Basic Text Processing

Regular Expressions

Regular expressions

A formal language for specifying text strings

How can we search for any of these?

- woodchuck
- woodchucks
- Woodchuck
- Woodchucks

Regular Expressions: Disjunctions

Letters inside square brackets []

Pattern	Matches
[Ww]oodchuck	Woodchuck, woodchuck
[0123456789]	Any digit

Ranges [A-Z]

Pattern	Matches	
[A-Z]	An upper case letter	Drenched Blossoms
[a-z]	A lower case letter	my beans were impatient
[0-9]	A single digit	Chapter 1: Down the Rabbit Hole

Regular Expressions: Negation in Disjunction

Negations [^Ss]

◦ Carat means negation only when first in []

Pattern	Matches	
[^A-Z]	Not an upper case letter	O**y**fn pripetchik
[^Ss]	Neither 'S' nor 's'	**I** have no exquisite reason"
[^e^]	Neither e nor ^	Look here
a^b	The pattern a carat b	Look up **a^b** now

Regular Expressions: More Disjunction

Woodchuck is another name for groundhog!

The pipe | for disjunction

Pattern	Matches
woodchuck	groundhog	woodchuck
yours	mine	<u>yours</u> mine
a	b	c	= [abc]
☆ [gG]roundhog	[Ww]oodchuck	Woodchuck

Regular Expressions: ? *+ .

Pattern	Matches	
	Optional previous char	color colour
	0 or more of previous char	oh! ooh! oooh! ooooh!
	1 or more of previous char	oh! ooh! oooh! ooooh!
		baa baaa baaaa baaaaa
		begin begun begun beg3n

Handwritten annotations in Pattern column:

coloou?r

oo*h!

o+h!

ba+

beg.n
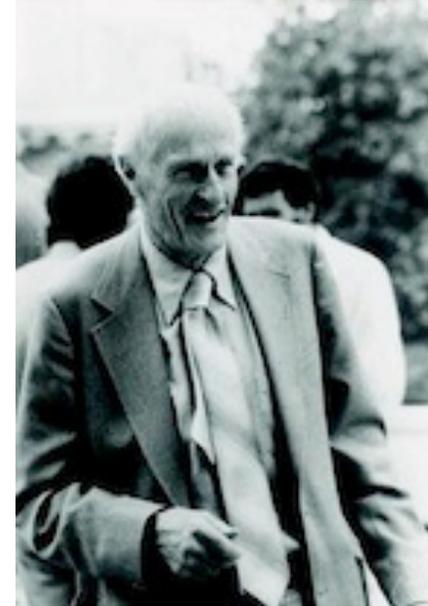
Stephen C Kleene

Kleene *, Kleene +

Regular Expressions: Anchors ^ $

Pattern	Matches
^[A-Z]	<u>P</u>alo Alto ~~palo Alto~~
^[^A-Za-z]	<u>1</u> <u>"</u>Hello"
\.$	The end<u>.</u>
[?.!]$	The end<u>?</u> The end<u>!</u>
.$	The end? The end

Example

[ ]
disjunction over characters

[^]
negated set

[-]
ranges

|
disjunction over strings

. any

* for 0 or more

+ for 1 or more

^ beginning of string

$ end of string

? optional

Find me all instances of the word "the" in a text.

[the]

h^t_e

[T+]he   The   the

?[the]?   ? +   ?

?e

the   the   ~~The~~

[^a-zA-z]   [t T]he   [^a-z A-z]   !The !

Errors

The process we just went through was based on fixing two kinds of errors:

1. Matching strings that we should not have matched (there, then, other)
   False positives (Type I errors)

   [T+]ne.t

2. Not matching things that we should have matched (The)
   False negatives (Type II errors)

   the

Error Types

	REAL "the"	NOT "the"
GUESS "the"	true positive	false positive
GUESS not "the"	false negative	true negative

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).

Basic Text Processing

Advanced Regular Expressions

Capture Groups

- Say we want to put angles around all numbers:

  the 35 boxes → *the <35> boxes*

- Use parens () to "capture" a pattern into a numbered register (1, 2, 3...)

- Use \1 to refer to the contents of the register

  `s/([0-9]+)/<\1>/`

Capture groups: multiple registers

`/the (.*)er they (.*), the \1er we \2/`

Matches

the faster they run, the faster we run
the sillier they sing, the sillier we sing

✗ the fastr they run, the sillier we sing

But suppose we don't want to capture?

Parentheses have a double function: grouping terms, and capturing

Non-capturing groups: add a ?: after paren:

`/(?:some|a few) (people|cats) like some \1/`

some   cats   like   some   cats

✗ some   Cats   like   some   some

Lookahead assertions

`(?= pattern)` is true if pattern matches, but is **zero-width; doesn't advance character pointer**

`(?! pattern)` true if a pattern does not match

How to match, at the beginning of a line, any single word that doesn't start with "Volcano":

$$\text{^(?! Volcano) [A-Za-z]+}$$

↑

at least once

Basic Text Processing

Regular Expression Applications

Simple Application: ELIZA

Early NLP system that imitated a Rogerian psychotherapist

- ◦ Joseph Weizenbaum, 1966.

Uses pattern matching to match, e.g.,:

- ◦ "I need X"

and translates them into, e.g.

- ◦ "What would it mean to you if you got X?

Simple Application: ELIZA

Men are all alike.
IN WHAT WAY

They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE

Well, my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE

He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED

How ELIZA works

s/.* I'M (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/

s/.* I AM (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/

s/.* all .*/IN WHAT WAY?/

s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE?/

Text Processing Tips and Tricks

- list comprehensions

- sorting with lambdas

- strip(), replace(), and split()

- sys.argv for reading in data

- json and CSV reading/writing