In [45]:
```
"""
This program was adapted from the Stanford NLP class SpamLord homework assig
    The code has been rewritten and the data modified, nevertheless
    please do not make this code or the data public.
This version has two patterns that were suggested in comments
    in order to get you started .
"""
import sys
import os
import re
import pprint

"""
TODO
For Part 1 of our assignment, add to these two lists of patterns to match
examples of obscured email addresses and phone numbers in the text.
For optional Part 3, you may need to add other lists of patterns.
"""
# email .edu patterns

# each regular expression pattern should have exactly two sets of parenthese
#   the first parenthesis should be around the someone part
#   the second parenthesis should be around the somewhere part
#   in an email address whose standard form is someone@somewhere.edu
epatterns = []
epatterns.append('([A-Za-z.]+)@([A-Za-z.]+)\.edu')
epatterns.append('([A-Za-z.]+)\s@\s([A-Za-z.]+)\.edu')
epatterns.append('(\b\w+\b)\s*(?:at|@)?\s*cs\.?\s*(?:dot|\.)?\s*(?:stanford|
epatterns.append('([A-Za-z.]+)\s\s@\s\s([A-Za-z.]+)\.edu')
epatterns.append('([A-Za-z.]+)@([A-Za-z.]+)\.edu\.')
epatterns.append('([A-Za-z.]+)@([A-Za-z.]+)\.EDU')
epatterns.append('([A-Za-z.]+)\\s@\\s([A-Za-z.]+)\\.edu')
epatterns.append('([A-Za-z.]+)@([A-Za-z.]+)\\.edu')
epatterns.append('([A-Za-z.]+)@([A-Za-z.]+)([A-Za-z]+)\\.edu')
epatterns.append('([A-Za-z0-9.]+)\\s*at\\s*([A-Za-z0-9.]+)\\.EDU')
epatterns.append('([A-Za-z0-9-]+)\\-@\\-([A-Za-z0-9-]+)\\.\\-e-d-u')
epatterns.append('([A-Za-z0-9.]+)\\s*[<][A-Za-z0-9.]+[>]\\s*@\\s*([A-Za-z0-9
epatterns.append('([A-Za-z0-9._]+)\\s*<at[\\s*A-Za-z0-9\\s*]+>\\s*([A-Za-z0-
epatterns.append('([A-Za-z0-9._]+)\\s*at\\s*[<][A-Za-z0-9.!\\s_-]+[>]\\s*([A
epatterns.append('obfuscate\(\'([A-Za-z0-9._]+)\.edu\',\'([A-Za-z0-9._]+)\''









# phone patterns
# each regular expression pattern should have exactly three sets of parenthe
#   the first parenthesis should be around the area code part XXX
```

```
#   the second parenthesis should be around the exchange part YYY
#   the third parenthesis should be around the number part ZZZZ
#   in a phone number whose standard form is XXX-YYY-ZZZZ
ppatterns = []
ppatterns.append('(\d{3})-(\d{3})-(\d{4})')
ppatterns.append('[\+]?(\d{3})\s(\d{3})-(\d{4})')
ppatterns.append('(\\d{3})[- ](\\d{3})[- ](\\d{4})')
ppatterns.append('\((\d{3})\)(\d{3})-(\d{4})')
ppatterns.append('\((\d{3})\)\s(\d{3})-(\d{4})')
ppatterns.append('\\[(\\d{3})\\][ ]*(\\d{3})-(\\d{4})')
ppatterns.append('(\d{3})\s(\d{3})\s(\d{4})')
ppatterns.append('\[(\d{3})\]\s(\d{3})-(\d{4})')
ppatterns.append('([0-9]{3})[- ]([0-9]{3})[- ]([0-9]{4})')
ppatterns.append('\\((\\d{3})\\)[ ]*(\\d{3})-(\\d{4})')




"""
This function takes in a filename along with the file object and
scans its contents against regex patterns. It returns a list of
(filename, type, value) tuples where type is either an 'e' or a 'p'
for e-mail or phone, and value is the formatted phone number or e-mail.
The canonical formats are:
     (name, 'p', '###-###-#####')
     (name, 'e', 'someone@something')
If the numbers you submit are formatted differently they will not
match the gold answers

TODO
For Part 3, if you have added other lists, you should add
additional for loops that match the patterns in those lists
and produce correctly formatted results to append to the res list.
"""
dic = {}
def process_file(name, f):
    # note that debug info should be printed to stderr
    # sys.stderr.write('[process_file]\tprocessing file: %s\n' % (path))
    res = []
    for line in f:
        # you may modify the line, using something like substitution
        #    before applying the patterns



        # email pattern list
        for epat in epatterns:
            # each epat has 2 sets of parentheses so each match will have 2
            matches = re.findall(epat,line)
```

```
            for m in matches:
                # string formatting operator % takes elements of list m
                #   and inserts them in place of each %s in the result strin
                # email has form  someone@somewhere.edu
                #email = '%s@%s.edu' % m
                email = '{}@{}.edu'.format(m[0],m[1])

                if not epat in dic:
                    dic[epat] = []
                dic[epat].append(email)
                res.append((name,'e',email))

        # phone pattern list
        for ppat in ppatterns:
            # each ppat has 3 sets of parentheses so each match will have 3
            matches = re.findall(ppat,line)
            for m in matches:
                # phone number has form  areacode-exchange-number
                #phone = '%s-%s-%s' % m
                phone = '{}-{}-{}'.format(m[0],m[1],m[2])

                if not ppat in dic:
                    dic[ppat] = []
                dic[ppat].append(phone)
                res.append((name,'p',phone))

    return res

"""
You should not edit this function.
"""
def process_dir(data_path):
    # save complete list of candidates
    guess_list = []
    # save list of filenames
    fname_list = []

    for fname in os.listdir(data_path):
        if fname[0] == '.':
            continue
        fname_list.append(fname)
        path = os.path.join(data_path,fname)
        f = open(path,'r', encoding='latin-1')
        # get all the candidates for this file
        f_guesses = process_file(fname, f)
        guess_list.extend(f_guesses)
    return guess_list, fname_list
```

```
"""
You should not edit this function.
Given a path to a tsv file of gold e-mails and phone numbers
this function returns a list of tuples of the canonical form:
(filename, type, value)
"""
def get_gold(gold_path):
    # get gold answers
    gold_list = []
    f_gold = open(gold_path,'r', encoding='latin-1')
    for line in f_gold:
        gold_list.append(tuple(line.strip().split('\t')))
    return gold_list

"""
You should not edit this function.
Given a list of guessed contacts and gold contacts, this function
    computes the intersection and set differences, to compute the true
    positives, false positives and false negatives.
It also takes a dictionary that gives the guesses for each filename,
    which can be used for information about false positives.
Importantly, it converts all of the values to lower case before comparing.
"""
def score(guess_list, gold_list, fname_list):
    guess_list = [(fname, _type, value.lower()) for (fname, _type, value) in
    gold_list = [(fname, _type, value.lower()) for (fname, _type, value) in
    guess_set = set(guess_list)
    gold_set = set(gold_list)

    # for each file name, put the golds from that file in a dict
    gold_dict = {}
    for fname in fname_list:
        gold_dict[fname] = [gold for gold in gold_list if fname == gold[0]]

    tp = guess_set.intersection(gold_set)
    fp = guess_set - gold_set
    fn = gold_set - guess_set

    pp = pprint.PrettyPrinter()
    #print 'Guesses (%d): ' % len(guess_set)
    #pp.pprint(guess_set)
    #print 'Gold (%d): ' % len(gold_set)
    #pp.pprint(gold_set)

    print ('True Positives (%d): ' % len(tp))
    # print all true positives
    pp.pprint(tp)
    print ('False Positives (%d): ' % len(fp))
    # for each false positive, print it and the list of gold for debugging
```

```
    for item in fp:
        fp_name = item[0]
        pp.pprint(item)
        fp_list = gold_dict[fp_name]
        for gold in fp_list:
            s = pprint.pformat(gold)
            print('   gold: ', s)
    print ('False Negatives (%d): ' % len(fn))
    # print all false negatives
    pp.pprint(fn)
    print ('Summary: tp=%d, fp=%d, fn=%d' % (len(tp),len(fp),len(fn)))

"""
You should not edit this function.
It takes in the string path to the data directory and the gold file
"""
def main(data_path, gold_path):
    guess_list, fname_list = process_dir(data_path)
    gold_list =  get_gold(gold_path)
    score(guess_list, gold_list, fname_list)


"""
commandline interface assumes that you are in the directory containing "data
It then processes each file within that data folder and extracts any
matching e-mails or phone numbers and compares them to the gold file
"""
if __name__ == '__main__':
    print ('Assuming ContactFinder.py called in directory with data folder')
    main('data/dev', 'data/devGOLD')
```

```
Assuming ContactFinder.py called in directory with data folder
True Positives (102):
{('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'),
('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', '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', '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'),
('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', '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 (21):
('thm', 'e', 'pkrokel@stanfor.edu')
   gold:  ('thm', 'e', 'pkrokel@stanford.edu')
   gold:  ('thm', 'p', '650-725-3383')
   gold:  ('thm', 'p', '650-725-3636')
   gold:  ('thm', 'p', '650-725-3938')
('rinard', 'e', 'rinard@lcs.mi.edu')
   gold:  ('rinard', 'e', 'rinard@lcs.mit.edu')
   gold:  ('rinard', 'p', '617-253-1221')
   gold:  ('rinard', 'p', '617-258-6922')
('cheriton', 'e', 'cheriton@cs.stanfor.edu')
   gold:  ('cheriton', 'e', 'cheriton@cs.stanford.edu')
   gold:  ('cheriton', 'e', 'uma@cs.stanford.edu')
```

```
   gold:  ('cheriton', 'p', '650-723-1131')
   gold:  ('cheriton', 'p', '650-725-3726')
('eroberts', 'e', 'eroberts@cs.stanfor.edu')
   gold:  ('eroberts', 'e', 'eroberts@cs.stanford.edu')
   gold:  ('eroberts', 'p', '650-723-3642')
   gold:  ('eroberts', 'p', '650-723-6092')
('nick', 'e', 'nick.parlante@cs.stanfor.edu')
   gold:  ('nick', 'e', 'nick.parlante@cs.stanford.edu')
   gold:  ('nick', 'p', '650-725-4727')
('fedkiw', 'e', 'fedkiw@cs.stanfor.edu')
   gold:  ('fedkiw', 'e', 'fedkiw@cs.stanford.edu')
('dlwh', 'e', 'd-l-w-h@s-t-a-n-f-o-r-d-.edu')
   gold:  ('dlwh', 'e', 'dlwh@stanford.edu')
('hanrahan', 'e', 'hanrahan@cs.stanfor.edu')
   gold:  ('hanrahan', 'e', 'hanrahan@cs.stanford.edu')
   gold:  ('hanrahan', 'p', '650-723-0033')
   gold:  ('hanrahan', 'p', '650-723-8530')
('widom', 'e', 'widom@cs.stanfor.edu')
   gold:  ('widom', 'e', 'siroker@cs.stanford.edu')
   gold:  ('widom', 'e', 'widom@cs.stanford.edu')
   gold:  ('widom', 'p', '650-723-0872')
   gold:  ('widom', 'p', '650-723-7690')
   gold:  ('widom', 'p', '650-725-2588')
('kunle', 'e', 'kunle@ogun.stanfor.edu')
   gold:  ('kunle', 'e', 'darlene@csl.stanford.edu')
   gold:  ('kunle', 'e', 'kunle@ogun.stanford.edu')
   gold:  ('kunle', 'p', '650-723-1430')
   gold:  ('kunle', 'p', '650-725-3713')
   gold:  ('kunle', 'p', '650-725-6949')
('jurafsky', 'e', 'stanford@jurafsky.edu')
   gold:  ('jurafsky', 'e', 'jurafsky@stanford.edu')
   gold:  ('jurafsky', 'p', '650-723-5666')
('psyoung', 'e', 'patrick.young@stanfor.edu')
   gold:  ('psyoung', 'e', 'patrick.young@stanford.edu')
('balaji', 'e', 'balaji@stanfor.edu')
   gold:  ('balaji', 'e', 'balaji@stanford.edu')
('zelenski', 'e', 'zelenski@cs.stanfor.edu')
   gold:  ('zelenski', 'e', 'zelenski@cs.stanford.edu')
   gold:  ('zelenski', 'p', '650-723-6092')
   gold:  ('zelenski', 'p', '650-725-8596')
('engler', 'e', 'engler@lcs.mi.edu')
   gold:  ('engler', 'e', 'engler@lcs.mit.edu')
   gold:  ('engler', 'e', 'engler@stanford.edu')
('shoham', 'e', 'shoham@stanfor.edu')
   gold:  ('shoham', 'e', 'shoham@stanford.edu')
   gold:  ('shoham', 'p', '650-723-3432')
   gold:  ('shoham', 'p', '650-725-1449')
('kosecka', 'e', 'kosecka@cs.gm.edu')
   gold:  ('kosecka', 'e', 'kosecka@cs.gmu.edu')
   gold:  ('kosecka', 'p', '703-993-1710')
```

```
   gold:  ('kosecka', 'p', '703-993-1876')
('widom', 'e', 'siroker@cs.stanfor.edu')
   gold:  ('widom', 'e', 'siroker@cs.stanford.edu')
   gold:  ('widom', 'e', 'widom@cs.stanford.edu')
   gold:  ('widom', 'p', '650-723-0872')
   gold:  ('widom', 'p', '650-723-7690')
   gold:  ('widom', 'p', '650-725-2588')
('kunle', 'e', 'darlene@csl.stanfor.edu')
   gold:  ('kunle', 'e', 'darlene@csl.stanford.edu')
   gold:  ('kunle', 'e', 'kunle@ogun.stanford.edu')
   gold:  ('kunle', 'p', '650-723-1430')
   gold:  ('kunle', 'p', '650-725-3713')
   gold:  ('kunle', 'p', '650-725-6949')
('nass', 'e', 'nass@stanfor.edu')
   gold:  ('nass', 'e', 'nass@stanford.edu')
   gold:  ('nass', 'p', '650-723-5499')
   gold:  ('nass', 'p', '650-725-2472')
('zm', 'e', 'manna@cs.stanfor.edu')
   gold:  ('zm', 'e', 'manna@cs.stanford.edu')
   gold:  ('zm', 'p', '650-723-4364')
   gold:  ('zm', 'p', '650-725-4671')
False Negatives (15):
{('dlwh', 'e', 'dlwh@stanford.edu'),
 ('engler', 'e', 'engler@stanford.edu'),
 ('hager', 'e', 'hager@cs.jhu.edu'),
 ('jks', 'e', 'jks@robotics.stanford.edu'),
 ('jurafsky', 'e', 'jurafsky@stanford.edu'),
 ('lam', 'e', 'lam@cs.stanford.edu'),
 ('levoy', 'e', 'ada@graphics.stanford.edu'),
 ('levoy', 'e', 'melissa@graphics.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', 'subh@stanford.edu'),
 ('subh', 'e', 'uma@cs.stanford.edu'),
 ('ullman', 'e', 'support@gradiance.com')}
Summary: tp=102, fp=21, fn=15
```

In [ ]:

Patterns and their Matchings

At the beginning I was getting 22 True positives and 95 False Negatives. After adding the below Patterns I was able to get 102 True Positives and 20 False Positives and 15 False Negatives

1. [A-Za-z0-9._]+)\s<at[|sA-Za-z0-9\s]+>|s([A-Za-z0-9._]+)\.edu :

["manning@cs.stanford.edu", "manning@cs.stanford.edu", "dbarros@cs.stanford.edu", "dbarros@cs.stanford.edu", "manning@cs.stanford.edu", "manning@cs.stanford.edu"]

Pattern description:

([A-Za-z0-9._]+): This part captures one or more alphanumeric characters, underscores, or dots. It is enclosed in parentheses, indicating that it is a capturing group. \s*: This part matches zero or more whitespace characters. <at[|s*A-Za-z0-9\s*]+>: This part matches the literal string "<at" followed by one or more whitespace characters, alphanumeric characters, or asterisks. It is enclosed in square brackets, indicating a character class, which means any of the characters within the brackets can match. |s*: This part matches zero or more whitespace characters. ([A-Za-z0-9._]+): This part is similar to the first capturing group and captures one or more alphanumeric characters, underscores, or dots. \.edu: This part matches the literal string ".edu". The backslash is used to escape the dot, which is a special character in regular expressions.

2.

"
(
$d3)$
\s(\d{3})-(\d{4})"

Pattern description:

- `\\(` : Matches a literal opening parenthesis "(".
- `(\\d{3})` : This is a capturing group that matches exactly three digits. It captures the first three digits within parentheses. The backslash and "d" (\d) represent any digit, and "{3}" specifies that there should be exactly three occurrences of a digit.
- `\\)` : Matches a literal closing parenthesis ")".
- `\\s` : Matches a single whitespace character.
- `(\\d{3})` : This is another capturing group that matches exactly three digits. It captures the next three digits after the whitespace.
- `-` : Matches a hyphen character.
- `(\\d{4})` : This is the third capturing group that matches exactly four digits. It captures the final four digits of the phone number.

Overall, the regular expression pattern is used to match phone numbers in the format: (###) ###-####, where # represents a digit. The parentheses, whitespace, and hyphen

are fixed parts of the pattern, while the three-digit and four-digit numbers are captured using capturing groups.

["650-723-7683", "650-725-1449", "650-725-3358", "650-723-4364", "650-725-4671", "410-516-5521", "410-516-5553", "650-725-4727", "650-723-9753", "650-725-1449", "650-725-3714", "650-725-6949", "650-725-3713", "650-725-6949", "650-723-1430", "650-725-6949", "617-258-6922", "617-253-1221", "650-725-3636", "650-725-3383", "650-725-3938", "650-725-3383", "650-723-3334", "650-725-1449", "650-723-0350", "650-725-1449", "650-721-6625", "650-723-4137", "650-725-3897", "650-725-4671", "650-724-6354", "650-723-4539", "650-724-3648", "703-993-1876", "703-993-1710", "650-725-4089", "650-723-0033", "650-725-3724", "650-724-6865", "650-723-8530", "650-723-0033"]

3. "
   (
   $d3$)
   [ ]*(\d{3})-(\d{4})"

Pattern description:

: $Matches a literal opening parenthesis$ ” (”. (
$d3$) : $This is a capturing group that matches exactly three digits. It captures the first t$
” $specifies that there should be exactly three occurrences of a digit.$
: Matches a literal closing parenthesis ")". [ ]*: Matches zero or more spaces. The square brackets and asterisk indicate a character class where a space can occur zero or more times. (\d{3}): This is another capturing group that matches exactly three digits. It captures the next three digits after the optional spaces. -: Matches a hyphen character. (\d{4}): This is the third capturing group that matches exactly four digits. It captures the final four digits of the phone number.

["650-723-7683", "650-725-1449", "650-725-3358", "650-723-4364", "650-725-4671", "410-516-5521", "410-516-5553", "650-725-4727", "650-723-9753", "650-725-1449", "650-725-3714", "650-725-6949", "650-725-3713", "650-725-6949", "650-723-1430", "650-725-6949", "617-258-6922", "617-253-1221", "650-725-3636", "650-725-3383", "650-725-3938", "650-725-3383", "650-723-3334", "650-725-1449", "650-814-1478", "650-723-1614", "650-723-4173", "650-725-3707", "650-725-6949", "650-723-0350", "650-725-1449", "650-721-6625", "650-723-4137", "650-724-9147", "650-725-4671", "650-725-2340", "650-725-4671", "650-725-3897", "650-725-4671", "650-724-6354", "650-723-4539", "650-724-3648", "703-993-1876", "703-993-1710", "650-725-4089", "650-723-0033", "650-725-3724", "650-

724-6865", "650-723-8530", "650-723-0033"]

4. "(\d{3})-(\d{3})-(\d{4})"

Pattern description:

(\d{3}): This is a capturing group that matches exactly three digits. The backslash and "d" (\d) represent any digit, and "{3}" specifies that there should be exactly three occurrences of a digit. -: Matches a hyphen character. (\d{3}): This is another capturing group that matches exactly three digits. -: Matches a hyphen character. (\d{4}): This is the third capturing group that matches exactly four digits.

["650-723-6045", "650-725-4671", "650-723-4377", "650-725-4671", "410-516-8000", "650-723-1131", "650-725-3726", "650-725-4802", "650-494-8016", "650-725-2588", "650-724-1915", "650-725-3726", "650-725-6949", "650-723-3642", "650-723-6092", "650-725-8596", "650-723-6092", "650-723-7690", "650-725-2588", "650-723-0872"]

5. "([0-9]{3})- - "

Pattern description:

([0-9]{3}): This is a capturing group that matches exactly three digits. The square brackets and "0-9" represent any digit, and "{3}" specifies that there should be exactly three occurrences of a digit. [-]: Matches either a hyphen or a space character. ([0-9]{3}): This is another capturing group that matches exactly three digits. [-]: Matches either a hyphen or a space character. ([0-9]{4}): This is the third capturing group that matches exactly four digits.

["650-723-6045", "650-725-4671", "650-723-4377", "650-725-4671", "410-516-8000", "650-723-1131", "650-725-3726", "650-725-4802", "650-494-8016", "650-725-2588", "650-724-1915", "650-725-3726", "650-725-6949", "650-725-9046", "650-723-3642", "650-723-6092", "650-725-8596", "650-723-6092", "650-723-7690", "650-725-2588", "650-723-0872", "650-723-5666", "650-723-3432", "650-725-1449"]

6. "(\d{3})- - "

Pattern description:

(\d{3}): This is a capturing group that matches exactly three digits. The backslash and "d" (\d) represent any digit, and "{3}" specifies that there should be exactly three

occurrences of a digit. [-]: Matches either a hyphen or a space character. (\d{3}): This is another capturing group that matches exactly three digits. [-]: Matches either a hyphen or a space character. (\d{4}): This is the third capturing group that matches exactly four digits.

["650-723-6045", "650-725-4671", "650-723-4377", "650-725-4671", "410-516-8000", "650-723-1131", "650-725-3726", "650-725-4802", "650-494-8016", "650-725-2588", "650-724-1915", "650-725-3726", "650-725-6949", "650-725-9046", "650-723-3642", "650-723-6092", "650-725-8596", "650-723-6092", "650-723-7690", "650-725-2588", "650-723-0872", "650-723-5666", "650-723-3432", "650-725-1449"]

7. "([A-Za-z.]+)@([A-Za-z.]+)\.edu":

Pattern description:

([A-Za-z.]+): This is a capturing group that matches one or more letters (both uppercase and lowercase) or dots. It captures the part of the email address before the "@" symbol. @: Matches the "@" symbol. ([A-Za-z.]+): This is another capturing group that matches one or more letters (both uppercase and lowercase) or dots. It captures the part of the email address after the "@" symbol, but before the ".edu" domain. \.edu: Matches the ".edu" domain part of the email address. The backslash is used to escape the dot, which is a special character in regular expressions.

["manna@cs.stanford.edu", "manna@cs.stanford.edu", "manna@cs.stanford.edu", "manna@cs.stanford.edu", "nick.parlante@cs.stanford.edu", "nick.parlante@cs.stanford.edu", "nick.parlante@cs.stanford.edu", "nick.parlante@cs.stanford.edu", "cheriton@cs.stanford.edu", "cheriton@cs.stanford.edu", "patrick.young@stanford.edu", "patrick.young@stanford.edu", "patrick.young@stanford.edu", "patrick.young@stanford.edu", "kunle@ogun.stanford.edu", "kunle@ogun.stanford.edu", "kunle@ogun.stanford.edu", "kunle@ogun.stanford.edu", "darlene@csl.stanford.edu", "darlene@csl.stanford.edu", "darlene@csl.stanford.edu", "darlene@csl.stanford.edu", "engler@lcs.mit.edu", "engler@lcs.mit.edu", "eroberts@cs.stanford.edu", "eroberts@cs.stanford.edu", "eroberts@cs.stanford.edu", "eroberts@cs.stanford.edu", "rinard@lcs.mit.edu", "rinard@lcs.mit.edu", "pkrokel@Stanford.edu", "pkrokel@Stanford.edu", "pkrokel@Stanford.edu", "pkrokel@Stanford.edu", "balaji@stanford.edu", "balaji@stanford.edu", "zelenski@cs.stanford.edu", "zelenski@cs.stanford.edu", "zelenski@cs.stanford.edu", "zelenski@cs.stanford.edu", "siroker@cs.stanford.edu", "siroker@cs.stanford.edu", "widom@cs.stanford.edu",

"widom@cs.stanford.edu", "fedkiw@cs.stanford.edu", "fedkiw@cs.stanford.edu", "fedkiw@cs.stanford.edu", "fedkiw@cs.stanford.edu", "kosecka@cs.gmu.edu", "kosecka@cs.gmu.edu", "kosecka@cs.gmu.edu", "kosecka@cs.gmu.edu", "hanrahan@cs.stanford.edu", "hanrahan@cs.stanford.edu", "hanrahan@cs.stanford.edu", "hanrahan@cs.stanford.edu", "nass@stanford.edu", "nass@stanford.edu", "nass@stanford.edu", "nass@stanford.edu", "nass@stanford.edu", "nass@stanford.edu", "nass@stanford.edu", "nass@stanford.edu", "shoham@stanford.edu", "shoham@stanford.edu", "shoham@stanford.edu", "shoham@stanford.edu"]

8. "([A-Za-z.]+)@([A-Za-z.]+)([A-Za-z]+)\.edu":

Pattern description:

([A-Za-z.]+): This is a capturing group that matches one or more letters (both uppercase and lowercase) or dots. It captures the part of the email address before the "@" symbol. @: Matches the "@" symbol. ([A-Za-z.]+): This is another capturing group that matches one or more letters (both uppercase and lowercase) or dots. It captures the part of the email address after the "@" symbol, but before the final part of the domain. ([A-Za-z]+): This is the third capturing group that matches one or more letters (both uppercase and lowercase). It captures the final part of the domain. \.edu: Matches the ".edu" domain part of the email address. The backslash is used to escape the dot, which is a special character in regular expressions.

["manna@cs.stanfor.edu", "manna@cs.stanfor.edu", "nick.parlante@cs.stanfor.edu", "nick.parlante@cs.stanfor.edu", "cheriton@cs.stanfor.edu", "patrick.young@stanfor.edu", "patrick.young@stanfor.edu", "kunle@ogun.stanfor.edu", "kunle@ogun.stanfor.edu", "darlene@csl.stanfor.edu", "darlene@csl.stanfor.edu", "engler@lcs.mi.edu", "eroberts@cs.stanfor.edu", "eroberts@cs.stanfor.edu", "rinard@lcs.mi.edu", "pkrokel@Stanfor.edu", "pkrokel@Stanfor.edu", "balaji@stanfor.edu", "zelenski@cs.stanfor.edu", "zelenski@cs.stanfor.edu", "siroker@cs.stanfor.edu", "widom@cs.stanfor.edu", "fedkiw@cs.stanfor.edu", "fedkiw@cs.stanfor.edu", "kosecka@cs.gm.edu", "kosecka@cs.gm.edu", "hanrahan@cs.stanfor.edu", "hanrahan@cs.stanfor.edu", "nass@stanfor.edu", "nass@stanfor.edu", "nass@stanfor.edu", "nass@stanfor.edu", "shoham@stanfor.edu", "shoham@stanfor.edu"]

9. "([A-Za-z.]+)\s@\s([A-Za-z.]+)\.edu":

Pattern description:

([A-Za-z.]+): This is a capturing group that matches one or more letters (both uppercase and lowercase) or dots. It captures the part of the email address before the "@" symbol. \s@\s: Matches a space character followed by the "@" symbol, followed by another space character. This ensures there is whitespace before and after the "@" symbol. ([A-Za-z.]+): This is another capturing group that matches one or more letters (both uppercase and lowercase) or dots. It captures the part of the email address after the "@" symbol, but before the ".edu" domain. \.edu: Matches the ".edu" domain part of the email address. The backslash is used to escape the dot, which is a special character in regular expressions.

["ullman@cs.stanford.edu", "ullman@cs.stanford.edu", "ashishg@stanford.edu", "ashishg@stanford.edu", "rozm@stanford.edu", "rozm@stanford.edu", "zelenski@cs.stanford.edu", "zelenski@cs.stanford.edu"]

10. "
$($
$d3)$
(\d{3})-(\d{4})":

Pattern description:

$: Matches a literal opening parenthesis " (". ($ $d3) : This is a capturing group that matches exactly three digits. The backslash and $ - digit are a code within parentheses.$
: Matches a literal closing parenthesis ")". (\d{3}): This is another capturing group that matches exactly three digits. This captures the next three digits after the area code. -: Matches a hyphen character. (\d{4}): This is the third capturing group that matches exactly four digits. This captures the final four digits of the phone number.

["650-814-1478", "650-723-1614", "650-723-4173", "650-725-3707", "650-725-6949", "650-724-9147", "650-725-4671", "650-725-2340", "650-725-4671"]

11. "([A-Za-z0-9.]+)\s*[<][A-Za-z0-9.]+[>]*|s@\s*([A-Za-z0-9._]+)\.edu":

Pattern description:

([A-Za-z0-9.]+): This is a capturing group that matches one or more alphanumeric characters or dots. It captures the part of the email address before the "@" symbol. It allows for a mix of uppercase and lowercase letters, digits, and dots. *\s: Matches zero or more whitespace characters. [<]: Matches a literal "<" character. [A-Za-z0-9.]+: Matches one or more alphanumeric characters or dots. It represents the part of the*

email address within the angle brackets ("<" and ">"). [>]: Matches a literal ">" character. |s@\s*: Matches the "@" symbol surrounded by zero or more whitespace characters on both sides. ([A-Za-z0-9._]+): This is another capturing group that matches one or more alphanumeric characters or underscores. It captures the domain part of the email address before the ".edu" domain. \.edu: Matches the ".edu" domain part of the email address. The backslash is used to escape the dot, which is a special character in regular expressions.

["latombe@cs.stanford.edu", "latombe@cs.stanford.edu", "asandra@cs.stanford.edu", "asandra@cs.stanford.edu", "liliana@cs.stanford.edu", "liliana@cs.stanford.edu"]

In [41]:
```
"""
This program was adapted from the Stanford NLP class SpamLord homework assig
    The code has been rewritten and the data modified, nevertheless
    please do not make this code or the data public.
This version has two patterns that were suggested in comments
    in order to get you started .
"""
import sys
import os
import re
import pprint

"""
TODO
For Part 1 of our assignment, add to these two lists of patterns to match
examples of obscured email addresses and phone numbers in the text.
For optional Part 3, you may need to add other lists of patterns.
"""
# email .edu patterns

# each regular expression pattern should have exactly two sets of parenthese
#   the first parenthesis should be around the someone part
#   the second parenthesis should be around the somewhere part
#   in an email address whose standard form is someone@somewhere.edu
epatterns = []
epatterns.append('([A-Za-z.]+)@([A-Za-z.]+)\.edu')
epatterns.append('([A-Za-z.]+)\s@\s([A-Za-z.]+)\.edu')
epatterns.append('(\b\w+\b)\s*(?:at|@)?\s*cs\.?\s*(?:dot|\.)?\s*(?:stanford|
epatterns.append('([A-Za-z.]+)\s\s@\s\s([A-Za-z.]+)\.edu')
epatterns.append('([A-Za-z.]+)@([A-Za-z.]+)\.edu\.')
epatterns.append('([A-Za-z.]+)@([A-Za-z.]+)\.EDU')
epatterns.append('([A-Za-z.]+)\\s@\\s([A-Za-z.]+)\\.edu')
epatterns.append('([A-Za-z.]+)@([A-Za-z.]+)\\.edu')
epatterns.append('([A-Za-z.]+)@([A-Za-z.]+)([A-Za-z]+)\\.edu')
epatterns.append('([A-Za-z0-9.]+)\\s*at\\s*([A-Za-z0-9.]+)\\.EDU')
epatterns.append('([A-Za-z0-9-]+)\\-@\\-([A-Za-z0-9-]+)\\.\\-e-d-u')
```

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

```
# phone patterns
# each regular expression pattern should have exactly three sets of parenthe
#   the first parenthesis should be around the area code part XXX
#   the second parenthesis should be around the exchange part YYY
#   the third parenthesis should be around the number part ZZZZ
#   in a phone number whose standard form is XXX-YYY-ZZZZ
ppatterns = []
ppatterns.append('(\d{3})-(\d{3})-(\d{4})')
ppatterns.append('[\+]?(\d{3})\s(\d{3})-(\d{4})')
ppatterns.append('(\\d{3})[- ](\\d{3})[- ](\\d{4})')
ppatterns.append('\((\d{3})\)(\d{3})-(\d{4})')
ppatterns.append('\((\d{3})\)\s(\d{3})-(\d{4})')
ppatterns.append('\\[(\\d{3})\\][ ]*(\\d{3})-(\\d{4})')
ppatterns.append('(\d{3})\s(\d{3})\s(\d{4})')
ppatterns.append('\[(\d{3})\]\s(\d{3})-(\d{4})')
ppatterns.append('([0-9]{3})[- ]([0-9]{3})[- ]([0-9]{4})')
ppatterns.append('\\((\\d{3})\\)[ ]*(\\d{3})-(\\d{4})')


#parentheses patterns

extrapatterns=[]
extrapatterns.append('([A-Za-z0-9.]+)\s*at\s*([A-Za-z0-9]{2,})\s*dot\s*([A-Z
extrapatterns.append('([A-Za-z0-9.]+)\s*at\s*([A-Za-z0-9]{2,})\s*([A-Za-z0-9
extrapatterns.append('([A-Za-z0-9.]+)\s*at\s*([A-Za-z0-9]{2,})\s*;\s*([A-Za-
```

```
"""
This function takes in a filename along with the file object and
scans its contents against regex patterns. It returns a list of
(filename, type, value) tuples where type is either an 'e' or a 'p'
for e-mail or phone, and value is the formatted phone number or e-mail.
The canonical formats are:
     (name, 'p', '###-###-#####')
     (name, 'e', 'someone@something')
If the numbers you submit are formatted differently they will not
match the gold answers

TODO
For Part 3, if you have added other lists, you should add
```

```
additional for loops that match the patterns in those lists
and produce correctly formatted results to append to the res list.
"""
dic = {}
def process_file(name, f):
    # note that debug info should be printed to stderr
    # sys.stderr.write('[process_file]\tprocessing file: %s\n' % (path))
    res = []
    for line in f:
        # you may modify the line, using something like substitution
        #    before applying the patterns


        line = re.sub('\s\(followed.+?\@', '@', line)
        # Removing everything between and including "(followed by" and repla

        if '-@-' in line:
            line = re.sub('-', '', line)
            # Removing all occurrences of '-' if the line contains '-@-' pat

        line = re.sub('\s+(?:DOT|dot|DOM)\s+', '.', line)
        line = re.sub('\s+(?:AT|WHERE)\s+', '@', line)
        # Replacing occurrences of 'DOT', 'dot', 'DOM' with '.' and 'AT', 'W
        line = re.sub('&#x40;', '@', line)
        # Substituting '&#x40;' with '@' using regex

        line = re.sub('\s+(?:at)\s+', '@', line)
        # Replacing occurrences of 'at' with '@'

        line = re.sub('\<del\>', '', line)
        # Replacing <del> tag with an empty string

        line = re.sub('\<at\ssymbol\>', '@', line)
        # Replacing '<at symbol>' with '@'



        # email pattern list
        for epat in epatterns:
            # each epat has 2 sets of parentheses so each match will have 2
            matches = re.findall(epat,line)



            if ('com' in epat or 'COM' in epat):                    #for dis
                for m in matches:
                    # string formatting operator % takes elements of list m
                    #   and inserts them in place of each %s in the result s
                    email='%s@%s.com' % m
                    res.append((name,'e',email))
```

```
            elif ('obfuscate' in epat):                                 #for c
                for m in matches:
                    email=m[1]+'@'+m[0]+'.edu'
                    res.append((name,'e',email))
            else:
                for m in matches:
                    # string formatting operator % takes elements of list m
                    #   and inserts them in place of each %s in the result s
                    # email has form  someone@somewhere.edu
                    #email = '%s@%s.edu' % m
                    email = '{}@{}.edu'.format(m[0],m[1])

                    if not epat in dic:
                        dic[epat] = []
                    dic[epat].append(email)
                    res.append((name,'e',email))

        # phone pattern list
        for ppat in ppatterns:
            # each ppat has 3 sets of parentheses so each match will have 3
            matches = re.findall(ppat,line)
            for m in matches:
                # phone number has form  areacode-exchange-number
                #phone = '%s-%s-%s' % m
                phone = '{}-{}-{}'.format(m[0],m[1],m[2])

                if not ppat in dic:
                    dic[ppat] = []
                dic[ppat].append(phone)
                res.append((name,'p',phone))
        for newpat in extrapatterns:
            # each newpat has 3 sets of parentheses so each match will have
            matches = re.findall(newpat,line)
            for m in matches:
                # string formatting operator % takes elements of list m
                #   and inserts them in place of each %s in the result strin
                email = '%s@%s.%s.edu' % m

                res.append((name,'e',email))
    return res

"""
You should not edit this function.
"""
def process_dir(data_path):
    # save complete list of candidates
    guess_list = []
    # save list of filenames
    fname_list = []
```

```
    for fname in os.listdir(data_path):
        if fname[0] == '.':
            continue
        fname_list.append(fname)
        path = os.path.join(data_path,fname)
        f = open(path,'r', encoding='latin-1')
        # get all the candidates for this file
        f_guesses = process_file(fname, f)
        guess_list.extend(f_guesses)
    return guess_list, fname_list

"""
You should not edit this function.
Given a path to a tsv file of gold e-mails and phone numbers
this function returns a list of tuples of the canonical form:
(filename, type, value)
"""
def get_gold(gold_path):
    # get gold answers
    gold_list = []
    f_gold = open(gold_path,'r', encoding='latin-1')
    for line in f_gold:
        gold_list.append(tuple(line.strip().split('\t')))
    return gold_list

"""
You should not edit this function.
Given a list of guessed contacts and gold contacts, this function
    computes the intersection and set differences, to compute the true
    positives, false positives and false negatives.
It also takes a dictionary that gives the guesses for each filename,
    which can be used for information about false positives.
Importantly, it converts all of the values to lower case before comparing.
"""
def score(guess_list, gold_list, fname_list):
    guess_list = [(fname, _type, value.lower()) for (fname, _type, value) in
    gold_list = [(fname, _type, value.lower()) for (fname, _type, value) in
    guess_set = set(guess_list)
    gold_set = set(gold_list)

    # for each file name, put the golds from that file in a dict
    gold_dict = {}
    for fname in fname_list:
        gold_dict[fname] = [gold for gold in gold_list if fname == gold[0]]

    tp = guess_set.intersection(gold_set)
    fp = guess_set - gold_set
    fn = gold_set - guess_set
```

```
    pp = pprint.PrettyPrinter()
    #print 'Guesses (%d): ' % len(guess_set)
    #pp.pprint(guess_set)
    #print 'Gold (%d): ' % len(gold_set)
    #pp.pprint(gold_set)

    print ('True Positives (%d): ' % len(tp))
    # print all true positives
    pp.pprint(tp)
    print ('False Positives (%d): ' % len(fp))
    # for each false positive, print it and the list of gold for debugging
    for item in fp:
        fp_name = item[0]
        pp.pprint(item)
        fp_list = gold_dict[fp_name]
        for gold in fp_list:
            s = pprint.pformat(gold)
            print('   gold: ', s)
    print ('False Negatives (%d): ' % len(fn))
    # print all false negatives
    pp.pprint(fn)
    print ('Summary: tp=%d, fp=%d, fn=%d' % (len(tp),len(fp),len(fn)))

"""
You should not edit this function.
It takes in the string path to the data directory and the gold file
"""
def main(data_path, gold_path):
    guess_list, fname_list = process_dir(data_path)
    gold_list =  get_gold(gold_path)
    score(guess_list, gold_list, fname_list)


"""
commandline interface assumes that you are in the directory containing "data
It then processes each file within that data folder and extracts any
matching e-mails or phone numbers and compares them to the gold file
"""
if __name__ == '__main__':
    print ('Assuming ContactFinder.py called in directory with data folder')
    main('data/dev', 'data/devGOLD')
```

```
Assuming ContactFinder.py called in directory with data folder
True Positives (113):
{('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'),
('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', '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', 'ullman@cs.stanford.edu'),
('ullman', 'p', '650-494-8016'),
('ullman', 'p', '650-725-2588'),
('ullman', 'p', '650-725-4802'),
('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 (37):
('ouster', 'e', 'ouster@cs.stanfor.edu')
   gold:  ('ouster', 'e', 'teresa.lynn@stanford.edu')
   gold:  ('ouster', 'e', 'ouster@cs.stanford.edu')
('rinard', 'e', 'rinard@lcs.mi.edu')
   gold:  ('rinard', 'e', 'rinard@lcs.mit.edu')
   gold:  ('rinard', 'p', '617-253-1221')
   gold:  ('rinard', 'p', '617-258-6922')
('jure', 'e', 'server@cs.stanfor.edu')
('thm', 'e', 'pkrokel@stanfor.edu')
   gold:  ('thm', 'e', 'pkrokel@stanford.edu')
   gold:  ('thm', 'p', '650-725-3383')
   gold:  ('thm', 'p', '650-725-3636')
   gold:  ('thm', 'p', '650-725-3938')
('levoy', 'e', 'melissa@graphics.stanfor.edu')
   gold:  ('levoy', 'e', 'ada@graphics.stanford.edu')
   gold:  ('levoy', 'e', 'melissa@graphics.stanford.edu')
   gold:  ('levoy', 'p', '650-723-0033')
   gold:  ('levoy', 'p', '650-724-6865')
   gold:  ('levoy', 'p', '650-725-3724')
   gold:  ('levoy', 'p', '650-725-4089')
('hager', 'e', 'hager@cs.jh.edu')
   gold:  ('hager', 'e', 'hager@cs.jhu.edu')
   gold:  ('hager', 'p', '410-516-5521')
   gold:  ('hager', 'p', '410-516-5553')
   gold:  ('hager', 'p', '410-516-8000')
('engler', 'e', 'engler@stanfor.edu')
   gold:  ('engler', 'e', 'engler@lcs.mit.edu')
   gold:  ('engler', 'e', 'engler@stanford.edu')
('cheriton', 'e', 'cheriton@cs.stanfor.edu')
   gold:  ('cheriton', 'e', 'cheriton@cs.stanford.edu')
   gold:  ('cheriton', 'e', 'uma@cs.stanford.edu')
   gold:  ('cheriton', 'p', '650-723-1131')
   gold:  ('cheriton', 'p', '650-725-3726')
('eroberts', 'e', 'eroberts@cs.stanfor.edu')
   gold:  ('eroberts', 'e', 'eroberts@cs.stanford.edu')
   gold:  ('eroberts', 'p', '650-723-3642')
   gold:  ('eroberts', 'p', '650-723-6092')
('nick', 'e', 'nick.parlante@cs.stanfor.edu')
   gold:  ('nick', 'e', 'nick.parlante@cs.stanford.edu')
   gold:  ('nick', 'p', '650-725-4727')
```

```
('jure', 'e', 'server@cs.stanford.edu')
('fedkiw', 'e', 'fedkiw@cs.stanfor.edu')
   gold:  ('fedkiw', 'e', 'fedkiw@cs.stanford.edu')
('lam', 'e', 'lam@cs.stanfor.edu')
   gold:  ('lam', 'e', 'lam@cs.stanford.edu')
   gold:  ('lam', 'p', '650-725-3714')
   gold:  ('lam', 'p', '650-725-6949')
('serafim', 'e', 'serafim@cs.stanfor.edu')
   gold:  ('serafim', 'e', 'serafim@cs.stanford.edu')
   gold:  ('serafim', 'p', '650-723-3334')
   gold:  ('serafim', 'p', '650-725-1449')
('subh', 'e', 'subh@stanfor.edu')
   gold:  ('subh', 'e', 'subh@stanford.edu')
   gold:  ('subh', 'e', 'uma@cs.stanford.edu')
   gold:  ('subh', 'p', '650-724-1915')
   gold:  ('subh', 'p', '650-725-3726')
   gold:  ('subh', 'p', '650-725-6949')
('subh', 'e', 'uma@cs.stanfor.edu')
   gold:  ('subh', 'e', 'subh@stanford.edu')
   gold:  ('subh', 'e', 'uma@cs.stanford.edu')
   gold:  ('subh', 'p', '650-724-1915')
   gold:  ('subh', 'p', '650-725-3726')
   gold:  ('subh', 'p', '650-725-6949')
('hanrahan', 'e', 'hanrahan@cs.stanfor.edu')
   gold:  ('hanrahan', 'e', 'hanrahan@cs.stanford.edu')
   gold:  ('hanrahan', 'p', '650-723-0033')
   gold:  ('hanrahan', 'p', '650-723-8530')
('widom', 'e', 'widom@cs.stanfor.edu')
   gold:  ('widom', 'e', 'siroker@cs.stanford.edu')
   gold:  ('widom', 'e', 'widom@cs.stanford.edu')
   gold:  ('widom', 'p', '650-723-0872')
   gold:  ('widom', 'p', '650-723-7690')
   gold:  ('widom', 'p', '650-725-2588')
('latombe', 'e', 'latombe@cs.stanfor.edu')
   gold:  ('latombe', 'e', 'asandra@cs.stanford.edu')
   gold:  ('latombe', 'e', 'latombe@cs.stanford.edu')
   gold:  ('latombe', 'e', 'liliana@cs.stanford.edu')
   gold:  ('latombe', 'p', '650-721-6625')
   gold:  ('latombe', 'p', '650-723-0350')
   gold:  ('latombe', 'p', '650-723-4137')
   gold:  ('latombe', 'p', '650-725-1449')
('kunle', 'e', 'kunle@ogun.stanfor.edu')
   gold:  ('kunle', 'e', 'darlene@csl.stanford.edu')
   gold:  ('kunle', 'e', 'kunle@ogun.stanford.edu')
   gold:  ('kunle', 'p', '650-723-1430')
   gold:  ('kunle', 'p', '650-725-3713')
   gold:  ('kunle', 'p', '650-725-6949')
('plotkin', 'e', 'server@infolab.stanford.edu')
('psyoung', 'e', 'patrick.young@stanfor.edu')
   gold:  ('psyoung', 'e', 'patrick.young@stanford.edu')
```

```
('balaji', 'e', 'balaji@stanfor.edu')
   gold:  ('balaji', 'e', 'balaji@stanford.edu')
('latombe', 'e', 'asandra@cs.stanfor.edu')
   gold:  ('latombe', 'e', 'asandra@cs.stanford.edu')
   gold:  ('latombe', 'e', 'latombe@cs.stanford.edu')
   gold:  ('latombe', 'e', 'liliana@cs.stanford.edu')
   gold:  ('latombe', 'p', '650-721-6625')
   gold:  ('latombe', 'p', '650-723-0350')
   gold:  ('latombe', 'p', '650-723-4137')
   gold:  ('latombe', 'p', '650-725-1449')
('zelenski', 'e', 'zelenski@cs.stanfor.edu')
   gold:  ('zelenski', 'e', 'zelenski@cs.stanford.edu')
   gold:  ('zelenski', 'p', '650-723-6092')
   gold:  ('zelenski', 'p', '650-725-8596')
('ouster', 'e', 'teresa.lynn@stanfor.edu')
   gold:  ('ouster', 'e', 'teresa.lynn@stanford.edu')
   gold:  ('ouster', 'e', 'ouster@cs.stanford.edu')
('engler', 'e', 'engler@lcs.mi.edu')
   gold:  ('engler', 'e', 'engler@lcs.mit.edu')
   gold:  ('engler', 'e', 'engler@stanford.edu')
('shoham', 'e', 'shoham@stanfor.edu')
   gold:  ('shoham', 'e', 'shoham@stanford.edu')
   gold:  ('shoham', 'p', '650-723-3432')
   gold:  ('shoham', 'p', '650-725-1449')
('kosecka', 'e', 'kosecka@cs.gm.edu')
   gold:  ('kosecka', 'e', 'kosecka@cs.gmu.edu')
   gold:  ('kosecka', 'p', '703-993-1710')
   gold:  ('kosecka', 'p', '703-993-1876')
('levoy', 'e', 'ada@graphics.stanfor.edu')
   gold:  ('levoy', 'e', 'ada@graphics.stanford.edu')
   gold:  ('levoy', 'e', 'melissa@graphics.stanford.edu')
   gold:  ('levoy', 'p', '650-723-0033')
   gold:  ('levoy', 'p', '650-724-6865')
   gold:  ('levoy', 'p', '650-725-3724')
   gold:  ('levoy', 'p', '650-725-4089')
('widom', 'e', 'siroker@cs.stanfor.edu')
   gold:  ('widom', 'e', 'siroker@cs.stanford.edu')
   gold:  ('widom', 'e', 'widom@cs.stanford.edu')
   gold:  ('widom', 'p', '650-723-0872')
   gold:  ('widom', 'p', '650-723-7690')
   gold:  ('widom', 'p', '650-725-2588')
('kunle', 'e', 'darlene@csl.stanfor.edu')
   gold:  ('kunle', 'e', 'darlene@csl.stanford.edu')
   gold:  ('kunle', 'e', 'kunle@ogun.stanford.edu')
   gold:  ('kunle', 'p', '650-723-1430')
   gold:  ('kunle', 'p', '650-725-3713')
   gold:  ('kunle', 'p', '650-725-6949')
('nass', 'e', 'nass@stanfor.edu')
   gold:  ('nass', 'e', 'nass@stanford.edu')
   gold:  ('nass', 'p', '650-723-5499')
```

```
   gold:  ('nass', 'p', '650-725-2472')
('zm', 'e', 'manna@cs.stanfor.edu')
   gold:  ('zm', 'e', 'manna@cs.stanford.edu')
   gold:  ('zm', 'p', '650-723-4364')
   gold:  ('zm', 'p', '650-725-4671')
('latombe', 'e', 'liliana@cs.stanfor.edu')
   gold:  ('latombe', 'e', 'asandra@cs.stanford.edu')
   gold:  ('latombe', 'e', 'latombe@cs.stanford.edu')
   gold:  ('latombe', 'e', 'liliana@cs.stanford.edu')
   gold:  ('latombe', 'p', '650-721-6625')
   gold:  ('latombe', 'p', '650-723-0350')
   gold:  ('latombe', 'p', '650-723-4137')
   gold:  ('latombe', 'p', '650-725-1449')
('dlwh', 'e', 'dlwh@stanfor.edu')
   gold:  ('dlwh', 'e', 'dlwh@stanford.edu')
('plotkin', 'e', 'server@infolab.stanfor.edu')
False Negatives (4):
{('jks', 'e', 'jks@robotics.stanford.edu'),
 ('pal', 'e', 'pal@cs.stanford.edu'),
 ('ullman', 'e', 'support@gradiance.com'),
 ('vladlen', 'e', 'vladlen@stanford.edu')}
Summary: tp=113, fp=37, fn=4
```

Part 3

1st Part:

extrapatterns=[] extrapatterns.append('([A-Za-z0-9.]+)\s*at*|s([A-Za-z0-9]{2,})\s*dot*|s([A-Za-z0-9_]{2,})\s*dot*|sedu') extrapatterns.append('([A-Za-z0-9.]+)\s*at*|s([A-Za-z0-9]{2,})\s*([A-Za-z0-9_]{2,})*|sedu') extrapatterns.append('([A-Za-z0-9.]+)\s*at*|s([A-Za-z0-9]{2,})\s*;*|s([A-Za-z0-9_]{2,})\s*;*|sedu')

The code defines an empty list called extrapatterns. Three regular expressions are then appended to this list. Here's a breakdown of each regex:

([A-Za-z0-9.]+)\s*at*|s([A-Za-z0-9]{2,})\s*dot*|s([A-Za-z0-9_]{2,})\s*dot*|sedu Matches patterns like name at domain dot subdomain dot edu. Captures the name, domain, and subdomain separately. ([A-Za-z0-9.]+)\s*at*|s([A-Za-z0-9]{2,})\s*([A-Za-z0-9_]{2,})*|sedu Matches patterns like name at domain subdomain edu. Captures the name, domain, and subdomain separately. ([A-Za-z0-9.]+)\s*at*|s([A-Za-z0-9]{2,})\s*;*|s([A-Za-z0-9_]{2,})\s*;*|sedu Matches patterns like name at domain ; subdomain ; edu. Captures the name, domain, and subdomain separately. These patterns can be used for matching and extracting information from text that follows these specific formats.

2nd Part: New Update process file Logic

The new patterns will iterates over each pattern in the `extrapatterns` list and performs the following steps:

1. It uses `re.findall()` to find all non-overlapping matches of the pattern `newpat` in the `line`.
2. For each match found, it enters a loop and assigns the match groups to the variable `m`. Since each `newpat` has 3 sets of parentheses, each `m` will be a list containing 3 items.
3. It uses string formatting to construct an email address by inserting the elements of `m` into the format string `'%s@%s.%s.edu'`. The elements of `m` correspond to the name, domain, and subdomain of the email address.
4. The constructed email address is then appended to the `res` list as a tuple, where the first element is the `name`, the second element is `'e'` (to indicate that it is an email), and the third element is the constructed email address.

In summary, the code extracts information from the `line` using the regular expressions in `extrapatterns` and constructs email addresses based on the matched patterns. It then adds these email addresses to the `res` list for further processing or storage.

The given code snippet performs various string replacements using regular expressions:

1. `line = re.sub('\s\(followed.+?\@', '@', line)`: This replaces everything between and including "(followed by" and "@" with just "@".

2. `if '-@-' in line: line = re.sub('-', '', line)`: If the line contains the pattern '-@-', it removes all occurrences of '-' from the line.

3. `line = re.sub('\s+(?:DOT|dot|DOM)\s+', '.', line)`: It replaces occurrences of 'DOT', 'dot', or 'DOM' surrounded by whitespace with '.' in the line.

4. `line = re.sub('\s+(?:AT|WHERE)\s+', '@', line)`: It replaces occurrences of 'AT' or 'WHERE' surrounded by whitespace with '@' in the line.

5. `line = re.sub('&#x40;', '@', line)`: It substitutes the character sequence '@' with '@' using a regular expression.

6. `line = re.sub('\s+(?:at)\s+', '@', line)`: It replaces occurrences of 'at' surrounded by whitespace with '@' in the line.

7. `line = re.sub('\<del\>', '', line)`: It replaces the '~~' tag with an empty

~~string.~~

8. `line = re.sub('\<at\ssymbol\>', '@', line)` : It replaces the '' with '@' in the line.

Each of these regular expression substitutions is applied to the `line` variable, modifying it accordingly.

The Final output has 113 True Positives and 37 False positives and 4 false negatives

In [ ]: