Recap: Language Models

- Why are language models useful?
- Why did I show samples of generated text?
- What are the main challenges in building n-gram language models?

Smoothing

- We often want to make estimates from sparse statistics:
  \[ P(w \mid \text{denied the}) = \frac{3 + \delta}{7 + \delta} \]
  \[ P(w \mid \text{allegations}) = \frac{2 + \delta}{7 + \delta} \]
  \[ P(w \mid \text{attack}) = \frac{1 + \delta}{7 + \delta} \]
  \[ P(w \mid \text{man}) = \frac{1 + \delta}{7 + \delta} \]
  \[ P(w \mid \text{outcome}) = \frac{1 + \delta}{7 + \delta} \]
- Smoothing flattens spiky distributions so they generalize better
- Vocabulary Size
  - Key issue for language models: open or closed vocabulary?
  - When would you want an open vocabulary?
  - When would you want a closed vocabulary?
  - How to set the vocabulary size \( V \)?
    - By external factors (e.g. speech recognizers)
    - Using statistical estimates?
    - Difference between estimating unknown token rate and probability of a given unknown word
  - For the homework:
    - OK to assume there is only one unknown word type UNK
    - UNK be quite common in new text!
    - UNK stands for all unknown word type

Smoothing: Add-One, Etc.

- One class of smoothing functions:
  - Add-one / delta: assumes a uniform prior
    \[ P_{\text{Add-}\delta}(w \mid w_{-1}) = \frac{c(w, w_{-1}) + \delta}{c(w_{-1}) + \delta} \]
  - Better to assume a unigram prior
    \[ P_{\text{Unigram}}(w \mid w_{-1}) = \frac{c(w, w_{-1}) + \delta \hat{P}(w)}{c(w_{-1}) + \delta} \]

Held-Out Data

- Important tool for getting models to generalize:
  - When we have a small number of parameters that control the degree of smoothing, we set them to maximize the (log-)likelihood of held-out data
  \[ LL(w_1, \ldots, w_n \mid M(\lambda_1, \ldots, \lambda_d)) = \sum \log P_{M(\lambda_1, \ldots, \lambda_d)}(w_i \mid w_{-i}) \]
  - Can use any optimization technique (line search or EM usually easiest)

Examples:

\[
\begin{align*}
P_{\text{Add-}\delta}(w \mid w_{-1}) &= \lambda_1 \hat{P}(w \mid w_{-1}) + \lambda_2 \hat{P}(w) \\
P_{\text{Unigram}}(w \mid w_{-1}) &= \frac{c(w, w_{-1}) + \delta \hat{P}(w)}{c(w_{-1}) + \delta} \\
P_{\text{Unigram}}(w) &= \frac{c(w) + \delta \hat{P}(w)}{c + \delta} \\
\end{align*}
\]
Held-Out Reweighting

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

Big things to notice:
- Add-one vastly overestimates the fraction of new bigrams
- Add-0.0000027 still underestimates the ratio \(2^*/1^*\)

Issue: which distribution are we smoothing?
One solution: use held-out data to predict the map of \(c\) to \(c^*\)

Good-Turing Reweighting I

We’d like to not need held-out data (why?)
Idea: leave-one-out validation
- Take each of the \(c\) training words out in turn
- \(c\) training sets of size \(c-1\), held-out of size 1
- What fraction of held-out words are unseen in training?
  - \(N_{c/c}\)
- What fraction of held-out words are seen \(k\) times in training?
  - \((k+1)N_{k+1}/c\)
- So in the future we expect \((k+1)N_{k+1}/c\) of the words to be those with training count \(k\)
  - Each should occur with probability:
    - \((k+1)N_{k+1}/c\)
    - \(N_k\) words with training count \(k\)
  - There are \(N_k\) words with training count \(k\)
  - Mass on New
    - \((k+1)N_{k+1}/c\)
    - \(N_k\)
    - \((k+1)N_{k+1}/c\)

Problem: what about “the”? (say \(c=4417\))
- 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

Kneser-Ney Smoothing I

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 unigram probability of \(w\)
- Instead, probability given that we are observing a novel continuation
- Every bigram type was a novel continuation the first time it was seen

Kneser-Ney Smoothing II

<table>
<thead>
<tr>
<th>Count in 22M Words</th>
<th>Actual (c^*) (Next 22M)</th>
<th>Add (c^*) c (c)</th>
<th>Add-0.0000027 (c^*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.446</td>
<td>0.446</td>
<td>0.446</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>2.24</td>
<td>2.24</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>3.23</td>
<td>3.23</td>
</tr>
<tr>
<td>5</td>
<td>4.21</td>
<td>4.21</td>
<td>4.21</td>
</tr>
</tbody>
</table>

Problem: what about “the”? (say \(c=4417\))
- 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

Kneser-Ney Smoothing III

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

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

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
Higher-Order Models

- Trigrams:
  - Unigrams, bigrams too little context
  - Trigrams much better (when there’s enough data)
  - 4-, 5-grams usually not worth the cost (which is more than it seems, due to how speech recognizers are constructed)
- 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!

What Actually Works?

- Having more data is always good…
  - … but so is picking a better smoothing mechanism!
  - N > 3 often not worth the cost (greater than you’d think)

Beyond N-Gram LMs

- Caching Models
  - Recent words more likely to appear again
  \[ P_{\text{next}}(w | \text{history}) = \lambda P(w | \text{w}_{-2},_{-1}) + (1 - \lambda) \frac{\lambda P(w \in \text{history})}{|\text{history}|} \]
  - Can be disastrous in practice for speech (why?)
- Skipping Models
  \[ P_{\text{next}}(w | \text{w}_{-2},_{-1}) = \lambda \hat{P}(w | \text{w}_{-2},_{-1}) + \lambda \hat{P}(w | \text{w}_{-1}) + \lambda \hat{P}(w | \text{w}_{-2}) \]
- Clustering Models: condition on word classes when words are too sparse
- Trigger Models: condition on bag of history words (e.g., maxent)
- Structured Models: use parse structure (we’ll see these later)

For Next Time

- Readings: M+S 6, J+M 6, Chen & Goodman (on web page)
- Next up: Text Categorization, Naïve-Bayes