IST-664
HW2 - Spamlord
Pan Chen
Prof. Nancy McCracken

Part 0: Examples demonstrated in the class

To begin with, I already have applied the regular expressions patterns demonstrated in the class, which are:

Email Patterns	**What does the pattern do**	**Examples**
([A-Za-z.]+)@([A-Za-z.]+)\.edu	allow a dot before @ sign, and allow a dot after @ sign	nick.parlante@cs.stanford.edu'
([A-Za-z.]+)\s@\s([A-Za-z.]+)\.edu	match the pattern where there is a space before and after @ sign	ashishg @ stanford.edu
Phone Patterns		
(\d{3})-(\d{3})-(\d{4})	Match phone number in the format of XXX-XXX-XXXX	650-723-1131

So far, these regular expressions produces: tp=23, fp=0, fn=94.

Part 1: Write more Regular Expressions

Here let's start write more regular expressions to be a real spamlord.

Phone Numbers

Let's deal with the phone numbers first…because I think these would be easier than the email addresses

1. ([2-9]\d{2})\D?\D?(\d{3})\D?(\d{4})

   a. Since I have never heard of an area code starting with number 1, I think I can exclude 1 from the first digit of my regular expression pattern.
   b. Also, some people might not use two "-" in their phone number format: they might use one "-" like XXX-XXXXXXX, or not use "-" at all like: XXXXXXXXXX, or they would use different symbols like parentheses: (XXX)XXX-XXXX, or they might use two symbols between two groups of number like (XXX)-XXX-XXXX To deal with these, I put '\D?\D?' in my regular expression, to match any 1 or 2 or none non-digit symbols between two groups of number. Therefore, cases like 2344444444, (234)444-4444, (234)-444-4444, 234-444(4444) would be matched.

c. To avoid an overkill, I put only one ‘/D?’ to separate the second and the third paren group of digits, so numbers like 234-444-(4444) will not be matched.

After I added this pattern to my ppatterns list, tp=76, fp=10, fn=41, and none of the phone number is in fn category which is a great relief.

Now, let's deal with the false positives.

2. Changed the previous pattern to ([2-9]\d{2})\D?\D?(\d{3})\D?(\d{4})\?[^0-9*]

a. I examined a guy named "latombe", because he/she produced 6/10 false positives, and it looked like its code “mso-list-template-ids:-695830552 -190277794 67698691 67698693 676986889 67698691 67698693 676986889 67698691 67698693;}” caused a lot of troubles...I highlighted the parts that were matched as false positives.

b. To eliminate these trouble makers, I added [^0-9*] at the the of the previous pattern, which means that if the end of the previous pattern is followed immediately with a number or is followed by a non-numerical character and a number, it would not be matched

After I made the change the previous pattern, tp=76, fp=2, fn=41

3. There are still two false positives, and they were both from this "nass" guy.It turned out, this guy’s code has a “href="http://www.amazon.com/exec/obidos/ASIN/1575860538/qid=1110995710/sr=2-1/ref=pd_bbs_b_2_1/104-3375742-1353522">” Again I highlighted the troublesome portion that produced false positive.

To deal with this, I updated the previous matching pattern to [^2-9]\D?([2-9]\d{2})\D?\D?(\d{3})\D?(\d{4})\D?[^0-9], that is, the pattern will not match the number string following a number other than 0 or 1 and a possible non-numeric character. so 75742-1353522 would not be matched as a phone number because 742-1353522 is following a 5.

Now all of the phone numbers are matched without any false positives! With two regular expressions: (\d{3})-(\d{3})-(\d{4}) and [^2-9]\D?([2-9]\d{2})\D?\D?(\d{3})\D?(\d{4})\D?[^0-9].

Email addresses

Upon scanning the false negative list, I decided to change “([A-Za-z.]+)\s@\s([A-Za-z.]+)\.edu” pattern in the example to “([A-Za-z.]+)\s*@\s*([A-Za-z.]+)\.\D\D\D”. It made the space before and after @ sign optional and it would match the patterns with more than one space on the two sides of the @ sign. I also changed ".edu" to "\D\D\D", so even if the domain was written like ".EDU", it would would be matched. As the result, emails like “support @ gradiance.edu” and “support@gradiance.EDU” should be matched with this updated pattern.

Manually adding patterns:

1. engler: engler WHERE stanford DOM edu; I wrote ([A-Za-z.]+)\sWHERE\s([A-Za-z.]+)\sDOM\sedu and it matched
2. lam: lam at cs.stanford.edu, this could be matched by using two parentheses, so I added the pattern ([A-Za-z.]+)\s*at\s*([A-Za-z.]+)\.edu and it matched. However it also produced two false positives.
3. latombe: asandra<del>@cs.stanford.edu; I then wrote "([A-Za-z.]+)<del>@([A-Za-z.]+)\.edu", and it matched.
4. levoy: ada&#x40;graphics.stanford.edu; I wrote ([A-Za-z.]+)\s*&#x40;\s*([A-Za-z.]+)\.edu to match patterns like that, and it matched.
5. manning: manning <at symbol> cs.stanford.edu; Similar to levoy, I tried ([A-Za-z.]+)\s<at symbol>\s([A-Za-z.]+)\.edu, to match it, and it matched.
6. ouster: ouster (followed by &ldquo;@cs.stanford.edu&rdquo;); teresa.lynn (followed by "@stanford.edu"); I tried ([A-Za-z.]+)\s*\(\D*@([A-Za-z.]+)\.edu and it matched both.
7. subh: subh AT stanford DOT edu; I wrote ([A-Za-z.]+) AT ([A-Za-z.]+)\s*DOT\s*edu to match that and it matched!

As the result, here are all the the patterns I wrote:

```
epatterns.append(([A-Za-z.]+)@([A-Za-z.]+)\.edu) #allow a dot before @ sign, and allow a dot after @ sign
epatterns.append('([A-Za-z.]+)\s*@\s*([A-Za-z.]+)\.\D\D\D') #match the pattern where there is a space befor and after @ sign
epatterns.append('([A-Za-z.]+)\sat\s([A-Za-z.]+)\.edu')
#lam at cs.stanford.edu
epatterns.append('([A-Za-z.]+)<del>@([A-Za-z.]+)\.edu')
#asandra<del>@cs.stanford.edu
epatterns.append('([A-Za-z.]+)\s*&#x40;\s*([A-Za-z.]+)\.edu') #ada&#x40;graphics.stanford.edu
epatterns.append('([A-Za-z.]+) AT ([A-Za-z.]+)\s*DOT\s*edu')
#subh AT stanford DOT edu
epatterns.append('([A-Za-z.]+)\s<at symbol>\s([A-Za-z.]+)\.edu')
#mailto:manning <at symbol> cs.stanford.edu
epatterns.append('([A-Za-z.]+)\s*\(\D*@([A-Za-z.]+)\.edu')
#ouster (followed by &ldquo;@cs.stanford.edu&rdquo;);
```

And Summary: tp=108, fp=2, fn=9...not too bad I guess?

Part 2 Option 2

A. List the examples that you found you could not match with the current regular expressions with two extracted parts, ending in .edu.

Those I could not match:

False Negatives:

1. dlwh: d-l-w-h-@-s-t-a-n-f-o-r-d-.-e-d-u"...I really don't think this could be matched by two sets of parentheses, unless there is a way to filter out the "-" in the strings I matched. (re.sub() maybe?)
2. hager: hager at cs dot jhu dot edu" I think this requires at least three parentheses to match: (hager),(cs),(jhu)
3. jks: jks at robotics;stanford;edu; Like hager I think this requires at least three parentheses to match.
4. jurafsky:<script type="text/javascript"><!-- function obfuscate(domain, name) { document.write('<a href="mai' + 'lto:' + name + '@' + domain + '">' + name + '@' + domain + '</' + 'a>'); }// --></script>"; I have no idea what this is but it looks like some javascript code to cover up his real email address, and I really don't know how to decipher this.
5. pal: pal at cs stanford edu, again this couldn't be matched by using two parentheses...the best the two parentheses pattern can do is to get "pal@cs stanford.edu".
6. serafim: serafim at cs dot stanford dot edu; again this couldn't be matched by using two parentheses.
7. subh: uma at cs dot stanford dot edu; again this couldn't be matched by using two parentheses in regex pattern
8. ullman: support at gradiance dt com; This one used "com" instead of "edu" so it didn't work with our current code.
9. vladlen: vladlen at <!-- die!--> stanford <!-- spam pigs!--> dot <!-- die!--> edu; this one is a tad aggressive and I'm afraid I can't match it with two parentheses. The best I can get with this one is vladlen@<!-- die!--> stanford <!-- spam pigs!-->.edu.

False Positives:

1. jure: Server at cs.stanford.edu was matched by ([A-Za-z.]+)\sat\s([A-Za-z.]+)\.edu as server@cs.stanford.edu...where "at" in the string literally means the word at, and Server at cs.stanford.edu is not an email address.
2. plotkin: Server at infolab.stanford.edu was matched by ([A-Za-z.]+)\sat\s([A-Za-z.]+)\.edu as server@infold.stanford.edu... where "at" in the string literally means at, and it's not an email address.

Try to design a way to obscure an email address that would be extremely difficult for spamlord to match with a regular expression.

In my opinion, it would be hard to developed a way to scramble my email address to a pattern that's both human readable and undetected by regular expression pattern match, because if the spamlord spent much efforts developing as many patterns as he could. Rather, I would like to develop a pattern that would trick the spamlord to match an address that is different from my actual email address, yet my scrambled address is still being readable to human.

For example, a simple “Please emailltopachen@syr.edu” would be sufficient for this purpose, because human reader would naturally see “emaill” as a typo and filter out "to", and send email to my correct email address pachen@syr.edu with no problem, while the spamlord might send the email to emailltopachen@syr.edu because that’s the pattern it would fetch. Even if it would eliminate the part of “email to” before my scrambled email address since it is a common expression, it would not be able to detect “emaillto” hopefully.