**IST 664 – NLP Homework 2**

**5/17/2021**

**David Doman**

**Introduction & Background**

This homework was based off an assignment from the Stanford NLP course known as “SpamLord!”. Spammers are a threat to everyday individuals and are always looking for ways to easily obtain e-mail addresses, phone numbers and many other personal information. Obfuscation has been a way for people to make it more difficult for spammers easily obtain their information. The practice of obfuscation, in simple terms, involves writing or presenting personal information in obscure or non-uniform ways to make simple sweeps of text more arduous for spammers. Common examples of obfuscations include writing e-mail addresses in the form of “ddoman at syr dot edu” or “DdOmAn @ SYR .eDu” or phone numbers in the form of “TEL 555864123” or “PHONE (555)-6257895”. These are all simple techniques that can help to avoid spammers from easily extracting information. Although this makes identification more difficult, it does not mean that spammers cannot create algorithms to identify these obfuscations and return the information in the regular or normal format like “[ddoman@syr.edu](mailto:ddoman@syr.edu)” and “555-689-7052”. Thus, the main idea of this assignment is to sweep through large amounts of text and attempt to identify as many obfuscations of e-mail addresses and phone numbers as possible and return them in their normal or standard form.

The code being used is set up to make a list of regular expression patterns that find e-mail addresses that have two matched parts: Userid and a domain (which is assumed to be .edu). This means that only two sets of parentheses will be used in the code to match e-mail addresses. For phone numbers, three parentheses will be utilized with the first set referring to the area code (XXX), the second set referring to the exchange portion (YYY) and the third part referring to the number part (ZZZZ). The standard form for a phone number is XXX-YYY-ZZZZ. The regular expression pattern starting points for both e-mail addresses and phone numbers are provided but will need to be expanded to identify more complex obfuscations in the text.

**Analysis Process**

I started with attempting to identify e-mail addresses although it may be more complex than dealing with phone numbers. In the following parts of this paper, “someone” and “somewhere” refer the following portions of a standard e-mail address: ***someone@somewhere.edu***

**E-mail Addresses**

The output for the various runs is provided below:

**Initial Run:**

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

True Positives (4):

{('ashishg', 'e', 'ashishg@stanford.edu'),

('ashishg', 'e', 'rozm@stanford.edu'),

('ullman', 'e', 'ullman@cs.stanford.edu'),

('zelenski', 'e', 'zelenski@cs.stanford.edu')}

False Positives (0):

False Negatives (113):

{('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'),

……………

Summary: tp=4, fp=0, fn=113

**Explanation:** This was the initial code that was provided for the assignment in the instructions and only resulted in 4 True Positives. This regular expression (RE) pattern allowed for uppercase or lowercase letters between A & Z and also allowed for a period, for the “someone” and “somewhere” portion of an e-mail address. The final part of the RE is simply “.edu”. The remaining 113 False Negatives still need to be identified.

**2nd Run:**

The initial run requires a space before and after the “@” sign. To remedy the situation where there is no space before or after the @ sign or maybe a space before but not after the “@” sign, I simply changed the “\s” to “\s\*”.

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

True Positives (23):

{('ashishg', 'e', 'ashishg@stanford.edu'),

('ashishg', 'e', 'rozm@stanford.edu'),

('balaji', 'e', 'balaji@stanford.edu'),

('cheriton', 'e', 'cheriton@cs.stanford.edu'),

('dabo', 'e', 'dabo@cs.stanford.edu'),

('engler', 'e', 'engler@lcs.mit.edu'),

('eroberts', 'e', 'eroberts@cs.stanford.edu'),

('fedkiw', 'e', 'fedkiw@cs.stanford.edu'),

('hanrahan', 'e', 'hanrahan@cs.stanford.edu'),

('kosecka', 'e', 'kosecka@cs.gmu.edu'),

('kunle', 'e', 'darlene@csl.stanford.edu'),

('kunle', 'e', 'kunle@ogun.stanford.edu'),

('nass', 'e', 'nass@stanford.edu'),

('nick', 'e', 'nick.parlante@cs.stanford.edu'),

('psyoung', 'e', 'patrick.young@stanford.edu'),

('rinard', 'e', 'rinard@lcs.mit.edu'),

('shoham', 'e', 'shoham@stanford.edu'),

('thm', 'e', 'pkrokel@stanford.edu'),

('ullman', 'e', 'ullman@cs.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')}

False Positives (0):

False Negatives (94):

{('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', 'e', 'uma@cs.stanford.edu'),

('cheriton', 'p', '650-723-1131'),

('cheriton', 'p', '650-725-3726'),

('dabo', 'p', '650-725-3897'),

('dabo', 'p', '650-725-4671'),

('dlwh', 'e', 'dlwh@stanford.edu'),

……………

Summary: tp=23, fp=0, fn=94

**Explanation:** This small change making the space before and after the @ sign not required resulted in 23 True Positives and 94 False Negatives.

**3rd Run:**

To account for “.edu” to be identified if written with uppercase letters, the following changes were made to the end portion of the RE. I initially tried a single “\D” but this returned a false positive for “engler” so I changed this to “\D\D” and it removed the False Positive.

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

True Positives (24):

{('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'),

('engler', 'e', 'engler@lcs.mit.edu'),

('eroberts', 'e', 'eroberts@cs.stanford.edu'),

('fedkiw', 'e', 'fedkiw@cs.stanford.edu'),

('hanrahan', 'e', 'hanrahan@cs.stanford.edu'),

('kosecka', 'e', 'kosecka@cs.gmu.edu'),

('kunle', 'e', 'darlene@csl.stanford.edu'),

('kunle', 'e', 'kunle@ogun.stanford.edu'),

('nass', 'e', 'nass@stanford.edu'),

('nick', 'e', 'nick.parlante@cs.stanford.edu'),

('psyoung', 'e', 'patrick.young@stanford.edu'),

('rinard', 'e', 'rinard@lcs.mit.edu'),

('shoham', 'e', 'shoham@stanford.edu'),

('thm', 'e', 'pkrokel@stanford.edu'),

('ullman', 'e', 'ullman@cs.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')}

False Positives (0):

False Negatives (93):

{('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'),

('dlwh', 'e', 'dlwh@stanford.edu'),

('engler', 'e', 'engler@stanford.edu'),

……………

Summary: tp=24, fp=0, fn=93

**Explanation:** While this only returned 1 additional True Positive, any change to the RE is that increases accuracy is worthwhile. The specific e-mail that was correctly matched was “uma@cs.stanford.EDU” from the cheriton text.

There were now 21 remaining e-mails that were still False Negatives. To determine exactly what needed to be altered in the RE to identify these, I examined the text files for each and attempted to resolve each case 1 by 1. I went through each case in alphabetical order. The following runs show the e-mails I was successfully able to identify. Those I was not able to identify will be listed at the end.

**4th Run:**

To resolve engler text for the case “engler WHERE stanford DOM edu”, the following RE pattern was run in addition to ('([A-Za-z.]+)\s\*@\s\*([A-Za-z.]+)\.\D\D'), which covered all the previous 3 runs. The summary of results is shown below as well:

('([A-Za-z.]+)\s\*WHERE\s\*([A-Za-z.]+)\s\*DOM\s\*edu')

Summary: tp=25, fp=0, fn=92

**Explanation:** The RE pattern above resulted in an additional e-mail being added to the True Positives**.**

('engler', 'e', 'engler@stanford.edu')

**5th Run:**

The latombe text contained 3 e-mails that included “<del>” before the “@” sign. These were resolved using the following RE pattern. The summary of results is shown below as well:

('([A-Za-z.]+)<del>@([A-Za-z.]+)\.edu')

Summary: tp=28, fp=0, fn=89

**Explanation:** This additional RE pattern resulted in the following e-mails being added to the True Positives**.**

('latombe', 'e', 'asandra@cs.stanford.edu')

('latombe', 'e', 'latombe@cs.stanford.edu')

('latombe', 'e', 'liliana@cs.stanford.edu')

**6th Run:**

The levoy text contained 2 e-mails that included “a&#x40” before the “@” sign. These were resolved using the following RE pattern. The summary of results is shown below as well:

('([a-z]+)&#x40;(graphics.stanford).edu')

Summary: tp=30, fp=0, fn=87

**Explanation:** This additional RE pattern resulted in the following e-mails being added to the True Positives**.**

('levoy', 'e', 'ada@graphics.stanford.edu')

('levoy', 'e', 'melissa@graphics.stanford.edu')

**7th Run:**

The manning text contained 2 e-mails that included “<at symbol>” instead of a “@” sign. These were resolved using the following RE pattern. The summary of results is shown below as well:

('([A-Za-z.]+)\s<at symbol>\s([A-Za-z.]+)\.edu')

Summary: tp=32, fp=0, fn=85

**Explanation:** This additional RE pattern resulted in the following e-mails being added to the True Positives**.**

('manning', 'e', 'dbarros@cs.stanford.edu')

('manning', 'e', ' manning@cs.stanford.edu ')

**8th Run:**

The ouster text contained 2 e-mails that included text between the “someone” and “@” sign. These were resolved using the following RE pattern. The summary of results is shown below as well:

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

Summary: tp=34, fp=0, fn=83

**Explanation:** This additional RE pattern resulted in the following e-mails being added to the True Positives**.**

('ouster', 'e', 'ouster@cs.stanford.edu')

('ouster', 'e', 'teresa.lynn@cs.stanford.edu')

The remaining 11 e-mail addresses were not able to be matched and are presented in part 2.a.

**Phone Numbers**

Working with the phone numbers was much easier than it was working with the e-mail addresses. The output for the runs is provided below. The results in the summary only include phone numbers. The previously matched e-mails are not included in the True Positive totals (only True Positive phone numbers are).

**Initial Run:**

('(\d{3})-(\d{3})-(\d{4})')

True Positives (19):

{('cheriton', 'p', '650-723-1131'),

('cheriton', 'p', '650-725-3726'),

('eroberts', 'p', '650-723-3642'),

('eroberts', 'p', '650-723-6092'),

('hager', 'p', '410-516-8000'),

('rajeev', 'p', '650-723-4377'),

('rajeev', 'p', '650-723-6045'),

('rajeev', 'p', '650-725-4671'),

('subh', 'p', '650-724-1915'),

('subh', 'p', '650-725-3726'),

('subh', 'p', '650-725-6949'),

('ullman', 'p', '650-494-8016'),

('ullman', 'p', '650-725-2588'),

('ullman', 'p', '650-725-4802'),

('widom', 'p', '650-723-0872'),

('widom', 'p', '650-723-7690'),

('widom', 'p', '650-725-2588'),

('zelenski', 'p', '650-723-6092'),

('zelenski', 'p', '650-725-8596')}

Summary: tp=19, fp=0, fn=98

**Explanation:** This was the initial code that was provided for the assignment in the instructions and resulted in 19 True Positives. This regular expression (RE) pattern allowed for matches on the standard form XXX-YYY-ZZZZ.

Given symbols like “-“, “( )”, etc. can be used in phone numbers, I figured that using “\D?” would be beneficial to account for these various symbols. This will allow for handling these non-digit characters. There typically could be one non-digit character to begin a phone number with either a “(“ or a “+”. To handle the instances where this might be the case I put [^0-9]\D? at the beginning of my RE pattern. This would handle any non-digit character, if there was one, at the start of a phone number. Next, I added “\D? \D?” between the two “\d{3}” ’s to account for the possibility of 2 characters, 1 character, or no characters being between the area code and the exchange portion. For example, formats like (XXX) YYY-ZZZZ, (XXX)-YYY-ZZZZ, etc. will be covered by this addition. I added another “\D?” after the second “d{3}” to account for a “-“ or a space or even none of these. Finally, after the “\d{4}” I added “D?[^0-9]” to account for a non-digit character such as a “)”. This RE pattern is shown below and the summary of results are also displayed.

**2nd Run:**

('[^0-9]\D?(\d{3})\D?\D?(\d{3})\D?(\d{4})\D?[^0-9]')

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'),

………

Summary: tp=72, fp=19, fn=45

**Explanation:** This matched all the phone numbers and resulted in 72 True Positives. Although, this RE pattern did result in 19 False Positives. All the False Negatives are e-mail address related so that value is actually zero for False Negative phone numbers.

Through trial-and-error and tweaking the RE Pattern that matched all the phone numbers, I came up with a slight change that still resulted in the same True Positives and eliminated the False Positives produced from Run 2. Below are the results for the 3rd run.

**3rd Run:**

('[^0-9]\D?([2-9]\d{2})\D?\D?(\d{3})\D?(\d{4})\D?[^0-9]')

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'),

………

Summary: tp=72, fp=0, fn=45

**Explanation:** This matched all the phone numbers again, resulting in 72 True Positives. Although, this RE pattern did not result in the 19 False Positives. Again, the 45 False Negatives are e-mail addresses.

The final result with the complete RE Patterns for the E-mail Addresses and Phone Numbers is shown below.

**Final RE Patterns:**

epatterns = []

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

epatterns.append('([A-Za-z.]+)\s\*WHERE\s\*([A-Za-z.]+)\s\*DOM\s\*edu')

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

epatterns.append('([a-z]+)&#x40;(graphics.stanford).edu')

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

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

ppatterns = []

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

ppatterns.append('[^0-9]\D?([2-9]\d{2})\D?\D?(\d{3})\D?(\d{4})\D?[^0-9]')

**Final Results:**

True Positives (106):

{('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', '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', '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 (0):

False Negatives (11):

{('dlwh', 'e', 'dlwh@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'),

('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'),

('vladlen', 'e', 'vladlen@stanford.edu')}

Summary: tp=106, fp=0, fn=11

As shown, 106 True Positives were achieved with the 11 False Negative E-mail Addresses that I was not able to match.

**2.a.**

E-mail Addresses that were not able to be matched using the current RE pattern format:

* **dlwh:** [d-l-w-h-@-s-t-a-n-f-o-r-d-.-e-d-u](mailto:d-l-w-h-@-s-t-a-n-f-o-r-d-.-e-d-u)
  + The dashes between each character made this unable to be identified using only 2 parentheses.
* **hager:** hager at cs dot jhu dot edu
  + More than 2 parentheses needed to identify.
* **jks:** jks at robotics;stanford;edu
  + The semi-colon caused this not to be identified.
* **jurafsky:** obfuscate('stanford.edu','jurafsky')
  + With the obfuscate along with parentheses and apostrophes, this was unable to be identified with only 2 parentheses.
* **lam:** lam at cs.stanford.edu
  + I think extra spaces may have contributed to this not being identified.
* **pal:** pal at cs stanford edu
  + I think extra spaces may have contributed to this not being identified.
* **serafim:** serafim at cs dot stanford dot edu
  + More than 2 parentheses needed to identify.
* **subh:** 
  + subh AT stanford DOT edu
    - More than 2 parentheses needed to identify.
  + uma at cs dot stanford dot edu
    - More than 2 parentheses needed to identify.
* **ullman:** support at gradiance dt com
  + More than 2 parentheses needed to identify.
* **vladlen**: vladlen at <!-- die!--> stanford <!-- spam pigs!--> dot <!-- die!--> edu
  + More than 2 parentheses need to deal with the complexity of this one.

**2.b.**

Through research about ways to obfuscate e-mails, I came across many individuals saying JavaScript, jQuery and CSS are viable ways to accomplish this. This seems like these methods have worked for many but some learning of how to do so is necessary.

I also came across a website that makes it remarkably simple to obfuscate an e-mail address. It does so by encoding the e-mail address using character entities, transforming the ascii e-mail address into its equivalent decimal entity.

Here is the link to the website I used to do this: [Email Address Encoder (wbwip.com)](http://www.wbwip.com/wbw/emailencoder.html).

You simply type in your e-mail address and it returns the e-mail address in decimal entity form.

This is an example of my Syracuse e-mail:

*Standard Format:* [ddoman@syr.edu](mailto:ddoman@syr.edu)

*Decimal Entity Format:* &#100;&#100;&#111;&#109;&#097;&#110;&#064;&#115;&#121;&#114;&#046;&#101;&#100;

If you were to replace, for example [ddoman@syr.edu](mailto:ddoman@syr.edu) with “&#100;&#100;&#111;&#109;&#097;&#110;&#064;&#115;&#121;&#114;&#046;&#101;&#100;&#117;” in the source code, this would be equivalent to [ddoman@syr.edu](mailto:ddoman@syr.edu) being displayed in the source code.