Smoothing

Smoothing – Definitions

- the N-gram matrix for any given training corpus is **sparse**
  - i.e., not all n-grams will be present
  - MLE produces bad estimates when the counts are small
- **smoothing** = re-evaluating zero and small probabilities, assigning very small probabilities for zero-N-grams
  - if non-occurring N-grams receive small probabilities, the probability mass needs to be redistributed!
- smoothing also sometimes called **discounting**

Types Vs. Tokens

- **token**: single item
- **type**: abstract class of items
- **example**: words in text
  - token: each word
  - type: each different word, i.e., wordform
- **sentence**: the man with the hat
  - tokens: the, man, with, the, hat # of tokens = 5
  - types: the, man, with, hat # of types = 4

Basics of n-grams

n-grams are used to model language, capturing some degree of grammatical properties
- Thus, we can state the probability of a word based on its history:
  1. \( P(w_n|w_1...w_{n-1}) \)
- n-gram probabilities are estimated as follows
  2. \( P(w_n|w_1...w_{n-1}) = \frac{C(w_1...w_n)}{C(w_1...w_{n-1})} \)
- To avoid data sparsity issues, bigrams and trigrams are commonly used
- We can use maximum likelihood estimation (MLE) to obtain basic probabilities:
  3. \( P(w_n|w_1...w_{n-1}) = \frac{C(w_1...w_n)}{C(w_1...w_{n-1})} \)
- But MLE probabilities do nothing to handle unseen data

Add-One Smoothing

**Idea**: pretend that non-existent bigrams are there once
- to make the model more just: assume that for each bigram we add one to the count
- ...turns out not to be a very good estimator
Add-One Smoothing

Laplace’s Law

- unigram probabilities:
  \( N = \text{number of tokens} \)
  \( C(x) = \text{frequency of } x \)
  \( V = \text{vocabulary size; number of types} \)
- standard probability for word \( w_k \): \( P(w_k) = \frac{C(w_k)}{N} \)
- adjusted count: \( \frac{N}{N+V} \)
  - \( \frac{N}{N+V} \) is a normalizing factor, \( N+V \) is the new “size” of the text
- \( P'(w_k) \): estimated probability
  - probability: \( P'(w_k) = \frac{C(w_k)+1}{N+1} \)

Test Corpus: Add-One Smoothing

<table>
<thead>
<tr>
<th>word</th>
<th>freq.</th>
<th>unsmoothed: ( \frac{C(w_i)}{V} )</th>
<th>add-one: ( \frac{C(w_i)+1}{N+1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>35</td>
<td>0.1383</td>
<td>0.0860</td>
</tr>
<tr>
<td>,</td>
<td>8</td>
<td>0.0316</td>
<td>0.0215</td>
</tr>
<tr>
<td>the</td>
<td>7</td>
<td>0.0277</td>
<td>0.0191</td>
</tr>
<tr>
<td>The</td>
<td>4</td>
<td>0.0158</td>
<td>0.0119</td>
</tr>
<tr>
<td>that</td>
<td>3</td>
<td>0.0119</td>
<td>0.0095</td>
</tr>
<tr>
<td>on</td>
<td>2</td>
<td>0.0079</td>
<td>0.0072</td>
</tr>
<tr>
<td>We</td>
<td>1</td>
<td>0.0040</td>
<td>0.0048</td>
</tr>
<tr>
<td>operator</td>
<td>0</td>
<td>0.0000</td>
<td>0.0024</td>
</tr>
</tbody>
</table>

Bigrams Example

<table>
<thead>
<tr>
<th>bigram</th>
<th>freq.</th>
<th>unsmoothed: ( \frac{C(w_i...w_j)}{V} )</th>
<th>add-one: ( \frac{C(w_i...w_j)+1}{N+1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>. END</td>
<td>35</td>
<td>1.0000</td>
<td>0.1800</td>
</tr>
<tr>
<td>START The</td>
<td>3</td>
<td>0.0857</td>
<td>0.0200</td>
</tr>
<tr>
<td>START You</td>
<td>2</td>
<td>0.0571</td>
<td>0.0150</td>
</tr>
<tr>
<td>is not</td>
<td>2</td>
<td>0.2857</td>
<td>0.0174</td>
</tr>
<tr>
<td>Your ire</td>
<td>1</td>
<td>0.5000</td>
<td>0.0120</td>
</tr>
<tr>
<td>You bring</td>
<td>1</td>
<td>0.3333</td>
<td>0.0119</td>
</tr>
<tr>
<td>not found</td>
<td>1</td>
<td>0.2500</td>
<td>0.0118</td>
</tr>
<tr>
<td>is the</td>
<td>0</td>
<td>0.0000</td>
<td>0.0058</td>
</tr>
<tr>
<td>This system</td>
<td>0</td>
<td>0</td>
<td>0.0060</td>
</tr>
</tbody>
</table>

Lidstone’s & Jeffreys-Perks Laws

Because Laplace’s law overestimates non-zero events, variations were created:

- Lidstone’s law: instead of adding one, add some smaller value \( \lambda \)
  \[
  P(w_1...w_n) = \frac{C(w_1...w_n)+\lambda}{N+V+\lambda}
  \]
- Jeffreys-Perks law: set \( \lambda \) to be \( \frac{1}{2} \) (the expectation of maximized MLE):
  \[
  P(w_1...w_n) = \frac{C(w_1...w_n)+\frac{1}{2}}{N+\frac{1}{2}}
  \]

Problems: How do we guess \( \lambda \)? And still not good for low frequency \( n \)-grams . . .
Towards Deleted Estimation

Held-Out Estimation

To get an idea as to whether a smoothing technique is effective, we can use held-out estimation.

- Split the data into training data and held-out data
- Use the held-out data to see how good the training estimates are

Using bigrams as an example:

- Say that there are \( N_r \) bigrams with frequency \( r \) in the training data
- Count up how often all these bigrams together occur in the held-out data; call this \( T_r \)
- Average frequency in held-out data is thus \( \frac{T_r}{N} \)

Deleted Estimation

Held-Out Estimation keeps the held-out data separate from the training data

- But what if we split the training data in half?
  - We could train on one half and validate on the other
  - And then we could switch the training and validation portions
- With both of these estimates, we can average them to obtain even more reliable estimates

\[
(6) \quad p_{del}(w_1 w_2) = \frac{T_1 + T_2}{N(N_r + T_r)}
\]

Witten-Bell Discounting

Problems with Add-One Smoothing:

- add-one smoothing leads to sharp changes in probabilities
- too much probability mass goes to unseen events

Witten-Bell: think of unseen events as ones not having happened yet

- the probability of this event – when it happens – can be modeled by the probability of seeing it for the first time

First Time Probability

How do we estimate probability of an N-gram occurring for the first time?

- count number of times of seeing an N-gram for the first time in training corpus
- think of corpus as series of events: one event for each token and one event for each new type
- e.g. unigrams:
  - corpus: a man with a hat
  - event: a new man new with new a hat ...
- number of events: \( N + T \)
Witten-Bell Probabilities

- total probability mass for unseen events:
  \[ \sum_{x:C(w_x)=0} p^*(w_x) = \frac{T}{N^2} \]
- probability for one unseen unigram: \( p^*(w_x) = \frac{T}{Z(w_x)N} \)
  - divide total prob. mass up for all unseen events
  - number of all unseen unigrams: \( Z = \sum_{x:C(w_x)=0} 1 \)
- discount total probability mass for unseen events from other events
  \[ p^*(w_x) = \frac{C(w_x)}{Z(w_x)N} \]
  for \( C(w_x) > 0 \)
- alternatively: smoothed counts:
  \[
  C^*(w_x) = \begin{cases} 
    \frac{T \cdot N}{Z(w_x)N} & \text{if } C(w_x) = 0 \\
    \frac{C(w_x)N}{Z(w_x)N} & \text{if } C(w_x) > 0 
  \end{cases}
  \]

**T(w) And Z(w) from Haikus**

\[ Z(w) = \text{number of unseen bigrams starting with } w \]
\[ Z(w) = V - T(w) = 165 - T(w) \]

<table>
<thead>
<tr>
<th>word</th>
<th>( T(w) )</th>
<th>( Z(w) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>1</td>
<td>164</td>
</tr>
<tr>
<td>START</td>
<td>13</td>
<td>152</td>
</tr>
<tr>
<td>is</td>
<td>6</td>
<td>159</td>
</tr>
<tr>
<td>Your</td>
<td>2</td>
<td>163</td>
</tr>
<tr>
<td>You</td>
<td>2</td>
<td>163</td>
</tr>
<tr>
<td>not</td>
<td>4</td>
<td>161</td>
</tr>
<tr>
<td>This</td>
<td>2</td>
<td>163</td>
</tr>
</tbody>
</table>

**Haiku Probabilities**

<table>
<thead>
<tr>
<th>bigram</th>
<th>unsmoothed</th>
<th>add-one</th>
<th>Witten-Bell</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td>1.0000</td>
<td>0.1800</td>
<td>0.9722</td>
</tr>
<tr>
<td>START The</td>
<td>0.0857</td>
<td>0.0200</td>
<td>0.0625</td>
</tr>
<tr>
<td>START You</td>
<td>0.0571</td>
<td>0.0150</td>
<td>0.0417</td>
</tr>
<tr>
<td>is not</td>
<td>0.2857</td>
<td>0.0174</td>
<td>0.1538</td>
</tr>
<tr>
<td>Your ire</td>
<td>0.5000</td>
<td>0.0120</td>
<td>0.2500</td>
</tr>
<tr>
<td>You bring</td>
<td>0.3333</td>
<td>0.0119</td>
<td>0.2000</td>
</tr>
<tr>
<td>not found</td>
<td>0.2500</td>
<td>0.0118</td>
<td>0.1250</td>
</tr>
<tr>
<td>is the</td>
<td>0</td>
<td>0.0058</td>
<td>0.0029</td>
</tr>
<tr>
<td>This system</td>
<td>0</td>
<td>0.0060</td>
<td>0.0031</td>
</tr>
</tbody>
</table>

**Good-Turing-Smoothing**

**Idea:** re-estimate probability mass assigned to N-grams with zero counts
- by looking at probability mass of all N-grams with count 1
- based on assumption of binomial distribution

Idea broken down:
- create classes \( N_c \) of N-grams which occur \( c \) times
- the size of class \( N_c \) is the frequency of frequency \( c \)
- This works well for N-grams.

**Good-Turing-Smoothing (2)**

- smoothed count \( c^* = \frac{c + 1}{N_c} \)
  - \( N_c = \sum_{b,c,d=c} 1 \)
- smoothed count for unseen events:
  \[ c^* = \frac{N}{N_c} \]

Haiku counts:
\[
\begin{array}{ccc}
  c & \frac{N_c}{c} & c^*  \\
  0 & 26980 & 0.0087  \\
  1 & 236 & 0.0593  \\
  2 & 7 & 0.4286  \\
  3 & 1 & 0  \\
  4 & 0 &  \\
\end{array}
\]
Problem: for highest count \( c \), \( N_{c+1} = 0 \)!!!
- i.e. \( c^* = (c + 1) \frac{N_{c+1}}{N_c} = (c + 1) \frac{0}{N_c} = 0 \)

Solution: discount only for small counts \( c \leq k \) (e.g. \( k = 5 \))
- \( c^* = c \) for \( c > k \)

New discounting:
\[
C^* = \frac{(c+1)N_{c+1}}{(c+1)N_{c+1} + \sum_{k=1}^{c} N_{k+1} - N_c} \quad \text{for } 1 \leq c \leq k
\]

Haiku Bigrams

\[
G-T = \frac{P(w_i | w_{i-2}w_{i-1})}{P(w_i | w_{i-1})}
\]
- \( k = 3 \)

<table>
<thead>
<tr>
<th>bigram</th>
<th>count</th>
<th>orig.</th>
<th>add-1</th>
<th>( W-B )</th>
<th>G-T</th>
</tr>
</thead>
<tbody>
<tr>
<td>. END</td>
<td>35</td>
<td>1.000</td>
<td>0.1800</td>
<td>0.9722</td>
<td>1.0000</td>
</tr>
<tr>
<td>START The</td>
<td>3</td>
<td>0.0857</td>
<td>0.0200</td>
<td>0.0625</td>
<td>0.0857</td>
</tr>
<tr>
<td>START You</td>
<td>2</td>
<td>0.0571</td>
<td>0.0150</td>
<td>0.0417</td>
<td>0.0122</td>
</tr>
<tr>
<td>is not</td>
<td>2</td>
<td>0.2857</td>
<td>0.0174</td>
<td>0.1538</td>
<td>0.1837</td>
</tr>
<tr>
<td>Your ire</td>
<td>1</td>
<td>0.5000</td>
<td>0.0120</td>
<td>0.2500</td>
<td>0.0297</td>
</tr>
<tr>
<td>You bring</td>
<td>1</td>
<td>0.3333</td>
<td>0.0119</td>
<td>0.2000</td>
<td>0.0198</td>
</tr>
<tr>
<td>not found</td>
<td>1</td>
<td>0.2500</td>
<td>0.0118</td>
<td>0.1250</td>
<td>0.0148</td>
</tr>
<tr>
<td>is the</td>
<td>0</td>
<td>0</td>
<td>0.0058</td>
<td>0.0029</td>
<td>0.0012</td>
</tr>
<tr>
<td>This system</td>
<td>0</td>
<td>0</td>
<td>0.0060</td>
<td>0.0031</td>
<td>0.0044</td>
</tr>
</tbody>
</table>

**Good-Turing-Smoothing**

- Ideally, we want to bias away from trigrams that are not used (e.g., bigrams and unigrams are subsets of trigrams)

(7)
\[
P(w_i | w_{i-2}w_{i-1}) = \lambda_1 P(w_i | w_{i-2}w_{i-1}) + \lambda_2 P(w_i | w_{i-1}) + \lambda_3 P(w_i)
\]
- \( \sum \lambda_i = 1 \)
- \( 0 \leq \lambda_i \leq 1 \)

Every trigram probability is a linear combination of the focus word’s trigram, bigram, and unigram.

- Use EM algorithm on held-out data to calculate \( \lambda \) values

**Backoff**

Idea: go back to “smaller” N-grams
- i.e. do not only use trigram prob. but also bigrams and unigrams
- no trigram found, use bigram; if no bigram found, use unigram
  ... can be used instead of smoothing

- need to weight contribution of specific N-gram:
  \[
P^*(w_i | w_{i-2}w_{i-1}) = \begin{cases} 
P(w_i | w_{i-2}w_{i-1}) & \text{if } C(w_{i-2}w_{i-1}w_i) > 0 \\
\alpha_1 P(w_i | w_{i-1}) & \text{if } C(w_{i-2}w_{i-1}w_i) = 0 \text{ and } C(w_{i-1}w_i) > 0 \\
\alpha_2 P(w_i) & \text{otherwise}
\end{cases}
\]

**Linear Interpolation**

Simple linear interpolation involves mixing different pieces of information to derive a probability
- Called deleted interpolation with subset relations (e.g., bigrams and unigrams are subsets of trigrams)

More generally, we can condition the word on its history and each \( \lambda \) can be based on the history, too

(8)
\[
P(w|h) = \sum \lambda_i P_i(w|h)
\]
- \( P_1 \) may focus on the trigram history, while \( P_2 \) uses the bigram, and so forth.
- So, instead of having one \( \lambda_1 \) for all trigrams, we have individualized it for each unique trigram
  - Useful, in that every trigram potentially behaves differently
  - But there’s a big sparse data problem
Equivalence bins

To overcome the sparse data problem, λ’s are calculated by putting them into equivalence bins

- One method (Chen and Goodman 1996) bases the bins on the number of different words which an n−1-gram has following it

\[
\lambda = \frac{C(w_1 \ldots w_{i-1})}{|w_i : C(w_1 \ldots w_i) > 0|}
\]

- \( w_i : C(w_1 \ldots w_i) > 0 \) means: the set of \( w_i \) such that the trigram exists

- So, great deal occurs 178 times, with 36 different words after it: average count = 4.94
- of that occurs 178 times, with 115 different words after it: 1.55
- These histories will thus prompt different λ values