Basic Text Processing

Regular Expressions

Regular expressions (regexes)

Q: How can we search for any of these?

- woodchuck
- woodchucks
- Woodchuck
- Woodchucks

A: Regexes! Patterns that include some subset of all possible strings and exclude everything else

A formal language for specifying text strings

Regular Expressions: Disjunctions

Letters inside square brackets []

(matches any of those characters)

Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	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	Oyfn 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

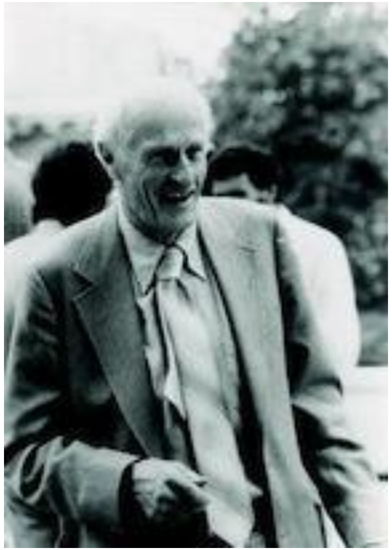
Woodchuck is another name for groundhog!

The pipe | for disjunction

Pattern	Matches
groundhog|woodchuck	woodchuck
yours|mine	yours
a|b|c	== [abc]
[gG]roundhog|[Ww]oodchuck	Woodchuck

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 ^ $

Related to regex parser:

Assert current position matches some pre-defined location (generally start or end of string)

Pattern	Matches
`^[A-Z]`	Evanston
`^[^A-Za-z]`	1 "Hello"
`\.$`	The end.
`.$`	The end? The end!

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:

1. Matching strings that we should not have matched (there, then, other)

   False positives (Type I errors)

2. Not matching things that we should have matched (The)

   False negatives (Type II errors)

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 task

For hard tasks, we often use machine learning classifiers

- But regular expressions are still used for pre-processing, or as features in the classifiers
- Can be very useful in capturing generalizations

Basic Text Processing

Regular Expressions

Basic Text Processing

More Regular Expressions: Substitutions and ELIZA

Substitutions

Substitution in Python and UNIX commands:

`s/regexp1/pattern/`

e.g.:

`s/colour/color/`

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 ran, the faster we ran

But not

the faster they ran, the faster we ate

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/
```

matches

- `some cats like some cats`

but not

- `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”:

`/^(?!Volcano)[A-Za-z]+/`

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?/

Basic Text Processing

More Regular Expressions: Substitutions and ELIZA