Assignment 2

Dixitha Kasturi
dkasturi@syr.edu

Topic : Regular expression matching

1.Write more regular expressions that correct false positives or fit false negatives. Do as best as you can using the epatterns and ppatterns lists in the program. For each regular expression that you write or extend, list

- example(s) of email or phone numbers that match that pattern,
- including example(s) of the obfuscated text that was matched from the file,
- a short English description of the expressions the pattern matches, demonstrating your understanding of regular expressions.
- and the results (TP, FP and FN) before and after you added the expression.

When you are done with as many as you can do, give the output of the program.

For this part I used 14 different regular expressions to parse the obfuscated email IDs and phone numbers. 10 for emails, 4 for numbers

Emails :

1. **'([A-Za-z0-9.]+)\s*&#[A-Za-z0-9.]+;\s*([A-Za-z0-9._]+)\.edu' :** This pattern finds emails which have IID @ something.domain.edu) eg: ada@graphics.stanford.edu.
2. **'([A-Za-z.]+)\s+@\s+([A-Za-z.]+)\.[A-Za-z]+' :** This pattern finds one or more uppercase ,lowercase letters and special characters. The expression then indicates that the at sign should be followed by one or more special characters, in case there are whitespaces eg: ullman @ cs.stanford.edu
3. **'([A-Za-z.]+)@([A-Za-z.]+)\.[A-Za-z]+' :** This pattern is similar to the last one minus the presence of special charcaters. eg: balaji@stanford.edu.edu
4. **'([a-z.]+)\b[<][a-zA-Z&;]+[>].?@([a-z.]+).edu' :** This pattern finds emails with lowercase words followed by open anchor bracket, someword, close anchor bracket and @ followed by the domain.edu eh: asandra<del>@cs.stanford.edu
5. **'([A-Za-z0-9.]+)\s*\([\s*A-Za-z0-9.&;#]*["|;}]@([A-Za-z0-9._]+)\.edu' :** This pattern find emails in which the first part has lowercase letters and special characters. then any other pattern or words with the specified special characters followed by quotes pipe or semi colon with domain.edu at the end. Eg:
6. **'^([a-z]+).?\bat\b\s(\W.+).edu+' :** This pattern finds commencing at the beginning of the phrase, any lowercase character The center section could then contain either a special character, such as the word 'at,' or a special character. eg: : vladlen at <!--die!--> stanford
7. **'(\w+)\b.[A-Z].*\b(stanford).[A-Za-z]+.edu' :** This pattern finds emails, with a word, followed by special characters which are in letters for instance ID WHERE source DOM edu Dom is dot and where is at. Eg: engler WHERE stanford DOM edu

8. **'([a-z]+).at <!--.+>.(stanford).+edu' :** This pattern finds email that begins with lowercase letters and ends with !-> and any other matching special characters be returned.
9. **'([A-Za-z0-9.]+)\s*at\s*([A-Za-z0-9.]+)\.EDU' :** This finds emails which have "at" in the middle with smaller or upper case letters in the beginning and following the at and capital EDU.
10. **'([a-zA-Z0-9]+)\s*[<][a-zA-Z0-9 .]+[>]\s*([a-zA-Z0-9.]+)([a-zA-Z0-9.]+)edu' :** This is similar to the 4th pattern but identifies <at symbol> instead.

Phone numbers:

1. '**.+(\d{3}).[^0-9](\d{3})[^0-9](\d{4})'** : This pattern finds any character that isn't a line terminator is matched by the expression. The first group then matches any three consecutive digits. Before repeating the first group, it asks you to match a single non-digit character. It then asks for a single non-numeric character to be matched once more before asking for four consecutive digits.
2. **'.?(\d{3})[^0-9](\d{3})[^0-9](\d{4})**' : Similar to the previous pattern, this has optional special character matching
3. **'(\d{3})-(\d{3})-(\d{4})'** : This pattern finds all number that are separated by a – in 3-3-4 count format
4. **'(?:[(])([0-9]{3})(?:[)])[ ]*([0-9]{3})-([0-9]{4})'** : This pattern finds numbers which have state code in parenthesis

2. (Option) Do both parts a and b in this option.
a. List the examples that you found you could not match with the current regular expressions with two extracted parts, ending in .edu. For each example, or set of examples that fit the same pattern, explain briefly why it won't work. If you can make expressions in regexpal that match, but don't work in the program to extract the email or phone numbers, list those here. If you had any false positives in Part 1, include a discussion here of why that rule generated them.

Out of all the cases, there were 108 true positives, 0 false positives and 9 false negatives:
True Positives: (108)
{('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', 'e', 'subh@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'),
('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 negatives: (9)
While most of the patterns gpt parsed, there were a few that need additional processing in strings to get them into the correct format. The following 9 were the ones that I couldnot process.

{('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', 'uma@cs.stanford.edu'),
('ullman', 'e', 'support@gradiance.com')}

a) 'd-l-w-h@s-t-a-n-f-o-r-d-.edu' - Additional – in between has to be parsed separately
b) hager at cs dot jhu dot edu – string manipulations are needed
c) jks at robotics;stanford;edu – string manipulations are needed
d) obfuscate('stanford.edu', 'jurafsky') - the email ID is obfuscated by the creator, it needs a lot of manipulatons
e) lam at cs.stanford.edu – string manipulations are needed
f) pal at cs stanford edu – string manipulations are needed
g) serafim@cs.edu - The semicolons that obscured the regular expression were the reason this expression couldn't be matched.
h) uma at cs dot stanford dot edu – string manipulationsare needed
i) support at gradiance dt com - This false negative is caused by a string with the pattern.com, and we have a constant with.edu on the format we append to our result list (email = '@.edu'.format(m[0],m[1]) in our code (email = '@.edu'.format(m[0],m[1]). As a result, we obtain a false positive and a negative.

To process all of these, a separate chunk of code had to be written which would divide the email into parts and each part had to be processed separately. It would require a lot of effort.

b. Then search the web and find a couple of additional examples of obscured email addresses or phone numbers and report on them, or try to design a way to obscure an email address that would be extremely difficult for ContactFinder to match with a regular expression. For the latter, try to have something more specific than things like "To send me email, try the simplest address that makes sense."

ContactFinder wouldn't be able to parse obfuscated emails that have html tags around the mail address in script

```
<SCRIPT LANGUAGE="JavaScript">
user = 'name';
site = 'domain.com';
document.write('<a href=\"mailto:' + user + '@' + site + '\">');
document.write(user + '@' + site + '</a>');
</SCRIPT>
```

In the document write part, we can use any type of special characters. Using this script makes it hard for web scrappers and crawlers to separate the code from the email.

The other method is that we can utilize a captcha approach, which is generally secure but requires the client to fill out a captcha each time an email needs to be linked. This is the safest way, and bots have a hard time cracking it. Accessibility, on the other hand, comes at a price.