CIS 668 Assignment 2 Report

Lichen Liang

Part 1:

Initially, the given code produced 41 true positives, 0 false positives, and 76 false negatives. I have separated the task into two parts: emails and phone numbers. That is, focusing on one type and commenting out the other.

Process as follows:

1. Check for a false negative in the console output
2. Go to that file and find the original text
3. Add/modify the regex and add to the pattern
4. Run the code and record how many false negatives are changed to true positives
5. Keep all process in an excel file

Table 1. Emails

	A	B	C	D	E
1	**File Name**	**Original Text**	**Epattern Added/Modified**	**Output**	**# of True Positives**
2	cheriton	uma@cs.stanford.EDU	'([A-Za-z.]+)@([A-Za-z.]+)\.EDU'	uma@cs.stanford.edu	22->23
3	dabo	dabo @ cs.stanford.edu	'([A-Za-z.]+)\s*@\s*([A-Za-z.]+)\.edu'	dabo@cs.stanford.edu	23->24
4	engler	engler WHERE stanford DOM edu	'([A-Za-z.]+) WHERE ([A-Za-z.]+) DOM edu'	engler@stanford.edu	24->25
5	dlwh	d-l-w-h-@-s-t-a-n-f-o-r-d-.-e-d-u			
6	hager	hager at cs dot jhu dot edu			
7	jsk	jks at robotics;stanford;edu			
8	jurafsky	obfuscate('stanford.edu','jurafsky')			
9	lam	lam at cs.stanford.edu			
10	latombe	latombe<del>@cs.stanford.edu	'([A-Za-z.]+)<del>@([A-Za-z.]+)\.edu'	latombe@cs.stanford.edu	25->28
11		asandra<del>@cs.stanford.edu	'([A-Za-z.]+)<del>@([A-Za-z.]+)\.edu'	asandra@cs.stanford.edu	
12		liliana<del>@cs.stanford.edu	'([A-Za-z.]+)<del>@([A-Za-z.]+)\.edu'	liliana@cs.stanford.edu	
13	levoy	ada@graphics.stanford.edu	'([A-Za-z.]+)@([A-Za-z.]+)\.edu'	ada@graphics.stanford.edu	28->30
14		melissa@graphics.stanford.edu	'([A-Za-z.]+)@([A-Za-z.]+)\.edu'	melissa@graphics.stanford.edu	
15	manning	manning <at symbol> cs.stanford.edu	'([A-Za-z.]+)\s<at symbol>\s([A-Za-z.]+)\.edu'	manning@cs.stanford.edu	30->32
16		dbarros <at symbol> cs.stanford.edu	'([A-Za-z.]+)\s<at symbol>\s([A-Za-z.]+)\.edu'	dbarros@cs.stanford.edu	
17	ouster	ouster (followed by “@cs.stanford.edu”)			
18		teresa.lynn (followed by "@stanford.edu")			
19	pal	pal at cs stanford edu			
20	serafim	serafim at cs dot stanford dot edu			
21	subh	subh AT stanford DOT edu	'([A-Za-z.]+) AT ([A-Za-z.]+) DOT edu'	subh@stanford.edu	32->33
22		uma at cs dot stanford dot edu			
23	ullman	support at gradiance dt com			
24	vladlen	vladlen at <!-- die!--> stanford <!-- spam pigs!--> dot <!-- die!--> edu			

Table 2. Phone Numbers

	A	B	C	D	E	F
1	File Name	Original Text	Ppattern Added/Modified	Output	# of True Positives	
2	aishig	(650)814-1478	'\((\d{3})\)(\d{3})\-(\d{4})'	650-814-1478	19->27	
3		(650)723-1614	'\((\d{3})\)(\d{3})\-(\d{4})'	650-723-1614		
4		(650)723-4173	'\((\d{3})\)(\d{3})\-(\d{4})'	650-723-4173		
5	horowitz	(650)725-3707	'\((\d{3})\)(\d{3})\-(\d{4})'	650-725-3707		
6		(650)725-6949	'\((\d{3})\)(\d{3})\-(\d{4})'	650-725-6949		
7	tim	(650)724-9147	'\((\d{3})\)(\d{3})\-(\d{4})'	650-724-9147		
8		(650)725-4671	'\((\d{3})\)(\d{3})\-(\d{4})'	650-725-4671		
9		(650)725-2340	'\((\d{3})\)(\d{3})\-(\d{4})'	650-725-2340		
10	bgirod	(650) 724-6354	'\((\d{3})\)\s*(\d{3})\-(\d{4})'	650-724-6354	27->66	
11		(650) 723-4539	'\((\d{3})\)\s*(\d{3})\-(\d{4})'	650-723-4539		
12		(650) 724-3648	'\((\d{3})\)\s*(\d{3})\-(\d{4})'	650-724-3648		
13	dabo					
14	hager					
15	hanrahan					
16	kosecka					
17	kunle					
18	lam					
19	latombe					
20	levoy					
21	manning					
22	nick					
23	ok					
24	rinard					
25	serafim					
26	thm					
27	zm					
28	jurafsky	+1 650 723 5666	'(\d{3})\s(\d{3})\s(\d{4})'	650-723-5666	66->68	
29	pal	+1 650 725 9046	'(\d{3})\s(\d{3})\s(\d{4})'	650-725-9046		
30	nass	[650] 723-5499	'\[(\d{3})\]\s*(\d{3})\-(\d{4})'	650-723-5499	68->70	
31		[650] 725-2472	'\[(\d{3})\]\s*(\d{3})\-(\d{4})'	650-725-2472		
32	shoham	650 723-3432	'(\d{3})\s(\d{3})\-(\d{4})'	650-723-3432	70->72	
33		650 725-1449	'(\d{3})\s(\d{3})\-(\d{4})'	650-725-1449		

In table 1 and 2, for each file, the original text, pattern, and output are displayed. The last column keeps track of how true positives are changed. Each color represents a type that can be matched by a pattern. For example, in table 2, file names aishig, horowitz, and tim, they are similar in the original input and therefore they can be matched using the same pattern. An exception is in table 2 yellow color. There are too many data matched at the same time (number of true positives increased from 27 to 66), so I only listed the file names. In table 1, there are 12 emails that we did not match because they are obscured emails, so they do not have a color.

Breakdown of Table 1:

- Cheriton: the ‘edu’ is in upper case. I added a new regex same as the default one but with ‘edu’ in upper case
- Dabo: there are two spaces before and after the @, so I modified a given regex by using \s* meaning 0 or more space.
- Engler: the text contains ‘WHERE’ and ‘DOM’. I added a new regex replacing @ and \. with ‘WHERE’ and ‘DOM’
- Latombe: the text has ‘<del>’ before the @ symbol. I added a new regex with ‘<del>’ before @.
- Levoy: the text replaced the @ with ‘a@’. I added a new regex with ‘a@’ in place of the @
- Manning: the text replaced @ with ‘ <at symbol> ’ I added a new regex with ‘\s<at symbol>\s’ in place of the @. Note there are two spaces.
- Subh: the text contains ‘AT’ and ‘DOT’. I added a new regex replacing @ and \. with ‘AT’ and ‘DOT’

Initially there are 22 true positives and now there are 33. Leaving 12 false negatives will be discussed in part 2.

Breakdown of Table 2:

- In files aishig, horowitz, and tim, the first three number are in parenthesis, followed by the fourth number. I added a new regex using \(and \) at the positions for the first three numbers.
- In files bgirod, dabo,…zm, the only difference is that there is a space after the closing parenthesis. I modified the previous regex by adding \s* after the \)
- In files jurafsky and pal, the numbers are separated by a single space, so I added a regex and used \s between the digits sets.
- In file shoham, different from the file jurafsky, the second space is replaced with a ‘-’, so I added a regex with \- before the last four digits.
- In file nass, the first three numbers are in square brackets and the others like file shoham, so I added new regex using \[and \] before and after the three digits.

Initially there are 19 true positives and now there are 72. All phone numbers have been matched.

Overall, there are 105 true positives matched.

Part 2:

a) Reasons why some are not matched

d-l-w-h-@-s-t-a-n-f-o-r-d-.-e-d-u	Each character is separated by a '-'
hager at cs dot jhu dot edu	Need more than two parentheses
jks at robotics;stanford;edu	Need more than two parentheses
obfuscate('stanford.edu','jurafsky')	Order of name and domain is in reverse order
lam at cs.stanford.edu	Need more than two parentheses
ouster (followed by “@cs.stanford.edu”)	Too many symbols and need more than two parentheses
teresa.lynn (followed by "@stanford.edu")	Unnecessary words and symbols
pal at cs stanford edu	Need more than two parentheses
serafim at cs dot stanford dot edu	Need more than two parentheses
support at gradiance dt com	Not ending in edu
vladlen at <!-- die!--> stanford <!-- spam pigs!--> dot <!-- die!--> edu	Too many symbols and words in between

b) Some more obscured email address I found interesting:

- An email address built by JavaScript:

```
<script language="JavaScript" type="text/javascript">
var part1 = "example11";
var part2 = Math.pow(2,6);
var part3 = String.fromCharCode(part2);
var part4 = "heartinternet.uk";
var part5 = part1 + String.fromCharCode(part2) + part4;
document.write("Please email us at <a href=" + "mai" + "l
to" + ":"
+ part5 + ">" + part1 + part3 + part4 + "</a>.");
</script>
```

The email address is separated into multiple parts and combined to form the complete email address. However, this only works when JavaScript is enabled. If not, the email address would not appear.

- Use CSS for part of an email address

```
<style type="text/css">
p#emaildomain::after {
content: "13\40heartinternet.uk";
}
span#emailname::before {
content: "example";
}
span#botblocker {
display: none;
}
</style>
<p id="emaildomain">Please email us at
<span id="emailname">
<span id="botblocker">botblocker@heartinternet.uk</span>
</span>.</p>
```

Using ::before, :: after, and display::none, the email address will appear just right to the user. The spam bots will pickup a wrong email address instead.