NLP models
- How do people learn/acquire language?

Administrivia
- Assignment 0 due today
  - article discussion
- Assignment 1 out soon
  - due Wednesday, 2/2 in class
  - no code submitted, but will require coding
- Send me an e-mail if you’d like me to e-mail announcements to another account besides your school account
- Send videos…

NLP models
- A lot of debate about how human’s learn language
  - Rationalist (e.g. Chomsky)
  - Empiricist
- From my perspective (and many people who study NLP)…
  - I don’t care :)
- Strong AI vs. weak AI: don’t need to accomplish the task the same way people do, just the same task
  - Machine learning
  - Statistical NLP
### Vocabulary
- **Word**
  - a unit of language that native speakers can identify
  - words are the blocks from which sentences are made
- **Sentence**
  - a string of words satisfying the grammatical rules of a language
- **Document**
  - A collection of sentences
- **Corpus**
  - A collection of related texts

### Corpora characteristics
- monolingual vs. parallel
- language
- annotated (e.g., parts of speech, classifications, etc.)
- source (where it came from)
- size

### Corpora examples
- Linguistic Data Consortium
  - [http://www.ldc.upenn.edu/Catalog/byType.jsp](http://www.ldc.upenn.edu/Catalog/byType.jsp)
- Dictionaries
  - WordNet – 206K English words
  - CELEX2 – 365K German words
- Monolingual text
  - Gigaword corpus
    - 4M documents (mostly news articles)
    - 1.7 trillion words
    - 11GB of data (4GB compressed)
- Parallel data
  - ~10M sentences of Chinese-English and Arabic-English
  - Europarl
    - ~1.5M sentences English with 10 different languages
Corpora examples

- Annotated
  - Brown Corpus
    - 1M words with part of speech tag
  - Penn Treebank
    - 1M words with full parse trees annotated
  - Other treebanks
    - Treebank refers to a corpus annotated with trees (usually syntactic)
    - Chinese: 51K sentences
    - Arabic: 143K words
    - many other languages...
    - BLUP: 300M words (automatically annotated)

Many others...

- Spam and other text classification
- Google n-grams
  - 2006 (2.4GB compressed)
  - 13M unigrams
  - 300M bigrams
  - ~1B 3, 4 and 5-grams
- Speech
- Video (with transcripts)

Corpus analysis

- Corpora are important resources
- Often give examples of an NLP task we’d like to accomplish
- Much of NLP is data-driven!
- A common and important first step to tackling many problems is analyzing the data you’ll be processing

What types of questions might we want to ask?

- How many...
  - documents, sentences, words
- On average, how long are the:
  - documents, sentences, words
- What are the most frequent words? pairs of words?
- How many different words are used?
- Data set specifics, e.g. proportion of different classes?
- ...
Corpora issues

- Somebody gives you a file and says there's text in it
- Issues with obtaining the text?
  - text encoding
  - language recognition
  - formatting (e.g. web, xml, ...)
  - misc. information to be removed
    - header information
    - tables, figures
    - footnotes

A rose by any other name...

- Word
  - a unit of language that native speakers can identify
  - words are the blocks from which sentences are made
- Concretely:
  - We have a stream of characters
  - We need to break into words
  - What is a word?
  - Issues/problem cases?
  - Word segmentation/tokenization?

Tokenization issues: ‘

- *Finland’s capital…*

? ?

Tokenization issues: ‘

- *Finland’s capital…*
  - Finland
  - Finland’s
  - Finland’s
  - Finland’s

What are the benefits/drawbacks?
Tokenization issues: ‘

Aren’t we ...

?’

Tokenization issues: ‘

Aren’t we ...

Aren’t

Are n’t

Are not

Tokenization issues: hyphens

Hewlett-Packard state-of-the-art

co-education lower-case

take-it-or-leave-it 26-year-old

?’

Tokenization issues: hyphens

Hewlett-Packard state-of-the-art

co-education lower-case

- Keep as is
- Merge together
  - HewlettPackard
  - stateoftheart
- Split on hyphen
  - lower case
  - co education

What are the benefits/drawbacks?
More tokenization issues

- Compound nouns: San Francisco, Los Angelos, ...
  - One token or two?
- Numbers
  - Examples
    - Dates: 3/12/91
    - Model numbers: B-52
    - Domain specific numbers: PGP key - 324e3df234cb23e
    - Phone numbers: (800) 234-2333
    - Scientific notation: 1.456 e-10

Tokenization: language issues

- Lebensversicherungsgesellschaftsangestellter
  - 'life insurance company employee'
  - Opposite problem we saw with English (San Francisco)
- German compound nouns are not segmented
- German retrieval systems frequently use a compound splitter module

Tokenization: language issues

- Many character based languages (e.g. Chinese) have no spaces between words
  - A word can be made up of one or more characters
  - There is ambiguity about the tokenization, i.e. more than one way to break the characters into words
  - Word segmentation problem
  - can also come up in speech recognition

Word counts

- Tom Sawyer
  - How many words?
    - 71,370 total
    - 8,018 unique
  - Is this a lot or a little? How might we find this out?
    - Random sample of news articles: 11K unique words
  - What does this say about Tom Sawyer?
    - Simpler vocabulary (colloquial, audience target, etc.)
### Word counts

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>3332</td>
</tr>
<tr>
<td>and</td>
<td>2972</td>
</tr>
<tr>
<td>a</td>
<td>1775</td>
</tr>
<tr>
<td>to</td>
<td>1725</td>
</tr>
<tr>
<td>of</td>
<td>1440</td>
</tr>
<tr>
<td>was</td>
<td>1161</td>
</tr>
<tr>
<td>it</td>
<td>1027</td>
</tr>
<tr>
<td>in</td>
<td>906</td>
</tr>
<tr>
<td>that</td>
<td>877</td>
</tr>
<tr>
<td>he</td>
<td>877</td>
</tr>
<tr>
<td>I</td>
<td>783</td>
</tr>
<tr>
<td>Ms</td>
<td>772</td>
</tr>
<tr>
<td>you</td>
<td>686</td>
</tr>
<tr>
<td>Tom</td>
<td>679</td>
</tr>
<tr>
<td>with</td>
<td>642</td>
</tr>
</tbody>
</table>

#### What are the most frequent words?

#### What types of words are most frequent?

### Zipf's “Law”

- Frequency of occurrence of words is inversely proportional to the rank in this frequency of occurrence.
- When both are plotted on a log scale, the graph is a straight line.

George Kingsley Zipf (1902-1950)

### Zipf's law

- **At a high level:**
  - a few words occur very frequently
  - a medium number of elements have medium frequency
  - many elements occur very infrequently

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency of frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3993</td>
</tr>
<tr>
<td>2</td>
<td>1292</td>
</tr>
<tr>
<td>3</td>
<td>664</td>
</tr>
<tr>
<td>4</td>
<td>410</td>
</tr>
<tr>
<td>5</td>
<td>243</td>
</tr>
<tr>
<td>6</td>
<td>199</td>
</tr>
<tr>
<td>7</td>
<td>172</td>
</tr>
<tr>
<td>8</td>
<td>131</td>
</tr>
<tr>
<td>9</td>
<td>82</td>
</tr>
<tr>
<td>10</td>
<td>91</td>
</tr>
<tr>
<td>11-50</td>
<td>540</td>
</tr>
<tr>
<td>51-100</td>
<td>99</td>
</tr>
<tr>
<td>&gt; 100</td>
<td>102</td>
</tr>
</tbody>
</table>

8K words in vocab 71K total occurrences

how many occur once? twice?
Zipf’s law

\[ f = C \frac{1}{r} \]

\[ C \approx N/10 \]

The product of the frequency of words \((f)\) and their rank \((r)\) is approximately constant.

Zipf Distribution

Illustration by Jacob Nielsen

Zipf’s law: Brown corpus

Zipf’s law: Tom Sawyer

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
<th>Rank</th>
<th>( f \cdot r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>332</td>
<td>1</td>
<td>3332</td>
</tr>
<tr>
<td>and</td>
<td>2972</td>
<td>2</td>
<td>5944</td>
</tr>
<tr>
<td>a</td>
<td>1775</td>
<td>3</td>
<td>5325</td>
</tr>
<tr>
<td>he</td>
<td>877</td>
<td>10</td>
<td>8770</td>
</tr>
<tr>
<td>but</td>
<td>410</td>
<td>20</td>
<td>8400</td>
</tr>
<tr>
<td>he</td>
<td>294</td>
<td>30</td>
<td>8820</td>
</tr>
<tr>
<td>Oh</td>
<td>116</td>
<td>90</td>
<td>10440</td>
</tr>
<tr>
<td>two</td>
<td>104</td>
<td>100</td>
<td>10400</td>
</tr>
<tr>
<td>name</td>
<td>21</td>
<td>400</td>
<td>8400</td>
</tr>
<tr>
<td>group</td>
<td>13</td>
<td>600</td>
<td>7800</td>
</tr>
<tr>
<td>friends</td>
<td>10</td>
<td>800</td>
<td>8000</td>
</tr>
<tr>
<td>family</td>
<td>8</td>
<td>1000</td>
<td>8000</td>
</tr>
<tr>
<td>sins</td>
<td>2</td>
<td>3000</td>
<td>6000</td>
</tr>
<tr>
<td>Applause</td>
<td>1</td>
<td>8000</td>
<td>8000</td>
</tr>
</tbody>
</table>
Sentences

- **Sentence**: a string of words satisfying the grammatical rules of a language.
- **Sentence segmentation**
  - How do we identify a sentence?
  - Issues/problem cases?
  - Approach?

Sentence segmentation: issues

- A first answer:
  - something ending in a: . ? !
  - gets 90% accuracy

  Dr. Kauchak gives us just the right amount of homework.

  Abbreviations can cause problems

The scene is written with a combination of unbridled passion and sure-handed control. In the exchanges of the three characters and the rise and fall of emotions, Mr. Weller has captured the heartbreaking inexorability of separation.

sometimes: ; and — might also denote a sentence split

"You remind me," she remarked, "of your mother."

Quotes often appear outside the ending marks
Sentence segmentation

- Place initial boundaries after: . ? !
- Move the boundaries after the quotation marks, if they follow a break
- Remove a boundary following a period if:
  - It is a known abbreviation that doesn’t tend to occur at the end of a sentence (Prof., vs.)
  - It is preceded by a known abbreviation and not followed by an uppercase word

Sentence length

What is the average sentence length, say for news text? 23

<table>
<thead>
<tr>
<th>Length</th>
<th>percent</th>
<th>cumul. percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6-10</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>11-15</td>
<td>14</td>
<td>25</td>
</tr>
<tr>
<td>16-20</td>
<td>17</td>
<td>42</td>
</tr>
<tr>
<td>21-25</td>
<td>17</td>
<td>59</td>
</tr>
<tr>
<td>26-30</td>
<td>15</td>
<td>74</td>
</tr>
<tr>
<td>31-35</td>
<td>11</td>
<td>86</td>
</tr>
<tr>
<td>36-40</td>
<td>7</td>
<td>92</td>
</tr>
<tr>
<td>41-45</td>
<td>4</td>
<td>96</td>
</tr>
<tr>
<td>46-50</td>
<td>2</td>
<td>98</td>
</tr>
<tr>
<td>51-100</td>
<td>1</td>
<td>99.99</td>
</tr>
<tr>
<td>101+</td>
<td>0.01</td>
<td>100</td>
</tr>
</tbody>
</table>

Regular expressions

- Regular expressions are a very powerful tool to do string matching and processing
- Allows you to do things like:
  - Tell me if a string starts with a lowercase letter, then is followed by 2 numbers and ends with “ing” or “ion”
  - Replace all occurrences of one or more spaces with a single space
  - Split up a string based on whitespace or periods or commas or …
  - Give me all parts of the string where a digit is proceeded by a letter and then the “#” sign

Regular expressions: literals

- We can put any string in a regular expression
  - /test/ matches any string that has “test” in it
  - /this class/ matches any string that has “this class” in it
  - /Test/ case sensitive: matches any string that has “Test” in it
Regular expressions: character classes

- A set of characters to match:
  - put in brackets: \[
  - [abc] matches a single character a or b or c
- For example:
  - /\[Th\]est/ matches any string with "Test" or "test" in it
- Can use – to represent ranges
  - \[a-z\] is equivalent to [abdefghijklmnopqrstuvwxyz]
  - \[A-D\] is equivalent to [ABCD]
  - \[0-9\] is equivalent to [0123456789]

For example:

- /\([0-9][0-9][0-9][0-9]\)/ matches any four digits, e.g. a year
- Can also specify a set NOT to match
  - ^ means all character EXCEPT those specified
  - \[^a\] all characters except 'a'
  - \[^0-9\] all characters except numbers
  - \[^A-Z\] not an upper case letter

- Meta-characters (not always available)
  - \w - word character (a-zA-Z_0-9)
  - \W - non-word-character (i.e. everything else)
  - \d - digit (0-9)
  - \s - whitespace character (space, tab, endline, …)
  - \S - non-whitespace
  - \b matches a word boundary (whitespace, beginning or end of line)
  - . - matches any character

For example:

- /19\d\d/ would match a year starting with 19
- /\s/ matches anything with a space
- /\s/ or /[^\s]/ matches anything with at least one non-space character
Regular expressions: repetition

- `*` matches zero or more of the preceding
  - `/ba*d/`
    - matches any string with
      - bad
      - badoad
      - boads

- `?` zero or 1 occurrence of the preceding
  - `/fights?/`
    - matches any string with “fight” or “fights” in it

- `{n,m}` matches n to m inclusive
  - `/ba{3,4}d/`
    - matches any string with
      - baaad
      - baaaad

Regular expressions: repetition revisited

- What if we wanted to match:
  - This very interesting
  - This is very very interesting
  - This is very very very interesting

- Would `/This is very+ interesting/` work?
  - No… + only corresponds to the ‘y’
  - `/This is (very )+interesting/`
Regular expressions: disjunction
- `|` has the lowest precedence and can be used
  - `/cats|dogs/`
    - matches: `cats`
    - `dogs`
    - does NOT match: `catsogs`
  - `/I like (cats|dogs)$/`
    - matches: `I like cats`
    - `I like dogs`

Some examples
- All strings that start with a capital letter
  - `/^[A-Z]/`
- IP addresses
  - `/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/`
- Matching a decimal number
  - `/[-+]?[0-9]*\.?[0-9]+/`
- All strings that end in ing
  - `/ing$/`
- All strings that end in ing or ed
  - `/ing|ed$/`

Some examples
- All strings that start with a capital letter
  - `/^[A-Z]/`
- IP addresses
  - `/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/`
- Matching a decimal number
  - `/[-+]?[0-9]*\.?[0-9]+/`
- All strings that end in ing
  - `/ing$/`
- All strings that end in ing or ed
  - `/ing|ed$/`

Regular expressions: memory
- All strings that begin and end with the same character
  - `()`
    - used for precedence
    - also records a matched grouping, which can be referenced later
  - `/\1$/`
    - all strings that begin and end with the same character
Regular expression: memory

- /She likes \(\w+\) and he likes \1/
- We can use multiple matches
  - /She likes \(\w+\) and \(\w+\) and he also likes \1 and \2/

Regular expressions: substitution

- Most languages also allow for substitution
  - s/banana/apple/
    - substitute first occurrence banana for apple
  - s/banana/apple/g
    - substitute all occurrences (global)
  - s/\w+/\1 /\1 /
  - s/\s+/ /g

Regular expressions by language

- Java:
  - String s = “this is a test”
  - s.matches(“test”)  
  - s.matches(“.*test.*”)  
  - s.matches(“this \sis .* test”)
  - s.split(“\s+”)
  - s.replaceAll(“\s+”, “ ”);

- perl:
  - s = “this is a test”
  - s =~ /test/  
  - s =~ /test$/  
  - s =~ /this\sis .* test/  
  - split/\s+/s  
  - s =~ s/\s+//g
Regular expressions by language

- **Python:**
  - `import re`
  - `s = "this is a test"`
  - `p = re.compile("test")`
  - `p.match(s)`
  - `p = re.compile(".*test.*")`
  - `re.split(\s+\", s)`
  - `re.sub(\s+\", "\", s)`

Regular expression by language

- **grep**
  - Command-line tool for regular expressions (general regular expression print/parser)
  - Returns all lines that match a regular expression
  - `grep "@" twitter.posts`
  - `grep "http:" twitter.posts`
  - Can’t use metacharacters (\d, \w), use [] instead

Regular expression resources

- **General regular expressions:**
  - Ch 2.1 of the book
  - [http://www.regular-expressions.info/](http://www.regular-expressions.info/)
  - Good general tutorials
  - Many language specific examples as well

- **Java**
  - See also the documentation for java.util.regex

- **Python**
  - [http://docs.python.org/howto/regex.html](http://docs.python.org/howto/regex.html)
  - [http://docs.python.org/library/regex.html](http://docs.python.org/library/regex.html)

Regular expression by language

- **sed**
  - Another command-line tool using that regexs to print and manipulate strings
  - Very powerful, though we’ll just play with it
  - Most common is substitution:
    - `sed "s/ is a / is not a/" twitter.posts`
    - `sed "s/ +/ /" twitter.posts`
  - Can also do things like delete all that match, etc.
Regular expression resources

- Perl
  - [http://perldoc.perl.org/perlre.html](http://perldoc.perl.org/perlre.html)
  - [http://perldoc.perl.org/perlre.html](http://perldoc.perl.org/perlre.html)

- grep
  - See the write-up at the end of Assignment 1

- sed
  - See the write-up at the end of Assignment 1