Recap: smoothing

- Add-one: simple, but steals way too much mass from seen items.
- Add-\(\alpha\): better. Optimize \(\alpha\) on held-out data.
- Good-Turing: also better. Estimate probability of \(n\)-count items using \(n + 1\)-count items.

Remaining problem

- In given corpus, suppose we never observe
  - Scottish beer drinkers
  - Scottish beer eaters
- If we build a trigram model smoothed with Add-\(\alpha\) or G-T, which example has higher probability?

- Previous smoothing methods assign equal probability to all unseen events.
- Better: use information from lower order \(n\)-grams (shorter histories).
  - beer drinkers
  - beer eaters
- Two ways: interpolation and backoff
Interpolation

• Higher and lower order n-gram models have different strengths and weaknesses
  – high-order n-grams are sensitive to more context, but have sparse counts
  – low-order n-grams consider only very limited context, but have robust counts
• Combine them

\[ P_I(w_3|w_1,w_2) = \lambda_1 P_1(w_3) + \lambda_2 P_2(w_3|w_2) + \lambda_3 P_3(w_3|w_1,w_2) \]

Interpolation parameters must sum to 1:

\[ 1 = \sum_{w_3} [\lambda_1 P_1(w_3) + \lambda_2 P_2(w_3|w_2) + \lambda_3 P_3(w_3|w_1,w_2)] = \lambda_1 + \lambda_2 + \lambda_3 \]

• In general, weighted combination of distributions is called a mixture model

Context-specific Interpolation

• We can trust some histories \( w_{i-n+1}, ..., w_{i-1} \) more than others (why?)
• Condition interpolation weights on history: \( \lambda_{w_{i-n+1}, ..., w_{i-1}} \)
• Interpolation parameters (fixed or context-specific) optimized on held-out data

Back-Off

• Trust the highest order language model that contains n-gram

\[ P_{BO}(w_i|w_{i-n+1}, ..., w_{i-1}) = \begin{cases} P^*(w_i|w_{i-n+1}, ..., w_{i-1}) & \text{if count}(w_{i-n+1}, ..., w_{i}) > 0 \\ \alpha(w_{i-n+1}, ..., w_{i-1}) P_{BO}(w_i|w_{i-n+2}, ..., w_{i-1}) & \text{else} \end{cases} \]

• Requires
  – adjusted prediction model \( P^*(w_i|w_{i-n+1}, ..., w_{i-1}) \)
  – backoff weights \( \alpha(w_1, ..., w_{n-1}) \)
Back-Off with Good-Turing Smoothing

- Good Turing smoothing adjusts counts $c$ to discounted counts $c^*$
  \[
  \text{count}^*(w_1, w_2) \leq \text{count}(w_1, w_2)
  \]
- We use $c^*$ for the prediction model (but $0^*$ remains $0$)
  \[
  P^*(w_2|w_1) = \frac{\text{count}^*(w_1, w_2)}{\text{count}(w_1)}
  \]
- This leaves probability mass for the backoff weight
  \[
  \alpha(w_1) = 1 - \sum_{w_2} P^*(w_2|w_1)
  \]

Example

- Suppose only 3 words seen following 'a'.

<table>
<thead>
<tr>
<th></th>
<th>$c$</th>
<th>$P_{ML}$</th>
<th>$c^*$</th>
<th>$P^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p(\text{big}</td>
<td>a)$</td>
<td>3</td>
<td>0.43</td>
<td>2.24</td>
</tr>
<tr>
<td>$p(\text{house}</td>
<td>a)$</td>
<td>3</td>
<td>0.43</td>
<td>2.24</td>
</tr>
<tr>
<td>$p(\text{new}</td>
<td>a)$</td>
<td>1</td>
<td>0.14</td>
<td>0.446</td>
</tr>
</tbody>
</table>

- $1 - (0.32 + 0.32 + 0.06) = 0.30$ is left for back-off $\alpha(a)$
- Note: actual values for $\alpha$ is slightly higher, since the predictions of the lower-order model to seen events at this level are not used (see J&M 4.7.1).

Diversity of Histories

- Consider the word York
  - fairly frequent word in Europarl corpus, occurs 477 times
  - as frequent as foods, indicates and providers
    → in unigram language model: a respectable probability
- However, it almost always directly follows New (473 times)
- Recall: unigram model only used, if the bigram model inconclusive
  - York unlikely second word in unseen bigram
  - in back-off unigram model, York should have low probability

Kneser-Ney Smoothing

- Kneser-Ney smoothing takes diversity of histories into account
- Count of distinct histories for a word
  \[
  N_{1+}(\bullet w_i) = |\{w_{i-1} : c(w_{i-1}, w_i) > 0\}|
  \]
- Recall: maximum likelihood estimation of unigram language model
  \[
  P_{ML}(w) = \frac{c(w_i)}{\sum_{w_i} c(w_i)}
  \]
- In Kneser-Ney smoothing, replace raw counts with count of histories
  \[
  P_{KN}(w_i) = \frac{N_{1+}(\bullet w_i)}{\sum_{w_i} N_{1+}(\bullet w_i)}
  \]
Kneser-Ney (Backoff version)

- Uses fixed discount $D < 1$, and backoff to prob based on history diversity.

\[
P_{BKN}(w_i|w_{i-n+1}, \ldots, w_{i-1}) = \begin{cases} 
C(w_i|w_{i-n+1}, \ldots, w_{i-1}) - D \\ 
\frac{C(w_i|w_{i-n+2}, \ldots, w_{i-1})}{\text{count}_n(w_{i-n+1}, \ldots, w_i)} 
\end{cases}
\]

\[
= \begin{cases} 
\frac{C(w_i|w_{i-n+1}, \ldots, w_{i-1}) - D}{C(w_i|w_{i-n+2}, \ldots, w_{i-1})} \\
\alpha(w_{i-n+1}, \ldots, w_{i-1}) \sum_{w_j} N_{1+}(w_j|w_{i-n+2}, \ldots, w_{i-1}) \\
\text{else}
\end{cases}
\]

Modified Kneser-Ney (Chen and Goodman, 1998)

- Uses interpolation rather than backoff, with recursive formulation:

\[
P_{MKN}(w_i|w_{i-n+1}, \ldots, w_{i-1}) = 
\frac{C(w_i|w_{i-n+1}, \ldots, w_{i-1}) - D}{C(w_i|w_{i-n+2}, \ldots, w_{i-1})} + \beta(w_{i-n+1}, \ldots, w_{i-1}) P_{MKN}(w_i|w_{i-n+2}, \ldots, w_{i-1})
\]

where $\beta$ depends on $D$ and $N_{1+}$ counts

- Uses separate $D$s for counts of 1, 2, and $\geq 3$.

- Best smoothing method until recently.

Word similarity

- Two words with $C(w_1) \gg C(w_2)$
  - salmon
  - swordfish

- Can $P($salmon|$caught two$) tell us something about $P($swordfish|$caught two$)?

- $n$-gram models: no.
Word similarity in language modeling

• Early version: class-based language models (J&M 4.9.2)
  – Define classes $c$ of words, by hand or automatically
  – $P_{CL}(w_i|w_{i-1}) = P(c_i|c_{i-1})P(w_i|c_i)$ (an HMM)

• Recent version: distributed language models
  – Current models have better perplexity than MKN.
  – Ongoing research to make them more efficient.
  – Examples: Recursive Neural Network LM (Mikolov et al., 2010), Log Bilinear LM (Mnih and Hinton, 2007) and extensions.

Distributed word representations

• Each word represented as high-dimensional vector (50-500 dims)
  E.g., salmon is $[0.1, 2.3, 0.6, -4.7, ...]$

• Similar words represented by similar vectors
  E.g., swordfish is $[0.3, 2.2, 1.2, -3.6, ...]$

Which words are similar?

• Appear in similar set of contexts
• Have similar probabilities across those contexts
• Modulo overall differences in uigram frequency

Which vectors are similar?

• Each vector is a point in high-dimensional space
• 2-dimensional example:
Distance measures

- Cosine distance often better for high-dim spaces
- Learned distance functions also possible

Training the model

- n-gram LM: collect counts, maybe optimize some parameters
  - (Relatively) quick, especially these days (minutes-hours)
- Distributed LM: learn the representation for each word
  - Solved with machine learning methods (e.g., neural networks)
  - Can be extremely time-consuming (hours-days)

Using the model

Want to compute $P(w_1 \ldots w_n)$ for a new sequence.

- n-gram LM: again, relatively quick
- Distributed LM: often prohibitively slow for real applications
- An active area of research for distributed LMs

Other Topics in Language Modeling

Many active research areas in language modeling:

- Factored/morpheme-based language models: back off to word stems, part-of-speech tags, and/or other morphemes in word
- Syntactic language models: using parse trees
- Domain adaptation: when only a small domain-specific corpus is available
- Time efficiency and space efficiency are both key issues (esp on mobile devices!)
Announcements

- Asgn 1 will hit the website this evening. Due two weeks from today, 3pm.
- Office hours: Thursdays 4-5pm.
  - Whoever is lecturing that day will stay in DHT after class to answer questions.
  - If you want to talk to the other lecturer (because your questions pertain to the part of the course they taught) please email ahead.

References

