HACK YALE

NLP IN PYTHON

WWW.HACKYALE.COM

HACK YALE

NLP IN PYTHON

WEEK 1

CLASSIFYING TEXT

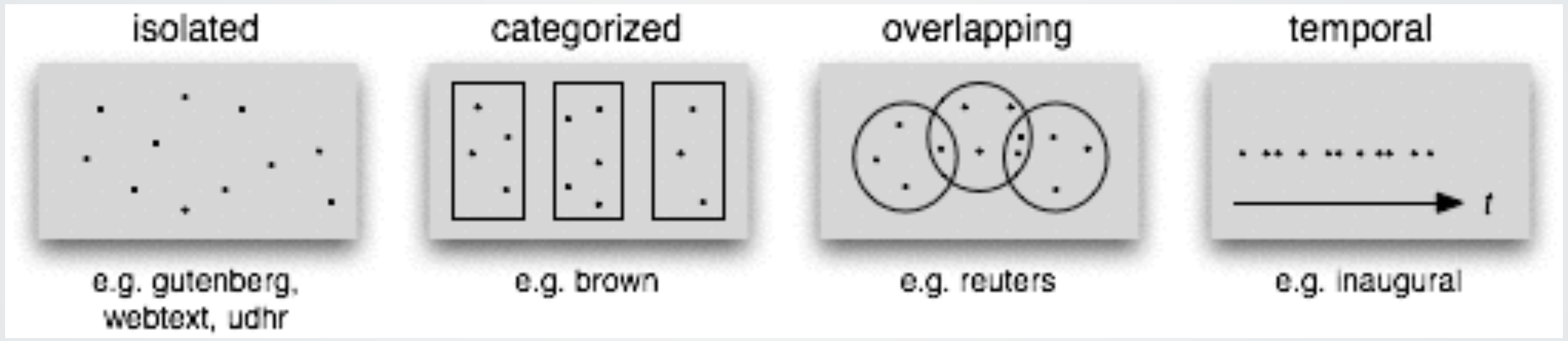
SOURCES

- Natural Language Processing with Python, by Steven Bird, Ewan Klein, and Edward Loper. O'Reilly Media, 978-0-596-51649-9.
- MOOC: Natural Language Processing with Dan Jurafsky and Christopher Manning (Stanford, Coursera)

REVIEW

HACK YALE

NLTK BOOK: CORPUS CHART



CREDIT: NLTK BOOK

HACK YALE

INAUGURAL CORPUS

```
from nltk.corpus import inaugural
import matplotlib

from nltk import Text
inaug = Text(inaugural.words())
inaug.collocations()
inaug.concordance(‘freedom’)
inaug.dispersion_plot([‘freedom’, ‘government’,
‘liberty’, ‘hope’])
```

WHAT IS TEXT CLASSIFICATION?

HACK YALE

TEXT CLASSIFICATION

- Building a model on some features of a language
- Assigning categories to unseen documents, given the ones you've seen
- Documents can mean anything:
  - Emails, news articles, tweets, etc.
  - Spam detection, article relevance, sentiment analysis, etc.

TEXT CLASSIFICATION

- On different levels
  - By word, phoneme, author, genre
- **Identifying patterns and making predictions**

TRAINING

- Supervised
  - Hand-labeled documents (i.e. movie reviews with pos. or neg.)
- Unsupervised
  - Completely unlabeled
- Semi-supervised
  - Mixture of both

TRAINING

- **Supervised**
  - Hand-labeled documents (i.e. movie reviews with pos. or neg.)
- Unsupervised
  - Completely unlabeled
- Semi-supervised
  - Mixture of both

BAYES RULE

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

NOISY CHANNEL MODEL

- There is some process that introduces noise to an underlying “true” form
- Misspelling:
  - The “true” spelling of a word goes through some noisy process that produces a misspelled word
  - Edit distance, position in word, proximity of keys on keyboard, homophones, confusion of left and right fingers

BAYES RULE

$$P(cat|obs) = \frac{P(obs|cat)P(cat)}{P(obs)}$$

cat = category

obs = observation

NLP EXAMPLES

Speech Analysis:

$$P(/keep/|"keep") = \frac{P("keep"|/keep/)P(/keep/)}{P("keep")}$$

Spelling Corrector:

$$P(the|thew) = \frac{P(thew|the)P(the)}{P(thew)}$$

DEFINITIONS

$$P(/keep/|\text{"}keep\text{"}) = \frac{P(\text{"}keep\text{"}|/keep/)P(/keep/)}{P(\text{"}keep\text{"})}$$

P("keep"|/keep/) - Noisy process

P(/keep/) - Prior probability of the lexicon

HACK YALE

NAIVE BAYES INFERENCE

- Independence assumptions
  - Noisy model can be interpreted without context
- Our case:
  - Labeled categories
  - Look at inputs separately
  - Independence assumption applies to words in the topics

LANGUAGE MODELS

HACK YALE

LANGUAGE MODEL

- N-gram
  - Simplest and “dumbest” model
  - Just counts words and their physical relation to each other in the document
    - Or phonemes, or letters, etc.
- Bag of words
  - Google n-grams example

N-GRAMS

- Applications
    - Author identification
        - Federalist Papers
    - Genre identification

N-GRAMS

Unigram (simple count):

“This movie is great!”

{‘this’: 1, ‘movie’: 1, …}

Bigram:

“# This movie is great!”

[(‘#’, ‘this’), (‘this’, ‘movie’), (‘movie’, ‘is’), …]

N-GRAMS

Trigram:

“# # This movie is great”

[(‘#’, ‘#’, ‘this’), (‘#’, ‘this’, ‘movie’), …]

4-grams, 5-grams, etc, etc.

Trigrams are usually good for English

N-GRAMS

- You have to strike a balance between
  - Capturing context (large n)
  - Creating a generalizable model (small n)
- Flaws:
  - Doesn't model long-distance dependencies
  - "The movie that I saw with my friends on Wednesday was great."

NLP EXAMPLE (BIGRAMS)

P("I hate cats")

= P(I | START) P(hate | I) P(cats | hate)

How do we measure this?

Relative Frequency Estimation (i.e., counting)

P(hate | I) = count "I hate" / count "I"

PROBABILITY OF A SENTENCE

INDEPENDENCE ASSUMPTION

Joint prob of a,b = prob of a given b and the prob of b

$P(a, b) = P(a \mid b)\, P(b)$

Markov Chain:

$P(a,b,c,d) = P(a \mid b,c,d)\, P(b \mid c,d)\, P(c \mid d)\, P(d)$

Independence Assumption (for Bigrams):

$P(a,b,c,d) = P(a \mid b)\, P(b \mid c)\, P(c \mid d)\, P(d)$

N-GRAMS

```
# -*- coding: utf-8 -*-
from nltk import bigrams, ConditionalFreqDist
from nltk.corpus import gutenberg

bi = bigrams(gutenberg.words(fileids="austen-
emma.txt"))
cfd = ConditionalFreqDist(bi)
```

PROBLEMS WITH N-GRAMS

- New words will have 0 counts
    - If your corpus only recognizes this sentence:
        - “This is great.”
    - And you get:
        - “This is amazing.”
        - P(this | START) P (is | this) P(amazing | is)
        - The probability of this sentence is 0.

HACK YALE

SMOOTHING

- Re-distributes probability mass from recognizable words to unrecognized words
- Ideally, pulls more mass from the n-grams we're not certain about (i.e., those with low counts)
- Basic: Laplace smoothing
  - +1 to every 0 count then normalize
  - Add-Lambda: use something other than 1!

SMOOTHING

- More advanced: Good-Turing
  - Give the probability of the 1-counts to the 0-counts, the 2-counts to the 1-counts, the 3-counts to the 2-counts, etc., etc.
    - Then normalize
  - You have to stop eventually
  - Sometimes you'll have categories that don't exist (i.e., there are no words counted 18 times).
    - What do you do for the 17-count words?

BASIC TEXT CLASSIFICATION

HACK YALE

IMPORTANT STEPS

- Define a features function
  - {‘feature’ : count/bool, ‘feature2’ : count/bool}
- Label words in your corpus with their categories
  - [(category, word), (category, word), (category, word)]
- Create a features set
  - [(list_of_features, category), …]

IMPORTANT STEPS

- Create a classifier object
- Remember that features are just a dict of the important aspects of some word/document
- Jack => {'last_letter' : k}

NAMES CORPUS

- 8000 names, separated into male and female
- Think of appropriate features
- Overfitting is a problem
  - Especially with a small data set

MOVIE REVIEWS

- Mess around with corpus.movie_reviews
- See sentiment.py
- Play around with numbers in NLTK Book example
  - (i.e., size of test set vs. training set, # of most frequent words we consider relevant)

ACCURACY VS. F-SCORE

	True	False
Chosen	True Positive	False Positive
Not Chosen	False Negative	True Negative

Accuracy:

How often do we choose correctly?

(i.e. True / False)

True Negative might be HUGE compared to True Positive, skewing the results.

ACCURACY VS. PRECISION & RECALL

	True	False
Chosen	**True Positive**	**False Positive**
Not Chosen	False Negative	True Negative

Precision:

Of the things our model selects, how many belong in the category?

True Positive / False Positive

ACCURACY VS. F-SCORE

	True	False
Chosen	**True Positive**	False Positive
Not Chosen	**False Negative**	True Negative

Recall:

Of the things that belong to the category, how many does our model select?

True Positive / False Negative

HACK YALE

BALANCED F1-SCORE

Combines Precision and Recall values:

$$\frac{2PR}{P + R}$$