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
`[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

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

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

python: re library
re.findall
re.search
re.split
- - -

Regular Expressions: ? * + .

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

.+ : Anything length ≥ 1

.{0,10}

Stephen C Kleene

Kleene *, Kleene +

Regular Expressions: Anchors ^ $

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

Text normalization

- Every NLP task needs text normalization
  - 1. Segment/tokenize words in running text
  - 2. Normalizing word formats
  - 3. Sentence segmentation (typically)

easy: case BlackLivesMatter => blacklivesmatter

harder: stemming cats => cat

cat

US

U.S.A.

Type vs Token

- I saw one cat and then more cats! and another cat
  1 2 3 4 5 6 7 8 9 10 11
- **N** = number of tokens = 11
- **V** = vocabulary = set of types

V = { I, saw, cat, one, then, and, more, cats, another }

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

Word frequencies

Word	Frequency (f)
the	1629
and	844
to	721
a	627
she	537
it	526
of	508
said	462
i	400
alice	385

Alice's Adventures in Wonderland, by Lewis Carroll

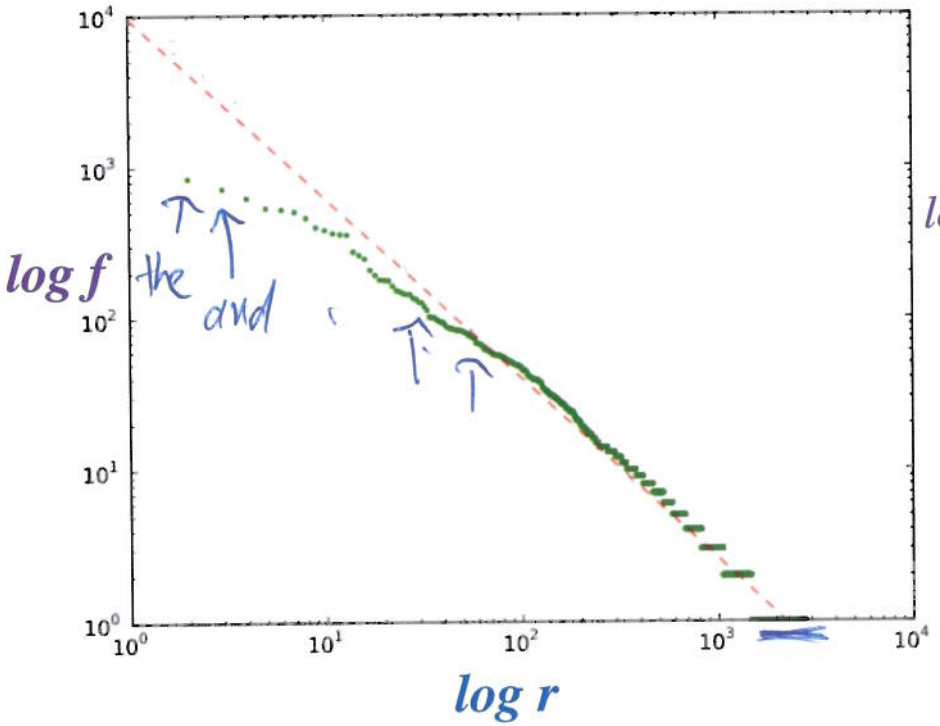
Zipf's Law

- When word types are ranked by frequency, then frequency (f) * rank (r) is roughly equal to some constant (k)

$$f \times r = k$$

Rank (r)	Word	Frequency (f)	$r \cdot f$
1	the	1629	1629
2	and	844	1688
3	to	721	2163
4	a	627	2508
5	she	537	2685
6	it	526	3156
7	of	508	3556
8	said	462	3696
9	i	400	3600
10	alice	385	3850
20	all	179	3580
30	little	128	3840
40	about	94	3760
50	again	82	4100
60	queen	68	4080
70	don't	60	4200
80	quite	55	4400
90	just	51	4590
100	voice	47	4700
200	hand	20	4000
300	turning	12	3600
400	hall	9	3600
500	kind	7	3500

Plot: log frequencies



recall:

$f * r = k$

$\log f + \log r = \log k$