Announcements

conda install -c conda-forge spacy=3.0
Words as dimensionality reduction

https://brenocon.com/blog/2012/07/the-60000-cat-deep-belief-networks-make-less-sense-for-language-than-vision/
Words

- One morning I shot an elephant in my pajamas
- I didn’t shoot an elephant
- Imma let you finish but Beyonce had one of the best videos of all time
- I do uh main- mainly business data processing
- 一天早上我穿着睡衣射了一只大象
@dbamman have you seen this :) http://popvssoda.com
<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>:-)</td>
<td>Smiley or happy face.</td>
</tr>
<tr>
<td>:))</td>
<td>Laughing, big grin, laugh with glasses, or wide-eyed surprise</td>
</tr>
<tr>
<td>:-((</td>
<td>Very happy or double chin</td>
</tr>
<tr>
<td>:-('</td>
<td>Frown, sad, angry, pouting</td>
</tr>
<tr>
<td>:'-)</td>
<td>Crying</td>
</tr>
<tr>
<td>:'-)</td>
<td>Tears of happiness</td>
</tr>
<tr>
<td>D:\</td>
<td>Horror, disgust, sadness, great dismay</td>
</tr>
<tr>
<td>:-*</td>
<td>Kiss</td>
</tr>
<tr>
<td>:-(</td>
<td>Wink, smirk</td>
</tr>
<tr>
<td>:P</td>
<td>Tongue sticking out, cheeky/playful, blowing a raspberry</td>
</tr>
</tbody>
</table>
Types and tokens

- Type = abstract descriptive concept
- Token = instantiation of a type

```
To be or not to be
```

6 tokens (to, be, or, not, to, be)
4 types (to, be, or, not)

- Types = the **vocabulary**; the unique tokens.
Types and tokens

• Type = abstract descriptive concept
• Token = instantiation of a type

How can we use types and tokens to measure vocabulary richness?
As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.
Whitespace

text.split(" ")

• As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.
<table>
<thead>
<tr>
<th></th>
<th>earth</th>
<th></th>
<th>earth</th>
</tr>
</thead>
<tbody>
<tr>
<td>368</td>
<td>earth</td>
<td>2</td>
<td>earth--to</td>
</tr>
<tr>
<td>135</td>
<td>earth,</td>
<td>2</td>
<td>earth--if</td>
</tr>
<tr>
<td>68</td>
<td>earth.</td>
<td>2</td>
<td>earth--and</td>
</tr>
<tr>
<td>26</td>
<td>earth</td>
<td>2</td>
<td>earth:</td>
</tr>
<tr>
<td>24</td>
<td>earth.</td>
<td>2</td>
<td>earth,'</td>
</tr>
<tr>
<td>18</td>
<td>earth.*</td>
<td>1</td>
<td>earth-worms,</td>
</tr>
<tr>
<td>16</td>
<td>earth;</td>
<td>1</td>
<td>earth-worm.</td>
</tr>
<tr>
<td>14</td>
<td>earth,</td>
<td>1</td>
<td>earth--which</td>
</tr>
<tr>
<td>9</td>
<td>earth's</td>
<td>1</td>
<td>earth--when</td>
</tr>
<tr>
<td>5</td>
<td>earth!*</td>
<td>1</td>
<td>earth--something</td>
</tr>
<tr>
<td>5</td>
<td>earth!</td>
<td>1</td>
<td>earth-smeared,</td>
</tr>
<tr>
<td>4</td>
<td>earth;</td>
<td>1</td>
<td>earth-scoops,</td>
</tr>
<tr>
<td>4</td>
<td>earth,*</td>
<td>1</td>
<td>earth's</td>
</tr>
<tr>
<td>3</td>
<td>earth.&quot;</td>
<td>1</td>
<td>earth--oh,</td>
</tr>
<tr>
<td>3</td>
<td>earth?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>earth!*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

"earth" in sample of 100 books (Project Gutenberg)
Punctuation

- We typically don’t want to just strip all punctuation, however.
  - Punctuation signals boundaries (sentence, clausal boundaries, parentheticals, asides)
  - Some punctuation has illocutionary force, like exclamation points (!) and question marks (?)
  - Emoticons are strong signals of e.g. sentiment
Regular expressions

• Most tokenization algorithms (for languages typically delimited by whitespace) use regular expressions to segment a string into discrete tokens.
Regular expressions

• A language for specifying search strings in text.

\[\text{waters}\]

As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.

Kleene 1951
Regular expressions

• A language for specifying search strings in text.

/ing?/

As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.
Regular expressions

• A language for specifying search strings in text.

/\(\text{waters}\?)\|\(\text{earth}\)\|\(\text{[Hh]ill}\)/

As much mud in the streets as if the \text{waters} had but newly retired from the face of the \text{earth}, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn \text{Hill}. 
## Regular expressions

<table>
<thead>
<tr>
<th>regex</th>
<th>matches</th>
<th>doesn’t match</th>
</tr>
</thead>
<tbody>
<tr>
<td>/the/</td>
<td>the, isothermally</td>
<td>The</td>
</tr>
<tr>
<td>/[Tt]he/</td>
<td>the, isothermally, The</td>
<td></td>
</tr>
<tr>
<td>/b[Tt]he\b/</td>
<td>the, The</td>
<td>—The</td>
</tr>
</tbody>
</table>
Regular expressions

• Bracket specifies alternations (match one of the elements inside brackets)

\[ \text{[Tt]he} = \text{The or the} \]

• Brackets can specify ranges

\[ \text{[a-z]} = \{a, b, c, \ldots, z\} \]
\[ \text{[0-9]} = \{0, 1, \ldots, 9\} \]
\[ \text{[A-Za-z]} = \{A, B, C, \ldots, Z, a, b, c, \ldots, z\} \]
# Regular expressions

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
<th>Sample regex</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>one or more</td>
<td>he+y</td>
<td>hey, heeeeeeey</td>
</tr>
<tr>
<td>?</td>
<td>optional</td>
<td>colou?r</td>
<td>color, colour</td>
</tr>
<tr>
<td>*</td>
<td>zero or more</td>
<td>toys*</td>
<td>toy, toys, toyssss</td>
</tr>
</tbody>
</table>
## Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>\b</td>
<td>Word boundary (zero width)</td>
</tr>
<tr>
<td>\d</td>
<td>Any decimal digit (equivalent to [0-9])</td>
</tr>
<tr>
<td>\D</td>
<td>Any non-digit character (equivalent to [^0-9])</td>
</tr>
<tr>
<td>\s</td>
<td>Any whitespace character (equivalent to [ \t\n\r\f\v])</td>
</tr>
<tr>
<td>\S</td>
<td>Any non-whitespace character (equivalent to [^ \t\n\r\f\v])</td>
</tr>
<tr>
<td>\w</td>
<td>Any alphanumeric character (equivalent to [a-zA-Z0-9_])</td>
</tr>
<tr>
<td>\W</td>
<td>Any non-alphanumeric character (equivalent to [^a-zA-Z0-9_])</td>
</tr>
<tr>
<td>\t</td>
<td>The tab character</td>
</tr>
<tr>
<td>\n</td>
<td>The newline character</td>
</tr>
</tbody>
</table>

[https://www.nltk.org/book/ch03.html](https://www.nltk.org/book/ch03.html)
Disjunction

• We can specify complex regular expressions by joining separate regexes with a disjunction operator |

/\(\text{waters}\)?\|\(\text{earth}\)\|\([\text{Hh}]\text{ill}\)/

As much mud in the streets as if the \texttt{waters} had but newly retired from the face of the \texttt{earth}, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn \texttt{Hill}.
Python

- `re.findall(regex, text)` finds all non-overlapping matches for a target regex.

- `re.findall(r"[Tt]he", "The dog barked at the cat")`

- `[“The”, “the”]`
import nltk
tokens=nltk.word_tokenize(text)

Tokenizes following the conventions of the Penn Treebank:

- punctuation split from adjoining words
- double quotes (""") changes to forward/backward quotes based on their location in word (``the``)
- verb contractions + 's split into separate tokens: (did_n’t, children_’)
import nltk
tokens=nltk.word_tokenize(text)

Penn Treebank tokenization is important because a lot of downstream NLP is trained on annotated data that uses Treebank tokenization!

I didn't see the parade.

I didn’t see the parade.
import spacy
nlp = spacy.load('en_core_web_sm')
tokens=[token.text for token in nlp(text)]

https://spacy.io/usage/spacy-101#annotations-token
Sentence segmentation

• Word tokenization presumes a preprocessing step of sentence segmentation — identifying the boundaries between sentences.

• Lots of NLP operates at the level of the sentence (POS tagging, parsing), so really important to get it right.

• Harder to write regexes to delimit these, since there are many cases where the usual delimiters (periods, question marks) serve double duty.
“Do you want to go?” said Jane.

Mr. Collins said he was going.

He lives in the U.S. John, however, lives in Canada.
Sentence segmentation

- NLTK: Punkt sentence tokenizer — unsupervised method to learn common abbreviations, collocations, sentence-initial words. Can be trained on data from new domain.  
  [Kiss, Tibor and Strunk, Jan (2006): Unsupervised Multilingual Sentence Boundary Detection (Computational Linguistics)]

- spaCy: Relies on dependency parsing to find sentence boundaries.

```python
import spacy
nlp = spacy.load('en_core_web_sm')
doc=nlp(text)
for sent in doc.sents:
    for token in sent:
        print(token.text)
```
Stemming and lemmatization

• Many languages have some inflectional and derivational morphology, where similar words have similar forms:
  
  organizes, organized, organizing

• Stemming and lemmatization reduce this variety to a single common base form.
Stemming

• Heuristics process for chopping off the inflected suffixes of a word
  
  organizes, organized, organizing → organ

• Lower precision, higher recall
Porter stemmer

• Sequence of rules for removing suffixes from words

  • EMENT → ∅
  • SSES → SS
  • IES → l
  • SS → ∅
  • S → ∅
Lemmatization

• Using morphological analysis to return the dictionary form of a word (the entry in a dictionary you’d find all forms under)

organizes, organized, organizing → organize

```python
import spacy
nlp = spacy.load('en_core_web_sm')
lemmas=[token.lemma_ for token in nlp(text)]
```
Difficulties

- When does **punctuation** disrupt the desired boundaries of a token?

<table>
<thead>
<tr>
<th>Emoticons</th>
<th>:) :D \o/ o_O</th>
</tr>
</thead>
<tbody>
<tr>
<td>URLs</td>
<td><a href="http://www.google.com">http://www.google.com</a></td>
</tr>
<tr>
<td>Prices</td>
<td>$19.99</td>
</tr>
<tr>
<td>Decimals</td>
<td>19.99</td>
</tr>
<tr>
<td>Hyphens</td>
<td>state-of-the-art</td>
</tr>
<tr>
<td>Usernames</td>
<td>@dbamman</td>
</tr>
<tr>
<td>Hashtags</td>
<td>#blacklivesmatter</td>
</tr>
</tbody>
</table>
# Keep usernames together (any token starting with @, followed by A-Z, a-z, 0-9)
regexes=(r"(?:@[\w_]+)",

# Keep hashtags together (any token starting with #, followed by A-Z, a-z, 0-9, _, or -)
 r"(?:\#+[\w_]+[\w\'-_]*[\w_]+)",

# Keep words with apostrophes, hyphens and underscores together
 r"(?:[a-z][a-z']\'_-]+[a-z])",

# Keep all other sequences of A-Z, a-z, 0-9, _ together
 r"(?:[\w_]+)",

# Everything else that's not whitespace
 r"(?:\S)"
)

big_regex="|".join(regexes)

my_extensible_tokenizer = re.compile(big_regex, re.VERBOSE | re.I | re.UNICODE)

def my_extensible_tokenize(text):
    return my_extensible_tokenizer.findall(text)
EvaluateTokenization
ForSentiment.ipynb

• Don’t just assume an out-of-the box tokenizer works exactly for your application.

• Sentiment analysis accuracy (even on IMDB data) can vary by ~5 points as a function of tokenization choices.
• Design a tokenizer for printed works that have been OCR’d

the inhabitants of the surrounding districts will, also, be thus
prevented. Moritz Wagner has lately published an interesting
garison essay on this subject, and has shown that the service
rendered by isolation in preventing crosses between newly-
formed varieties is probably greater even than I supposed.
Homework 1

- Complete `Text_Complexity_TODO.ipynb` before class on Thursday (deadline 2pm 9/2), submit through bCourses (ipynb and pdf).

- Explores measures of textual complexity using simple counts of words, sentences and syllables.