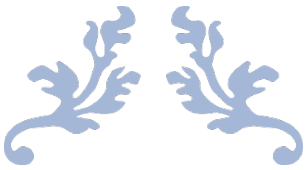

# CIS 668 – NATURAL LANGUAGE PROCESSING

Assignment 2

OCTOBER 12, 2017

ABHIRAM SRINIVASAN
SUID - 407650467

1.

First when we run the python without any regular expression patterns we observe that there are
Summary: tp=0, fp=0, fn=117

Now we will satisfy some email patterns

**epatterns.append('([A-Za-z.]+)\s*@\s*([A-Za-z.]+)\.[Ee][Dd][Uu]')**
This regular expression satisfies many combinations of email addresses.
In this regular expression, the first parentheses accepts any alphabets which are capital or small from the range A to Z followed by a period, we have added a plus to match one or more of the preceding tokens. Following the ‘)’ we have a pattern which accepts 0 or more spaces which is implied by \s*. After the @ symbol we have a pattern which accepts 0 or more spaces. The second parentheses accepts any alphabets which are capital or small from the range A to Z followed by a period, we have added a ‘+’ to match one or more of the preceding tokens. The ‘\.’ matches a period after the second parentheses (domain name). And finally, we have a pattern to match edu or EDU.

Some examples of what the above regex matches are as bellow
ashishg @ stanford.edu
balaji@stanford.edu
uma@cs.stanford.EDU
dabo @ cs.stanford.edu
eroberts@cs.stanford.edu
patrick.young@stanford.edu

After performing this regex we get the following summary
Summary: tp=24, fp=0, fn=93

**epatterns.append('([A-Za-z]+)\sWHERE\s([A-Za-z]+)\sDOM\sedu')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case, the ‘+’ indicates that we can match one or more of the preceding tokens. The ‘\s’ indicates a single space. The WHERE identifies the exact word in the email. The second parentheses can accept alphabets which can be capital or lower case, the ‘+’ indicates that we can match one or more of the preceding tokens. This is followed by a space, DOM and edu.

Example of the above regex matches are
engler WHERE stanford DOM edu

After performing this regex we get the following summary
Summary: tp=25, fp=0, fn=92

**epatterns.append('([A-Za-z]+)\s<at symbol>\s([A-Za-z.]+)\.edu')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case, the '+' indicates that we can match one or more of the preceding tokens. The '\s' indicates a single space. The <at symbol> matches the exact word in the email. The second parentheses can accept any alphabets which can be capital or lower case followed by a period. This is followed by a period edu.

Example of the above regex matches are
manning <at symbol> cs.stanford.edu
dbarros <at symbol> cs.stanford.edu

After performing this regex we get the following summary
Summary: tp=27, fp=0, fn=90

**epatterns.append('([A-Za-z]+)\sat\s<!-- die!-->\s([A-Za-z]+)\s<!-- spam pigs!-->\sdot\s<!-- die!-->\s[Ee][Dd][Uu]')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case, the '+' indicates that we can match one or more of the preceding tokens. The '\s' indicates a single space. The HTML tags/comments are written as it is to match the given pattern. The second parentheses we can accept alphabets which can be capital or lower case, the '+' indicates that we can match one or more of the preceding tokens. We match 'dot' and 'at' as it is. This is followed by edu.

Example of the above regex matches are
vladlen at <!-- die!--> stanford <!-- spam pigs!--> dot <!-- die!--> edu

After performing this regex we get the following summary
Summary: tp=28, fp=0, fn=89

**epatterns.append('([A-Za-z]+)<del>@([A-Za-z.]+)\.edu')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case, the '+' indicates that we can match one or more of the preceding tokens. It is followed by <del> to match the pattern and the second parentheses can accept alphabets which can be capital or lower case followed by a period, the '+' indicates that we can match one or more of the preceding tokens. This is followed by a period edu.

Example of the above regex matches are
asandra<del>@cs.stanford.edu
latombe<del>@cs.stanford.edu
liliana<del>@cs.stanford.edu

After performing this regex we get the following summary
Summary: tp=31, fp=0, fn=86

**epatterns.append('([A-Za-z]+)@([A-Za-z.]+)\.edu')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case, the '+' indicates that we can match one or more of the preceding tokens. This is followed by '@' to match the pattern in the email. The second parentheses can accept alphabets which can be capital or lower case followed by a period, the '+' indicates that we can match one or more of the preceding tokens. This is followed by a period edu.

Example of the above regex matches are
ada@graphics.stanford.edu
melissa@graphics.stanford.edu

After performing this regex we get the following summary
Summary: tp=33, fp=0, fn=84

**epatterns.append('([A-Za-z]+)\s[Aa][Tt]\s([A-Za-z.]+)\.edu')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case, the '+' indicates that we can match one or more of the preceding tokens. The '\s' indicates a single space. Then we have a pattern which matches 'at' which can be either capital or lowercase. The second parentheses can accept alphabets which can be capital or lower case followed by a period, the '+' indicates that we can match one or more of the preceding tokens. This is followed by a period edu.

Example of the above regex matches are
lam at cs.stanford.edu

After performing this regex we get the following summary
Summary: tp=34, fp=2, fn=83

**epatterns.append('([A-Za-z]+)\s[Aa][Tt]\s([A-Za-z]+)\sDOT\sedu')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case, the '+' indicates that we can match one or more of the preceding tokens. The '\s' indicates a single space. Then we have a pattern which matches 'at' which can be either capital or lowercase. The second parentheses we can accept alphabets which can be capital or lower case, the '+' indicates that we can match one or more of the preceding tokens. It is then followed by DOT to match the pattern and then followed by edu.

Example of the above regex matches are
subh AT stanford DOT edu

After performing this regex we get the following summary
Summary: tp=35, fp=2, fn=82

**epatterns.append('([A-Za-z]+)\s+\(followed by\s?(?:“)?@([A-Za-z.-]+)+\.edu')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case, the '+' indicates that we can match one or more of the preceding tokens. The '\s' indicates a single space. In the second parentheses we have the word followed by and the ?:&ldquo which is written the same way to match the pattern. We then have a pattern to accept alphabets which can be capital or lower case, followed by a period and hyphen, the '+' indicates that we can match one or more of the preceding tokens. This is followed by a period edu.

Example of the above regex matches are
ouster (followed by “@cs.stanford.edu”)

After performing this regex we get the following summary
Summary: tp=36, fp=2, fn=81

**epatterns.append('([A-Za-z.]+)\s+\(followed by\s?(?:“)?"?@([A-Za-z.-]+)+\.edu')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case which is followed by a period, the '+' indicates that we can match one or more of the preceding tokens. The '\s' indicates a single space. We then have 'followed by (:&ldquo). This is written exactly to match the words in the pattern. We have '?' which can optionally match the preceding pattern given. After the @ symbol we can accept alphabets which can be capital or lower case which is followed by a period and a hyphen. This is then followed by a period and edu.

Example of the above regex matches are
teresa.lynn (followed by "@stanford.edu")

After performing this regex we get the following summary
Summary: tp=37, fp=2, fn=80

```
#Regular expression which matches email ids like
#ashishg @ stanford.edu
#balaji@stanford.edu
#uma@cs.stanford.EDU
#dabo  @  cs.stanford.edu
#eroberts@cs.stanford.edu
#patrick.young@stanford.edu
#After performing this we get
#Summary: tp=24, fp=0, fn=93
epatterns.append('([A-Za-z.]+)\s*@\s*([A-Za-z.]+)\.[Ee][Dd][Uu]')

#Regular expression which satisfies email in Engler doc
#This satisfies an email like
#engler WHERE stanford DOM edu
epatterns.append('([A-Za-z]+)\sWHERE\s([A-Za-z]+)\sDOM\sedu')

#Regular expression which satisfies email in Manning doc
# this satisfies email like
#manning <at symbol> cs.stanford.edu
#dbarros <at symbol> cs.stanford.edu
epatterns.append('([A-Za-z]+)\s<at symbol>\s([A-Za-z.]+)\.edu')

#Regular expression which satisfies email in Vladlen
#This satisfies an email like
#vladlen at <!-- die!--> stanford <!-- spam pigs!--> dot <!-- die!--> edu
epatterns.append('([A-Za-z]+)\sat\s<!-- die!-->\s([A-Za-z]+)\s<!-- spam pigs!-->\sdot\s<!-- die!-->\s[Ee][Dd][Uu]')

#Regular expression which satisfies email in lantombe doc
#email satisfies
#asandra<del>@cs.stanford.edu
#latombe<del>@cs.stanford.edu
#liliana<del>@cs.stanford.edu
epatterns.append('([A-Za-z]+)<del>@([A-Za-z.]+)\.edu')

#Regular expression which satisfies email in levoy doc
#email satisfies
#ada&#x40;graphics.stanford.edu
#melissa&#x40;graphics.stanford.edu
epatterns.append('([A-Za-z]+)&#x40;([A-Za-z.]+)\.edu')

#Regular expression which satisfies email in lam doc
#email satisfies
#lam at cs.stanford.edu
epatterns.append('([A-Za-z]+)\s[Aa][Tt]\s([A-Za-z.]+)\.edu')
```

```
#Regular expression which satisfies email in subh doc
#email satisfies
#subh AT stanford DOT edu
epatterns.append('([A-Za-z]+)\s[Aa][Tt]\s([A-Za-z]+)\sDOT\sedu')

#Regular expression which satisfies email in ouster doc
#email satisfies
#ouster (followed by &ldquo;@cs.stanford.edu&rdquo;)
epatterns.append('([A-Za-z]+)\s+\(followed by\s?(?:&ldquo;)?@([A-Za-z.-]+)+\.edu')

#Regular expression which satisfies email in ouster doc
#email satisfies
#teresa.lynn (followed by "@stanford.edu")
epatterns.append('([A-Za-z.]+)\s+\(followed by\s?(?:&ldquo;)?"?@([A-Za-z.-]+)+\.edu')
```

Now we will satisfy some phone patterns

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

In this regular expression, we have three parentheses. In the first parentheses, we satisfy 3 digits then we have a hyphen. In the second parentheses we satisfy 3 digits and then we have a hyphen. In the third parentheses we satisfy 4 digits.

Example of the above regex matches are
650-725-8596
650-723-6092
650-723-7690

After performing this regex we get the following summary
Summary: tp=56, fp=2, fn=61

**ppatterns.append('[ \(](\d{3})[ \)]+\s*(\d{3})[ \-]+(\d{4})')**
In this regular expression, we have '\(' to match ( bracket, we have '\)' to match ) bracket. We have '\s' to match a single space. '+' indicates that we can match one or more of the preceding tokens. we use '*' to indicate that we can use 0 or more preceding characters.We satisfy an hyphen by using '\-'. We satisfy three digits by using d{3} and 4 digits by using d{4}.

Example of the above regex matches are
(650) 724-3648
(650)723-4173

After performing this regex we get the following summary
Summary: tp=107, fp=2, fn=10

**ppatterns.append('\[(\d{3})\]\s(\d{3})-(\d{4})')**
In this regular expression, we '\[' this to match an opening [ bracket and perform '\]' to match a closing ]. We have '\s' to match a single space. We satisfy an hyphen by using '\-'. We satisfy three digits by using d{3} and 4 digits by using d{4}.

Example of the above regex matches are
[650] 723-5499

After performing this regex we get the following summary
Summary: tp=109, fp=2, fn=8

```
ppatterns = []
#ppatterns.append('(\d{3})-(\d{3})-(\d{4})')
#Regular expression which satisfies phone numbers
#examples
#650-725-8596
#650-723-6092
#650-723-7690
ppatterns.append('(\d{3})-(\d{3})-(\d{4})')

#Regular expression which satisfies phone numbers
#examples
#(650) 724-3648
#(650)723-4173
ppatterns.append('[ \(](\d{3})[ \)]+\s*(\d{3})[ \-]+(\d{4})')

#Regular expression which satisfies phone numbers
#examples
#[650] 723-5499
ppatterns.append('\[(\d{3})\]\s(\d{3})-(\d{4})')
```

3.

I choose question 3 as I want to further reduce the number of false negatives by adding python code. I created a new parts in the process_filename which satisfies email id with .com and email id which accepts 3 parts.

I created a new list which takes in compatterns. I have written a for loop in the process_file function which has comments.

```
# we are adding this for loop
# to satisfy all the email address with .com
# hence we have %s@%s.com
# these would satisfy two parentheses which take in
# someone@somewhere.com
for cpat in compatterns:
    matches = re.findall(cpat,line)
    for m in matches:
        email = '%s@%s.com' % m
        res.append((name,'e',email))
```

**compatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+)\sdt\scom')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case. the '+' indicates that we can match one or more of the preceding tokens. '\s' matches a single space. We write at as it is. The second parentheses can

alphabets which can be capital or lower case. It is followed by a space then we write dt space and com as it is.

Example of the above regex matches are
support at gradiance dt com

After performing this regex we get the following summary
Summary: tp=110, fp=2, fn=7

```
# com patterns
# each regular expression pattern should have exactly two sets of parentheses
# the first parenthesis should be around the someone part
# the second parenthesis should be around the somewhere part
# in an email address whose standard form is someone@somewhere.edu
compatterns = []

#Regular expression which satisfies email
#Example of email id
# support at gradiance dt com
compatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+)\sdt\scom')
```

To match patterns with 3 parts I created a new list which takes in tpatterns. I have written a for loopin the process_file function which has comments

```
    # we are adding this for loop
    # to satisfy all the email address with abc@xyz.jk.edu
    # hence we have %s@%s.%s.edu
    # these would satisfy three parentheses which take in
    # someone@somewhere1.somewhere2.edu
    for tpat in tpatterns:
        matches = re.findall(tpat,line)
        for m in matches:
            email = '%s@%s.%s.edu' % m
            res.append((name,'e',email))
```

**tpatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+)\s([A-Za-z]+)\sedu')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case. the '+' indicates that we can match one or more of the preceding tokens. '\s' matches a single space. We write at as it is. Then in our second parentheses we we can accept alphabets which can be capital or lower case. This is followed by a space. In our third parentheses we can accept alphabets which can be capital or lower case. This is followed by a space and edu.

Example of the above regex matches are
pal at cs stanford edu

After performing this regex we get the following summary
Summary: tp=111, fp=2, fn=6

**tpatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+);([A-Za-z]+);edu')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case. the '+' indicates that we can match one or more of the preceding tokens. '\s' matches a single space. We write at as it is. Then in our second parentheses we we can accept alphabets which can be capital or lower case. We then have a ';' to match the pattern. In the third parentheses we can accept alphabets which can be capital or lower case. We then have a ';' followed by edu.

Example of the above regex matches are
jks at robotics;stanford;edu

After performing this regex we get the following summary
Summary: tp=112, fp=2, fn=5

**tpatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+)\sdot\s([A-Za-z]+)\sdot\sedu')**
In this regular expression, in our first parentheses we can accept alphabets which can be capital or lower case. the '+' indicates that we can match one or more of the preceding tokens. '\s' matches a single space. We write at as it is. Then in our second parentheses we we can accept alphabets which can be capital or lower case. We write 'dot' as it is to match our pattern. In the third parentheses, we can accept alphabets which can be capital or lower case. We then have a space and it is followed by dot and edu to match the pattern.

Example of the above regex matches are
hager at cs dot jhu dot edu
serafim at cs dot stanford dot edu
uma at cs dot stanford dot edu

After performing this regex we get the following summary
Summary: tp=115, fp=2, fn=2

```
# three patterns
# each regular expression pattern should have exactly three sets of parentheses
# the first parentheses should be around the someone part
# the second parentheses should be around the somewhere1 part
# the three parentheses should be around the somewhere2 part
# in an email address whose standard form is someone@somewhere1.somewhere2.edu
tpatterns = []

#Regular expression which satisfies email
#Example of email id
#pal at cs stanford edu
tpatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+)\s([A-Za-z]+)\sedu')

#Regular expression which satisfies email
#Example of email id
#jks at robotics;stanford;edu
tpatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+);([A-Za-z]+);edu')

#Regular expression which satisfies email
#Example of email id
# hager at cs dot jhu dot edu
# serafim at cs dot stanford dot edu
# uma at cs dot stanford dot edu
tpatterns.append('([A-Za-z]+)\sat\s([A-Za-z]+)\sdot\s([A-Za-z]+)\sdot\sedu')
```

Finally, we have these

```
False Positives (2):
{('jure', 'e', 'server@cs.stanford.edu'),
 ('plotkin', 'e', 'server@infolab.stanford.edu')}
False Negatives (2):
{('dlwh', 'e', 'dlwh@stanford.edu'), ('jurafsky', 'e', 'jurafsky@stanford.edu')}
Summary: tp=115, fp=2, fn=2
```

We have 2 false positives and 2 false negatives.

In jure and plotkin file the email address are as follow
**Apache/2.2.4 (Fedora) Server at cs.stanford.edu Port 80**
**Apache/2.0.54 (Fedora) Server at infolab.stanford.edu Port 80**
We cannot write a pattern matching these email address because we need more than three parentheses to match them. It has words like Fedora, Port no, Server which cannot be matched. Hence, we need to enhance the framework more to satisfy this pattern.

In dlwh file the email address are as follows
**d-l-w-h-@-s-t-a-n-f-o-r-d-.-e-d-u**
We cannot write a pattern matching this email address because we need more have hyphens in between each character. It can be solved by doing string replacement and hence we it cannot be satisfied by just writing a regular expression.

In the jurafsky file the email address are as follows
**obfuscate('stanford.edu','jurafsky');**
We cannot write a pattern matching this email address because it is a javascript call and we cannot toggle the function definition directly. When we write a regular expression for this pattern we get an output like standford@jurafsky.edu. The someone and somewhere parts are swapped and hence we cannot write a regular expression satisfying this pattern directly.

Some True Positives examples are

```
Abhirams-MacBook-Pro:SpamLord abhiram$ python SpamLord.base\ copy.py data/dev data/devGOLD
True Positives (115):
{('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', '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', 'p', '650-723-5666'),
```