REGEX EMAILS

Ali Ho & Kendra Osburn | NLP | 5/9/19

Overall, seven distinct methods were attempted to solve this problem. Initially, patterns were added to the epatterns array (attempts 1-3). This proved ineffective as certain patterns would overwrite other patterns and produce inaccurate false positives. The next change was combining all patterns into one master regex line and only adding additional patterns for extreme outliers. This worked well until the final 12.

The final way we approached this was to come up with a uniform regex at the top to capture all email addresses. For edge cases, we added substitutions to the **process_file** function (using the python regex package re) so the edge case emails would be formatted in a way that our "upper regex" (the original regex at the top) would catch and capture these re-formatted emails. This makes this file and these functions more reusable across different scenarios (e.g. not just with '.edu' addresses). Here was our process:

PHONE NUMBERS

Started with a fresh copy of ContactFinder.base.py

STARTING POINT
tp=0, fp=0, fn=117

CAPTURING
Phone numbers separated by a - xxx-yyy-zzzz
REGEX
`'(\d{3})-(\d{3})-(\d{4})'`
IN ENGLISH
Return three sets of digits (the first and second set have 3 digits, the third set has 4 digits) separated by '-'
RESULT
tp=19, fp=0, fn=98

CAPTURING

Phone numbers separated by a .
xxx.yyy.zzzz

REGEX

```
'(\d{3}).(\d{3}).(\d{4})'
```

IN ENGLISH

Return three sets of digits (the first and second set have 3 digits, the third set has 4 digits) separated by '.'

RESULT

tp=31, fp=9, fn=86

CAPTURING

Phone numbers formatted like this (xxx) yyy-zzzz

REGEX

```
'\((\d{3})\)\s(\d{3})-+(\d{4})'
```

IN ENGLISH

Return three sets of digits (the first and second set have 3 digits, the third set has 4 digits) the first set surrounded by '( )' followed by the second and third set separated by '-'

RESULT

tp=70, fp=9, fn=47

CAPTURING

Phone numbers formatted like this [xxx] yyy-zzzz

REGEX

```
'\[(\d{3})\]\s(\d{3})-+(\d{4})'
```

IN ENGLISH

Return three sets of digits (the first and second set have 3 digits, the third set has 4 digits) the first set surrounded by '[ ]' followed by the second and third set separated by '-'

RESULT
tp=72, fp=9, fn=45

CAPTURING
Trying to un-capture the false positives, numbers like filenames or urls **(e.g. <a href="**http://www.amazon.com/exec/obidos/ASIN/1575860538/qid=1110995710/sr=2-1/ref=pd_bbs_b_2_1/104-3375742-1353522**">)** *Note: This line replaces two previous lines*
REGEX
`'(\d{3})(?:[\W]{1})(\d{3})(?:[\W]{1})(\d{4})'`
IN ENGLISH
Return three sets of digits (the first and second set have 3 digits, the third set has 4 digits) each separated by a single (but unspecified) non-alphanumeric character
RESULT
tp=72, fp=0, fn=45

EMAILS

We refactored the **process_file** function to take three parenthesis so it can capture '.com' as well as '.edu'

BEFORE:

```
email = '{}@{}.edu'.format(m[0],m[1])
```

AFTER:

```
email = '{}@{}.{}'.format(m[0], m[1], m[2])
```

CAPTURING
Traditional emails with the format **someone@something.edu**
REGEX
`'([A-Za-z]+)@([A-Za-z]+)\.([A-Za-z]+)'`

IN ENGLISH

Two sets of one or more (+) of characters (A-Z or a-z) separated by an @
Both before a . and another set of one or more (+) characters (A-z or a-z)

RESULT

tp=76, fp=16, fn=41

Example of some of the false positives:

```
False Positives (16):
('eroberts', 'e', 'eroberts@cs.stanford')
   gold:  ('eroberts', 'e', 'eroberts@cs.stanford.edu')
   gold:  ('eroberts', 'p', '650-723-3642')
   gold:  ('eroberts', 'p', '650-723-6092')
('nick', 'e', 'parlante@cs.stanford')
   gold:  ('nick', 'e', 'nick.parlante@cs.stanford.edu')
   gold:  ('nick', 'p', '650-725-4727')
('cheriton', 'e', 'uma@cs.stanford')
   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')
```

CAPTURING

Traditional emails with the format **someone@something.somethingelse.edu**
NOTE: This line replaced the previous line, otherwise the previous line flagged false positives

REGEX

```
'([A-Za-z.]+)@([A-Za-z.]+)\.([A-Za-z]+)'
```

IN ENGLISH

Same as original, with the added . character (which, inside the brackets, does not need an escape character).

RESULT

tp=92, fp=0, fn=25

CAPTURING

Emails with the format **someone @ something.somethingelse.edu**
NOTE: This line replaced the previous line, otherwise the previous line flagged false positives

REGEX

```
'([A-Za-z.]+)\s*@\s*([A-Za-z.]+)\.([A-Za-z]+)'
```

IN ENGLISH

Same as original, with the added \s* regex around the @ sign. This means a space can (but doesn't have to be) around the @ sign. (\s denotes space, * denotes zero or more)

RESULT

tp=96, fp=0, fn=21

CAPTURING

manning: manning <at symbol> cs.stanford.edu
Note: this line went inside the process_file function

REGEX

```
line = re.sub('\<at\ssymbol\>', '@', line)
```

IN ENGLISH

This uses the python regex package (re) to substitute `<at symbol>` for @

RESULT

tp=98, fp=0, fn=19

CAPTURING

latombe: latombe<del>@cs.stanford.edu
Note: this line went inside the process_file function

REGEX

```
line = re.sub('\<del\>', '', line)
```

IN ENGLISH

This uses the python regex package (re) to remove <del> by substituting <del> for an empty space

RESULT

tp=101, fp=0, fn=16

CAPTURING

levoy: ada@graphics.stanford.edu
Note: this line went inside the process_file function

REGEX

```
line = re.sub('&#x40;', '@', line)
```

IN ENGLISH

This uses the python regex package (re) to substitute `&#x40` for @

RESULT

tp=103, fp=0, fn=14

CAPTURING

subh: Email: subh AT stanford DOT edu
engler: engler WHERE stanford DOM edu
Note: this line went inside the process_file function

REGEX

```
line = re.sub('\s+(?:DOT|dot|DOM)\s+', '.', line)
line = re.sub('\s+(?:AT|WHERE)\s+', '@', line)
```

IN ENGLISH

This uses the python regex package (re) to substitute `&#x40` for @

RESULT

tp=105, fp=0, fn=12

EMAIL EDGE CASES

Here is where things get particularly tricky. There are many ways to address these unusual emails that were precisely intended to deter crazy regex hackers like us, but there are only so many hours and so many outliers, so we had to draw the line somewhere. Here is our best attempt to catch these edge cases (after four other attempts):

CAPTURING
lam: E-mail: lam at cs.stanford.edu
REGEX
`'([A-Za-z._0-9]+)\s*(?:@|\sat\s)\s*([A-Za-z._0-9]+)(?:\.|\s)([A-Za-z]+)'`
IN ENGLISH
One or more characters, zero or more spaces, EITHER an @ sign or an ‘ at ‘ (at surrounded by two spaces, otherwise this would capture things like ‘cat’ or ‘hatter’)
RESULT
tp=109, fp=40, fn=8

At first glance, it appears as if we can/should simply add \s around our ‘at’ in our upper regex. Unfortunately, this lead to problems and 40 false positives. Sample shown below:

```
('manning', 'e', 'funding@stanford.comes')
   gold:  ('manning', 'e', 'dbarros@cs.stanford.edu')
   gold:  ('manning', 'e', 'manning@cs.stanford.edu')
   gold:  ('manning', 'p', '650-723-7683')
   gold:  ('manning', 'p', '650-725-1449')
   gold:  ('manning', 'p', '650-725-3358')
('kunle', 'e', 'is@the.center')
   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')
```

This is taking the ‘ at ‘ from the sentence ‘funding at stanford comes’ and the ‘ at ‘ from the sentence ‘is at the center.’ Clearly, this is not what the code should actually be capturing.

The first attempt to remedy this was forcing the final characters after the last period to be only 3 characters long.

```
'([A-Za-z._0-9]+)\s*(?:@|\sat\s)\s*([A-Za-z._0-9]+)(?:\.|\s)([A-Za-z]{3})'
```

This produced 35 false positives, including the following:

```
('fedkiw', 'e', 'of@least.two')
   gold:  ('fedkiw', 'e', 'fedkiw@cs.stanford.edu')
('manning', 'e', 'directed@students.who')
   gold:  ('manning', 'e', 'dbarros@cs.stanford.edu')
   gold:  ('manning', 'e', 'manning@cs.stanford.edu')
   gold:  ('manning', 'p', '650-723-7683')
   gold:  ('manning', 'p', '650-725-1449')
   gold:  ('manning', 'p', '650-725-3358')
('levoy', 'e', 'taken@the.top')
   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')
```

This is taking any occurance of a three letter word happening after the word ' at ' (e.g. in the sentence 'of at least two') and considering it an email, which it is not.

The second attempt to remedy this was forcing the final characters after the last period to be either 'com' or 'edu' (followed by a \W so we wouldn't replicate the 'cat' or 'hatter' issue above).

CAPTURING
lam: E-mail: lam at cs.stanford.edu
REGEX
`'([A-Za-z._0-9]+)\s*(?:@|\sat\s)\s*([A-Za-z._0-9]+)(?:\.|\s)(com|edu)\W'`
IN ENGLISH
One or more characters, digits or periods, zero or more spaces, EITHER an @ sign or an ' at ' (at surrounded by two spaces, otherwise this would capture things like 'cat' or 'hatter'), zero or more spaces, one or more characters, digits or periods, either a period or a space, ending with either 'com' or 'edu' followed by a non-alphanumeric character (so it won't capture things like .computer)
RESULT
tp=108, fp=2, fn=9

```
False Positives (2):
```

```
('plotkin', 'e', 'server@infolab.stanford.edu')
('jure', 'e', 'server@cs.stanford.edu')
```

Upon further inspection, these two false positives were edge cases on their own and definitely something we'd want to weed out -- we do not want or need to be emailing a server.

```
line = re.sub('[sS]erver', '', line)
```

CAPTURING
jurafsky: jks at robotics;stanford;edu
REGEX
`'([A-Za-z._0-9]+)\s*(?:@|\sat\s)\s*([A-Za-z.;_0-9]+)\s*(?:\.|\s|;)((?:com|edu))\W'`
IN ENGLISH
Same as above with the addition of ';' to the 'somewhere' character capture and ';' as another option (in addition to . or space) before 'com' or 'edu'
RESULT
tp=108, fp=1, fn=9

```
False Positives (1):
('jks', 'e', 'jks@robotics;stanford.edu')
   gold:  ('jks', 'e', 'jks@robotics.stanford.edu')
```

Added this line to the for epat in epatterns loop to turn any "somewhere;somewhereelse" with a ; into a "somewhere.somewhereelse"

CAPTURING
jurafsky: jks at robotics;stanford;edu
REGEX
`somewhere = re.sub('(?:\s|\;)', '.', m[1])` `email = '{}@{}.{}'.format(m[0],somewhere, m[2])`
IN ENGLISH

Added this line to the for epat in epatterns loop to turn any “somewhere;somewhereelse” with a ; into a “somewhere.somewhereelse”

RESULT

tp=109, fp=0, fn=8

CAPTURING

vladlen: vladlen at <!-- die!--> stanford <!-- spam pigs!--> dot <!-- die!--> edu

REGEX

```
'([A-Za-z._0-9]+)\s*(?:@|\sat\s)\s*([A-Za-z.;_0-9]+)\s*(?:\.|\s|;)\s*((?:com|edu))\W'
line = re.sub('\s?\<\!.+?\>\s?', ' ', line)
```

IN ENGLISH

Added \s* to our upper regex to allow for ‘. edu’
Added the re.sub line to remove everything between and including <!-- and -- >

RESULT

tp=110, fp=0, fn=7

CAPTURING

ouster: ouster (followed by “@cs.stanford.edu”)

REGEX

```
line = re.sub('\s\(followed.+?\@', '@', line)
```

IN ENGLISH

Added the re.sub line to remove everything between and including “followed by” and replaced it with an @

RESULT

tp=112, fp=0, fn=5

CAPTURING

ullman: email to support at gradiance dt com

REGEX

```
line = re.sub('\s+(?:DOT|dot|DOM|dt)\s+', '.', line)
```

IN ENGLISH

Added the ‘dt’ to the resub line already in place to substitute ‘dot’ for .

RESULT

tp=113, fp=0, fn=4

CAPTURING

dlwh: d-l-w-h-@-s-t-a-n-f-o-r-d-.-e-d-u

REGEX

```
'([A-Za-z._0-9-]+)@([A-Za-z.;_0-9-]+)\.(-e-d-u)'

somewhere = re.sub('(?:\s|\;)', '.', m[1])
someone = re.sub('-', '', m[0])
somewhere = re.sub('-', '', somewhere)
tld = re.sub('-', '', m[2])
email = '{}@{}.{}'.format(someone,somewhere, tld)
```

IN ENGLISH

Removed every instance of a ‘-’ for every “email” that passed correctly through the upper level regex constructed specifically for this email

RESULT

tp=114, fp=0, fn=3

CAPTURING

dlwh: d-l-w-h-@-s-t-a-n-f-o-r-d-.-e-d-u

REGEX

```
if '-@-' in line:
    line = re.sub('-', '', line)
```

IN ENGLISH
Much shorter solution to this problem
RESULT
tp=114, fp=0, fn=3

CAPTURING
pal: email: pal at cs stanford edu
REGEX
`'([A-Za-z._0-9]+)\s*(?:@|\sat\s)\s*([A-Za-z.;_0-9]+\s?[A-Za-z.;_0-9]+)\s*(?:\.|\s|;)\s*((?:com|edu))\W'`
IN ENGLISH
Added the possibilities of spaces between the periods with \s*
RESULT
tp=114, fp=1, fn=3

```
False Positives (1):
('vladlen', 'e', 'vladlen@stanford...edu')
   gold:  ('vladlen', 'e', 'vladlen@stanford.edu')
```

Corrected this line to remove the trailing space:

```
line = re.sub('\s?\<\!.+?\>', '', line)
```

Added this line to remove any excess periods:

```
somewhere = somewhere.rstrip('.')
```

RESULT
tp=115, fp=0, fn=2

Note: The below should not be considered an “edge case” -- It just slipped by. It was very simply addressed by minor edits to the upper regex.

CAPTURING

cheriton: uma at cs.Stanford.EDU

REGEX

```
'([A-Za-z._0-9]+)\s*(?:@|\sat\s)\s*([A-Za-z.;_0-9]+\s?[A-Za-z.;_0-9]+)\s*(?:\.|\s|;)\s*((?:com|[eE][dD][uU]))\W'
```

IN ENGLISH

Changed ‘edu’ to [eE][dD][uU] so it can account for edu, EDU, Edu, eDu ,EDu etc.

RESULT

tp=116, fp=0, fn=1

CAPTURING

jurafsky: obfuscate('stanford.edu','jurafsky');

REGEX

```
'obfuscate\(\'(.+?)\.(.+?)\'\,\'(.+?)\'\)'

if len(m[1]) == 3:
  someone = m[2]
  somewhere = m[0]
  tld = m[1]
```

IN ENGLISH

Added the first line to the upper regex to catch this “obfuscate” function.
Added the conditional inside the epat loop to swap positions of what the upper regex returned (e.g. upper regex returned (stanford)(edu)(jurafsky) and the conditional turned this into the pattern the checker was looking for)

RESULT

tp=117, fp=0, fn=0

MICROSOFT EMAIL EXAMPLES

The below emails are considered “good” or “acceptable” emails by microsoft. This list was found here.

These were added to the “devGOLD” file

```
zmicrosoft  e     email@domain.com
zmicrosoft  e     firstname.lastname@domain.com
zmicrosoft  e     email@subdomain.domain.com
zmicrosoft  e     firstname+lastname@domain.com
zmicrosoft  e     email@domain-one.com
zmicrosoft  e     1234567890@domain.com
zmicrosoft  e     _______@domain.com
zmicrosoft  e     firstname-lastname@domain.com
zmicrosoft  e     email@domain.name
zmicrosoft  e     email@domain.co.jp
zmicrosoft  e     "email"@domain.com
zmicrosoft  e     email@123.123.123.123
zmicrosoft  e     email@[123.123.123.123]
```

This was added as a new file (**zmicrosoft**, the z being added simply for ease of tracking) in the /dev folder

```
email@domain.com  Valid email
firstname.lastname@domain.com Email contains dot in the address field
email@subdomain.domain.com    Email contains dot with subdomain
firstname+lastname@domain.com Plus sign is considered valid character
email@domain-one.com    Dash in domain name is valid
1234567890@domain.com   Digits in address are valid
email@domain-one.com    Dash in domain name is valid
zmicrosoft  e     email@domain-one.com
_______@domain.com      Underscore in the address field is valid
firstname-lastname@domain.com Dash in address field is valid
email@domain.name "name" is valid Top Level Domain name
email@domain.co.jp      Dot in Top Level Domain name also considered valid
(use co.jp as example here)
"email"@domain.com      Quotes around email is considered valid
```

STARTING POINT
tp=117, fp=0, fn=0

These passed easily with the regex we already had:

```
('zmicrosoft', 'e', 'email@domain.com'),
('zmicrosoft', 'e', 'email@subdomain.domain.com'),
('zmicrosoft', 'e', 'firstname.lastname@domain.com')
```

Note: Took a slightly different approach and went one by one

CAPTURING
microsoft: firstname+lastname@domain.com
REGEX
`([A-Za-z._0-9+]+)\s*(?:@|\sat\s)\s*([A-Za-z.;_0-9]+\s? (cont')...`
IN ENGLISH
Added the "+" character to our upper regex
RESULT
tp=121, fp=0, fn=0

CAPTURING
microsoft: email@domain-one.com
REGEX
`'([A-Za-z._0-9+]+)\s*(?:@|\sat\s)\s*([A-Za-z.;_0-9]+\s?[A-Za-z.;_0-9-]+)\s* (cont')`
IN ENGLISH
Added the "-" character to our upper regex
RESULT
tp=122, fp=0, fn=0

Passed as-is

```
('zmicrosoft', 'e', '1234567890@domain.com')
('zmicrosoft', 'e', '_______@domain.com'),
('zmicrosoft', 'e', 'firstname-lastname@domain.com'),
```

RESULT

tp=125, fp=0, fn=0

CAPTURING

microsoft: email@domain.name & email@domain.co.jp

REGEX

```
(cont’) \s*((?:com|jp|[eE][dD][uU]|name))\W')
```

IN ENGLISH

Added ‘name’ and ‘jp’ to our accepted list of TLDs

RESULT

tp=127, fp=0, fn=0

CAPTURING

microsoft: “email”@domain.com

REGEX

```
ONE: '(\"(?:[A-Za-z._0-9+-]+)\")\@([A-Za-z._0-9+-]+)\.(com)'
TWO: '(\"?[A-Za-z._0-9+-]+\"?)\s*(?:@|\sat\s)\s* (cont’)
```

IN ENGLISH

Originally, we thought a whole new pattern was needed (ONE)
Then, we realized we could just add \”? (translation, zero or one “) to the capture statement for “someone” (TWO)

RESULT

tp=128, fp=2, fn=0

CAPTURING

microsoft: email@123.123.123.123 & email@[123.123.123.123]

REGEX

```
ONE: '(\"(?:[A-Za-z._0-9+-]+)\")\@([A-Za-z._0-9+-]+)\.(com)'
TWO: '(\"?[A-Za-z._0-9+-]+\"?)\s*(?:@|\sat\s)\s* (cont')
```

IN ENGLISH

Originally, we thought a whole new pattern was needed (ONE)
Then, we realized we could just add \"? (translation, zero or one ") to the capture statement for "someone" (TWO). Unfortunately, this gave us false positives, so we ended up back with our first attempt (a new pattern -- ONE).

RESULT

tp=128, fp=0, fn=0

The last two on the microsoft "good email" list we didn't have time to account for were the two with IP addresses.

```
email@123.123.123.123
email@[123.123.123.123]
```

We left that as a challenge for next time.

Final output including the additional microsoft emails:

```
True Positives (128):
{('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'),
 ('zmicrosoft', 'e', '"email"@domain.com'),
 ('zmicrosoft', 'e', '1234567890@domain.com'),
 ('zmicrosoft', 'e', '_______@domain.com'),
 ('zmicrosoft', 'e', 'email@domain-one.com'),
 ('zmicrosoft', 'e', 'email@domain.co.jp'),
 ('zmicrosoft', 'e', 'email@domain.com'),
 ('zmicrosoft', 'e', 'email@domain.name'),
 ('zmicrosoft', 'e', 'email@subdomain.domain.com'),
 ('zmicrosoft', 'e', 'firstname+lastname@domain.com'),
 ('zmicrosoft', 'e', 'firstname-lastname@domain.com'),
 ('zmicrosoft', 'e', 'firstname.lastname@domain.com')}
False Positives (0):
False Negatives (0):
set()
Summary: tp=128, fp=0, fn=0
```