Words & Regexes

CS 485, Fall 2023
Applications of Natural Language Processing
https://people.cs.umass.edu/~brenocon/cs485_f23/

Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst

- Thanks for survey responses!
- Questions
  - Python version 3 (more to come)
  - Midterm: TBA
- Near-term roadmap

Survey bags-of-words

Why are you interested in this class? Why are you interested in NLP?

Survey bags-of-words

What natural languages do you speak or read?

English and pig latin

English, barely spanish.

- Today: to do NLP you need to get started with text data

1. Text normalization: cleaning up text to reasonable sequences of words (tokens)

2. Regular expressions: a computational tool to help, and even do rule-based NLP

Text normalization

- Every NLP task needs text normalization
  - 1. Segment/tokenize words in running text
  - 2. Normalizing word formats
  - 3. Sentence segmentation
    and/or
    paragraphs/sections/chapters/etc.

Example preprocessing pipeline

Raw Text

```
103d CONGRESS
  1st Session
                                  H. R. 3

                    [Report No. 103-375, Part I]

  To amend the Federal Election Campaign Act of 1971 to provide for a
  voluntary system of spending limits and benefits for congressional
              election campaigns, and for other purposes.
```

Text cleaning
and term extraction

Unigrams

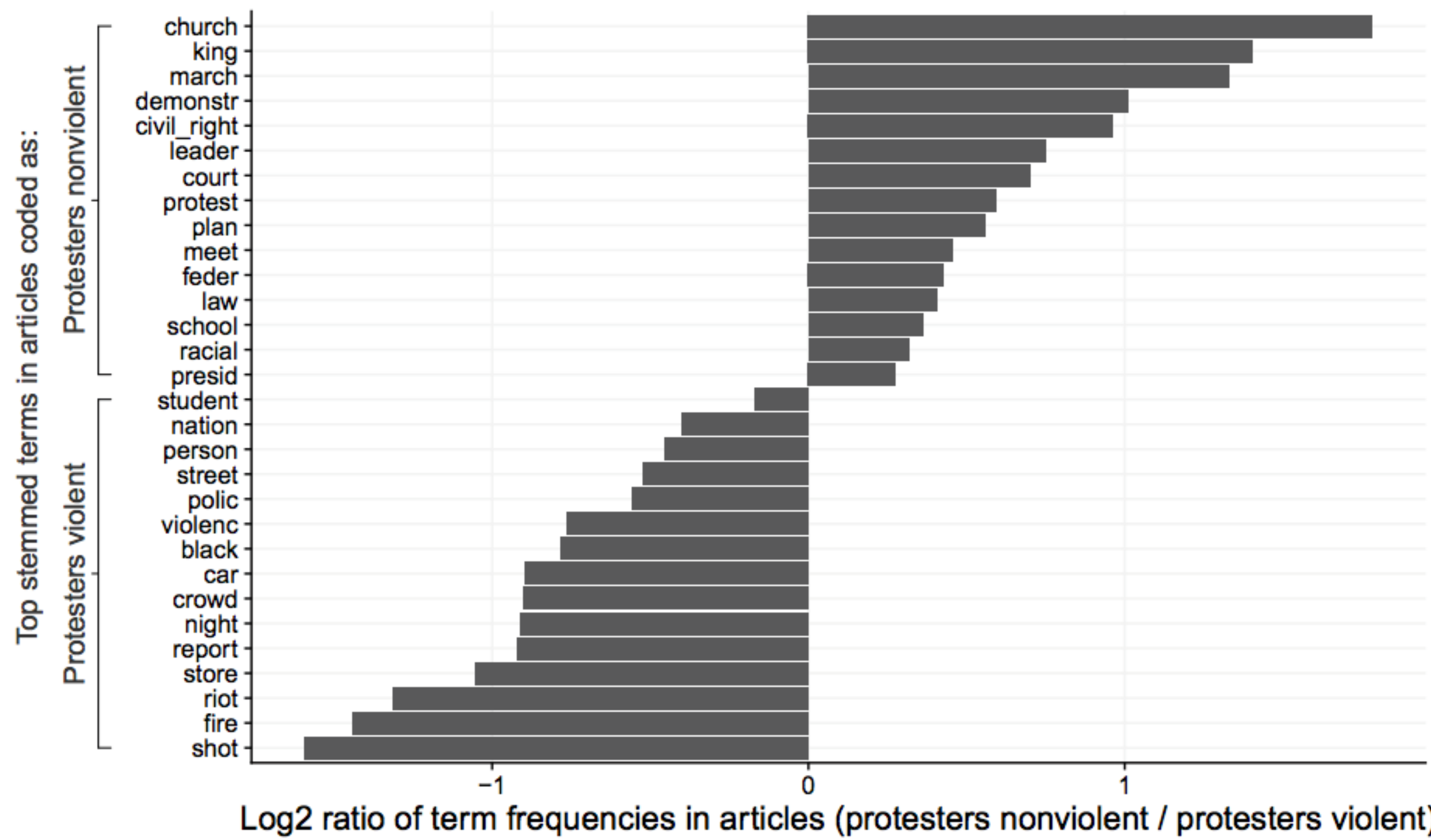
Word	Count
candidate	215
section	158
Federal	154
election	140
committee	120
under	115
that	114
...	...

Downstream
analysis

Text classification
Information extraction
...

Word statistics example
Corpus: news articles from late 1960s

FIGURE 10. Ratio of Term Frequencies in Articles About Protests Coded as Protester Nonviolent or Protester Violent



Wasow 2020

```
DISCHARGE CONDITION:  The patient was able to oxygenate
on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.

DISCHARGE STATUS:  The patient will be discharged to
[**Hospital1 **]
for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
```

Text data (MIMIC III EHR)

All-caps headers delineate sections: should be parsed out as structure

Unstructured, linguistic data
Has *semantic* structure: describes properties and relationships among entities

```
DISCHARGE CONDITION:  The patient was able to oxygenate
on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.

DISCHARGE STATUS:  The patient will be discharged to
[**Hospital1 **]
for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
```

Text data
(MIMIC III EHR)

Semi-structured, regular ordering
MEDICINE_NAME NUMBER UNITS MODIFIERS

All-caps headers delineate sections: should be parsed out as structure

Unstructured, linguistic data
Has *semantic* structure: describes properties and relationships among entities

```
DISCHARGE CONDITION:  The patient was able to oxygenate
on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.

DISCHARGE STATUS:  The patient will be discharged to
[**Hospital1 **]
for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
```

Text data (MIMIC III EHR)

1. Easy to structure: write hard-coded, custom string processor

2. Harder: develop more complex processor

3. Hardest: full natural language

Semi-structured, regular ordering
MEDICINE_NAME NUMBER UNITS MODIFIERS

(1) and **(2)**: regular expressions are often useful!

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`

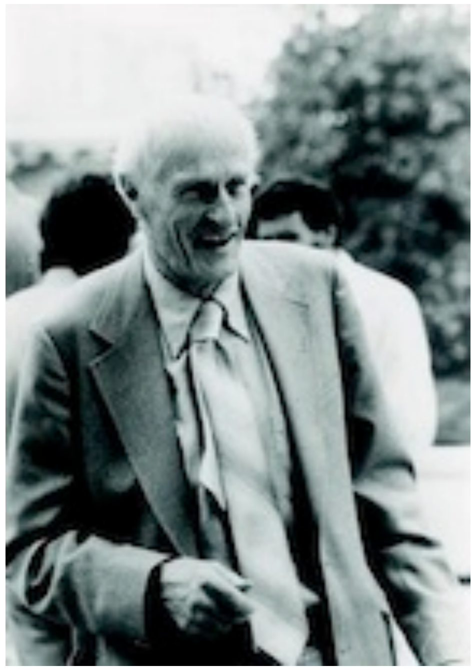
Regular Expressions: More Disjunction

- 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	

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 +

Demo: regexes in *grep*

```
grep -Poi '#[^ ]*livesmatter'
```

```
grep -Poi '#[a-z0-9]*livesmatter'
```

```
cat 2016-03-01.text.txt | grep -Poi '#[a-z0-9]*livesmatter' | sort | uniq -c | less
```

```
      3 #ALLLIVESMATTER
      1 #AllJonasLivesMatter
     81 #AllLivesMatter
      1 #Alllivesmatter
      2 #AmericanLivesMatter
      2 #ArmenianLivesMatter
     20 #BLACKLIVESMATTER
      1 #BLACKLivesMatter
      4 #BLackLivesMatter
      1 #BearLivesMatter
      1 #BeerLivesMatter
      1 #BeigeLivesMatter
    948 #BlackLivesMatter
      1 #BlackLlivesMatter
      1 #BlackMuslimLivesMatter
      1 #BlackTransLivesMatter
      1 #BlacklLivesMatter
      2 #BlacklivesMatter
     26 #Blacklivesmatter
      1 #Blackslivesmatter
      2 #BlueLIvesMatter
     90 #BlueLivesMatter
      1 #Bluelivesmatter
      1 #BookoutLivesMatter
      2 #BrownLivesMatter
      1 #BugLivesMatter
      1 #CatsLivesMatter
      1 #Chickenlivesmatter
```

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).

Simple Tokenization in UNIX

- (Inspired by Ken Church's UNIX for Poets.)
- Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt     Change all non-alpha to newlines
     | sort                           Sort in alphabetical order
     | uniq –c                        Merge and count each type
```

```
1945 A          25 Aaron
  72 AARON       6 Abate
  19 ABBESS      1 Abates
   5 ABBOT       5 Abbess
 ... ...         6 Abbey
                 3 Abbot
               .... …
```

[Slide: SLP3]

Issues in Tokenization

- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??

[Slide: SLP3]

Tokenization: language issues

- French
  - ***L'ensemble*** → one token or two?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Want ***l'ensemble*** to match with ***un ensemble***
- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German information retrieval needs **compound splitter**

[Slide: SLP3]

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

[Slide: SLP3]

Preprocessing: Text cleaning

```
DISCHARGE CONDITION:  The patient was able to oxygenate
on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.

DISCHARGE STATUS:  The patient will be discharged to
[**Hospital1 **]
for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
```

```
The patient was able to oxygenate on
room air at 93% at the time of discharge.  She was profoundly
weak, but was no longer tachycardic and had a normal blood
pressure.  Her respirations were much improved albeit with
transmitted upper airway sounds.
```

- Remove unwanted structure to just the sentences/ paragraphs you want to analyze
- Get text out of weird formats like HTML, PDFs, or idiosyncratic formatting (e.g. in these EHRs)

This step is usually specific to your dataset

Preprocessing: Tokenization

```
The patient was able to oxygenate on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.
```

```
['The', 'patient', 'was', 'able', 'to', 'oxygenate',
'on', 'room', 'air', 'at', '93', '%', 'at', 'the',
'time', 'of', 'discharge', '.', 'She', 'was',
'profoundly', 'weak', ',', 'but', 'was', 'no',
'longer', 'tachycardic', 'and', 'had', 'a',
'normal', 'blood', 'pressure', '.', 'Her',
'respirations', 'were', 'much', 'improved',
'albeit', 'with', 'transmitted', 'upper', 'airway',
'sounds', '.']
```

- Words are (usually) the basic units of analysis in NLP.
- In English, words are delineated as tokens via space and punctuation conventions, recognizable via moderately simple rules
- Tokenization: from text string to sequence of word strings
- Sentence splitting: harder but sometimes done too

There are good off-the-shelf tokenizers (NLTK, SpaCy, CoreNLP, Twokenizer)

Preprocessing: Normalization

- Often:
  - Lowercase words (“She” -> “she”)
- Sometimes:
  - Remove numbers (“93” -> “NUMBER_NN”)
  - Correct misspellings / alternate spellings (“color” -> “colour”)
- Problem specific:
  - Resolve synonyms / aliases (if you know them already)
  - Remove “stopwords”
    - Punctuation and grammatical function words (“if”, “the”, “by”), and
    - Very common words in your domain that don’t add much meaning

How many words?

N = number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

Church and Gale (1990): $|V| > O(N^{½})$

	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

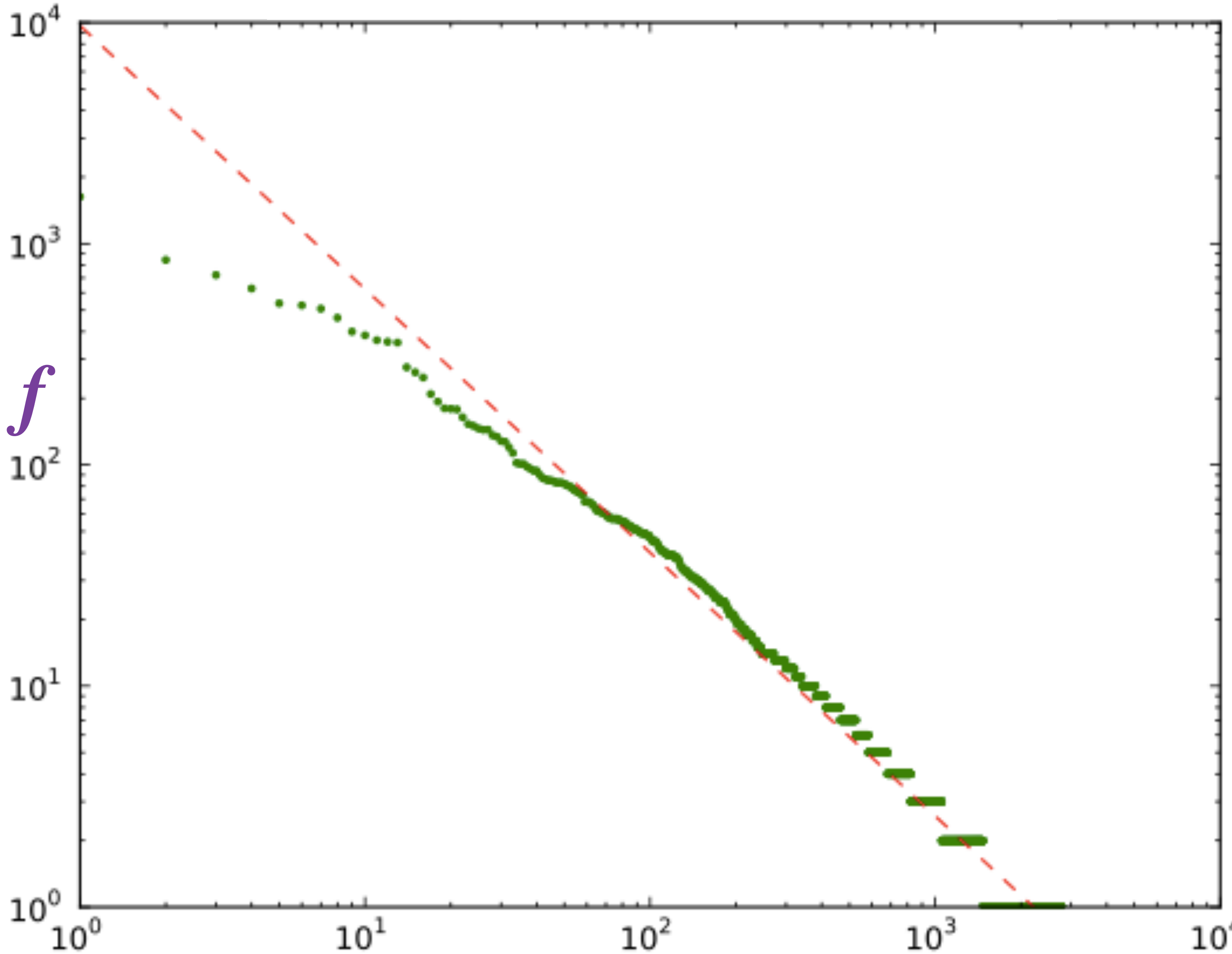
Zipf's Law

- 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



log f

log r

recall:

$f * r = k$

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

Normalization

- Need to “normalize” terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match ***U.S.A.*** and ***USA***
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***
- For sentiment analysis, MT, Information extraction
  - Case is helpful (***US*** versus ***us*** is important)

Lemmatization

- Reduce inflections or variant forms to base form
  - *am, are, is* $\rightarrow$ *be*
  - *car, cars, car's, cars'* $\rightarrow$ *car*
- *the boy's cars are different colors* $\rightarrow$ *the boy car be different color*
- Lemmatization: have to find correct dictionary headword form
- Machine translation
  - Spanish quiero ('I want'), quieres ('you want') same lemma as querer 'want'

Morphology

- **Morphemes**:
  - The small meaningful units that make up words
  - **Stems**: The core meaning-bearing units
  - **Affixes**: Bits and pieces that adhere to stems
    - Often with grammatical functions

Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Porter's algorithm
The most common English stemmer

Step 1a

```
sses → ss   caresses → caress
ies  → i    ponies   → poni
ss   → ss   caress   → caress
s    → ø    cats     → cat
```

Step 1b

```
(*v*)ing → ø  walking   → walk
              sing      → sing
(*v*)ed  → ø  plastered → plaster
...
```

Step 2 (for long stems)

```
ational→ ate  relational→ relate
izer→ ize     digitizer → digitize
ator→ ate     operator  → operate
...
```

Step 3 (for longer stems)

```
al    → ø  revival    → reviv
able  → ø  adjustable → adjust
ate   → ø  activate   → activ
...
```

Consider the IR query matching problem.
What are the precision/recall tradeoffs of the Porter stemmer?