and ‘ _’

**This pattern generated false positive as support@gradiance.edu . On checking the corresponding file indicated by interpretor , I found that actual output should have been support@gradiance.edu. Although the pattern(support@gradiance) was correct but the SpamLord program is written in such way that for every email match, it concatenated ‘.edu’ at the end whereas actual email had ‘.com’**
**I have provided a solution (using python programming) in part 2 for this problem by editing process_file in SpamLord.py**

6.) **epatterns.append('([A-Za-z0-9._]({2,})\s+at\s+([A-Za-z0-9._]+)\.edu')**
This pattern matches email divided in 2 parts where these two part are separated by ‘at’ and at least one whitespaces (before and after ‘at’ ) Also email ends with ‘.edu’string. Each part can have alpha-numeric characters with special characters like ‘. ‘ and ‘ _’

With this pattern I was able to match emails such as:

| **True Positives (1):** | **Obfuscated version:** |
|---|---|
| {('lam', 'e', 'lam@cs.stanford.edu')} | ‘E-mail: lam at cs.stanford.edu’ |

**Since we are detecting emails which would be embedded in some web technologies such as html. It is very likely that some kind of tags like <p> </>would have been used to obfuscate or even html comments like** <!-- -->

A tag can be before @ or after @ or before . or after . or combination of this. I tried different combinations as a pattern like below won’t work due to more than three parenthesis-
epatterns.append('([A-Za-z0-9._]+)\s*([<][A-Za-z0-9._]+[>])*\s*@\s*([<][A-Za-z0-9._]+[>])*\s*([A-Za-z0-9._]+)\s*([<][A-Za-z0-9._]+[>])*\s*\.\s*([<][A-Za-z0-9._]+[>])*\s*edu')

One which gave atleast one true positive:

**7.) epatterns.append('([A-Za-z0-9._]+)\s*[<][A-Za-z0-9._]+[>]\s*@\s*([A-Za-z0-9._]+)\.edu')**

This pattern matches email divided in 2 parts where these two parts are separated by ‘@’ and whitespaces (before and after ‘@’ although not mandatory) Also email ends with ‘.edu’ string. Each part can have alpha-numeric characters with special characters like ‘. ‘ and ‘ _’ We have tag starting with angular bracket ‘<‘ followed by some alphanumeric text and ending with ‘>’ in between first part and ‘@’

With this pattern I was able to match emails such as:

| **True Positives (3):** | **Obfuscated version:** |
|---|---|
| {('latombe', 'e', 'asandra@cs.stanford.edu'), | ‘asandra<del>@cs.stanford.edu’ |
| ('latombe', 'e', 'latombe@cs.stanford.edu'), | ‘latombe<del>@cs.stanford.edu’ |
| ('latombe', 'e', 'liliana@cs.stanford.edu')} | ‘liliana<del>@cs.stanford.edu’ |

For html comments, I tried above pattern with inclusion of ‘\s*’ for whitespaces ‘!’ ‘-‘. It did not work for any combination of html comments with @ . edu so I tried different combination of @ . edu with combination of html comments.

One which gave atleast one true positive:

**8.) epatterns.append('([A-Za-z0-9._]+)\s*at\s*[<][A-Za-z0-9.!\s_-]+[>]\s*([A-Za-z0-9._]+)\s*[<][A-Za-z0-9.!\s_-]+[>]\s*dot\s*[<][A-Za-z0-9.!\s_-]+[>]\s*edu')**

This pattern matches email divided in 2 parts where these two parts are separated by ‘at’ and whitespaces (before and after ‘@’ although not mandatory) Also email ends with ‘edu’ string. Each part can have alpha-numeric characters with special characters like ‘. ‘ and ‘ _’ We have html comment starting with angular bracket ‘<‘ followed by some alphanumeric text ‘!’ ‘-‘ and ending with ‘>’ in between first part and ‘at’ , after ‘at’, before ‘dot’ and after ‘dot'

With this pattern I was able to match emails such as:

| **True Positives (1):** | **Obfuscated version:** |
|---|---|
| {('vladlen', 'e', 'vladlen@stanford.edu')} | vladlen at <!-- die!--> stanford <!-- spam pigs!--> dot <!-- die!--> edu |

**Till now I was able to detect 31 emails out of 45**

```
C:\Users\HP\Desktop\NLP\Class 8\SpamLord\SpamLord>python SpamLord.py data/dev
data/devGOLD
True Positives (103):
{('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'),
 ('bgirod', 'p', '650-724-6354'),
 ('cheriton', 'e', 'cheriton@cs.stanford.edu'),
 ('cheriton', 'e', 'uma@cs.stanford.edu'),
 ('cheriton', 'p', '650-723-1131'),
 ('cheriton', 'p', '650-725-3726'),
 ('dabo', 'e', 'dabo@cs.stanford.edu'),
 ('dabo', 'p', '650-725-3897'),
 ('dabo', 'p', '650-725-4671'),
 ('engler', 'e', 'engler@lcs.mit.edu'),
 ('engler', 'e', 'engler@stanford.edu'),
 ('eroberts', 'e', 'eroberts@cs.stanford.edu'),
 ('eroberts', 'p', '650-723-3642'),
 ('eroberts', 'p', '650-723-6092'),
 ('fedkiw', 'e', 'fedkiw@cs.stanford.edu'),
 ('hager', 'p', '410-516-5521'),
 ('hager', 'p', '410-516-5553'),
 ('hager', 'p', '410-516-8000'),
 ('hanrahan', 'e', 'hanrahan@cs.stanford.edu'),
 ('hanrahan', 'p', '650-723-0033'),
 ('hanrahan', 'p', '650-723-8530'),
 ('horowitz', 'p', '650-725-3707'),
 ('horowitz', 'p', '650-725-6949'),
 ('jurafsky', 'p', '650-723-5666'),
 ('kosecka', 'e', 'kosecka@cs.gmu.edu'),
 ('kosecka', 'p', '703-993-1710'),
 ('kosecka', 'p', '703-993-1876'),
 ('kunle', 'e', 'darlene@csl.stanford.edu'),
 ('kunle', 'e', 'kunle@ogun.stanford.edu'),
 ('kunle', 'p', '650-723-1430'),
 ('kunle', 'p', '650-725-3713'),
 ('kunle', 'p', '650-725-6949'),
 ('lam', 'e', 'lam@cs.stanford.edu'),
 ('lam', 'p', '650-725-3714'),
 ('lam', 'p', '650-725-6949'),
 ('latombe', 'e', 'asandra@cs.stanford.edu'),
 ('latombe', 'e', 'latombe@cs.stanford.edu'),
 ('latombe', 'e', 'liliana@cs.stanford.edu'),
 ('latombe', 'p', '650-721-6625'),
 ('latombe', 'p', '650-723-0350'),
 ('latombe', 'p', '650-723-4137'),
 ('latombe', 'p', '650-725-1449'),
 ('levoy', 'p', '650-723-0033'),
 ('levoy', 'p', '650-724-6865'),
 ('levoy', 'p', '650-725-3724'),
 ('levoy', 'p', '650-725-4089'),
 ('manning', 'p', '650-723-7683'),
 ('manning', 'p', '650-725-1449'),
 ('manning', 'p', '650-725-3358'),
 ('nass', 'e', 'nass@stanford.edu'),
 ('nass', 'p', '650-723-5499'),
 ('nass', 'p', '650-725-2472'),
 ('nick', 'e', 'nick.parlante@cs.stanford.edu'),
 ('nick', 'p', '650-725-4727'),
 ('ok', 'p', '650-723-9753'),
 ('ok', 'p', '650-725-1449'),
 ('pal', 'p', '650-725-9046'),
 ('psyoung', 'e', 'patrick.young@stanford.edu'),
 ('rajeev', 'p', '650-723-4377'),
 ('rajeev', 'p', '650-723-6045'),
 ('rajeev', 'p', '650-725-4671'),
 ('rinard', 'e', 'rinard@lcs.mit.edu'),
 ('rinard', 'p', '617-253-1221'),
 ('rinard', 'p', '617-258-6922'),
 ('serafim', 'p', '650-723-3334'),
 ('serafim', 'p', '650-725-1449'),
 ('shoham', 'e', 'shoham@stanford.edu'),
 ('shoham', 'p', '650-723-3432'),
 ('shoham', 'p', '650-725-1449'),
 ('subh', 'e', 'subh@stanford.edu'),
 ('subh', 'p', '650-724-1915'),
 ('subh', 'p', '650-725-3726'),
 ('subh', 'p', '650-725-6949'),
 ('thm', 'e', 'pkrokel@stanford.edu'),
 ('thm', 'p', '650-725-3383'),
 ('thm', 'p', '650-725-3636'),
 ('thm', 'p', '650-725-3938'),
 ('tim', 'p', '650-724-9147'),
 ('tim', 'p', '650-725-2340'),
 ('tim', 'p', '650-725-4671'),
 ('ullman', 'e', 'ullman@cs.stanford.edu'),
 ('ullman', 'p', '650-494-8016'),
 ('ullman', 'p', '650-725-2588'),
 ('ullman', 'p', '650-725-4802'),
 ('vladlen', 'e', 'vladlen@stanford.edu'),
 ('widom', 'e', 'siroker@cs.stanford.edu'),
 ('widom', 'e', 'widom@cs.stanford.edu'),
 ('widom', 'p', '650-723-0872'),
 ('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')}**

**False Positives (2):**
**{('jure', 'e', 'server@cs.stanford.edu'),**
**('plotkin', 'e', 'server@infolab.stanford.edu')}**

**False Negatives (14):**
**{('dlwh', 'e', 'dlwh@stanford.edu'),**
**('hager', 'e', 'hager@cs.jhu.edu'),**
**('jks', 'e', 'jks@robotics.stanford.edu'),**
**('jurafsky', 'e', 'jurafsky@stanford.edu'),**
**('levoy', 'e', 'ada@graphics.stanford.edu'),**
**('levoy', 'e', 'melissa@graphics.stanford.edu'),**
**('manning', 'e', 'dbarros@cs.stanford.edu'),**
**('manning', 'e', 'manning@cs.stanford.edu'),**
**('ouster', 'e', 'ouster@cs.stanford.edu'),**
**('ouster', 'e', 'teresa.lynn@stanford.edu'),**
**('pal', 'e', 'pal@cs.stanford.edu'),**
**('serafim', 'e', 'serafim@cs.stanford.edu'),**
**('subh', 'e', 'uma@cs.stanford.edu'),**
**('ullman', 'e', 'support@gradiance.com')}**
**Summary: tp=103, fp=2, fn=14**

Based on documentation:
I tried to observe pattern of remaining 14 emails and found that-

**1.)** epatterns.append('([A-Za-z0-9._-]+)\s*@\s*([A-Za-z0-9._-]+)\.-e-d-u'**)**
Pattern: '`d-l-w-h-@-s-t-a-n-f-o-r-d-.-e-d-u`'
**Every character is followed by '-' except last character**
**I tried above pattern but it generated false positive as 'd-l-w-h-@-s-t-a-n-f-o-r-d-.edu'**
So we need to strip '-'in both parts of email (using python programming in part 2)

2.) epatterns.append('([A-Za-z0-9._]+)\s*&#[A-Za-z0-9]+;\s*([A-Za-z0-9._]+)\.edu')
Pattern: 'ada&#x40;graphics.stanford.edu'
**have some form of special characters (html ascii characters) in place of '@' which terminate with ; and start with &#**

Based on above logic, I made current pattern
True Positives (2):
{('levoy', 'e', 'ada@graphics.stanford.edu'),
('levoy', 'e', 'melissa@graphics.stanford.edu')}

3.) epatterns.append('([A-Za-z0-9._]+)\s*<at[\s*A-Za-z0-9\s*]+>\s*([A-Za-z0-9._]+)\.edu')
Pattern: 'manning <at symbol> cs.stanford.edu'
**have 'at symbol' in place of '@' which starts with <at and ends with >**

Based on above logic, I made current pattern
True Positives (2):
{('manning', 'e', 'dbarros@cs.stanford.edu'),
('manning', 'e', 'manning@cs.stanford.edu')}

4.) epatterns.append('([A-Za-z0-9._]+)\s*\([\s*A-Za-z0-9.&;#_]*["|;}]@([A-Za-z0-9._]+)\.edu')
Pattern: ouster (followed by &ldquo;@cs.stanford.edu&rdquo;)
and
teresa.lynn (followed by "@stanford.edu")
**has domain part encapsulated within double quotes denoted by &ldquo; and &rdquo;**
**Also userid is followed by –'(followed by' whereas @ starts after either ';' or '"'**

Based on above logic, I made current pattern
True Positives (1):
{('ouster', 'e', 'ouster@cs.stanford.edu')
('ouster', 'e', 'teresa.lynn@stanford.edu')}

5.) epatterns.append('obfuscate\(\'([A-Za-z0-9._]+)\.edu\',\'([A-Za-z0-9._]+)\'')
**Pattern: <script> obfuscate('stanford.edu','jurafsky'); </script>**
**Email is obfuscated within a script through a function named as 'obfuscate' which takes two argument- domain and then userid.**

Based on above logic, **I tried above pattern but it generated false positive as 'tanford@jurafsky.edu' Problem here lies in SpamLord.py program which always considers first part of match as userid and second part as domain but obfuscate function returns userid and domain in reverese order.**
So we change order of both parts for this pattern in process_file before generating correct email (using python programming in part 2)

**Final Output after Part-1**
**True Positives (109):**
**{('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'),**
**('bgirod', 'p', '650-724-6354'),**
**('cheriton', 'e', 'cheriton@cs.stanford.edu'),**
**('cheriton', 'e', 'uma@cs.stanford.edu'),**
**('cheriton', 'p', '650-723-1131'),**
**('cheriton', 'p', '650-725-3726'),**

```
('dabo', 'e', 'dabo@cs.stanford.edu'),
('dabo', 'p', '650-725-3897'),
('dabo', 'p', '650-725-4671'),

('engler', 'e', 'engler@lcs.mit.edu'),
('engler', 'e', 'engler@stanford.edu'),
('eroberts', 'e', 'eroberts@cs.stanford.edu'),
('eroberts', 'p', '650-723-3642'),
('eroberts', 'p', '650-723-6092'),
('fedkiw', 'e', 'fedkiw@cs.stanford.edu'),
('hager', 'p', '410-516-5521'),
('hager', 'p', '410-516-5553'),
('hager', 'p', '410-516-8000'),
('hanrahan', 'e', 'hanrahan@cs.stanford.edu'),
('hanrahan', 'p', '650-723-0033'),
('hanrahan', 'p', '650-723-8530'),
('horowitz', 'p', '650-725-3707'),
('horowitz', 'p', '650-725-6949'),
('jurafsky', 'e', 'jurafsky@stanford.edu'),
('jurafsky', 'p', '650-723-5666'),
('kosecka', 'e', 'kosecka@cs.gmu.edu'),
('kosecka', 'p', '703-993-1710'),
('kosecka', 'p', '703-993-1876'),
('kunle', 'e', 'darlene@csl.stanford.edu'),
('kunle', 'e', 'kunle@ogun.stanford.edu'),
('kunle', 'p', '650-723-1430'),
('kunle', 'p', '650-725-3713'),
('kunle', 'p', '650-725-6949'),
('lam', 'e', 'lam@cs.stanford.edu'),
('lam', 'p', '650-725-3714'),
('lam', 'p', '650-725-6949'),
('latombe', 'e', 'asandra@cs.stanford.edu'),
('latombe', 'e', 'latombe@cs.stanford.edu'),
('latombe', 'e', 'liliana@cs.stanford.edu'),
('latombe', 'p', '650-721-6625'),
('latombe', 'p', '650-723-0350'),
('latombe', 'p', '650-723-4137'),
('latombe', 'p', '650-725-1449'),
('levoy', 'e', 'ada@graphics.stanford.edu'),
('levoy', 'e', 'melissa@graphics.stanford.edu'),
('levoy', 'p', '650-723-0033'),
('levoy', 'p', '650-724-6865'),
('levoy', 'p', '650-725-3724'),
('levoy', 'p', '650-725-4089'),
('manning', 'e', 'dbarros@cs.stanford.edu'),
('manning', 'e', 'manning@cs.stanford.edu'),
('manning', 'p', '650-723-7683'),
('manning', 'p', '650-725-1449'),
('manning', 'p', '650-725-3358'),
('nass', 'e', 'nass@stanford.edu'),
('nass', 'p', '650-723-5499'),
('nass', 'p', '650-725-2472'),
('nick', 'e', 'nick.parlante@cs.stanford.edu'),
('nick', 'p', '650-725-4727'),
('ok', 'p', '650-723-9753'),
('ok', 'p', '650-725-1449'),
('ouster', 'e', 'ouster@cs.stanford.edu'),
('ouster', 'e', 'teresa.lynn@stanford.edu'),
('pal', 'p', '650-725-9046'),
('psyoung', 'e', 'patrick.young@stanford.edu'),
('rajeev', 'p', '650-723-4377'),
('rajeev', 'p', '650-723-6045'),
('rajeev', 'p', '650-725-4671'),
('rinard', 'e', 'rinard@lcs.mit.edu'),
('rinard', 'p', '617-253-1221'),
('rinard', 'p', '617-258-6922'),
('serafim', 'p', '650-723-3334'),
('serafim', 'p', '650-725-1449'),
('shoham', 'e', 'shoham@stanford.edu'),
('shoham', 'p', '650-723-3432'),
('shoham', 'p', '650-725-1449'),
('subh', 'e', 'subh@stanford.edu'),
('subh', 'p', '650-724-1915'),
('subh', 'p', '650-725-3726'),
('subh', 'p', '650-725-6949'),
('thm', 'e', 'pkrokel@stanford.edu'),
('thm', 'p', '650-725-3383'),
('thm', 'p', '650-725-3636'),
('thm', 'p', '650-725-3938'),
('tim', 'p', '650-724-9147'),
('tim', 'p', '650-725-2340'),
('tim', 'p', '650-725-4671'),
('ullman', 'e', 'support@gradiance.com'),
('ullman', 'e', 'ullman@cs.stanford.edu'),
('ullman', 'p', '650-494-8016'),
('ullman', 'p', '650-725-2588'),
('ullman', 'p', '650-725-4802'),
('vladlen', 'e', 'vladlen@stanford.edu'),
('widom', 'e', 'siroker@cs.stanford.edu'),
('widom', 'e', 'widom@cs.stanford.edu'),
('widom', 'p', '650-723-0872'),
('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')}**

**False Positives (2):**
{('jure', 'e', 'server@cs.stanford.edu'),
('plotkin', 'e', 'server@infolab.stanford.edu'),
('d-l-w-h-@-s-t-a-n-f-o-r-d-.edu'),
('tanford@jurafsky.edu')**,**
('support@gradiance.edu')}

**False Negatives (8):**
{('dlwh', 'e', 'dlwh@stanford.edu'),
('jurafsky', 'e', 'jurafsky@stanford.edu'),
('ullman', 'e', 'support@gradiance.com'),
('hager', 'e', 'hager@cs.jhu.edu'),
('jks', 'e', 'jks@robotics.stanford.edu'),
('pal', 'e', 'pal@cs.stanford.edu'),
('serafim', 'e', 'serafim@cs.stanford.edu'),
('subh', 'e', 'uma@cs.stanford.edu')}
**Summary: tp=109, fp=5, fn=8**

**Note-Last 5 emails in false negative list require further processing (python programming) in part 2 as current version of SpamLord.py program accepts only two parenthesis form regular expression and these 5 emails require 3 parenthesis which can't be handled by ppattern. Reason for 3 parenthesis requirement is that these 5 emails have their domain part further divided into two parts(total 3 including userid) and seprated by obfurscated '.' Like 'dot' ';'**

**Part 2-Python Programming to handle 8 false negatives and 2 false positives:**
**Problem 1:** As discussed above we need to adjust process file to accept three parenthesis regular expression

```
for tpat in tpatterns:
        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.%s.edu' % m
           res.append((name,'e',email))
```

We also need to generate new patterns for last 5 emails-
**tpatterns=[]**

1.) tpatterns.append('([A-Za-z0-9._]+)\s*at\s*([A-Za-z0-9_]{2,})\s*dot\s*([A-Za-z0-9_]{2,})\s*dot\s*edu')
for Emails: serafim at cs dot stanford dot edu
uma at cs dot stanford dot edu
hager at cs dot jhu dot edu

2.) tpatterns.append('([A-Za-z0-9._]+)\s*at\s*([A-Za-z0-9_]{2,})\s*([A-Za-z0-9_]{2,})\s*edu')
for email: pal at cs stanford edu

3.) tpatterns.append('([A-Za-z0-9._]+)\s*at\s*([A-Za-z0-9_]{2,})\s*;\s*([A-Za-z0-9_]{2,})\s*;\s*edu')
for email: jks at robotics;stanford;edu

**Problem 2:** We need to strip '-'in both parts of email userid and domain when epattern –
epatterns.append('([A-Za-z0-9._-]+)\s*@\s*([A-Za-z0-9._-]+)\.-e-d-u')
is processed

```
matches = re.findall(epat,line)
        if ('-e-d-u' in epat):
           for m in matches:
              b=m[0].replace('-','')
              c=m[1].replace('-','')
              email=b+'@'+c+'.edu'
              res.append((name,'e',email))
```

This changed false negative- ('d-l-w-h-@-s-t-a-n-f-o-r-d-.edu') to 'dlwh', 'e', 'dlwh@stanford.edu'

**Problem 3:** SpamLord.py program which always considers first part of match as userid and second part as domain but obfuscate function returns userid and domain in reverese order.
So we need to change order of both parts for this epattern-
epatterns.append('obfuscate\(\'([A-Za-z0-9._]+)\.edu\',\'([A-Za-z0-9._]+)\'')
in process_file before generating correct email

```
        elif ('obfuscate' in epat):
           for m in matches:
              email=m[1]+'@'+m[0]+'.edu'
              res.append((name,'e',email))
```

This changed false negative-'tanford@jurafsky.edu' to 'tanford@jurafsky.edu'

**Process 4:** SpamLord program is written in such way that for every email match(user id and domain) withbregular expression, it concatenated ‘.edu’ at the end although regular expression may contain ‘.com’

So for this epattern-

```
        epatterns.append('([A-Za-z0-9._]+)\s*at\s*([A-Za-z0-9._]+)\s*dt\s*com')
```

we need to change domain to ‘.com’

```
        elif ('com' in epat or 'COM' in epat):
          for m in matches:
            email='%s@%s.com' % m
            res.append((name,'e',email))
```

This changed false negative- ‘support@gradiance.edu’ **to** 'support@gradiance.com'

**Process 5:** Avoiding process of epatterns that matches with regular expression but have ‘server’ as userid

```
      else:
          for m in matches:
            #line.replace('-','')
            # string formatting operator % takes elements of list m
            #   and inserts them in place of each %s in the result string
            if ('Server' != m[0]):
              email = '%s@%s.edu' % m
              res.append((name,'e',email))
```

This removes false positives like 'server@cs.stanford.edu' and 'server@infolab.stanford.edu'

**Final output after part 2- python processing**
**True Positives (117):**

```
{('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'),
 ('bgirod', 'p', '650-724-6354'),
 ('cheriton', 'e', 'cheriton@cs.stanford.edu'),
 ('cheriton', 'e', 'uma@cs.stanford.edu'),
 ('cheriton', 'p', '650-723-1131'),
 ('cheriton', 'p', '650-725-3726'),
 ('dabo', 'e', 'dabo@cs.stanford.edu'),
 ('dabo', 'p', '650-725-3897'),
 ('dabo', 'p', '650-725-4671'),
 ('dlwh', 'e', 'dlwh@stanford.edu'),
 ('engler', 'e', 'engler@lcs.mit.edu'),
 ('engler', 'e', 'engler@stanford.edu'),
 ('eroberts', 'e', 'eroberts@cs.stanford.edu'),
 ('eroberts', 'p', '650-723-3642'),
 ('eroberts', 'p', '650-723-6092'),
 ('fedkiw', 'e', 'fedkiw@cs.stanford.edu'),
 ('hager', 'e', 'hager@cs.jhu.edu'),
 ('hager', 'p', '410-516-5521'),
 ('hager', 'p', '410-516-5553'),
 ('hager', 'p', '410-516-8000'),
 ('hanrahan', 'e', 'hanrahan@cs.stanford.edu'),
 ('hanrahan', 'p', '650-723-0033'),
 ('hanrahan', 'p', '650-723-8530'),
 ('horowitz', 'p', '650-725-3707'),
 ('horowitz', 'p', '650-725-6949'),
 ('jks', 'e', 'jks@robotics.stanford.edu'),
 ('jurafsky', 'e', 'jurafsky@stanford.edu'),
 ('jurafsky', 'p', '650-723-5666'),
 ('kosecka', 'e', 'kosecka@cs.gmu.edu'),
 ('kosecka', 'p', '703-993-1710'),
 ('kosecka', 'p', '703-993-1876'),
 ('kunle', 'e', 'darlene@csl.stanford.edu'),
 ('kunle', 'e', 'kunle@ogun.stanford.edu'),
 ('kunle', 'p', '650-723-1430'),
 ('kunle', 'p', '650-725-3713'),
 ('kunle', 'p', '650-725-6949'),
 ('lam', 'e', 'lam@cs.stanford.edu'),
 ('lam', 'p', '650-725-3714'),
 ('lam', 'p', '650-725-6949'),
 ('latombe', 'e', 'asandra@cs.stanford.edu'),
 ('latombe', 'e', 'latombe@cs.stanford.edu'),
 ('latombe', 'e', 'liliana@cs.stanford.edu'),
 ('latombe', 'p', '650-721-6625'),
 ('latombe', 'p', '650-723-0350'),
 ('latombe', 'p', '650-723-4137'),
 ('latombe', 'p', '650-725-1449'),
 ('levoy', 'e', 'ada@graphics.stanford.edu'),
 ('levoy', 'e', 'melissa@graphics.stanford.edu'),
 ('levoy', 'p', '650-723-0033'),
 ('levoy', 'p', '650-724-6865'),
 ('levoy', 'p', '650-725-3724'),
 ('levoy', 'p', '650-725-4089'),
 ('manning', 'e', 'dbarros@cs.stanford.edu'),
 ('manning', 'e', 'manning@cs.stanford.edu'),
 ('manning', 'p', '650-723-7683'),
 ('manning', 'p', '650-725-1449'),
 ('manning', 'p', '650-725-3358'),
 ('nass', 'e', 'nass@stanford.edu'),
 ('nass', 'p', '650-723-5499'),
 ('nass', 'p', '650-725-2472'),
 ('nick', 'e', 'nick.parlante@cs.stanford.edu'),
 ('nick', 'p', '650-725-4727'),
 ('ok', 'p', '650-723-9753'),
 ('ok', 'p', '650-725-1449'),
 ('ouster', 'e', 'ouster@cs.stanford.edu'),
 ('ouster', 'e', 'teresa.lynn@stanford.edu'),
 ('pal', 'e', 'pal@cs.stanford.edu'),
 ('pal', 'p', '650-725-9046'),
 ('psyoung', 'e', 'patrick.young@stanford.edu'),
 ('rajeev', 'p', '650-723-4377'),
 ('rajeev', 'p', '650-723-6045'),
 ('rajeev', 'p', '650-725-4671'),
 ('rinard', 'e', 'rinard@lcs.mit.edu'),
 ('rinard', 'p', '617-253-1221'),
 ('rinard', 'p', '617-258-6922'),
 ('serafim', 'e', 'serafim@cs.stanford.edu'),
 ('serafim', 'p', '650-723-3334'),
 ('serafim', 'p', '650-725-1449'),
 ('shoham', 'e', 'shoham@stanford.edu'),
 ('shoham', 'p', '650-723-3432'),
 ('shoham', 'p', '650-725-1449'),
 ('subh', 'e', 'subh@stanford.edu'),
 ('subh', 'e', 'uma@cs.stanford.edu'),
 ('subh', 'p', '650-724-1915'),
 ('subh', 'p', '650-725-3726'),
 ('subh', 'p', '650-725-6949'),
 ('thm', 'e', 'pkrokel@stanford.edu'),
 ('thm', 'p', '650-725-3383'),
 ('thm', 'p', '650-725-3636'),
 ('thm', 'p', '650-725-3938'),
 ('tim', 'p', '650-724-9147'),
 ('tim', 'p', '650-725-2340'),
 ('tim', 'p', '650-725-4671'),
 ('ullman', 'e', 'support@gradiance.com'),
 ('ullman', 'e', 'ullman@cs.stanford.edu'),
 ('ullman', 'p', '650-494-8016'),
 ('ullman', 'p', '650-725-2588'),
 ('ullman', 'p', '650-725-4802'),
 ('vladlen', 'e', 'vladlen@stanford.edu'),
 ('widom', 'e', 'siroker@cs.stanford.edu'),
 ('widom', 'e', 'widom@cs.stanford.edu'),
 ('widom', 'p', '650-723-0872'),
  ('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')}

 False Positives (0):
 set()

 False Negatives (0):
 set()

 Summary: tp=117, fp=0, fn=0
```