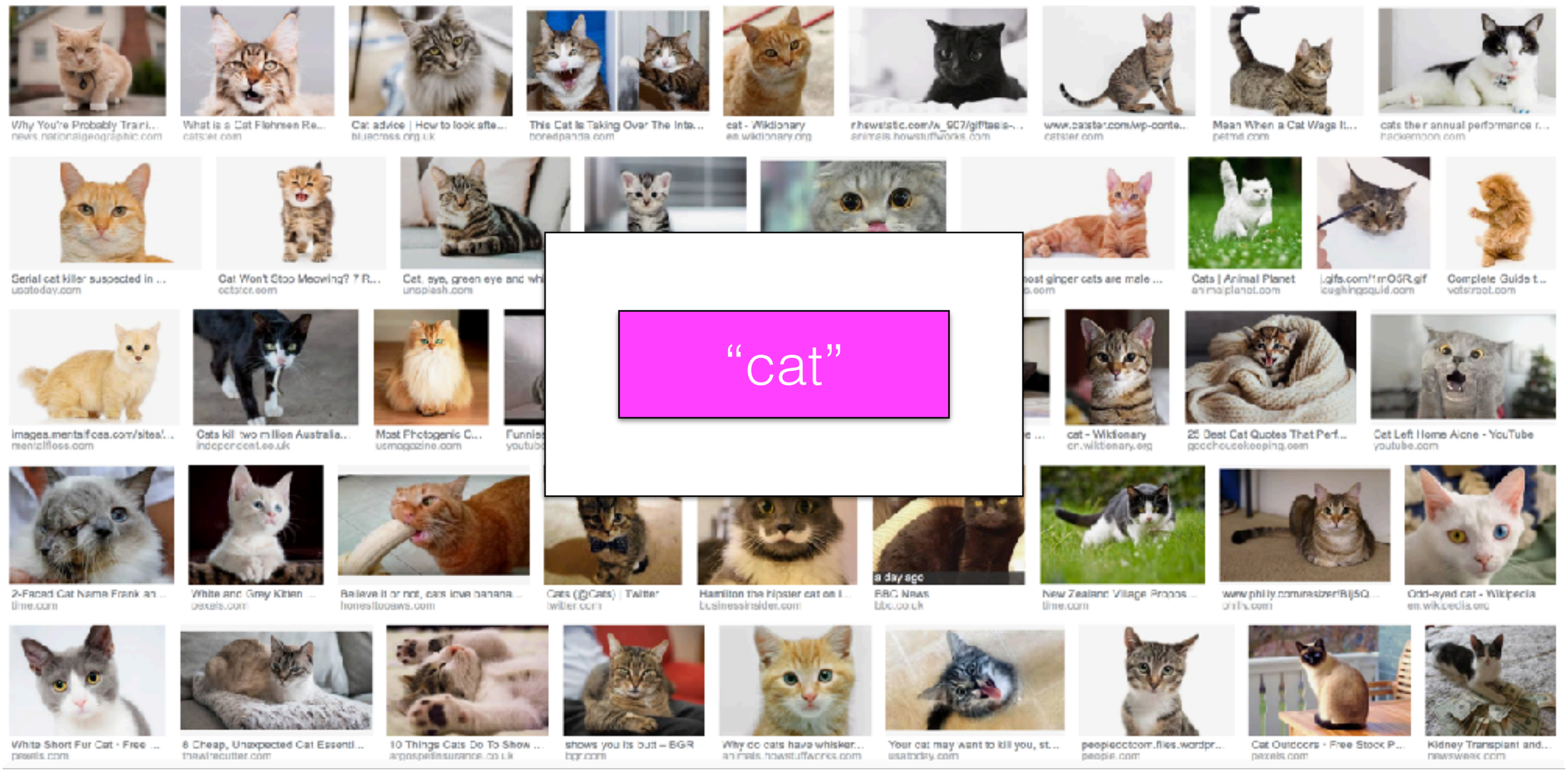
Applied Natural Language Processing

Info 256

Lecture 2: Words (Jan 24, 2019)

David Bamman, UC Berkeley

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

Words

@dbamman have you seen this :) http://popvssoda.com

Tokenization before Twitter:

```
@
dbamman
have
you
seen
this
:
)
http
:
//popvssoda.com
```

Emoticons

Icon											Meaning
:-) :)	:-] :]	:-3 :3	:-> :>	8-) 8)	:-} :}	:o)	:c)	:^)	=]	=)	Smiley or happy face.[4][5][6]
:-D :D	8-D 8D	x-D xD	X-D XD	=D	=3	B^D					Laughing,[4] big grin,[5][6] laugh with glasses,[7] or wide-eyed surprise[8]
:-))											Very happy or double chin[7]
:-(:(	:-c :c	:-< :<	:-[:[	:-||	>:[	:{	:@	>:(			Frown,[4][5][6] sad,[9] angry,[7] pouting
:'-(:'(											Crying[9]
:'-) :')											Tears of happiness[9]
D-':	D:<	D:	D8	D;	D=	DX					Horror, disgust, sadness, great dismay[5][6] (right to left)
:-O :O	:-o :o	:-0	8-0	>:O							Surprise,[3] shock,[4][10] yawn[11]
:-* :*	:×										Kiss
;-) ;)	*-) *)	;-] ;]	;^)	:-,	;D						Wink,[4][5][6] smirk[10][11]
:-P :P	X-P XP	x-p xp	:-p :p	:-Þ :Þ	:-þ :þ	:-b :b	d:	=p	>:P		Tongue sticking out, cheeky/playful,[4] blowing a raspberry

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?

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.

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.

what do we lose with whitespace tokenization?

368	earth
135	earth,
68	earth.
26	earth
24	earth.
18	earth."
16	earth;
14	earth,
9	earth's
5	earth!"
5	earth!
4	earth;
4	earth,"
3	earth.”
3	earth?
3	earth!"

2	earth--to
2	earth--if
2	earth--and
2	earth:
2	earth,'
1	earth-worms,
1	earth-worm.
1	earth--which
1	earth--when
1	earth--something
1	earth-smeared,
1	earth-scoops,
1	earth's
1	earth--oh,

“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.

/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.

```
/(waters?)|(earth)|([Hh]ill)/
```

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

regex	matches	doesn't match
`/the/`	the, isothermally	The
`/[Tt]he/`	the, isothermally, The	
`/\b[Tt]he\b/`	the, The	—The

Regular expressions

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

[Tt]he = The or the

- Brackets can specify ranges

[a-z] = {a, b, c, …, z}
[0-9] = {0, 1, …, 9}
[A-Za-z] = {A, B, C, …, Z, a, b, c, …, z}

Regular expressions

Term	Meaning	Sample regex	Matches
+	one or more	he+y	hey, heeeeeey
?	optional	colou?r	color, colour
*	zero or more	toys*	toy, toys, toysss

Symbols

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

https://www.nltk.org/book/ch03.html

Disjunction

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

`/(waters?)|(earth)|([Hh]ill)/`

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.

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)
```

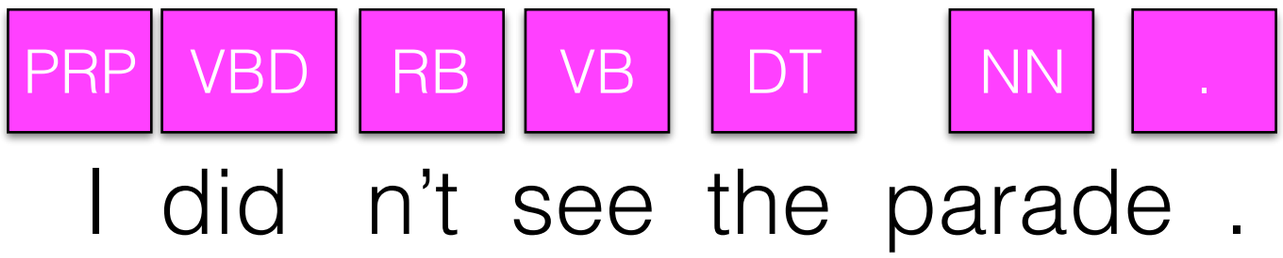
Tokenizes following the conventions of the Penn Treebank:

- punctuation split from adjoining words
- double quotes (“) changes to forward/ backward quotes based on on their location in word (``the”)
- verb contractions + ’s split into separate tokens: (`did_n’t, children_’s`)

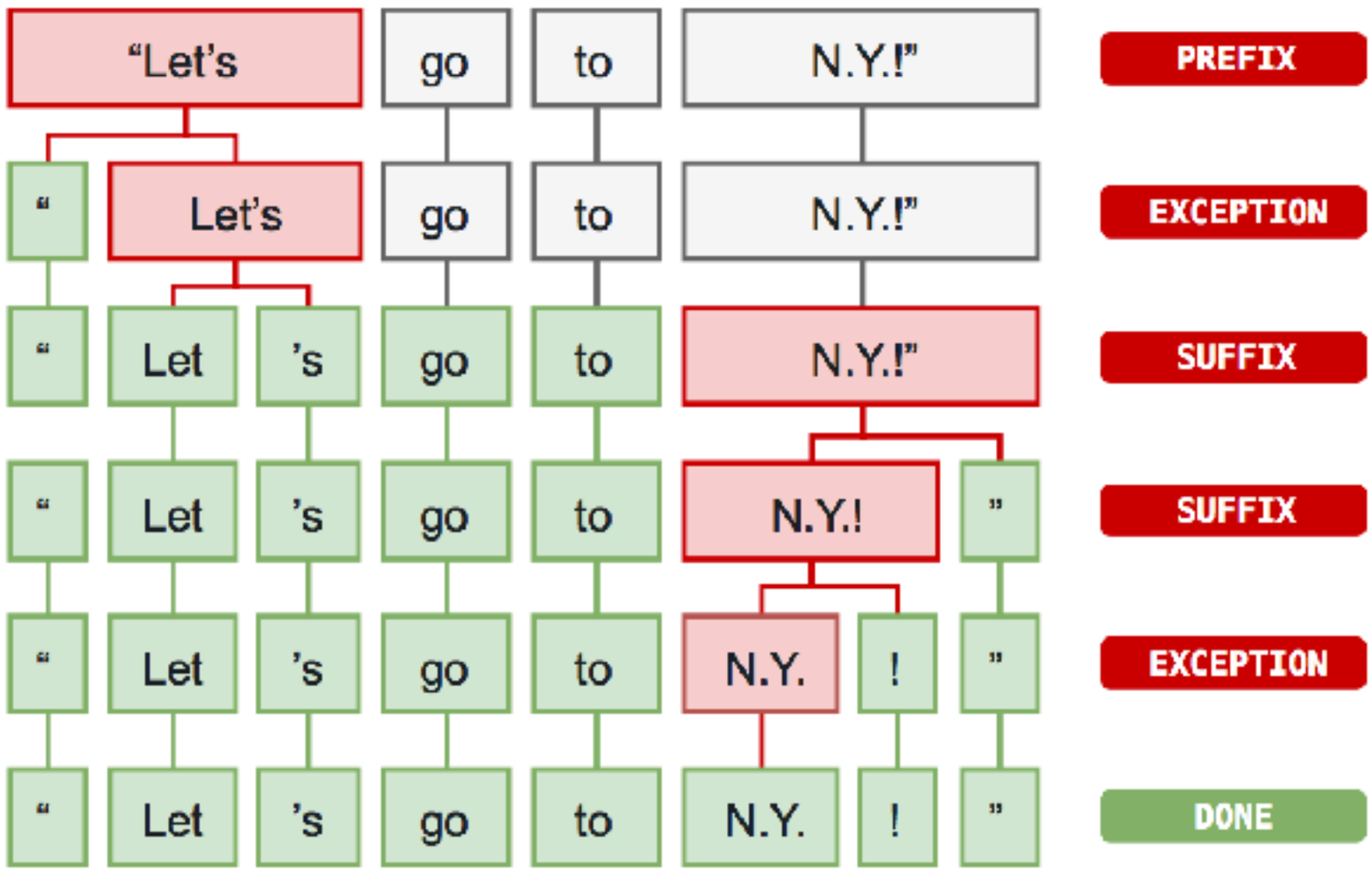
ftp://ftp.cis.upenn.edu/pub/treebank/public_html/tokenization.html

```
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!



```
import spacy
nlp = spacy.load('en')
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.

Sentence segmentation

- “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.

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

- Heuristic 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 → I
  - 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

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

Difficulties

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

Emoticons	:) :D \o/ o_O
URLs	http://www.google.com
Prices	$19.99
Decimals	19.99
Hyphens	state-of-the-art
Usernames	@dbamman
Hashtags	#blacklivesmatter

```
# 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)
```

Shorter version of http://sentiment.christopherpotts.net/code-data/happyfuntokenizing.py

EvaluateTokenizationForSentiment.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.