CS-AD 220 – Spring 2016

Natural Language Processing

Session 3: 4-Feb-16

Prof. Nizar Habash

Some slides are adapted from Jurafsky and Martin's course slides on Speech and Language Processing

NYUAD Course CS-AD 220 – Spring 2016
Natural Language Processing

Assignment #1
Unix Tools and Regular Expressions
Assigned Feb 4, 2016

Due Feb 18, 2016 (11:59pm)

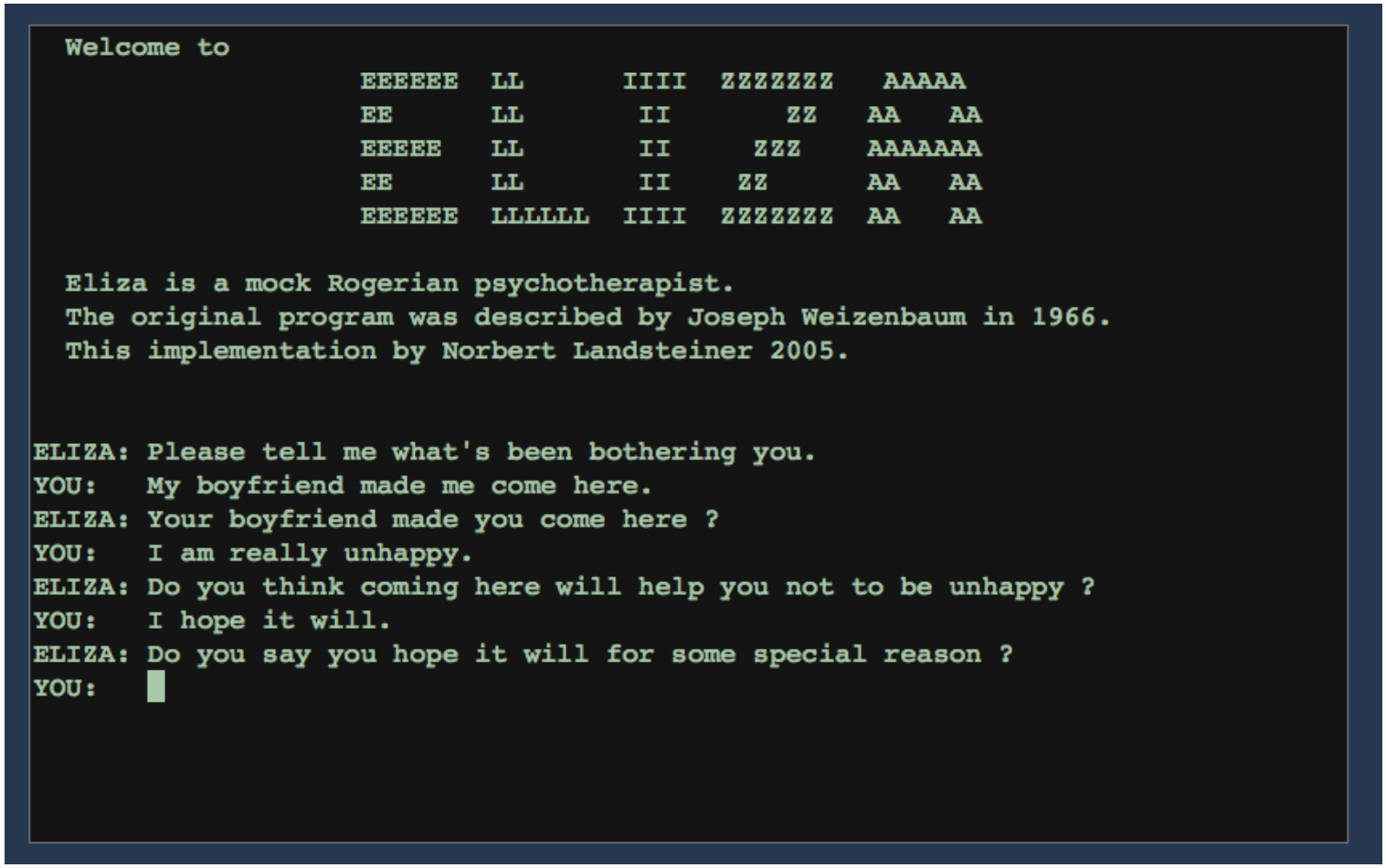
I. Grading & Submission

This assignment is about the use of regular expressions (regex) and a set of Unix tools for quick text processing. The assignment accounts for 10% of the full grade. Section III below has a set of questions. The student needs to answer them all. The specific number of points for each question is provided. The student should submit a **PDF** file containing the answers to each question and sub-question in order. The student should also include the commands and the result of applying the commands by copying and pasting from the terminal. Each student must work alone. This is not a group effort.

The assignment is due on Feb 18 before midnight (11:59pm). For late submissions, 10% will be deducted from the homework grade for any portion of each late day. The student should upload the answer to NYU Classes (Assignment #1).

Assignment #1 posted on NYU Classes

Please tell me what's been bothering you.



http://www.masswerk.at/elizabot/eliza.html

Eliza

- Eliza is a mock Rogerian psychotherapist.
  - Rogerian psychotherapy, or person-centered therapy, is a form of talk-psychotherapy developed by psychologist Carl Rogers in the 1940s & 1950s.
- The original program was described by Joseph Weizenbaum in 1966.
- Primitive natural language processing using simple scripts and **regular expressions**
- Chatterbots or chatbots
- Eliza effect
  - the tendency to unconsciously assume computer behaviors are analogous to human behaviors.
- Turing test

Mitsuku

- Let's chat with Mitsuku!
- http://www.mitsuku.com

Mitsuku

- Let's chat with Mitsuku!
- http://www.mitsuku.com
- Winner of Loebner prize in 2013
  - Modern form of the Turing test for Artificial Intelligence

Regular Expressions

- In computer science, RE (or regex, regexp) is a language used for specifying text search string.
- A *regular expression* is a *formula* in a *special language* that is used for specifying a simple class of *string*.
- Formally, a regular expression is an algebraic notation for characterizing a set of strings.
- RE search requires
  - a *pattern* that we want to search for, and
  - a *corpus* of texts to search through.

Regular Expressions

- A RE search function will search through the corpus returning all texts that contain the pattern.
  - In a Web search engine, they might be the entire documents or Web pages.
  - In a word-processor, they might be individual words, or lines of a document.
    - E.g., the UNIX `grep` command
- Substitutions allow us to manipulate text by matching on specific patterns to replace them.

Regular Expressions
Basic Regular Expression Patterns

- The use of the brackets `[]` to specify a disjunction of characters.

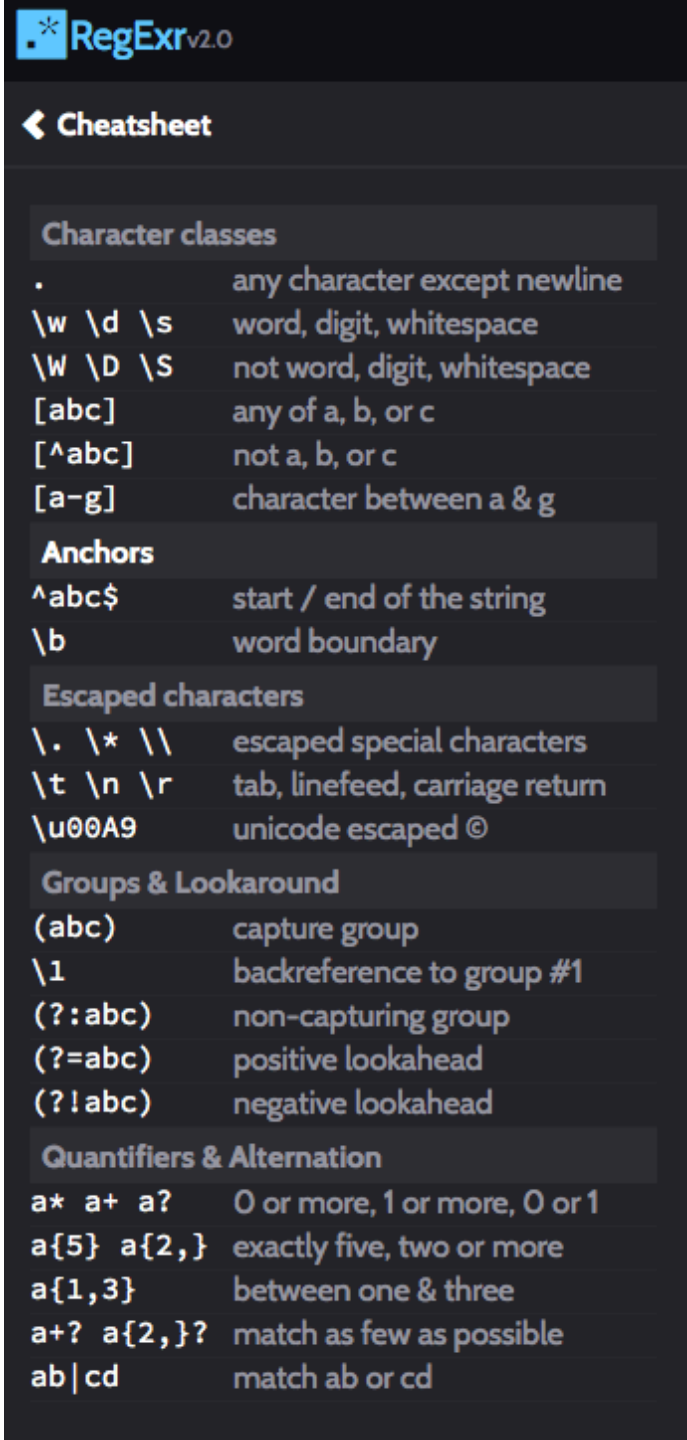
RE	Match	Example Patterns
`/[wW]oodchuck/`	Woodchuck or woodchuck	"Woodchuck"
`/[abc]/`	'a', 'b', *or* 'c'	"In uomini, in soldati"
`/[1234567890]/`	any digit	"plenty of 7 to 5"

- The use of the brackets `[]` plus the dash − to specify a range.

RE	Match	Example Patterns Matched
`/[A-Z]/`	an uppercase letter	"we should call it 'Drenched Blossoms'"
`/[a-z]/`	a lowercase letter	"my beans were impatient to be hoed!"
`/[0-9]/`	a single digit	"Chapter 1: Down the Rabbit Hole"

RegExr

- http://www.regexr.com
- Sandbox to tryout regular expressions



Regular Expressions
Basic Regular Expression Patterns

- Uses of the caret ^ for negation or just to mean ^

RE	Match (single characters)	Example Patterns Matched
[^A-Z]	not an uppercase letter	"Oyfn pripetchik"
[^Ss]	neither 'S' nor 's'	"I have no exquisite reason for't"
[^\.]	not a period	"our resident Djinn"
[e^]	either 'e' or '^'	"look up ^ now"
a^b	the pattern 'a^b'	"look up a^ b now"

- The question-mark ? marks optionality of the previous expression.

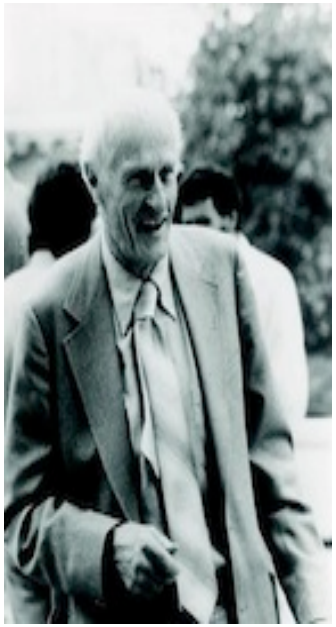
RE	Match	Example Patterns Matched
woodchucks?	woodchuck or woodchucks	"woodchuck"
colou?r	color or colour	"colour"

- The use of period . to specify any character

RE	Match	Example Patterns
/beg.n/	any character between *beg* and *n*	begin, beg'n, begun

Regular Expressions: ? * + .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n



Stephen C Kleene

Kleene *, Kleene +

Regular Expressions: Anchors ^ $

Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Za-z]	"Hello"
\.$	The end.
.$	The end?

Regular Expressions
Disjunction, Grouping, and Precedence

- Disjunction

```
/cat|dog/
```

- Precedence

```
/gupp(y|ies)/
```

- Operator precedence hierarchy

```
()
+ ? { }
the  ^my  end$
|
```

Regular Expressions
A Simple Example

- To find the English article *the*

```
/the/

/[tT]he/

/\b[tT]he\b/

/[^a-zA-Z][tT]he[^a-zA-Z]/

/(^|[^a-zA-Z])[tT]he([^a-zA-Z]|$)/
```

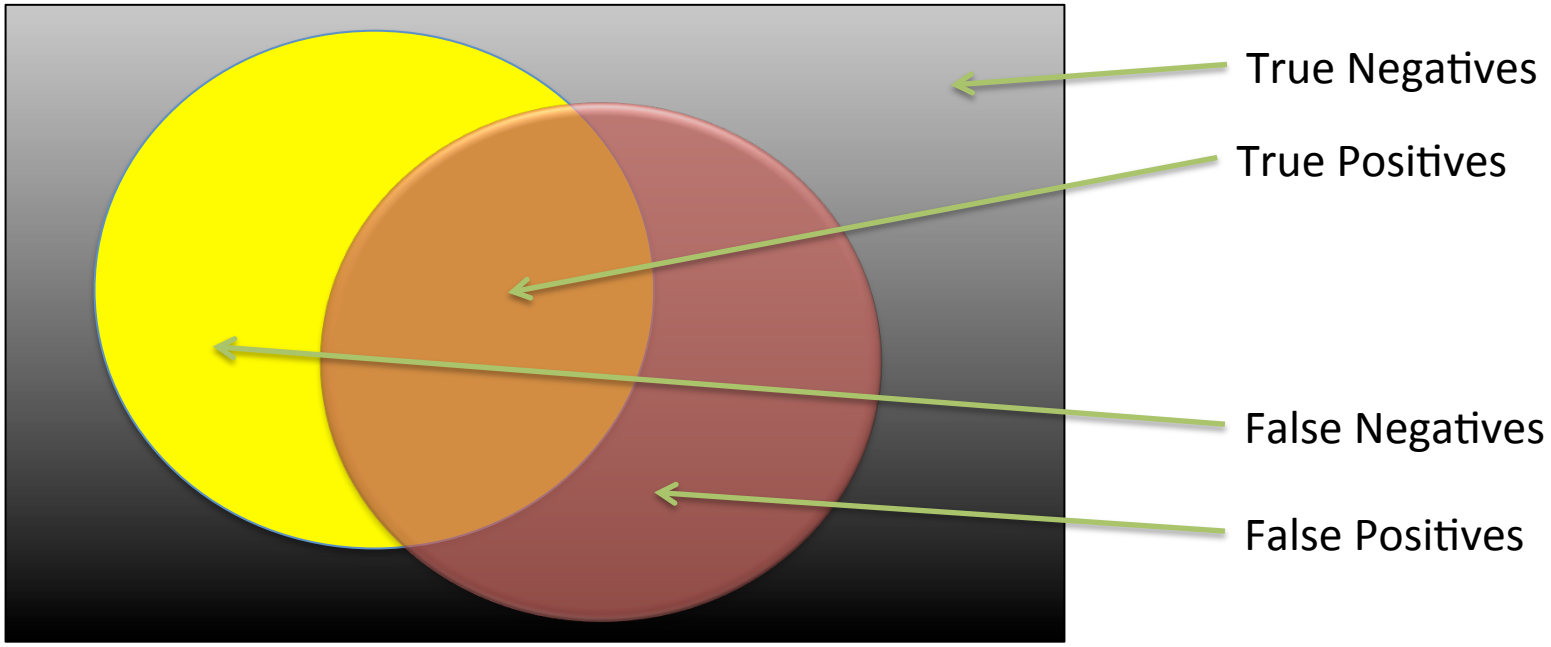
Errors

- The process we just went through was based on fixing two kinds of errors
  - Matching strings that we should not have matched (there, then, other)
    - False positives (Type I)
  - Not matching things that we should have matched (The)
    - False negatives (Type II)

Errors cont.

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
  - Increasing accuracy or **precision** (minimizing false positives)
  - Increasing coverage or **recall** (minimizing false negatives).

Errors



- Gold = the true answers (Yellow and Orange above)
- Prediction = the guessed answers (Red and Orange above)
- Intersection of Gold and Prediction (Orange) = True Positive
- Precision = Intersection / Prediction = True Positive / (True Positive + False Positive)
- Recall = Intersection / Gold = True Positive / (True Positive + False Negative)

Regular Expressions
A More Complex Example

- "any PC with more than ***500 MHz*** and ***32 Gb*** of disk space for less than ***$1000***"

```
/$[0-9]+/

/$[0-9]+\.[0-9][0-9]/

/\b$[0-9]+(\.[0-9][0-9])?\b/

/\b[0-9]+ *(MHz|[Mm]egahertz|GHz|[Gg]igahertz)\b/

/\b[0-9]+ *(Mb|[Mm]egabytes?)\b/

/\b[0-9](\.[0-9]+)? *(Gb|[Gg]igabytes?)\b/

/\b(Win95|Win98|WinNT|Windows *(NT|95|98|2000)?)\b/

/\b(Mac|Macintosh|Apple)\b/
```

Regular Expressions
Advanced Operators

Aliases for common sets of characters

RE	Expansion	Match	Example Patterns
\d	[0-9]	any digit	Party␣of␣5
\D	[^0-9]	any non-digit	Blue␣moon
\w	[a-zA-Z0-9␣]	any alphanumeric or space	Daiyu
\W	[^\w]	a non-alphanumeric	!!!!
\s	[␣\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in␣Concord

Regular Expressions
Advanced Operators

Regular expression operators for counting

RE	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	exactly zero or one occurrence of the previous char or expression
{n}	*n* occurrences of the previous char or expression
{n,m}	from *n* to *m* occurrences of the previous char or expression
{n,}	at least *n* occurrences of the previous char or expression

Regular Expressions
Advanced Operators

Some characters that need to be backslashed

RE	Match	Example Patterns Matched
*	an asterisk "*"	"K*A*P*L*A*N"
\.	a period "."	"Dr. Livingston, I presume"
\?	a question mark	"Would you light my candle?"
\n	a newline	
\t	a tab	

Unix Commands

- **cat**: concatenate and print files on screen
  - cat Bible-English.txt
- **more**: view file page by page
  - more Bible-English.txt
  - <ctrl-c> to get out of the viewing mode
- **head -<number>**: show head of file
  - head -5 Bible-English.txt
- **tail -<number>**: show tail of file
  - tail -5 Bible-English.txt
- **|** : the [pipe] is used to string commands
  - pass output of one command as input to another command
  - cat Bible-English.txt| head -100 |tail -5
- **>** : create a new file with the output of the command
  - cat Bible-English.txt| head -100 |tail -5 > new.txt
  - cat new.txt

Unix Commands

- **wc**: word count
  - Produce line, word and character counts
  - wc Bible-English.txt

    31102 789635 4138512 Bible-English.txt

    31,102 lines

    789,635 words

    4,138,512 characters

Unix Commands

- **sort** : sort lines
- **sort -u** : sort and unique
- **sort -nr** : sort line with reverse numerical order
- **uniq -c** : unique adjacent lines and count
- cat Bible-English.txt|sort|uniq -c|sort -nr |head -3

72 And the LORD spake unto Moses, saying,

12 One young bullock, one ram, one lamb of the first year, for a burnt offering:

12 And the word of the LORD came unto me, saying,

Next Time

- Discuss assignment #1 in some detail
  - Come prepared with questions
- More Regex and Unix tools!