Natural Language Processing

(start this now:)

```
pip install nltk
import nltk
nltk.download(‘all’)
```

What is NLP?

- Conversational Agents
- Dialogue Systems
- Machine Translation
- Question Answering
- Speech Recognition
- Speech Synthesis
- Lexical Semantics
- Compositional Semantics
- Sentiment Analysis
- What else?

What is NLP?

- Ambiguity
    - example: “Court to Try Shooting Defendant”, “Hospitals are sued by seven foot doctors”
    - example: “I made her duck”
        - I cooked waterfowl for her
        - I cooked waterfowl belonging to her
        - I created the (papier mache?) duck she owns
        - I caused her to quickly lower her head or body
        - I waved my magic wand and turned her into undifferentiated waterfowl
        - Lexical
            - word sense disambiguation
            - part of speech tagging
        - syntactic disambiguation
            - “her” vs “duck”
            - “lead pipe” vs “lead me on”
- speech act interpretation
    - “Any questions on that.”

Excellent source of examples: Jurafsky & Martin

What is NLP?

- Knowledge of language:
    - Phonetics&Phonology
        - linguistic sounds
    - Morphology
        - meaningful components of words
    - Semantics
        - meaning
    - Pragmatics
        - meaning wrt goals and intentions
    - Discourse
        - linguistic units larger than a single utterance

Uses

- Search
  - Information Retrieval
- Grouping
- Filtering
- Understanding
- …

Datasets

- Collection of documents: corpus
- Each document is a collection of tokens
- Each token is a…?
    - word
        - every word? what about ‘the’?
        - particular ‘version’ of the word?
            - banks working with that bank on the east bank were banking on a banker’s strike
    - n-gram
        - “I”
        - “I left”
        - “I left my”
        - “I left my heart”
        - “I left my heart in”

Considerations

- Part of Speech tagging
- Stopwords
- Sentence Segmentation,
    - That’s trivial, said Mr. Smartypants.
- N-grams
- Normalization
    - Typed features (case)
    - Lexical/Syntactic (stemming/lemmatization/pos-tagging)
        - Stemming
        - Lemmatization

TFIDF

- Term Frequency Inverse Document Frequency
  - “How apparently important was this term in this document?”
    - Term Frequency: # occurrences of t in this doc
  - “How common is this term in general?”
    - Document Frequency: # docs containing t / # docs
    - ^ Inverse that, log it (if you want), add 1 to the denominator (to avoid divide-by-zero):
    - idf(t, D): log(N / (1 + # of documents containing t)
- Normalize the first by the second
- Intuition

Bayes Theorem (again)

* Bayes' Theorem allows us to switch around the events X and Y in a P(X | Y) situation, provided we know certain other probabilities.
* Derivation:
  * P(A, B) = P(B, A) #Definition
  * P(A | B) * P(B) = P(B | A) * P(A) #Because P(A | B) * P(B) = P(A, B)
  * P(A | B) = P(B | A) * P(A) / P(B) #OMG
* In English: Posterior = Prior * Likelihood / Evidence

Intuitive?

* We have a test that's 99% accurate (99% Sensitive, 99% Specific). Only 0.5% of the population have the condition (say, for example, lactose intolerance).
* We pick Sally at random, and we test her. She tests positive. Does she have the condition?

* Answer: Probably not!
* It's only 33% likely. It's more likely that she doesn't have the condition than that she does!
* P(condition | pos) = P(pos | condition) * P(condition) / P(pos)
* = P(pos | condition) * P(condition) / (P(pos | condition) * P(condition) + P(pos | no-condition) * P(no-condition))
* = 0.99 * 0.005 / (0.99 * 0.005 + 0.01 * 0.995)
* ≈ .33

Naive Bayes: Class | Feature

* So for one feature:
  * P(class | feature) = P(feature | class) * P(class) / P(feature)
    * Don’t worry about the denominator for a minute, trust me...
  * = (# feat in class / # in class) * (# class / # total data points)
* For multiple features:
  * P(feature | class) * P(class)
  * = P(c, F1, F2, ..., Fn) (“joint”)
  * = P(c) * P(F1 | c) * P(F2 | c, F1) * P(F3 | c, F1, F2) * ...
  * But if we assume independence of features given class... P(Fi | c, Fj)
    = P(Fi | c)
  * = P(c) * P(F1 | c) * P(F2 | c) * P(F3 | c) *...
  * Which is way more tractable. Sweet!

Naive Bayes: argmax(Class)

* Now that we can calculate the numerator for some class, we calculate it for every class and compare them. The largest one is the most likely class, and therefore the class we should predict.
    * For each class c:
        * pseudo_prob_data_is_in_this_class = P(c) * P(F1 | c) * P(F2 | c) * P(F3 | c) *...
    * Prediction = the c which led to max pseudo_prob_data_is_in_this_class
* The denominator stays the same for all classes, so we don't have to calculate it (or even think about how we might calculate it--save that for another time and lots of caffeine). Awesome.
* "Training" is simply the process of counting up occurrences (all the little terms in "(# feat in class / # in class) * (# class / # total data points)") so we can use them to predict. (In some models/ algorithms, training is much, much more complicated.)

Naive Bayes

Naive Bayes Classifier: $\underset{Y}{\text{argmax}}\ P(Y|\vec{X}) = \underset{Y}{\text{argmax}}\ P(x_1|Y)P(x_2|Y)...P(x_3|Y)P(Y)$

$$= \underset{Y}{\text{argmax}}\ P(Y)\prod_{i=1}^{k} P(x_i|Y)$$

Naive Bayes details

- Log-transformation
- Unknown words
    - Laplace Smoothing
- Online learning
- Extensions
- When to use Naive Bayes
    - Pros:
        - Good with “wide data” (p >> n)
            - Good if n is small or n is quite big
        - Fast to train
        - Good at online learning, streaming data
        - Simple to implement, not necessarily memory-bound (DB implementations)
        - Multi-class
    - Cons:
        - Naive assumption means correlated features aren’t actually treated right
        - Sometimes outperformed by other models

Laplace Smoothing

- Helpful short video :)

- https://www.youtube.com/watch?v=evtCdmjcZ4I