Outline of Today’s Lecture

- Language Modeling
- Large Vocabulary Issues
books & sources

- Huang, Acero, Hon, “Spoken Language Processing”,
- Deller et. al. “Discrete-time Processing of speech signals”
- The HTK Book, Cambridge.

Scribe Assignments

- 1/3: Day 1: Jeff Bilmes
- 1/5: Day 2: Xin Lei
- 1/10: Day 3: Tom Mackel
- 1/12: Day 4: Scott Philips
- 1/19: Day 5: Kevin Duh
- 1/24: Day 6: Andrei Alexandrescu
- 1/31: Day 8: Amar Subramanya
- 2/2: Day 9: Andrei Alexandrescu
- 2/7: Day 10: Dustin Lennon
- 2/9: Day 11: Jeremy Holleman
- 2/14: Day 12: Mei-yang
- 2/16: Day 13: Amar Subramanya
- 2/23: Day 14: Kevin Duh
- 2/28: Day 15: Travis Wilkins
- 3/2: Day 16: Tom Mackel
- 3/7: Day 17: Jeremy Holleman
- 3/9: Day 18: John Cutter
Final Projects

• There will be a deadline every week for the rest of the quarter.
• S/NS people must do a final project.
• Due dates:
  – Friday, Feb 18th: 1-page email abstract due
  – Friday, Feb 25th: 2-page progress report due
  – Friday, March 4th: 2-page progress report due
  – Friday, March 11th: Final 4-page writeup due
  – Tuesday, March 15th: Final Projects
• Stop by office hours (Tuesdays, 2:30-4:00pm) if you are unsure of what to do, or send me email (I am available most early evenings to meet).

Announcements

• READING: Chapter 9 (speech issues) and Chapter 11 (Language Modeling)
• Read chapters 12/13
• Short not-found Google queries:
  – perplexity banana
  – fish xenon
  – avocado transistor
  – snoopy meows
  – zit axe(ax)
  – isomorphic joystick (homomorphic)
  – radius flu
Perplexity

- Let $Q()$ be a language model obtained in some way independent of a text string $w_{1:N}$
- $Q()$ is valid distribution, normalizes to 1.
  \[ \sum_{w_{1:N}} Q(w_{1:N}) = 1, \quad \forall N \]
- Perplexity of $w_{1:N}$ relative to $Q()$ is:
  \[ \text{ppl}_{Q}(w_{1:N}) \triangleq Q(w_{1:N})^{-1/N} \]
- So, perplexity is inverse geometric mean of the probabilities (we’ll see why in a sec.)

Perplexity & Entropy

- Entropy:
  \[ H(p) = -\sum p_i \log p_i \]
- Cross Entropy:
  \[ H_q(p) = -\sum q_i \log q_i \geq H(p) \]
- by KL-divergence
- For stationary ergodic sources:
  \[ H(W) = \lim_{N \to \infty} -\frac{1}{N} \log Q(w_{1:N}), \quad w_{1:N} \sim p(w_{1:N}) \]
- So perplexity is $2^H$, where $H$ is the entropy rate of the word stochastic process
- if $H$ is large, language is considered difficult.
- ppl is as difficult as a language with $|V|=\text{ppl}$, and with uniform distribution over words.
Language Modeling

- Wall-street journal 1.1M word corpus, \(|V| = 45.4k\): (train .89M tokens, test .22M)

<table>
<thead>
<tr>
<th>Probability Form</th>
<th>Model</th>
<th>ppl</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p(w_i))</td>
<td>unigram</td>
<td>1386</td>
</tr>
<tr>
<td>(p(w_i</td>
<td>w_{i-1}))</td>
<td>bigram</td>
</tr>
<tr>
<td>(p(w_i</td>
<td>w_{i-1},w_{i-2}))</td>
<td>trigram</td>
</tr>
<tr>
<td>(p(w_i</td>
<td>w_{i-1},w_{i-2},w_{i-3}))</td>
<td>4-gram</td>
</tr>
<tr>
<td>(p(w_i</td>
<td>w_{i-1},w_{i-2},w_{i-3},w_{i-4}))</td>
<td>5-gram</td>
</tr>
<tr>
<td>(p(w_i</td>
<td>w_{i-1},w_{i-2},w_{i-3},w_{i-4},w_{i-5}))</td>
<td>6-gram</td>
</tr>
<tr>
<td>(p(w_i</td>
<td>w_{i-1}))</td>
<td>d2-bigram</td>
</tr>
<tr>
<td>(p(w_i</td>
<td>w_{i-3}))</td>
<td>d3-bigram</td>
</tr>
<tr>
<td>(p(w_i</td>
<td>w_{i-4}))</td>
<td>d4-bigram</td>
</tr>
<tr>
<td>(p(w_i</td>
<td>w_{i-5}))</td>
<td>d5-bigram</td>
</tr>
<tr>
<td>(p(w_i</td>
<td>w_{i-3},w_{i-4},w_{i-5}))</td>
<td>d(3-5) 4-gram</td>
</tr>
</tbody>
</table>

Language Modeling

- Trigram ML solution:

\[
Q(w_t|w_{t-1},w_{t-2}) = \frac{C(w_t,w_{t-1},w_{t-2})}{C(w_{t-1},w_{t-2})}
\]

- But \(C(w_t,w_{t-1},w_{t-2})\) still sparse, many tri-grams won’t exist in reasonable size training data, underestimated (or zero) probabilities
- Even given sufficient training, \(|V|^3\) table size is large (memory issues)
- There are many solutions (we survey some of them here, others see EE517 next quarter)
- All solutions are some form of “smoothing” (replace zeros with non-zeros in distribution)
Language Modeling

• We will cover:
  – deleted interpolation
  – basics of adaptive language models
  – Language model “backoff” procedures

• Towards deleted interpolation: consider model of form:
\[
Q(w_t|w_{t-1}, w_{t-2}) = \lambda_3 f(w_t|w_{t-1}, w_{t-2}) \\
+ \lambda_2 f(w_t|w_{t-1}) \\
+ \lambda_1 f(w_t)
\]
\[
\lambda_1 + \lambda_2 + \lambda_3 = 1
\]
Note that same \(\lambda\)'s are used for all words in this discussion.

Language Modeling

• What if use ML to determine \(\lambda\) values?
\[
(\lambda_1, \lambda_2, \lambda_3)^* = \arg\max_{\lambda_1, \lambda_2, \lambda_3} \sum_t \log Q(w_t|w_{t-1}, w_{t-2})
\]

• If same training data used for \(\lambda\) and LM training, optimal solution always (why?):
\[
\lambda_1 = \lambda_2 = 0, \quad \lambda_3 = 1
\]

• This gets worse since ideally we would like the \(\lambda\) values to depend on the words (or at least a function of the words)
Language Modeling

• Basic idea:
  – divided training set (kept set & held out set)
  – train $f(w_3|w_2,w_1)$, $f(w_3|w_2)$, $f(w_3)$ on kept set
  – train $\lambda$ on held out set
  – have $\lambda$ depend on words via counts
    • if $C(w_2,w_1)$ is big, choose high weight for $\lambda_3$, else choose high weight for $\lambda_2$
  • Simplify to work with two coefficients at a time:
    \[ p(w_3|w_2,w_1) = \lambda_3' f(w_3|w_2,w_1) + \lambda'_2 (\lambda_2 f(w_3|w_2) + \lambda_1 f(w_3)) \]
    \[ \lambda_3 = \lambda_3' \quad \lambda'_1 + \lambda'_2 = 1 \]
    \[ \lambda_2 = \lambda_4 \lambda_2' \quad \lambda'_3 + \lambda'_4 = 1 \]

Language Modeling

• word-dependent lambda
  \[ p(w_t|h_t) = \lambda(h_t) f(w_t|h_t) + (1 - \lambda(h_t)) f(w_t) \]
  But now there are too many $\lambda$’s. We use $C(h_t)$, depending on count (in held out set), choose different mixture weights.
  \[ C() \in \{0, 1, \ldots, M\} \]
  • Simplify: \[ C() \in \{R_1, R_2, \ldots, R_N\} \in \mathbb{R} \]
    – $N \ll M$
    – $R_1 = [0, \max(R_1)]$
    – $R_i = [\min(R_i), \max(R_i)]$
  • Counts are binned into ranges for simplicity
**Language Modeling**

- **Procedure:**
  - divide data into kept & held-out set
  - compute \( f(w_t|h_t) \) and \( f(w_t) \) from kept set
  - compute counts on held out set \( N(h_t) \)
  - For each range \( R_r, r \in \{1, 2, \ldots, N\} \), find \( \lambda_r \) that maximizes the following:

\[
\sum_{h:C(h) \in R_r} \sum_{w_t} p_H(w_t|h) \log [\lambda_r f(w_t|h) + (1 - \lambda_r) f(w_t|h)]
\]

\[
p_H(w_t|h) = \frac{N(w_t,h)}{N(h)}
\]

- This is cross-entropy minimization (i.e., change \( p_K \) via \( \lambda_r \))

\[
- \sum p_H \log p_K \geq - \sum p_H \log p_H
\]

\[
\min_{\lambda_r} D(p_H||p_K)
\]

**Adaptive Language Modeling**

- **Idea:** LM \( p(w_t|h_t) \) should not stay fixed but should change depending on recent (hypothesized) events.
- **Ideal case:** \( p(w_t|h_t) = 1 \) if \( w_t \) is correct word.
- **Simple approach:** cache-based LMs:

\[
p(w_t|h_t) = \sum_i p_o(w_t|i)p(i|\phi(w_{t-1}), \phi(w_{t-2}))
\]

\[
\phi(w) = \text{word class of word } w
\]

\[
p_o(w_t|i) = \lambda_c p_{\text{cache}}(w_t|i) + (1 - \lambda_c)p_{\text{class}}(w_t|i)
\]

\[
p_{\text{class}}(w_t|i) = \frac{C(w_t,i)}{C(i)}
\]

\[
p_{\text{cache}}(w_t|i) = \text{something that gives high probability for words recently seen in } i^{th} \text{ cache}
\]
Adaptive Language Modeling

• Example:
  – $i^{th}$ cache consists of set of words $S_i$
  – Let $k_t$ be a K-length history at time $t$

$$p_{\text{cache}}(w_t|i) = \left| \{ w : w \in k_t \cap S_i \} \right| / K$$

• But other strategies exist (next quarter, EE517)

Backoff Language Modeling

• Idea: Form $p(w_t|h_t)$ from lower-order distributions when count falls below some threshold (but make sure to keep everything normalized)

• If the counts exceed a threshold use discounted ML distribution, otherwise “backoff” to weighted lower-order distribution.

• Do this recursively

$$p_{\text{DO}}(w_t|w_{t-1}, w_{t-2}) =
\begin{cases} 
  d_c(w_t, w_{t-1}, w_{t-2}) \frac{c(w_t, w_{t-1}, w_{t-2})}{c(w_t, w_{t-1})} & \text{if } c(w_t, w_{t-1}, w_{t-2}) > \tau \\
  \alpha(w_{t-1}, w_{t-2}) p_{\text{BO}}(w_t|w_{t-1}) & \text{else}
\end{cases}$$

$\alpha$ Backoff weight (BOW) $\tau$ Discount Coefficient
Backoff Language Modeling

• Discount coefficient:
  – if \( d_c(w) = 1 \) for all \( w \), then we get ML solution since no mass gets distributed to lower-order models
  – \( d() \) determines amount of mass to “steal” away from higher-order ML solution and give to BO lower-order model.
  – There are *many* different forms of \( d() \). Much research in LM has been on finding appropriate form (see Chen paper).
  – “smoothing algorithms” is equivalent to finding appropriate form of \( d() \). This equation “smoothes” out ML solution so that no zeros are in distribution.
  – BO equation doesn’t require full table even though it does not give 0 probability to anything, much less memory.

Backoff Language Modeling

• Ex: additive smoothing:

\[
d_{c}(w_{t},w_{t-1},w_{t-2}) p_{\text{ML}}(w_{t}|w_{t-1},w_{t-2}) = \frac{C(w_{t},w_{t-1},w_{t-2}) + \delta}{C(w_{t-1},w_{t-2}) + |V|\delta}
\]

• if \( \delta=0 \), ML solution (no backoff)
• As \( \delta \) gets bigger, approach uniform distribution. Ensures no zeros exist, but not a good solution.
• Better: Good/Turing smoothing or Katz smoothing
  – Any n-gram that occurs \( r \)-times, pretend that it occurs \( r^{*} \) times, where:
    \[
    r^{*} = (r + 1)^{n_r+1} / n_r
    \]
  \[
  n_r = \{ (w_{t},w_{t-1},w_{t-2}): c(w_{t},w_{t-1},w_{t-2}) = r \}
  \]
  = number of n-grams that have occurred exactly \( r \) times in training data
Backoff Language Modeling

• Good/Turing Smoothing

\[ r^* = (r + 1)^{\frac{n_r + 1}{n_r}} \]

• So if something has occurred 0 times, we will pretend it has occurred prop. to ratio of meta counts.

• We do this up to a limit, to get discount coeff:

\[ d_r = 1 \quad r \geq k \quad \text{gtmax, upper GT discounting cutoff} \]

\[ d_r = \frac{r^*/r - \frac{(k+1)n_{k+1}}{n_k}}{1 - \frac{(k+1)n_{k+1}}{n_1}} \approx \frac{r^*/r}{r} \]

where \( r = C(w_t, w_{t-1}, w_{t-2}) \)

Backoff Language Modeling

• Recursion:

\[ p_{bo}(w_{t}|w_{t-1}, w_{t-2}) = \begin{cases} 
    d_c(w_t, w_{t-1}, w_{t-2}) \frac{c(w_{t-1}, w_{t-2})}{c(w_t, w_{t-1})} & \text{if } c(w_t, w_{t-1}, w_{t-2}) > \tau \\
    \alpha(w_{t-1}, w_{t-2}) p_{bo}(w_{t}|w_{t-1}) & \text{else}
\end{cases} \]

• Recursively apply same formula.
  – If we start with unigram, backoff to uniform distribution.
  – If we start with bi-gram backoff to BO unigram, etc.
Backoff Language Modeling

- Normalization:
  \[ \sum_{w_t} p_{BO}(w_t|w_{t-1}, w_{t-2}) = 1 \quad \forall w_{t-1}, w_{t-2} \]

- The backoff weight (BOW, or \( \alpha \)) ensures that we have a valid normalized distribution.
  \[ \alpha(w_{t-2}, w_{t-1}) = \frac{1 - \sum_{w:c>\tau} d_c p_{ML}}{\sum_{w:c\leq\tau} p_{BO} p(w_t|w_{t-1})} \]
  \[ = \frac{1 - \sum_{w:c>\tau} d_c p_{ML}}{1 - \sum_{w:c>\tau} p_{BO} p(w_t|w_{t-1})} \]

- This is crucial to get right, otherwise perplexity will be wrong.

Towards Large Vocabulary

- Recall fundamental equation:
  \[ W^* = \arg \max_W p(X|W)p(W) \]

- For isolated phrase recognition:
  - each phrase has clearly defined begin/end
  - ok for command & control applications (car, cell phone)
  - Can do “whole word” models (have a separate HMM for each phrase, don’t need to model phones within words)
    - so no part of \( p(x|"hat") \) is shared with \( p(x|"cat") \), so HMM states have no real meaning other than part of a word in some cntx.
  - Possible to do separate EM training on each phrase
  - One of N possible phrases must be spoken
  - Fundamental equation and Bayes decision theory applies perfectly well.
Towards Large Vocabulary

- Many forms of speech (increasing order of difficulty):
  - isolated word/phrase
  - read speech
  - “lecture” speech
  - conversational speech – between two acquaintances
  - meeting speech – between people in meetings
  - group speech – between members of an informal group
  - casual conversational speech – between close friends or family members
- For anything but isolated phrase, we need different approach since W would contain all possible sentences of all words.
- Key thing: output is structured, need to account for (and ideally exploit/take advantage of) this in some way (rather than ignore it).

Towards Large Vocabulary

- Different speech is different
- Coarticulation
  - neighboring speech sounds influence each other (change in acoustic-phonetic context of a speech segment (phone, syllable, etc.) due to anticipation or preservation of an adjacent segment
  - Multi-dimensional articulator space: each phone has target, as speech becomes more fluent, we only approach rather than hit each target.

From Lecture 2: coarticulation example: “In Paris” vs. “In the house”, the ‘n’ is anticipating the “th”, so quite a different sound.
Towards Large Vocabulary

- Even word boundaries are difficult to define
- General Bayes decision theory is less applicable
  - $W$ is too many classes (e.g., 100k words, $N$ words, $N$ unbounded)
  - exhaustive examination of all possible sentences clearly infeasible
- Instead:
  - explore multiple word hypotheses simultaneously
  - try to discard as soon as possible (or even don’t consider) all hypotheses that are improbable
  - Like a greedy strategy, but instead we remove what doesn’t look good: Could be called a “renouncing” or an “abjuring” algorithm
  - Can also be seen as a “search” procedure (like in AI).
- Obviously, can’t have a separate model $p(X|M)$ for each $M$, so must have submodels (for subwords) and share between different words

Towards Large Vocabulary

- Submodels:
  - in HMM, There are hidden variables in $p(X|C)$ model, use these as shared subword models
  - Words are decomposed into subword units, such as:
    - syllables
    - phones
    - subphones (e.g., 1/3 of a phone)
    - a phone in context (a phone preceded and followed by other phones)
    - a portion thereof
  - Most common: context-dependent phones
    - 3-state triphones (each unit is a phone in the context of a preceding and following phone, and we use 3 HMM states for the triphone).
    - Some labs (IBM) uses 5 phones of context on left & right
    - Tradeoff: more means more accurate, but less well estimated.
    - Other options: parallel streams of hidden semi-asynchronous articulatory states (research topic)
Towards Large Vocabulary

• Sharing: (mostly we share acoustic units)
  – subword units must be shared (words share phones, phones share states)
  – But sharing doesn’t mean we lose track of where we are in the word. Example, “Banana” has (roughly) three /ae/ phones.

Even though same phone is used in word, different positions act differently (e.g., last /ae/ ends the word, but first /ae/ doesn’t).

Tri-phones

• $P(x|q_t)$ where $q_t$ is a triphone
• $x-y+z$ notation: phone $y$ with
  – left context of $x$
  – right context of $z$
• Example: “Beat it” : sil b iy t ih t sil
  – sil
  – sil-b+iy
  – b-iy-t
  – iy-t+ih
  – t-ih+t
  – ih-t+sil
  – sil

• Word internal vs. cross-word triphones
  – if tri-phones span word boundaries, problem is much more difficult (as we will see).
Triphones

- what do they capture?
  - more temporal context
  - coarticulatory effects
  - more states in an HMM so make it more powerful
  - sometimes need lots of states: 45 phones, $45^3 = 91k$ triphones, 3 states per phone $273k$ HMM state, not counting the language model. $p(x|q)$ is mixture of $10$ $39$-D Diagonal Covariance Gaussians, $790$ params/Gaussian, $\Rightarrow 215M$ prms.
  - This is too many even on today’s computers.
- Need to reduce this: Solution, observation distribution tying (or just tying)

Sharing/Parameter Tying

- Make it such that $p(x|Q=i) = p(x|Q=j)$ for the right i and j.
- How to do this for: $p(x|a-b+c)$
  - backing off (like in language modeling)
    - a-b+c model backs off to b+c or a-b etc.
  - smoothing (also like in language modeling)
    - $p(x|c-a+t) = a_1f(x|c-a+t) + a_2f(x|c-a) + a_3f(x|a)$
  - clustering (the preferred method)
    - Decision tree clustered tri-phones both bottom up and top down clustering procedures.
    - Two approaches: top down & bottom up
Triphone state tying

- Middle state of each tri-phone might be tied
  - compare with tying from before for duration modeling.
  - But states of each triphone still distinct.

HMM state tying algorithms

- Bottom up (approach used in HTK)
  - start where all contexts are distinct, and have very weak acoustic model (e.g., signal Gaussian component)
  - For tri-phones, cluster beginning, middle, and end state of a tri-phone separately.
  - Need distance measure between Gaussians $p(x|i)$ and $p(x|j)$. Symmetric KL-divergence.

$$d(i, j) = \frac{1}{2} \left( D(i||j) + D(j||i) \right)$$

For diagonal covariance Gaussians
HMM state tying algorithms

• Start with trained model for all contexts
  – x-y+z, 50 phones, $50^3 = 125k$ phones
• Merge together closest Gaussian pair (for merged Gaussian, form approximate Gaussian from mixture)
  – supposedly ok,
    since they’re close anyway.
  
\[ u_{ij} = \frac{1}{2}(\mu_i + \mu_j) \]

\[ \sigma_{ij}^2 = \frac{1}{2} \left[ \sigma_i^2 + \sigma_j^2 + \sigma_i \sigma_j \right] - \mu_{ij} \]

• Stop when sufficient number of clusters, or when $d(i,j) > \tau$, for some $\tau \forall i,j$.
• Ensure trainability by checking probability. Stopping criterion:
  \[ \gamma(i) = \sum_t \gamma_t(i) > \text{threshold} \]

HMM state tying algorithms

• Once clusters are formed, turn single component Gaussians into mixtures by splitting them during EM training (called “mixing up” in HTK)
• Problem: need to have reliable Gaussians to start:
  – with many tri-phones, even Diagonal covariance Gaussians might not be well estimated since so little data
• Solution: Top down clustering using phonetic decision trees
  – we’ve seen this before in mapping from baseform to surface form pronunciation models
  – We build a decision tree using same approach, using a set of phonetic questions as options to build the tree top down
  – example questions:
    • is left phone a nasal?
    • Is right phone a liquid?
    • Is left phone a fricative?
    • etc.
  – At leaf nodes, model for all phones that satisfy each question, and that is used as the Gaussian mixture.
HMM state tying algorithms

• Advantages:
  – will always find some context dependent model (will never “backoff” or smooth to no context)
  – Utilizes expert phonetic knowledge (phonetic questions)
  – Tree can be formed so that leaf nodes have sufficient data
  – more than tri-phone context by asking questions farther out

• But how to construct the tree?
  – before we used entropy minimization at the leaf nodes, here we have not only distribution but also data to worry about. We use likelihood maximization (which is the same as entropy if we have infinite data and accurate models)

\[ H(x|q) = - \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} \log p(x_i|q), \quad x_i \sim p(x|q) \forall i \]

• Key is to form the likelihood (estimate) so that we need not do a complete pass over the data to re-evaluate at each DT split. This is indeed possible (see HTK book)
Large Vocabulary

• Large Vocabulary ASR (LVCSR)
  – subword units are shared (otherwise too many)
  – explore hypotheses in parallel, discarding unlikely hypotheses
  \[ W^* = \arg\max_W p(X|W)p(W) \]

• Viterbi approximation makes this possible:

\[
p(x_{1:T}|W_{1:N}) = \sum_{q_{1:T}} p(x_{1:T}, q_{1:T}|W_{1:N}) \\
\approx \max_{q_{1:T}} p(x_{1:T}, q_{1:T}|W_{1:N})
\]