

# 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
<code>[wW]oodchuck</code>	Woodchuck, woodchuck
<code>[1234567890]</code>	Any digit

- Ranges `[A-Z]`

Pattern	Matches	
<code>[A-Z]</code>	An upper case letter	<u>D</u> renched Blossoms
<code>[a-z]</code>	A lower case letter	<u>m</u> y beans were impatient
<code>[0-9]</code>	A single digit	Chapter <u>1</u> : Down the Rabbit Hole

# Regular Expressions: Negation in Disjunction

- Negations `[ ^Ss ]`
  - Carat means negation only when first in []

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

# Regular Expressions: More Disjunction

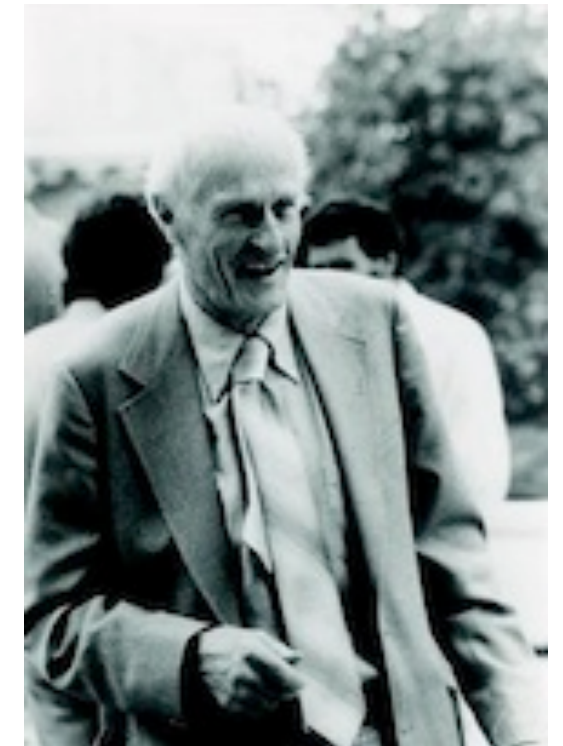
- Woodchucks is another name for groundhog!
- The pipe | for disjunction

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



# Regular Expressions: ? \* + .

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



Stephen C Kleene

Kleene \*, Kleene +

# Regular Expressions: Anchors <sup>^</sup> \$

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



# Example

- Find me all instances of the word “the” in a text.

the

Misses capitalized examples

[tT]he

Incorrectly returns other or theology

[^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).

# Summary

- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers
  - Can be very useful in capturing generalizations