COURSE WEBSITE:

http://cslu.ohsu.edu/~gormanky/courses/CS662
COURSE STRUCTURE
WHAT IS NATURAL LANGUAGE PROCESSING?
LANGUAGE IS AMBIGUOUS

Possible paraphrase: Upstate bison that other upstate bison bully, they also bully upstate NY bison.
Pope Francis on Saturday appointed a victim of sexual abuse and a senior cardinal known for his zero-tolerance approach to a new group charged with advising the Catholic Church on how to respond to the problem of sexual abuse of children.

(WSJ, 3/22/2014)
A WHIG HISTORY
THE DREAM OF THE ‘50S

Much of the early work in NLP focused on systems for understanding natural language. (Natural language must have seemed a particularly promising modality for user input in the teletype era.) Systems—when they worked—parsed and interpreted natural language input and generated natural language output.

SHRDLU (Winograd 1971):

Person: GRASP THE PYRAMID.
Computer: I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN.
Person: FIND A BLOCK WHICH IS TALLER THAN THE ONE YOU ARE HOLDING AND PUT IT INTO THE BOX.
Computer: BY "IT", I ASSUME YOU MEAN THE BLOCK WHICH IS TALLER THAN THE ONE I AM HOLDING.
Computer: OK. (does it)
Person: WHAT DOES THE BOX CONTAIN?
Computer: THE BLUE PYRAMID AND THE BLUE BLOCK.
Person: WHAT IS THE PYRAMID SUPPORTED BY?
Computer: THE BOX.
Person: HOW MANY BLOCKS ARE NOT IN THE BOX?
Computer: FOUR OF THEM.
Person: IS AT LEAST ONE OF THEM NARROWER THAN THE ONE WHICH I TOLD YOU TO PICK UP?
Computer: YES, THE RED CUBE.

[source: http://hci.stanford.edu/~winograd/shrdlu/]
The cognitive revolution sparked by Chomsky (among others) revealed the enormous complexity of human languages.

The 1964 ALPAC report (more on that later) emphasized the need for basic (rather than applied) research in computational linguistics (but, unfortunately shifted funding elsewhere).

Meanwhile, machine learning was just getting started: the perceptron was published in 1958 and the term machine learning was coined in 1959.
SOME EARLY SUCCESSES

1965-1970: Efficient algorithms for parsing

1976-1980: Noisy channel models for statistical automatic speech recognition

1987-1993: Evaluation goes mainstream

1992: The Penn Treebank
A new goal emerges in the ‘80s–‘90s: automated, linguistically-informed annotation of text.

Not all useful tasks require full understanding, and when they do, it is still easier to design an pipeline of increasingly abstract annotations.
SOME NLP TASKS AND APPLICATIONS
Do you really think it is weakness that yields to temptation? I tell you that there are terrible temptations which it requires strength, strength and courage to yield to. - Oscar Wilde

Do you really think it is weakness that yields to temptation? I tell you that there are terrible temptations which it requires strength, strength and courage to yield to. - Oscar Wilde
We are trying to understand the difference.
"This movie doesn’t care about cleverness, wit or any other kind of intelligent humor."

[source: http://nlp.stanford.edu:8080/sentiment/]
PROBABILITY THEORY FOR NLP
ARGMAX/ARGMIN

\[ \text{argmax}_x f(x) \]

The value(s) of \( x \) at which \( f(x) \) is at its maximum; e.g.,

\[ \text{argmax}_x \sin x = \frac{\pi}{2} \]
JOINT PROBABILITY

The probability that two or more random variables each take on certain respective values

\[ p(11) = p(\text{red} = 6) \cdot p(\text{blue} = 5) + p(\text{red} = 5) \cdot p(\text{blue} = 6) \]

\[ = \frac{1}{6} \cdot \frac{1}{6} + \frac{1}{6} \cdot \frac{1}{6} + \frac{1}{36} \]

\[ = \frac{1}{18} \]
CONDITIONAL PROBABILITY

The probability that one or more random variables have a respective value given that one or more other random variables have a respective value.
\[ p(A \mid B) := \frac{p(A \cap B)}{p(B)} \]

\[ p(\text{red} = 2) = \frac{1}{6} \]

\[ p(\text{red + blue} \leq 5) = \frac{5}{18} \]

\[ p(\text{red} = 2 \mid \text{red + blue} \leq 5) = ? \]

\[ p(\text{red} = 2, \text{red + blue} \leq 5) = \frac{1}{12} \]

\[ p(\text{red} = 2 \mid \text{red + blue} \leq 5) = \frac{1/12}{5/18} = \frac{3}{10} \]
STATISTICAL INDEPENDENCE

Random variables $A$, $B$ are independent iff

$$P(A \mid B) = P(A)$$

$$P(B \mid A) = P(B)$$
BAYES’ THEOREM

\[ p(A \mid B) = \frac{p(B \mid A) p(A)}{p(B)} \]
THE NOISY CHANNEL: WEAYER’S INTUITION

Warren Weaver, 1949 Rockefeller Foundation memorandum
Translation:

“When I look at an article in Russian, I say: this is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.”
Given a ciphertext $ct$, select plaintext $pt$ that maximizes the conditional probability of the plaintext

$$\arg \max_{pt} p(pt \mid ct)$$

Bayes’ theorem allows us to rewrite this as

$$\arg \max_{pt} \frac{p(ct \mid pt) p(pt)}{p(ct)}$$

The denominator is constant in $ct$, so

$$\frac{p(ct \mid pt) p(pt)}{p(ct)} \propto p(ct \mid pt) p(pt)$$

Thus we can rewrite the objective as

$$\arg \max_{pt} p(pt \mid ct) = \arg \max_{pt} p(ct \mid pt) p(pt)$$
DATA STRUCTURES FOR CONDITIONAL FREQUENCIES AND PROBABILITIES
REPRESENTING CONDITIONAL PROBABILITIES IN MEMORY

- If the space of outcomes is finite, small, and dense, a square matrix is a possibility (“sparse matrix” packages don’t apply here; we’re not going to do any linear algebra)

- Represent them as a tuple key to a hashtable (assuming outcomes are immutable, hashable): problems with this?
  - Cannot efficiently convert from conditional frequencies to conditional probabilities
  - Cannot efficiently generate the conditional probabilities for all \( a \in A \) once a particular \( b \in B \) has been observed
HASHTABLE OF
HASHTABLES: NAÏVE

```python
def make_cpdf(cdist):
   cpdf = {}
   for (a, b, p_a_given_b) in cdist:
      try:
         cpdf[b][a] = p_a_given_b
      except KeyError:
         cpdf[b] = {a: p_a_given_b}
   return cpdf

def cpdf_get(a, b, cpdf):
   try:
      return cpdf[b][a]
   except KeyError:
      return 0
```
INTRODUCTING DEFAULTDICT

```python
>>> from collections import defaultdict
>>> def inner_fnc():
...     print 'Hello, world!'
...     return 0
...

>>> d = defaultdict(inner_fnc)
>>> x = d['a']
Hello, world!
>>> print x
0
```
HASHTABLE OF HASHTABLES: PYTHONIC

```python
from functools import partial

def make_cpdf(cdist):
    cpdf = defaultdict(partial(defaultdict, int))
    for (a, b, p_a_given_b) in cdist:
        cpdf[b][a] = p_a_given_b
    return cpdf


def cpdf_get(a, b, cpdf):
    return cpdf[b][a]
```
UNDERFLOW

• Can a computer compute a number smaller than it can represent (i.e., store in memory)?

```python
>>> 2 ** -1000
9.332663618503219e-302
>>> 2 ** -2000
0.0
```

• Solution: represent small probabilities as negative log probabilities...enter BitWeight