Lecture 2: Corpora and Text Processing

USC VSoE CSCI 544: Applied Natural Language Processing

Jonathan May -- 梅約納

August 25, 2017

Slides from Kristy Hollingshead-Seitz, Nathan Schneider, Dan Jurafsky

Homework 1 is available

- Due Friday, 9/8 (Note: Homework 2 will become available Friday, 9/1)
- Warmup to get familiar with Python, Vocareum, solving language problems
- No special knowledge from lectures should be needed to work on it
- But feel free to ask questions in Piazza!

Quiz

- See Piazza

Lot of Work on the Terminal Today

- Get set up in Vocareum, on the 'In-class' assignment
- Or download code/data from my webpage to your own machine
  - Mac: you'll probably need to install gnu stuff with homebrew/ports; preface non-working commands with 'g'
  - Windows: cygwin is usually pretty good but it's been a while for me...
- We're going to code 'competitively': post your answers in Piazza

Pre-Statistical NLP

- NLP (really, more like Computational Linguistics) through the 80s was mostly about modeling specific linguistic phenomena with code
- Linguists/highly trained coders wrote fine-grained detailed rules to capture various aspects
  - E.g. "swallow" is a verb of ingestion, taking an animate subject and a physical object that is edible...
- Very time-consuming, expensive, limited coverage, but high precision
- Academically satisfying, but not good at producing systems beyond the demo phase

h/t K. Hollingshead-Seitz

The Alternative?

- Empirical approach: learn by observing language as it's used "in the wild"
- Many different names:
  - Corpus Linguistics
  - Empirical NLP
  - Statistical NLP
- Central tools:
  - Corpus
  - Thing to count with (i.e. statistics)

Advantages

- Generalize patterns as they actually exist (i.e. bottom-up, not top-down)
- Little need for knowledge (just count)
- Systems are robust and adaptable (change domain by changing corpus)
- Systems degrade more gracefully (corner cases captured in data)
- Evaluations are (more) meaningful

Limitations

- Bound by data – can't model what you can't see
- Big Data methods fail when the data is small
  - I just got finished trying to build an Oromo-English translation system in three weeks with 50,000 words of the bible...most MT is built on 10mw+
- Tends to be more computationally expensive (but less human-expensive)
  - Usually a good trade-off, but it's application-dependent
- But, a lot of effort was spent in coming up with rules that work well in certain corners
  - Mostly English
  - Mostly formal
- Do what you need to, leverage the resources you have, consider trade-offs

It's all about the corpus!

- Corpus (pl. corpora): a collection of (natural language) text systematically gathered and organized in some manner
- Features
  - Size
  - Balanced/domain
  - Written/Spoken
  - Raw/Annotated
  - Free/Pay
- Famous (Text) Examples
  - Brown Corpus: 1m words balanced English text, POS tags
  - Wall Street Journal: 1m words English news text, syntax trees
  - Canadian Hansards: 10m words French/English parliamentary text, aligned at sentence level
  - Clueweb 12: 100+b words English web text
  - Google books ngrams: 500B words

How Big Does It Need To Be?

- We'd like to get examples of all linguistic phenomena, ideally several times so we know how likely they are to occur
- How big should a corpus be to get every possible sentence in English?
  - Every possible idea?
  - Every 5-word phrase?
  - Every word?
- None of these are possible!

Sparsity and Zipf's Law

- Let's look at the frequencies of different words in a large text corpus
- Assume "word" is a string of letters separated by spaces (oversimplification!)

h/t Nathan Schneider

Vocareum Demo

words and counts

Reminder

- Count the word types:
  - `sed 's/ /\n/g' corpus | sort | uniq -c | grep –v "^$" | sort -k1nr`
  - `how many 1-count types? What % of all types?`
  - `how many tokens?`
- Count the ngram types:
  - `paste 1grams <(tail -n+2 1grams) | sort | uniq –c | sort –k1nr`
  - `how many 1-count types? What % of all types?`

Word Counts

Most frequent words in the English Europarl corpus (out of 24m word **tokens**)

any word	
Frequency	Token
1,698,599	the
849,256	of
793,731	to
640,257	and
508,560	in
407,638	that
400,467	is
394,778	a
263,040	I

nouns	
Frequency	Token
124,598	European
104,325	Mr
92,195	Commission
66,781	President
62,867	Parliament
57,804	Union
53,683	report
53,547	Council
45,842	States

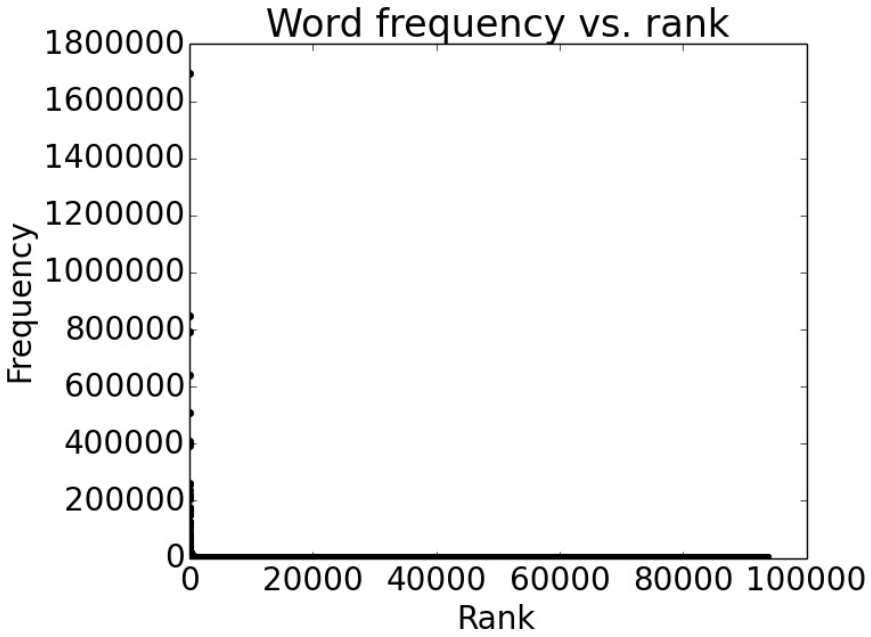
Word Counts

But also, out of 93,638 distinct words (**word types**), 36,231 occur only once. Examples:

- cornflakes, mathematicians, fuzziness, jumbling
- pseudo-rapporteur, lobby-ridden, perfunctorily,
- Lycketoft, UNCITRAL, H-0695
- policyfor, Commissioneris, 145.95, 27a

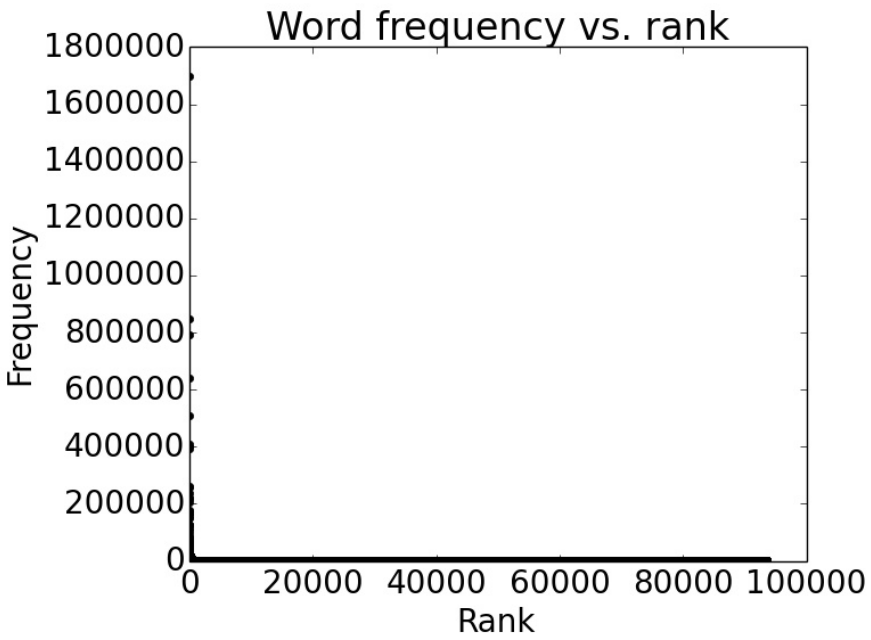
Plotting word frequencies

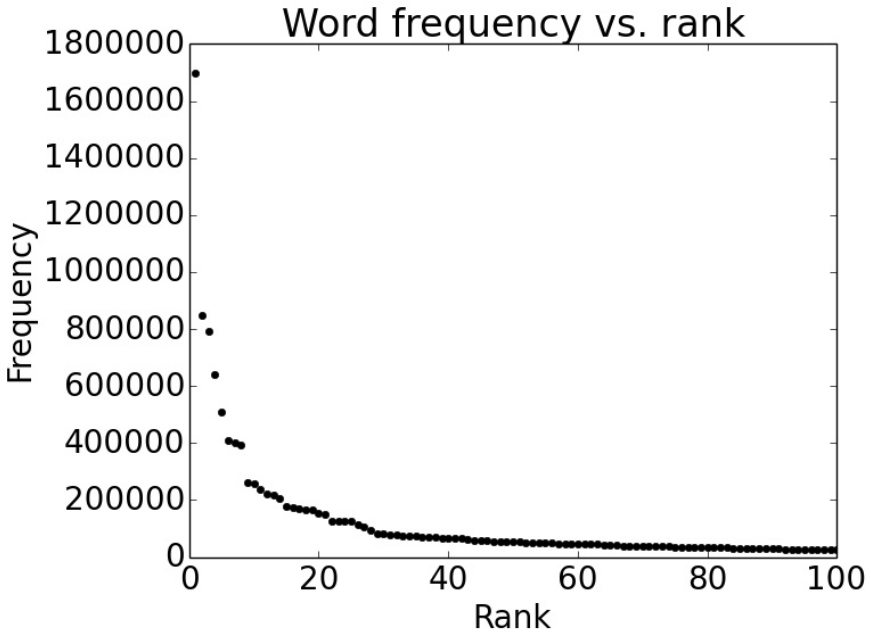
Order words by frequency. What is the frequency of nth ranked word?



Plotting word frequencies

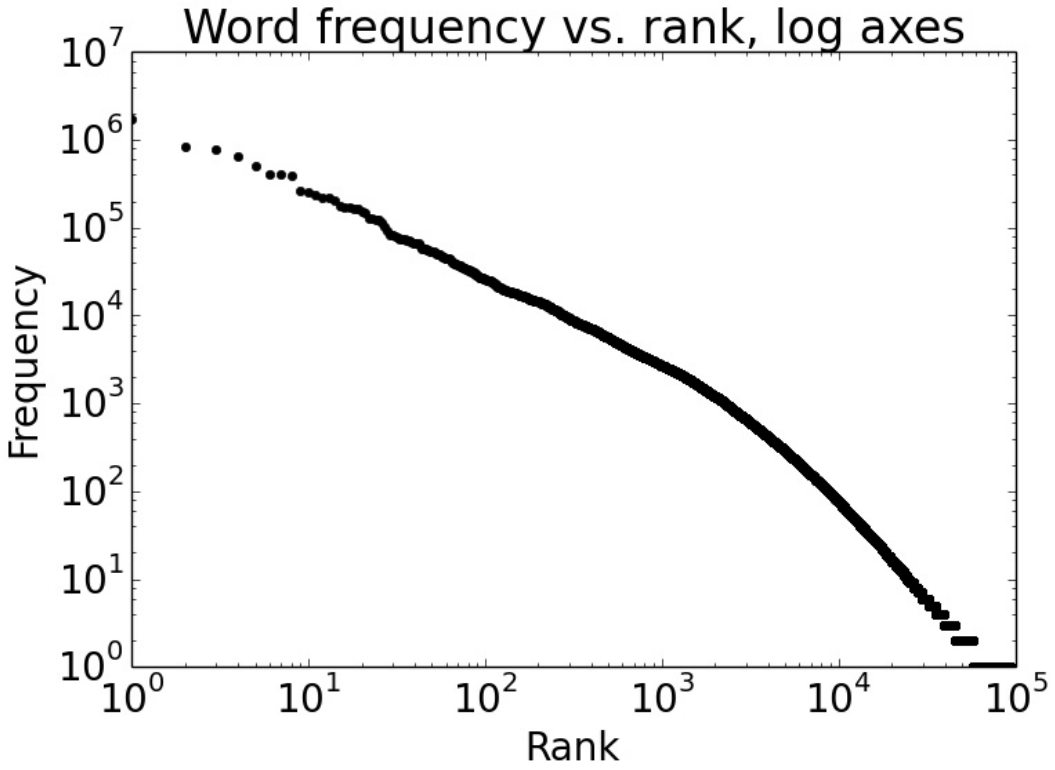
Order words by frequency. What is the frequency of nth ranked word?

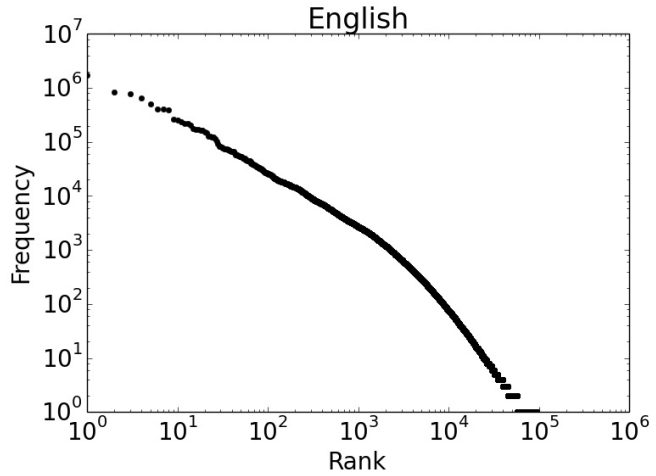


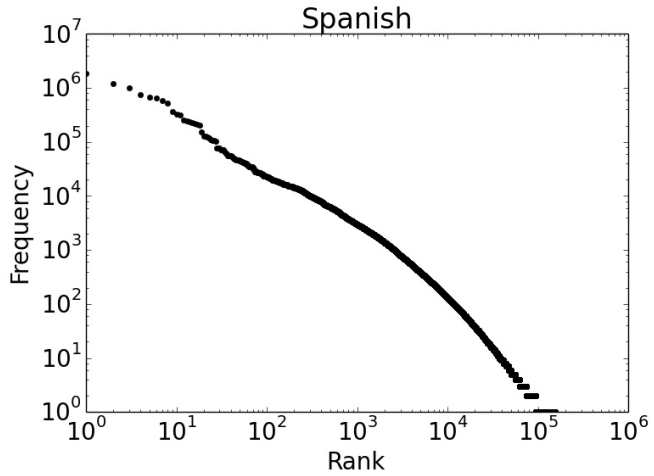


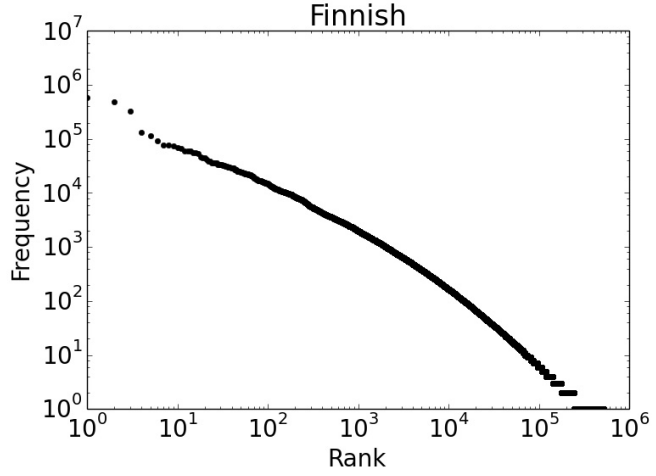
Rescaling the axes

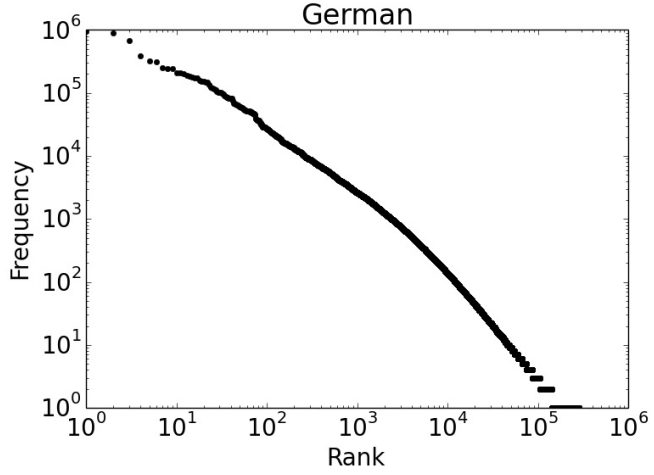
To really see what's going on, use logarithmic axes:




English
Frequency
Rank


Spanish
Frequency
Rank


Finnish
Frequency
Rank


German
Frequency
Rank

Zipf's law

Summarizes the behaviour we just saw:

$$f \times r \approx k$$

- $f =$ frequency of a word
- $r =$ rank of a word (if sorted by frequency)
- $k =$ a constant

Zipf's law

Summarizes the behaviour we just saw:

$$f \times r \approx k$$

- $f =$ frequency of a word
- $r =$ rank of a word (if sorted by frequency)
- $k =$ a constant

Why a line in log-scales? $fr = k \quad \Rightarrow \quad f = \frac{k}{r} \quad \Rightarrow \quad \log f = \log k - \log r$

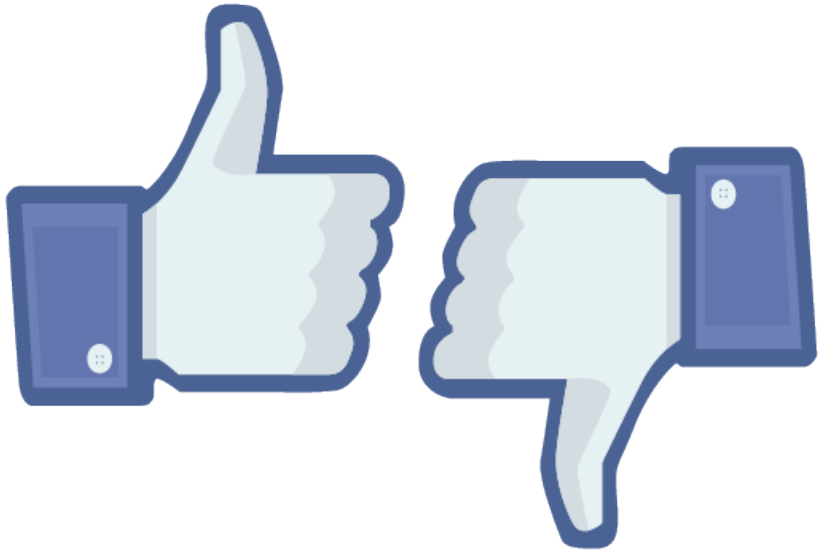
Implications of Zipf's Law

- Regardless of how large our corpus is, there will be a lot of infrequent (and zero-frequency!) words.
- In fact, the same holds for many other levels of linguistic structure (e.g., syntactic rules in a CFG).
- This means we need to find clever ways to estimate probabilities for things we have rarely or never seen.

Zipf's Law and n-grams

- Sparsity keeps getting worse
- Moby Dick
  - 110927 tokens; 18971 types (17% of tokens); 11874 1-count (62.6% of types)
  - 110927 3-grams; 103531 types (93% of tokens); 99150 1-count (95.7% of types)

Sentiment Analysis



Goal: Predict the **opinion** expressed in a piece of text. E.g., $+$ or $-$. (Or a rating on a scale.)

Sentiment Analysis

Filled with horrific dialogue, laughable characters, a laughable plot, ad really no interesting stakes during this film, "Star Wars Episode I: The Phantom Menace" is not at all what I wanted from a film that is supposed to be the huge opening to the segue into the fantastic Original Trilogy. The positives include the score, the sound effects, and most of the

KJ Proulx (/user/id/896976177/)
★ Super Reviewer

Extraordinarily faithful to the tone and style of the originals, The Force Awakens brings back the Old Trilogy's heart, humor, mystery, and fun. Since it is only the first piece in a new three-part journey it can't help but feel incomplete. But everything that's already there, from the stunning visuals, to the thrilling action sequences, to the charismatic new characters,

Matthew Samuel Mirliani (/user/id/896467979/)
★ Super Reviewer

RottenTomatoes.com

Vocareum Quiz: What is the review given to each (1-5 stars)

Sentiment Analysis

★★

Filled with horrific dialogue, laughable characters, a laughable plot, ad really no interesting stakes during this film, "Star Wars Episode I: The Phantom Menace" is not at all what I wanted from a film that is supposed to be the huge opening to the segue into the fantastic Original Trilogy. The positives include the score, the sound effects, and most of the

KJ Proulx (/user/id/896976177/)
★ Super Reviewer

★★★★★

Extraordinarily faithful to the tone and style of the originals, The Force Awakens brings back the Old Trilogy's heart, humor, mystery, and fun. Since it is only the first piece in a new three-part journey it can't help but feel incomplete. But everything that's already there, from the stunning visuals, to the thrilling action sequences, to the charismatic new characters,

Matthew Samuel Mirliani (/user/id/896467979/)
★ Super Reviewer

RottenTomatoes.com

Sentiment Analysis

★★

Filled with horrific dialogue, laughable characters, a laughable plot, ad really no interesting stakes during this film, "Star Wars Episode I: The Phantom Menace" is not at all what I wanted from a film that is supposed to be the huge opening to the segue into the fantastic Original Trilogy. The positives include the score, the sound effects, and most of the

KJ Proulx (/user/id/896976177/)
★ Super Reviewer

★★★★★

Extraordinarily faithful to the tone and style of the originals, The Force Awakens brings back the Old Trilogy's heart, humor, mystery, and fun. Since it is only the first piece in a new three-part journey it can't help but feel incomplete. But everything that's already there, from the stunning visuals, to the thrilling action sequences, to the charismatic new characters,

Matthew Samuel Mirliani (/user/id/896467979/)
★ Super Reviewer

`RottenTomatoes.com` + intuitions about positive/negative cue words

So, you want to build a sentiment analyzer

Questions to ask yourself:

1. What is the input for each prediction?

2. What are the possible outputs?

3. How will it decide?

4. How will you measure its effectiveness?

BEFORE you build a system, choose a dataset for evaluation!

Why is data-driven evaluation important?

- Good science requires controlled experimentation.
- Good engineering requires benchmarks.
- Your intuitions about typical inputs are probably wrong.

Sometimes you want multiple evaluation datasets: e.g., one for **development** as you hack on your system, and one reserved for final **testing**.

Where can you get a corpus?

- Many corpora are prepared specifically for linguistic/NLP research with text from content providers (e.g., newspapers). In fact, there is an entire subfield devoted to developing new **language resources**.

- You may instead want to collect a new one, e.g., by scraping websites. (There are tools to help you do this.)

sentiment corpus: https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

Annotations

To evaluate and compare sentiment analyzers, we need reviews with **gold labels** ($+$ or $-$) attached. These can be

- derived automatically from the original data artifact (**metadata** such as star ratings), or

- added by a human annotator who reads the text
  - Issue to consider/measure: How consistent are human annotators? If they often have trouble deciding or agreeing, how can this be addressed?

An evaluation measure

Once we have a dataset with gold (correct) labels, we can give the text of each review as input to our system and measure how often its output matches the gold label.

Simplest measure:

$$\textbf{accuracy} = \frac{\#\text{ correct}}{\#\text{ total}}$$

More measures later in the course!

Vocareum Demo

Extracting and labeling data

Building an evaluation set

Inspecting for ideas

Reminder

- Cleaning up data
  - boilerplate python: very useful!
  - beautiful soup:
    - `soup = bs(infile, "lxml")`
    - `soup.find_all('review_text')`
- Adding labels
  - `paste elec.pos.txt <(yes "pos" | head -2198) > elec.pos.tagged`

Reminder

- Dividing into train, dev, blind test
  - `cat elec.pos.tagged elec.neg.tagged | shuf > elec.mix.tagged`
  - `head -800 elec.mix.tagged > elec.devtest`
  - `tail -n+801 elec.mix.tagged > elec.train`
  - `head -400 elec.devtest > elec.dev`
  - `tail -400 elec.devtest > elec.test`
  - `for i in train dev test; do cut f1 elec.$i > elecdata/$i.input; cut -f2 elec.$i > elecdata/$i.label; done`

- World's simplest scorer and test
  - `cat <(yes "pos" | head -200) <(yes "neg" | head -200) | shuf > dev.random`
  - `paste dev.random elecdata/dev.label | awk '$1==$2{hit++}END{printf("%f\n",hit/NR)}'`

Catching our breath

We now have:

✓ a definition of the sentiment analysis task (inputs and outputs)

✓ a way to measure a sentiment analyzer (accuracy on gold data)

So we need:

- an algorithm for predicting sentiment

A simple sentiment classification algorithm

Use a **sentiment lexicon** to count positive and negative words:

Positive:

absolutely	beaming	calm
adorable	beautiful	celebrated
accepted	believe	certain
acclaimed	beneficial	champ
accomplish	bliss	champion
achieve	bountiful	charming
action	bounty	cheery
active	brave	choice
admire	bravo	classic
adventure	brilliant	classical
affirm	bubbly	clean
...		...

Negative:

abysmal	bad	callous
adverse	banal	can't
alarming	barbed	clumsy
angry	belligerent	coarse
annoy	bemoan	cold
anxious	beneath	collapse
apathy	boring	confused
appalling	broken	contradictory
atrocious		contrary
awful		corrosive
		corrupt
		...

From `http://www.enchantedlearning.com/wordlist/`

Simplest rule: Count positive and negative words in the text. Predict whichever is greater.

Vocareum Demo

Human Experiment

Build a classifier

Some possible problems with simple counting

1. Hard to know whether words that *seem* positive or negative tend to actually be used that way.
   - sense ambiguity
   - sarcasm/irony
   - text could mention expectations or opposing viewpoints, in contrast to author's actual opinon

2. Opinion words may be describing (e.g.) a character's attitude rather than an evaluation of the film.

3. Some words act as semantic modifiers of other opinion-bearing words/phrases, so interpreting the full meaning requires sophistication:

I **can't** stand this movie

vs.

I **can't** believe how great this movie is

What if we have more data?

Perhaps corpora can help address the first objection:

1. Hard to know whether words that *seem* positive or negative tend to actually be used that way.

A data-driven method: Use frequency counts from a **training corpus** to ascertain which words tend to be positive or negative.

- Why separate the training and test data (held-out test set)? Because otherwise, it's just data analysis; no way to estimate how well the system will do on new data in the future.

Choice of training and evaluation data

We know that the way people use language varies considerably depending on **context**. Factors include:

- *Mode of communication:* speech (in person, telephone), writing (print, SMS, web)
- *Topic:* chitchat, politics, sports, physics, . . .
- *Genre:* news story, novel, Wikipedia article, persuasive essay, political address, tweet, . . .
- *Audience:* formality, politeness, complexity (think: child-directed speech), . . .

In NLP, **domain** is a cover term for all these factors.

Choice of training evaluation data

- Statistical approaches typically assume that the training data and the test data are sampled from the same distribution.
  - I.e., if you saw an example data point, it would be hard to guess whether it was from the training or test data.

- Things can go awry if the test data is appreciably different: e.g.,
  - different tokenization conventions
  - new vocabulary
  - longer sentences
  - more colloquial/less edited style
  - different distribution of labels

- **Domain adaptation** techniques attempt to correct for this assumption when something about the source/characteristics of the test data is known to be different.

Why do we need text corpora?

Two main reasons:

1. To evaluate our systems on
   - Good science requires controlled experimentation.
   - Good engineering requires benchmarks.

2. To help our systems work well (data-driven methods/machine learning)
   - When a system's behavior is determined solely by manual rules or databases, it is said to be **rule-based**, **symbolic**, or **knowledge-driven** (early days of computational linguistics)
   - **Learning**: collecting statistics or patterns automatically from corpora to govern the system's behavior (dominant in most areas of contemporary NLP)
     - **supervised learning**: the data provides example input/output pairs (main focus in this course)
     - core behavior: **training**; refining behavior: **tuning**

Corpus Wrangling

- We use corpora to extract features, rules, and statistics for our models
- Even the cleanest corpora are noisy and most corpora aren't the cleanest
- The following techniques are useful for getting 'clean' information from your data

Basic Text Processing

Regular Expressions

demos using regexpal.com

these slides from Jurafsky

Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks
- Motivation: don't just find "happy" in a movie review, find all forms
  - happiness
  - happiest
  - but not unhappy

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

- Woodchuck 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 +

Regular Expressions: Anchors ^ $

Pattern	Matches
^[A-Z]	Palo Alto
^[^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
  - 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

Basic Text Processing

Word tokenization

Text Normalization

- Every NLP task needs to do text normalization:
  1. Segmenting/tokenizing words in running text
  2. Normalizing word formats
  3. Segmenting sentences in running text

How many words?

- I do uh main- mainly business data processing
  - Fragments, filled pauses
- Seuss's cat in the hat is different from other cats!
  - **Lemma**: same stem, part of speech, rough word sense
    - cat and cats = same lemma
  - **Wordform**: the full inflected surface form
    - cat and cats = different wordforms

How many words?

they lay back on the San Francisco grass and looked at the stars and their

- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.
- How many?
  - 15 tokens (or 14)
  - 13 types (or 12) (or 11?)

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

We already did some basic tokenization

```
gsed 's/ /\n/g' sawyr11.txt | sort | uniq | less
```

We can see some problems...

```
"'Bout
"A
"AIN'T
...
America
America,
Douglas
Douglas'
Douglas'.
Douglas'."
Douglas,
Douglass,
```

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. → ??

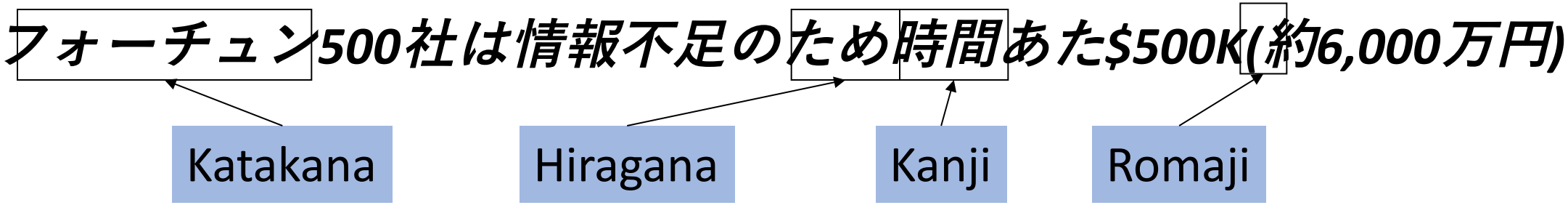
Tokenization: language issues

- French
  - ***L'ensemble*** $\rightarrow$ 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**

Tokenization: language issues

- Chinese and Japanese no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats



End-user can express query entirely in hiragana!

Word Tokenization in Chinese

- Also called **Word Segmentation**
- Chinese words are composed of characters
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:
  - Maximum Matching (also called Greedy)

Maximum Matching Word Segmentation Algorithm

- Given a wordlist of Chinese, and a string.

1) Start a pointer at the beginning of the string
2) Find the longest word in dictionary that matches the string starting at pointer
3) Move the pointer over the word in string
4) Go to 2

Max-match segmentation illustration

- Thecatinthehat — the cat in the hat
- Thetabledownthere — the table down there

  theta bled own there

- Doesn’t generally work in English!

- But works astonishingly well in Chinese
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
- Modern probabilistic segmentation algorithms even better

What About in English?

- Usually a set of rules that are appropriate for the application
- Can be implemented with regex!
- Cheat (that you should use): nltk!
-
```
>>> import nltk

>>> nltk.word_tokenize("This isn't much of--or indeed at all--
a test.")
['This', 'is', "n't", 'much', 'of', '--
', 'or', 'indeed', 'at', 'all', '--', 'a', 'test', '.']
```