Questions that linguistics should answer

- What kinds of things do people say?
- What do these things say/ask/request about the world?

Example: *In addition to this, she insisted that women were regarded as a different existence from men unfairly.*

- Text corpora give us data with which to answer these questions
- What words, rules, statistical facts do we find?
- Can we build programs that learn effectively from this data, and can then do NLP tasks?
- They are an externalization of linguistic knowledge

Corpora

- A corpus is a body of naturally occurring text, normally one organized or selected in some way
  - Greek: one corpus, two corpora
- A balanced corpus tries to be representative across a language or other domain
- Balance is something of a chimaera: What is balanced?
  - Who spends what percent of their time reading the sports pages?

The Brown corpus

- 1 million words, which seemed huge at the time.
  - Sorting the words to produce a word list took 17 hours of (dedicated) processing time, because the computer (an IBM 7070) had the equivalent of only about 40 kilobytes of memory, and so the sort algorithm had to store the data being sorted on tape drives.
- Its significance has increased over time, but also awareness of its limitations.
- Tagged for part of speech in the 1970s
  - The/AT General/JJ-TL Assembly/NN-TL ./, which/WDT adjourns/VBZ today/NR ./, has/HVZ performed/VBN

Recent corpora

- British National Corpus. 100 million words, tagged for part of speech. Balanced.
- Newswire (NYT or WSJ are most commonly used): Something like 600 million words is fairly easily available.
- Legal reports; UN or EU proceedings (parallel multilingual corpora – same text in multiple languages)
- The Web (in the billions of words, but need to filter for distinctness).
- Penn Treebank: 2 million words (1 million WSJ, 1 million speech) of parsed sentences (as phrase structure trees).

Large and strange, sparse, discrete distributions

- Both features and assigned classes regularly involve multinomial distributions over huge numbers of values (often in the tens of thousands).
- The distributions are very uneven, and have fat tails
- Enormous problems with data sparseness: much work on smoothing distributions/backoff (shrinkage), etc.
- We normally have inadequate (labeled) data to estimate probabilities
- Unknown/unseen things are usually a central problem
- Generally dealing with discrete distributions though

Sparsity

- How often does an every day word like *kick* occur in a million words of text?
  - *kick*: about 10 [depends vastly on genre, of course]
  - *wrist*: about 5
- Normally we want to know about something bigger than a single word, like how often you *kick a ball*, or how often the conative alternation *he kicked at the balloon* occurs.
- How often can we expect that to occur in 1 million words?
  - Almost never.
- “There’s no data like more data” [if of the right domain]
Probabilistic language modeling

- Assigns probability $P(t)$ to a word sequence $t = w_1 w_2 \cdots w_n$
- Chain rule and joint/conditional probabilities for text $t$:
  $$P(t) = P(w_1 \cdots w_n) = \prod_{i=1}^{n} P(w_i | w_1 \cdots w_{i-1})$$
  
  where
  $$P(w_k | w_1 \cdots w_{k-1}) = \frac{P(w_1 \cdots w_k)}{P(w_1 \cdots w_{k-1})} \approx \frac{C(w_1 \cdots w_k)}{C(w_1 \cdots w_{k-1})}$$

- The chain rule leads to a history-based model: we predict following things from past things
- We cluster histories into equivalence classes to reduce the number of parameters to estimate

$n$-gram models: the classic example of a statistical model of language

- Each word is predicted according to a conditional distribution based on a limited context
- Conditional Probability Table (CPT): $P(X|\text{both})$
  - $P(\text{of}|\text{both}) = 0.066$
  - $P(\text{to}|\text{both}) = 0.041$
  - $P(\text{in}|\text{both}) = 0.038$
- From 1940s onward (or even 1910s – Markov 1913)
- a.k.a. Markov (chain) models

Markov models = $n$-gram models

- Deterministic FSMs with probabilities
  - eats:0.01
  - broccoli:0.002
  - fish:0.1
  - chicken:0.15
- No long distance dependencies
  - "The future is independent of the past given the present"
- No notion of structure or syntactic dependency
- But lexical
- (And: robust, have frequency information, …)

$n$-gram models

- Core language model for the engineering task of better predicting the next word:
  - Speech recognition
  - OCR
  - Context-sensitive spelling correction
- It is only recently that they have been improved on for these tasks (Chelba and Jelinek 1998; Charniak 2001).
- But linguistically, they are appalling simple and naive

$n$-th order Markov models

- First order Markov assumption = bigram
  $$P(w_k | w_1 \cdots w_{k-1}) \approx P(w_k | w_{k-1}) = \frac{P(w_{k-1} w_k)}{P(w_{k-1})}$$
- Similarly, $n$-th order Markov assumption
- Most commonly, trigram (2nd order):
  $$P(w_k | w_1 \cdots w_{k-1}) \approx P(w_k | w_{k-1}) = \frac{P(w_{k-1} w_k)}{P(w_{k-2}, w_{k-1})}$$
Why mightn’t $n$-gram models work?

- Relationships (say between subject and verb) can be arbitrarily distant and convoluted, as linguists love to point out:
  - The man that I was watching without pausing to look at what was happening down the street, and quite oblivious to the situation that was about to befall him confidently strode into the center of the road.

Why do they work?

- That kind of thing doesn’t happen much
- Collins (1997):
  - 74% of dependencies (in the Penn Treebank - WSJ) are with an adjacent word (95% with one ≤ 5 words away), once one treats simple NPs as units:
  - Below, 4/6 = 66% based on words

Evaluation of language models

- Best evaluation of probability model is task-based
- As substitute for evaluating one component, standardly use corpus per-word cross entropy:

$$H(X, p) = -\frac{1}{n} \sum_{i=1}^{n} \log_2 P(w_i|w_1, \ldots, w_{i-1})$$

- Or perplexity (measure of uncertainty of predictions):

$$PP(X, p) = 2^{H(X, p)} = \left[ \prod_{i=1}^{n} P(w_i|w_1, \ldots, w_{i-1}) \right]^{-1/n}$$

- Needs to be assessed on independent, unseen, test data

Relative frequency = Maximum Likelihood Estimate

$$P(w_2|w_1) = \frac{C(w_1, w_2)}{C(w_1)}$$

(or similarly for higher order or joint probabilities)

Makes training data as probable as possible

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>8</td>
<td>1087</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>want</td>
<td>3</td>
<td>0</td>
<td>786</td>
<td>0</td>
<td>6</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>860</td>
<td>3</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>19</td>
<td>2</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>Chinese</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>food</td>
<td>19</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>lunch</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Selected bigram counts (Berkeley Restaurant Project – J&M)
I want to eat Chinese food lunch

Selected bigram probabilities (Berkeley Restaurant Project – J&M)

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>.0023</td>
<td>.32</td>
<td>0</td>
<td>.0038</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>want</td>
<td>.0025</td>
<td>.0</td>
<td>.65</td>
<td>0</td>
<td>.0049</td>
<td>.0066</td>
<td>.0049</td>
</tr>
<tr>
<td>to</td>
<td>.00092</td>
<td>.0031</td>
<td>.26</td>
<td>.0092</td>
<td>0</td>
<td>.0037</td>
<td></td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>.0021</td>
<td>0</td>
<td>.02</td>
<td>.0021</td>
<td>.055</td>
</tr>
<tr>
<td>Chinese</td>
<td>.004</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.56</td>
<td>.0047</td>
<td></td>
</tr>
<tr>
<td>food</td>
<td>.013</td>
<td>0</td>
<td>.011</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>lunch</td>
<td>.0087</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.0022</td>
<td>0</td>
</tr>
</tbody>
</table>

Limitations of Maximum Likelihood Estimator

Problem: often infinitely surprised when unseen word appears ($P(\text{unseen}) = 0$)

- Problem: this happens commonly.
- Probabilities of zero count words are too low
- Probabilities of nonzero count words are too high
- Estimates for high count words are fairly accurate
- Estimates for low count words are unstable
- We need smoothing!

Adding one = Laplace's law (1851)

$$P(w_2|w_1) = \frac{C(w_1, w_2) + 1}{C(w_1) + V}$$

- $V$ is the vocabulary size (assume fixed, closed vocabulary)
- This is the Bayesian (MAP) estimator you get by assuming a uniform unit prior on events (= a Dirichlet prior)

Add one = Laplace's law (1851)

Original versus add-one predicted counts
Add one estimator

- Problem: gives too much probability mass to unseens.
- Not good for large vocab, comparatively little data (i.e., NLP)
- e.g. 10,000 word vocab, 1,000,000 words of training data, but comes across occurs 10 times. Of those, 8 times next word is as
  - \( P_{\text{MLE}}(\text{comes across}) = 0.8 \)
  - \( P_{+1}(\text{comes across}) = \frac{8+1}{8+11000} \approx 0.0009 \)
- Quick fix: Lidstone's law (Mitchell's (1997) "\( m \)-estimate"):
  \[
  P(w_2|w_1) = \frac{C(w_1, w_2) + \lambda}{C(w_1) + \lambda V}
  \]
  for \( \lambda < 1 \), e.g., 1/2 or 0.05

The frequency of previously unseen events

How do you know how likely you are to see a new word type in the future (in a certain context)?

- Examine some further text and find out [empirical held out estimators = validation]
- Use things you've seen once to estimate probability of unseens:
  \[
  P(\text{unseen}) = \frac{N_1}{N}
  \]
  where \( N_1 \) is number of things seen once. (Good-Turing: Church and Gale 1991; Gale and Sampson 1995)

Good-Turing smoothing

Derivation reflects leave-one out estimation (Ney et al. 1997):

- For each word token in data, call it the test set; remaining data is training set
- See how often word in test set has \( r \) counts in training set
- This will happen every time word left out has \( r + 1 \) counts in original data
- So total count mass of \( r \) count words is assigned from mass of \( r + 1 \) count words \([= N_{r+1} \times (r + 1)]\)
- Needs smoothing; accurate when lots of data

Absolute discounting

- Idea is that we want to discount counts of seen things a little, and reallocate this probability mass to unseens
- By subtracting a fixed count, probability estimates for commonly seen things are scarcely affected, while probabilities of rare things are greatly affected
- If the discount is around \( \delta = 0.75 \), then seeing something once is not so different to not having seen it at all
  \[
  P(w_2|w_1) = \frac{C(w_1, w_2) - \delta}{C(w_1)} \quad \text{if} \quad C(w_1, w_2) > 0
  
  P(w_2|w_1) = \frac{V - N_0}{N_0 C(w_1)} \quad \text{otherwise}
  \]

Smoothing: Rest of the story (1)

- Other methods: backoff (Katz 1987), cross-validation, Witten-Bell discounting, ... (Chen and Goodman 1998; Goodman 2001)
- Simple, but surprisingly effective: Simple linear interpolation (deleted interpolation; mixture model; shrinkage):
  \[
  \hat{P}(w_3|w_1, w_2) = \lambda_3 P_3(w_3|w_1, w_2) + \lambda_2 P_2(w_3|w_2) + \lambda_1 P_1(w_3)
  \]
  - The \( \lambda_i \) can be estimated on held out data
  - They can be functions of (equivalence-classed) histories
  - For open vocabulary, need to handle words unseen in any context (just use UNK, spelling models, etc.)
**Smoothing: Rest of the story (2)**

- Recent work emphasizes constraints on the smoothed model.
- Kneser and Ney (1995): Backoff $n$-gram counts are not proportional to frequency of $n$-gram in training data but to expectation of how often it should occur in novel trigram - since one only uses backoff estimate when trigram not found.
- (Smoothed) maximum entropy (a.k.a. loglinear) models again place constraints on the distribution (Rosenfeld 1996, 2000).

---

**Size of language models with cutoffs**

Seymore and Rosenfeld (ICSLP, 1996): 58,000 word dictionary, 45 M words of training data, WSJ, Sphinx II

<table>
<thead>
<tr>
<th>Bi/Tri-gram cutoff</th>
<th># Bigrams</th>
<th># Trigrams</th>
<th>Memory (Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/0</td>
<td>4,627,551</td>
<td>16,838,937</td>
<td>104</td>
</tr>
<tr>
<td>0/1</td>
<td>4,627,551</td>
<td>3,581,187</td>
<td>51</td>
</tr>
<tr>
<td>1/1</td>
<td>1,787,935</td>
<td>3,581,187</td>
<td>29</td>
</tr>
<tr>
<td>10/10</td>
<td>347,647</td>
<td>367,928</td>
<td>4</td>
</tr>
</tbody>
</table>

80% of unique trigrams occur only once!

- Note the possibilities for compression (if you're confident that you'll be given English text and the encoder/decoder can use very big tables).

---

**More LM facts**

- Seymore, Chen, Eskenazi and Rosenfeld (1996)
- HUB-4: Broadcast News 51,000 word vocab, 130M words training. Katz backoff smoothing (1/1 cutoff).
- Perplexity 231
- 0/0 cutoff: 3% perplexity reduction
- 7-grams: 15% perplexity reduction
- Note the possibilities for compression, if you're confident that you'll be given English text (and the encoder/decoder can use very big tables).