Announcements

- Project #1
  - Due: today

- Reading Report #3
  - Due: today

Objectives

- Continue gaining insight into smoothing NLP models.
- Improve on Held-out Estimation with Good-Turing smoothing
- Fix problems with Good-Turing smoothing

Questions

- How many species are in the ocean?
- How many words did Shakespeare know?
- How many unseen word types are there?

Context

Held-Out Estimation

- From real bigram counts in 22M words of AP newswire data [Church and Gale 95]:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( r \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td>( r^2 \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td>( r^3 \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td>( r^4 \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td>( r^5 \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
</tbody>
</table>

- Why do we care about \( r \)? "the rich"
- Why is \( r \) lower on average?
Held-Out Estimation

- From real bigram counts in 22M words of AP newswire data [Church and Gale 91]:

<table>
<thead>
<tr>
<th>Trial</th>
<th>Held-out Count</th>
<th>Actual Count</th>
<th>Add-one Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.88</td>
<td>5.00875</td>
<td>2.0000075</td>
</tr>
<tr>
<td>2</td>
<td>2.28</td>
<td>0.00415</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>0.00549</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2.22</td>
<td>0.00668</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.21</td>
<td>0.00822</td>
<td></td>
</tr>
</tbody>
</table>

- Big things to notice:
  - Add-one vastly overestimates the fraction of new bigrams
  - Add-0.000027 still underestimates the ratio $r^*/r^*$

- Can we get the properties of held-out estimation with less data?

Activity

Given a large data set,

1. Compute the frequencies (done)
2. Compute the frequencies-of-frequencies

Activity (cont.)

3. Remove the occurrences of “oaks” one-by-one.
   - How many does this leave in training each time?
4. Remove the occurrences of “malaysian” and “riots” one-by-one.
   - How many (total) “malaysians” and “riots” are left each time?
5. “gibbons”, “aspin”, “chats”, “allusions”?
6. Generalize:
   - a) How many held-out tokens (trials) are there?
   - b) What fraction of held-out tokens (across all trials) are seen k times in training? Let’s see ...
Leave-one-out

Take the first token out and use it as "validation" to count how many times it occurs in training.

Replace it and use the next token as "validation".

…and the next…

…and the next…

…and the next…

…and the next…

…and the next…
Leave-one-out

...and the next...

Leave-one-out

...and the next...

Leave-one-out

...and so on...

Leave-one-out

Across all trials (in the mental exercise)...

How many held-out tokens occurred 0 times in training? 10 / 47902

How many held-out tokens occurred 1 time in training? 8 / 47902

Across all trials (in the mental exercise)...

How many held-out tokens occurred 0 times in training? 10 / 47902

How many held-out tokens occurred 1 time in training? 8 / 47902

How many held-out tokens occurred 2 times in training? 6 / 47902
Leave-one-out

Across all trials in the mental exercise, what fraction of held-out tokens occurred $k$ times in training? Let's break it down:

- How many word types for count $k$?
  \[ N_{k+1} \]
- How many tokens per type? $a = 1$?
- Total number of held-out tokens occurring $k$ times in training? $(k+1) \times N_{k+1}$
- As a fraction of trials (N)? $(k+1) \times N_{k+1} / N$

Conclusion from the mental exercise

- Now suppose we are given a fresh data set
  - split into a training portion and a test portion
- Based on our conclusion, all word types that occur $k$ times in training we expect to see
  \[(k + 1) \times N_{k+1} / N\]
  percent of the time in future text (i.e., the test set).
- How much of this percentage (mass) should be “assigned” to each of those types (occurring $k$ times)?
  - How many types are there that occur $k$ times in training?
    \[ N_k \]
  - \[ (k+1) \times N_{k+1} / N \]

What about $r^*$?

\[
p^* = \frac{(k + 1)N_{k+1}}{N} / N_k
\]
\[
= \frac{(k + 1)N_{k+1}}{N_k}
\]
\[
= \frac{r^*}{N}
\]

Expected Count $r^* = \frac{(k + 1)N_{k+1}}{N_k}$
Good-Turing: Unseen events

- How many held-out tokens are seen zero times in training?
  - \( N_0^* = N_0 \)
- What fraction?
  - \( N_1 / N \)
  - This is \( P_{GT}(\text{unseen event}) \)
  - For unigrams that’s simply \( P_{GT}(\text{OOV}) \)

Good-Turing Summary

\[
p^* = P_{GT}(w) = \begin{cases} 
(k + 1)N_{w+k} / N & \text{if } w \text{ seen in training } k > 0 \text{ times} \\
N_k / N & \text{otherwise } (k=0)
\end{cases}
\]

Generalizes to \( P_{GT}(\text{bigram}), P_{GT}(\text{trigram}) \), etc.

Did it Work?

- Hypothesis: smoothed counts should be \( r_{GT} = (r+1)N_{w+r}/N_r \)

<table>
<thead>
<tr>
<th>(MLE)</th>
<th>Held out</th>
<th>( r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Held out</td>
<td>9.2% &amp; 9.2%</td>
<td></td>
</tr>
<tr>
<td>1.25 &amp; 1.26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.24 &amp; 2.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.24 &amp; 3.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.12 &amp; 4.12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Almost the same as held-out estimation without requiring extra data!

Implementation

- Recommended reading: Bill Gale explains how to implement
  - See the paper titled “Simple Good-Turing”
  - Will be provided as optional reading in Learning Suite
  - Some details follow ...

Plots

Start with Zipf:
Problem #1 (for Large k)

\[ r^* = ((k + 1) \cdot N_{k+1}/N_k) \]

Your question: what about the 0s?

Solution #1: Fit a Power-Law

- Simple Good-Turing [Gale and Sampson]:
  - Fit a monotone decreasing, non-zero function
    - \[ N_k = ab^k \quad [0 < b < 1] \]
    - A “power-law” curve
    - Use the value at the fitted function instead of \( N_k \) for unreliable \( k \)
  - Common choice: beyond some \( k \) (the first one that is non-decreasing or zero)

\[ \begin{align*}
\text{N}_1 & \quad \text{N}_2 \\
\text{N}_1 & \quad \text{N}_2
\end{align*} \]

\[ \text{Solution: Averaging Transform (Gale)} \]

For \( k \) such that \( N_k \neq 0 \),

\[ Z_k = \frac{2N_k}{k_{\text{next}} - k_{\text{prev}}} \]

\[ \begin{align*}
\text{y} = -0.8763x + 6.1212 \\
\text{log}(N_k) & \quad \text{log}(k)
\end{align*} \]

Problem #2 (for Large k)

- Account for Zeros
  - Averaging Transform (Gale)

\[ \begin{align*}
\text{y} &= -1.8064x + 10.456 \\
\text{log}(Z_k) & \quad k
\end{align*} \]

\[ \begin{align*}
\text{Problem #2} & \quad \text{log}(Z_k)
\end{align*} \]

\[ \begin{align*}
\text{Problem #2} & \quad \text{log}(Z_k)
\end{align*} \]
Size of the set of unknowns

- $U = \#\text{unknown} = \#\text{possible} - \#\text{seen}$
- Let $V = \{\text{seen word types}\} \cup \{00V\}$
- Let $T = V \cup \{\text{stop}\}$
- $\#\text{possible} = \text{sum of the following:}$
  - $|T|$
  - $|V| \cdot |T|$
  - $|V^2| \cdot |T|$
  - $|V^3| \cdot |T|$

Solutions

- Actually applied in this order:
  1. Averaging transform
     $$Z_k = \frac{2N_k}{k_{\text{new}} - k_{\text{prev}}}$$
  2. Power-law fit: $\alpha, \beta$ such that $Z_k = \alpha k^\beta$
  3. Use power-law beyond some threshold $K$
     $$S_i = \begin{cases} U & k = 0 \\ N_i & 0 < k \leq K \\ \frac{\alpha k^\beta}{K} & K < k \end{cases}$$
  4. Now use Good-Turing!
     $$G(k) = G_i = \frac{(k+1)S_{i+1}}{S_i}$$

Note: since $N_i = U, G_i = N_i / U$

From Re-estimated Counts to Probabilities

$$P_{GT}(\text{hist, w}) = \frac{G(\text{hist, w})}{N}$$

- Where $C(\text{hist, w})$ is the number of times the $n$-gram $\langle \text{hist, w} \rangle$ appeared in training.
- $G()$ is the function we just defined!

The Denominator

$$P_{GT}(w | \text{hist}) = \frac{G(C(\text{hist, w}))}{\sum_{v \in T} G(C(\text{hist, v}))} = \frac{G(C(\text{hist, w}))}{M(\text{hist})}$$

- Pre-compute marginal counts $M(\text{hist})$ for denominator
- Problem: for each history, sum over $v \in T$
  - Slow! $O(|T| \cdot N)$
- Instead, compute via two cases:
  - Case 1: For any "unseen" hist
    - Assume uniform distribution of reserved mass
    - $P(w | \text{hist}) = \frac{1}{M}$
  - Case 2: For any seen hist
    - Compute $\sum_{v \in T} G(C(\text{hist, v}))$
The Denominator (2)

\[ P_{GT}(w | hist) = \frac{G(C(hist, w))}{\sum_{v \in \textit{seen}} G(C(hist, v))} \]

- Case 2: For any given “seen” hist:
  \[ M(hist) = \sum_{v \in \textit{seen}} G(C(hist, v)) + \sum_{(v', v'') \in \textit{seen}} G(C(hist, v')) \]
  - Assume uniform distribution of reserved mass
  \[ M(hist) = \{ \text{hist} | ((v' < \text{hist}, v' > \text{seen}) \} \]
  - Conclusion: we only have to iterate over seen n-grams!

Summary: “Simple” Good-Turing Training

1. Count n-grams in training data and store counts: \( C(hist, w) \)
2. Compute and store count-of-counts, \( N_k \)
3. Calculate \( Z_k \) via the averaging transform (for all \( k \) where \( N_k \neq 0 \))
4. Fit \( \{ \text{counts}, Z_k \} \) pairs (where \( N_k = 0 \)) to a power law function with coefficient \( \alpha \) and exponent \( \beta \)
   - Store \( \alpha \) and discard \( \beta \) and \( Z_k \)
5. Compute and store the threshold \( K \) after which the power law function will be used instead.
   - Recommendation: Pick a \( K \) such that for no \( k \leq K, N_k = 0 \)
6. Compute the number of unseen n-grams: \( N_\text{unseen} \)
7. Define \( G_k \):
   \[ G_k = \begin{cases} 1 & \text{for } k = 0 \\ \frac{\log(1 + k)}{\log(1 + 1)} & \text{for } 0 < k < K \end{cases} \]
   \[ G_k = \frac{k + 1}{N_k} \]
   \[ k < K \]
8. Pre-compute the marginal counts \( M(hist) \) for every seen hist.

Final: Use of Good-Turing LM

- When using the LM (generating sentences, computing perplexity, or re-ranking n-best lists), you need \( P(w) \):
  \[ P(w) = \prod_{i} P_{GT}(w_i | hist_i) \]
- We need to consult our local model’s conditionalProbability() method:
  - What if \( hist \) was never seen?
    \[ P(w | hist) = 1/|T| \]
  - Otherwise:
    \[ P_{\text{other hist}} = \frac{G(C(hist, w))}{M(hist)} \]

Next

- Advanced classifiers
- Optional topics:
  - Open vocabulary G-T language models