Speech in a Slide

- Frequency gives pitch; amplitude gives volume
- Frequencies at each time slice processed into observation vectors

---

\[ a_{12} a_{13} a_{12} a_{14} a_{14} \]
The Noisy-Channel Model

- We want to predict a sentence given acoustics:
  \[ w^* = \arg \max_w P(w|a) \]
- The noisy channel approach:
  \[ w^* = \arg \max_w P(w|a) \]
  \[ = \arg \max_w P(a|w)P(w)/P(a) \]
  \[ \propto \arg \max_w P(a|w)P(w) \]

Acoustic model: HMMs over word positions with mixtures of Gaussians as emissions
Language model: Distributions over sequences of words (sentences)

Acoustically Scored Hypotheses

- the station signs are in deep in english -14732
- the stations signs are in deep in english -14735
- the station signs are in deep into english -14739
- the station ’s signs are in deep in english -14740
- the station signs are in deep in the english -14741
- the station signs are indeed in english -14757
- the station ’s signs are indeed in english -14760
- the station signs are indians in english -14790
- the station signs are indian in english -14799
- the stations signs are indians in english -14807
- the stations signs are indians and english -14815
### Translation: Codebreaking?

- "Also knowing nothing official about, but having guessed and inferred considerable about, the powerful new mechanized methods in cryptography—methods which I believe succeed even when one does not know what language has been coded—one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. 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.'"

- Warren Weaver (1955:18, quoting a letter he wrote in 1947)
MT Overview

MT System Components

\[
\arg\max_{e} P(e|f) = \arg\max_{e} P(f|e)P(e)
\]
Other Noisy-Channel Processes

- **Handwriting recognition**
  \[ P(\text{text} | \text{strokes}) \propto P(\text{text})P(\text{strokes} | \text{text}) \]

- **OCR**
  \[ P(\text{text} | \text{pixels}) \propto P(\text{text})P(\text{pixels} | \text{text}) \]

- **Spelling Correction**
  \[ P(\text{text} | \text{typos}) \propto P(\text{text})P(\text{typos} | \text{text}) \]

- **More…**

Probabilistic Language Models

- **Goal:** Assign useful probabilities \( P(x) \) to sentences \( x \)
  - Input: many observations of training sentences \( x \)
  - Output: system capable of computing \( P(x) \)

- **Probabilities should broadly indicate likelihood of sentences**
  - \( P(\text{I saw a van}) \gg P(\text{eyes awe of an}) \)
  - *Not grammaticality:* \( P(\text{artichokes intimidate zippers}) \approx 0 \)
  - In principle, “likely” depends on the domain, context, speaker…

- **One option: empirical distribution over training sentences?**
  - Problem: doesn’t generalize (at all)

- **Two ways of generalizing**
  - Decomposition: break sentences into small steps which can be recombined in new ways (conditional independence)
  - Smoothing: allow for the possibility of unseen events
N-Gram Language Models

- No loss of generality: break sentence probability down
  \[ P(w_1 \ldots w_n) = \prod_i P(w_i | w_1 \ldots w_{i-1}) \]

- Too many histories!
  - \[ P(\ldots | \text{No loss of generality: break sentence}) \]
  - \[ P(\ldots | \text{the water is so transparent that}) \]

- N-gram models: assume each word depends only on a short linear history
  \[ P(w_1 \ldots w_n) = \prod_i P(w_i | w_{i-k} \ldots w_{i-1}) \]

Unigram Models

- Simplest case: unigrams
  \[ P(w_1 \ldots w_n) = \prod_i P(w_i) \]

- Generative process: pick a word, pick a word, ...

- As a graphical model:

- To make this a proper distribution over sentences, we have to generate a special STOP symbol last. (Why?)

Examples:
- [fifth, an, of, futures, the, an, incorporated, a, a, the, inflation, most, dollars, quarter, in, is, mass.]
- [thrift, did, eighty, said, hard, 'm, july, bullish]
- [that, or, limited, the]
- []
- [after, any, on, consistently, hospital, lake, of, of, other, and, factors, raised, analyst, too, allowed, mexico, never, consider, fall, bungled, davison, that, obtain, price, lines, the, to, sass, the, the, further, board, a, details, machinists, the, companies, which, rivals, an, because, longer, oakes, percent, a, they, three, edward, it, currier, an, within, in, three, wrote, is, you, s., longer, institute, dentistry, pay, however, said, possible, to, rooms, hiding, eggs, approximate, financial, canada, the, so, workers, advancers, half, between, nasdaq]
Bigram Models

- Big problem with unigrams: \( P(\text{the the the}) >> P(\text{I like ice cream})! \)
- Condition on previous word:

\[
P(w_1 \ldots w_n) = \prod_i P(w_i | w_{i-1})
\]

- Obvious that this should help – in probabilistic terms, we’re using weaker conditional independence assumptions (what’s the cost?)
- Any better?
  - [texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen]
  - [outside, new, car, parking, lot, of, the, agreement, reached]
  - [although, common, shares, rose, forty, six, point, four, hundred, dollars, from, thirty, seconds, at, the, greatest, play, disingenuous, to, be, reset, annually, the, buy, out, of, american, brands, wyng, for, mr., womack, currently, sharedata, incorporated, believe, chemical, prices, undoubtedly, will, be, as, much, is, scheduled, to, conscientious, teaching]
  - [this, would, be, a, record, november]

More N-Gram Examples

- To him swallowed confess hear both. Which. Of save on trial for are ay device and note life have
- Every enter now severely so, let
- All be late speaks; or! a more to leg less first you enter
- Are where excent and sight have rise excellency took off.. Sleep knave we. near; vile like
- Unigram
Regular Languages?

- N-gram models are (weighted) regular languages
  - Many linguistic arguments that language isn’t regular.
    - Long-distance effects: “The computer which I had just put into the machine room on the fifth floor crashed.”
    - Recursive structure
  - Why CAN we often get away with n-gram models?

- PCFG LM (later):
  - [This, quarter, ’s, surprisingly, independent, attack, paid, off, the, risk, involving, IRS, leaders, and, transportation, prices, .]
  - [It, could, be, announced, sometime, .]
  - [Mr., Toseland, believes, the, average, defense, economy, is, drafted, from, slightly, more, than, 12, stocks, .]

Model Parameters

- The parameters of an n-gram model:
  - The conditional probability estimates, we’ll call them \( \theta \)
  - Obvious estimate is the relative frequency estimate (aka the maximum likelihood estimate)
    \[
    \hat{P}(w|w_{-1}) = \frac{c(w_{-1}, w)}{\sum_{w'} c(w_{-1}, w')}
    \]

- General method
  - Take a training set \( X \) and a test set \( X' \)
  - Compute an estimate \( \theta \) from \( X \)
  - Use it to assign probabilities to other sentences, such as \( X' \)

- Some quantities of interest
  - Training likelihood
    \[
    L(X|\theta) = \prod_{x \in X} P(x|\theta)
    \]
  - Test likelihood
    \[
    L(X'|\theta) = \prod_{x' \in X'} P(x'|\theta)
    \]
Is This Working?

- The game isn’t to pound out fake sentences!
  - Obviously, generated sentences get “better” as we increase the model order
  - More precisely: using ML estimators, higher order is always better likelihood on train, but not test

- What we really want to know is:
  - Will our model prefer good sentences to bad ones?
  - Bad ≠ ungrammatical!
  - Bad ≈ unlikely
  - Bad = sentences that our acoustic model really likes but aren’t the correct answer

Measuring Model Quality

- Word Error Rate (WER)

  Correct answer: Andy saw a part of the movie
  Recognizer output: And he saw apart of the movie

  The “right” measure:
  - Task error driven
  - For speech recognition
  - For a specific recognizer!

  For general evaluation, we want a measure which references only good text, not mistake text (why?)

$WER: \frac{4}{7} = 57\%$
Measuring Model Quality

- The Shannon Game:
  - How well can we predict the next word?
    - When I order pizza, I wipe off the ____
    - Many children are allergic to ____
    - I saw a ____
  - Unigrams are terrible at this game. (Why?)

- “Entropy”: really per-word test log likelihood (misnamed)
  \[ H(X|\theta) = -\frac{1}{|X|} \sum_{x \in X} \log_2 P(x|\theta) \sum_{x \in X} |x| \sum_{i} \log P(x_i|x_{i-1}, \theta) \]

Measuring Model Quality

- Problem with “entropy”:
  - 0.1 bits of improvement doesn’t sound so good
  - Solution: perplexity
  \[ \text{perp}(X, \theta) = 2^{H(X|\theta)} \]
  - Interpretation: average branching factor in model

- Big notes:
  - It’s easy to get bogus perplexities by having bogus probabilities that sum to more than one over their event spaces. 30% of you will do this on HW1.
  - Even though our models require a stop step, averages are per actual word, not per derivation step.
Sparsity

- Problems with n-gram models:
  - New words appear all the time:
    - Synaptitude
    - 132,701.03
    - multidisciplinarization
  - New bigrams: even more often
  - Trigrams or more – still worse!

- Zipf’s Law
  - Types (words) vs. tokens (word occurrences)
  - Broadly: most word types are rare ones
  - Specifically:
    - Rank word types by token frequency
    - Frequency inversely proportional to rank
  - Not special to language: randomly generated character strings have this property (try it!)

Parameter Estimation

- Maximum likelihood estimates won’t get us very far
  \[
  \hat{P}(w|w_{-1}) = \frac{c(w_{-1}, w)}{\sum_{w'} c(w_{-1}, w')}
  \]

- Need to smooth these estimates

- General method (procedurally)
  - Take your empirical counts
  - Modify them in various ways to improve estimates

- General method (mathematically)
  - Often can give estimators a formal statistical interpretation
  - … but not always
  - Stuff that works not always the same as stuff we can explain (yet!)
Smoothing

- We often want to make estimates from sparse statistics:

\[ P(w \mid \text{denied the}) \]
3 allegations
2 reports
1 claims
1 request
7 total

- Smoothing flattens spiky distributions so they generalize better

\[ P(w \mid \text{denied the}) \]
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total

- Very important all over NLP, but easy to do badly!
- We’ll illustrate with bigrams today (h = previous word, could be anything).

Priors on Parameters

- Most obvious formal solution: use MAP estimate instead of ML estimate for a multinomial P(X)

- Maximum likelihood estimate: \( \max P(X|\theta) \)

\[ \theta_{\text{ML}} = \frac{c(x)}{\sum_{x'} c(x')} \]

- MAP estimate: \( \max P(\theta|X) \)
  - Dirichlet priors are a convenient choice
    - Specified by a center \( \theta' \) and strength \( k \), Dir(\( \theta',k \)) or Dir(k\( \theta' \))
    - Mean is center, higher strength means lower variance
  - MAP estimate is then

\[ \theta_{\text{MAP}} = \frac{c(x) + k\theta_x - 1}{\sum_{x'}(c(x') + k\theta_x - 1)} \]
Smoothing: Add-One, Etc.

- With a uniform prior, get estimates of the form
  \[ P_{\text{add}}(x) = \frac{c(x) + \delta}{\sum_x (c(x') + \delta)} \]

  - Add-one smoothing especially often talked about

- For a bigram distribution, can use a prior centered on the empirical unigram:
  \[ P_{\text{dir}}(w|w_{-1}) = \frac{c(w_{-1}, w) + k\hat{P}(w)}{\sum_{w'} c(w_{-1}, w')} + k \]

- Can consider hierarchical formulations in which trigram is centered on smoothed bigram estimate, etc [MacKay and Peto, 94]

- Basic idea of conjugacy is convenient: prior shape shows up as pseudo-counts

- Problem: works quite poorly!

Linear Interpolation

- Problem: \( \hat{P}(w|w_{-1}, w_{-2}) \) is supported by few counts
- Classic solution: mixtures of related, denser histories, e.g.:
  \[ \lambda \hat{P}(w|w_{-1}, w_{-2}) + \lambda' \hat{P}(w|w_{-1}) + \lambda'' \hat{P}(w) \]

  - The mixture approach tends to work better than the Dirichlet prior approach for several reasons
    - Can flexibly include multiple back-off contexts, not just a chain
    - Good ways of learning the mixture weights with EM (later)
    - Not entirely clear why it works so much better

- All the details you could ever want: [Chen and Goodman, 98]
Held-Out Data

- Important tool for calibrating how models generalize:

```
Training Data   Held-Out Data   Test Data
```

- Set a small number of hyperparameters that control the degree of smoothing by maximizing the (log-)likelihood of held-out data
- Can use any optimization technique (line search or EM usually easiest)

- Examples:

\[
P_{dir}(w|w_{-1}, k) = \frac{c(w_{-1}, w) + k\hat{P}(w)}{\left(\sum_{w'} c(w_{-1}, w')\right) + k}
\]

\[
P_{lin}(w|w_{-1}, \lambda, \lambda', \lambda'') = \lambda\hat{P}(w|w_{-1}, w_{-2}) + \lambda'\hat{P}(w|w_{-1}) + \lambda''\hat{P}(w)
\]

Held-Out Reweighting

- What’s wrong with unigram-prior smoothing?
- Let’s look at some real bigram counts [Church and Gale 91]:

<table>
<thead>
<tr>
<th>Count in 22M Words</th>
<th>Actual c* (Next 22M)</th>
<th>Add-one’s c*</th>
<th>Add-0.0000027’s c*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.448</td>
<td>27e-10</td>
<td>~1</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>37e-10</td>
<td>~2</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>47e-10</td>
<td>~3</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>57e-10</td>
<td>~4</td>
</tr>
<tr>
<td>5</td>
<td>4.21</td>
<td>67e-10</td>
<td>~5</td>
</tr>
</tbody>
</table>

| Mass on New         | 9.2%                 | ~100%        | 9.2%               |
| Ratio of 2/1        | 2.8                  | 1.5          | ~2                 |

- Big things to notice:
  - Add-one vastly overestimates the fraction of new bigrams
  - Add-0.0000027 vastly underestimates the ratio 2^1/1^*
  - One solution: use held-out data to predict the map of c to c^*

```
P_{dir}(w|w_{-1}, k) = \frac{c(w_{-1}, w) + k\hat{P}(w)}{\left(\sum_{w'} c(w_{-1}, w')\right) + k}
```

```
P_{lin}(w|w_{-1}, \lambda, \lambda', \lambda'') = \lambda\hat{P}(w|w_{-1}, w_{-2}) + \lambda'\hat{P}(w|w_{-1}) + \lambda''\hat{P}(w)
```
Good-Turing Reweighting I

- We’d like to not need held-out data (why?)
- Idea: leave-one-out validation
  - $N_k$: number of types which occur $k$ times in the entire corpus
  - Take each of the $c$ tokens out of corpus in turn
  - $c$ “training” sets of size $c-1$, “held-out” of size 1
  - How many held-out tokens are unseen in training?
    - $N_1$
  - How many held-out tokens are seen $k$ times in training?
    - $(k+1)N_{k+1}$
  - There are $N_k$ words with training count $k$
  - Each should occur with expected count
    - $(k+1)N_{k+1}/N_k$
  - Each should occur with probability:
    - $(k+1)N_{k+1}/(cN_k)$

<table>
<thead>
<tr>
<th>&quot;Training&quot;</th>
<th>&quot;Held-Out&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1$</td>
<td>$N_0$</td>
</tr>
<tr>
<td>$2N_2$</td>
<td>$N_1$</td>
</tr>
<tr>
<td>$3N_3$</td>
<td>$N_2$</td>
</tr>
<tr>
<td>$...$</td>
<td>$...$</td>
</tr>
<tr>
<td>$3511 N_{3511}$</td>
<td>$N_{3510}$</td>
</tr>
<tr>
<td>$4417 N_{4417}$</td>
<td>$N_{4416}$</td>
</tr>
</tbody>
</table>

Good-Turing Reweighting II

- Problem: what about “the”? (say $k=4417$)
  - For small $k$, $N_k > N_{k+1}$
  - For large $k$, too jumpy, zeros wreck estimates
  - Simple Good-Turing [Gale and Sampson]: replace empirical $N_k$ with a best-fit power law once count counts get unreliable
Good-Turing Reweighting III

- Hypothesis: counts of $k$ should be $k^* = (k+1)N_{k+1}/N_k$

<table>
<thead>
<tr>
<th>Count in 22M Words</th>
<th>Actual $c^*$ (Next 22M)</th>
<th>GT’s $c^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.448</td>
<td>0.446</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>1.26</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>2.24</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>3.24</td>
</tr>
<tr>
<td>Mass on New</td>
<td>9.2%</td>
<td>9.2%</td>
</tr>
</tbody>
</table>

- Katz Smoothing
  - Use GT discounted bigram counts (roughly – Katz left large counts alone)
  - Whatever mass is left goes to empirical unigram

$$P_{katz}(w|w') = \frac{c^*(w', w)}{c(w')} + \alpha(w')\bar{P}(w)$$

Kneser-Ney: Discounting

- Kneser-Ney smoothing: very successful but slightly ad hoc estimator
- Idea: observed n-grams occur more in training than they will later:

<table>
<thead>
<tr>
<th>Count in 22M Words</th>
<th>Avg in Next 22M</th>
<th>Good-Turing $c^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.448</td>
<td>0.446</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>1.26</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>2.24</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>3.24</td>
</tr>
</tbody>
</table>

- Absolute Discounting
  - Save ourselves some time and just subtract 0.75 (or some $d$)
  - Maybe have a separate value of $d$ for very low counts

$$P_{ad}(w|w') = \frac{c(w', w) - d}{c(w')} + \alpha(w')\bar{P}(w)$$
Kneser-Ney: Continuation

- Something’s been very broken all this time
  - Shannon game: There was an unexpected ____?
    - delay?
    - Francisco?
  - “Francisco” is more common than “delay”
  - … but “Francisco” always follows “San”

- Solution: Kneser-Ney smoothing
  - In the back-off model, we don’t want the probability of w as a unigram
  - Instead, want the probability that w is allowed in this novel context
  - For each word, count the number of bigram types it completes

\[ P_c(w) \propto |w' : c(w', w) > 0| \]

Kneser-Ney

- Kneser-Ney smoothing combines these two ideas
  - Absolute discounting
    \[ P(w|w') = \frac{c(w', w) - d}{c(w')} + \alpha(w')P'(w) \]
  - Lower order models take a special form
    \[ P_c(w) \propto |w' : c(w', w) > 0| \]

- KN smoothing repeatedly proven effective
  - But we’ve never been quite sure why
  - And therefore never known how to make it better
  - [Teh, 2006] shows KN smoothing is a kind of approximate inference in a hierarchical Pitman-Yor process (and better approximations are superior to basic KN)
**What Actually Works?**

- **Trigrams:**
  - Unigrams, bigrams too little context
  - Trigrams much better (when there’s enough data)
  - 4-, 5-grams often not worth the cost (which is more than it seems, due to how speech recognizers are constructed)
- **Note:** for MT, 5+ often used!
- **Good-Turing-like methods for count adjustment**
  - Absolute discounting, Good-Turing, held-out estimation, Witten-Bell
- **Kneser-Ney equalization for lower-order models**
- See [Chen+Goodman] reading for tons of graphs!

![Graphs from Joshua Goodman](image)

**Data >> Method?**

- Having more data is better…
- … but so is using a better model
- Another issue: N > 3 has huge costs in speech recognizers
Beyond N-Gram LMs

- Lots of ideas we won’t have time to discuss:
  - Caching models: recent words more likely to appear again
  - Trigger models: recent words trigger other words
  - Topic models

- A few recent ideas
  - Syntactic models: use tree models to capture long-distance syntactic effects [Chelba and Jelinek, 98]
  - Discriminative models: set n-gram weights to improve final task accuracy rather than fit training set density [Roark, 05, for ASR; Liang et. al., 06, for MT]
  - Structural zeros: some n-grams are syntactically forbidden, keep estimates at zero [Mohri and Roark, 06]
  - Bayesian document and IR models [Daume 06]