Language Modeling

Generalization and zeros

The Shannon Visualization Method

- Choose a random bigram \((<s>, w)\) according to its probability
- Now choose a random bigram \((w, x)\) according to its probability
- And so on until we choose \(</s>\)
- Then string the words together

\(<s>\) I

I want

want to

to eat

eat Chinese

Chinese food

food \(</s>\)

I want to eat Chinese food
Approximating Shakespeare

1 gram
- To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
- Hill he late speaks; or! a more to leg less first you enter

2 gram
- Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
- What means, sir. I confess she? then all sorts. he is trim, captain.

3 gram
- Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ‘tis done.
- This shall forbid it should be branded, if renown made it empty.

4 gram
- King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in;
- It cannot be but so.

Shakespeare as corpus

- \( N = 884,647 \) tokens, \( V = 29,066 \)
- Shakespeare produced 300,000 bigram types out of \( V^2 = 844 \) million possible bigrams.
  - So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- Quadrigrams worse: What's coming out looks like Shakespeare because it \textit{is} Shakespeare
The Wall Street Journal is not Shakespeare

<table>
<thead>
<tr>
<th>1 gram</th>
<th>Months the my and issue of year foreign new exchange’s september were recession exchange new endorsed a acquire to six executives</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 gram</td>
<td>Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her</td>
</tr>
<tr>
<td>3 gram</td>
<td>They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions</td>
</tr>
</tbody>
</table>

Can you guess the author of these random 3-gram sentences?

- They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and gram Brazil on market conditions
- This shall forbid it should be branded, if renown made it empty.
The perils of overfitting

- N-grams only work well for word prediction if the test corpus looks like the training corpus
  - In real life, it often doesn’t
  - We need to train robust models that generalize!
  - One kind of generalization: Zeros!
    - Things that don’t ever occur in the training set
      - But occur in the test set

Zeros

- Training set:
  ... denied the allegations
  ... denied the reports
  ... denied the claims
  ... denied the request

- Test set
  ... denied the offer
  ... denied the loan

P(“offer” | denied the) = 0
Zero probability bigrams

- Bigrams with zero probability
  - mean that we will assign 0 probability to the test set!
  - And hence we cannot compute perplexity (can’t divide by 0)!

 Language Modeling

Smoothing: Add-one (Laplace) smoothing
The intuition of smoothing (from Dan Klein)

- When we have sparse statistics:
  \[ P(w \mid \text{denied the}) \]
  3 allegations
  2 reports
  1 claims
  1 request
  7 total

- Steal probability mass to generalize better
  \[ P(w \mid \text{denied the}) \]
  2.5 allegations
  1.5 reports
  0.5 claims
  0.5 request
  2 other
  7 total

Add-one estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!

\[
P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}
\]

- MLE estimate:

\[
P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}
\]

- Add-1 estimate:
Maximum Likelihood Estimates

- The maximum likelihood estimate
  - of some parameter of a model M from a training set T
  - maximizes the likelihood of the training set T given the model M
- Suppose the word “bagel” occurs 400 times in a corpus of a million words
- What is the probability that a random word from some other text will be “bagel”?
- MLE estimate is $400/1,000,000 = .0004$
- This may be a bad estimate for some other corpus
  - But it is the estimate that makes it most likely that “bagel” will occur 400 times in a million word corpus.

Add-One Smoothing

<table>
<thead>
<tr>
<th></th>
<th>100</th>
<th>100/300</th>
<th>101</th>
<th>101/326</th>
</tr>
</thead>
<tbody>
<tr>
<td>xya</td>
<td>100</td>
<td>100/300</td>
<td>101</td>
<td>101/326</td>
</tr>
<tr>
<td>xyb</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
</tr>
<tr>
<td>xyc</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
</tr>
<tr>
<td>xyd</td>
<td>200</td>
<td>200/300</td>
<td>201</td>
<td>201/326</td>
</tr>
<tr>
<td>xye</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xyz</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
</tr>
<tr>
<td>Total xy</td>
<td>300</td>
<td>300/300</td>
<td>326</td>
<td>326/326</td>
</tr>
</tbody>
</table>

Following examples from Kai-Wei Chang
Problem with Add-One Smoothing

We’ve been considering just 26 letter types ...

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xya</td>
<td>1</td>
<td>1/3</td>
<td>2</td>
</tr>
<tr>
<td>xyb</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
</tr>
<tr>
<td>xyc</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
</tr>
<tr>
<td>xyd</td>
<td>2</td>
<td>2/3</td>
<td>3</td>
</tr>
<tr>
<td>xye</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xyz</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total xy</strong></td>
<td>3</td>
<td>3/3</td>
<td>29</td>
</tr>
</tbody>
</table>

Problem with Add-One Smoothing

Suppose we’re considering 20000 word types

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1/3</td>
<td>2</td>
<td>2/20003</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/20003</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/20003</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
<td>2/3</td>
<td>3</td>
<td>3/20003</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/20003</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/20003</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>3</td>
<td>3/3</td>
<td>20003</td>
<td>20003/20003</td>
</tr>
</tbody>
</table>
Problem with Add-One Smoothing

Suppose we’re considering 20000 word types

<table>
<thead>
<tr>
<th>Word Type</th>
<th>Count</th>
<th>Probability</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1/3</td>
<td>2</td>
</tr>
<tr>
<td>see the abbot</td>
<td>1</td>
<td>1/3</td>
<td>2</td>
</tr>
<tr>
<td>see the abduct</td>
<td>1</td>
<td>1/3</td>
<td>2</td>
</tr>
<tr>
<td>see the above</td>
<td>3</td>
<td>3/3</td>
<td>20003</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3</td>
<td>20003</td>
</tr>
</tbody>
</table>

“Novel event” = event never happened in training data.
Here: 19998 novel events, with total estimated probability 19998/20003.
Add-one smoothing thinks we are extremely likely to see novel events, rather than words we’ve seen.

Add-Lambda Smoothing

- A large dictionary makes novel events too probable.
- To fix: Instead of adding 1 to all counts, add $\lambda = 0.01$?
  - This gives much less probability to novel events.
- But how to pick best value for $\lambda$?
  - That is, how much should we smooth?
### Add-0.001 Smoothing

Doesn’t smooth much

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xya</td>
<td>1</td>
<td>1/3</td>
<td>1.001</td>
<td>0.331</td>
</tr>
<tr>
<td>xyb</td>
<td>0</td>
<td>0/3</td>
<td>0.001</td>
<td>0.0003</td>
</tr>
<tr>
<td>xyc</td>
<td>0</td>
<td>0/3</td>
<td>0.001</td>
<td>0.0003</td>
</tr>
<tr>
<td>xyd</td>
<td>2</td>
<td>2/3</td>
<td>2.001</td>
<td>0.661</td>
</tr>
<tr>
<td>xye</td>
<td>0</td>
<td>0/3</td>
<td>0.001</td>
<td>0.0003</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>xyz</td>
<td>0</td>
<td>0/3</td>
<td>0.001</td>
<td>0.0003</td>
</tr>
<tr>
<td>Total xy</td>
<td>3</td>
<td>3/3</td>
<td>3.026</td>
<td>1</td>
</tr>
</tbody>
</table>

### Add-1000 Smoothing

Smooths too much

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xya</td>
<td>1</td>
<td>1/3</td>
<td>1001</td>
<td>1/26</td>
</tr>
<tr>
<td>xyb</td>
<td>0</td>
<td>0/3</td>
<td>1000</td>
<td>1/26</td>
</tr>
<tr>
<td>xyc</td>
<td>0</td>
<td>0/3</td>
<td>1000</td>
<td>1/26</td>
</tr>
<tr>
<td>xyd</td>
<td>2</td>
<td>2/3</td>
<td>1002</td>
<td>1/26</td>
</tr>
<tr>
<td>xye</td>
<td>0</td>
<td>0/3</td>
<td>1000</td>
<td>1/26</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>xyz</td>
<td>0</td>
<td>0/3</td>
<td>1000</td>
<td>1/26</td>
</tr>
<tr>
<td>Total xy</td>
<td>3</td>
<td>3/3</td>
<td>26003</td>
<td>1</td>
</tr>
</tbody>
</table>
Add-Lambda Smoothing

• A large dictionary makes novel events too probable.

• To fix: Instead of adding 1 to all counts, add $\lambda = 0.01$?
  • This gives much less probability to novel events.

• But how to pick best value for $\lambda$?
  • That is, how much should we smooth?
  • E.g., how much probability to "set aside" for novel events?
    • Depends on how likely novel events really are!
    • Which may depend on the type of text, size of training corpus, ...
  • Can we figure it out from the data? (advanced topics)

Setting Smoothing Parameters

• How to pick best value for $\lambda$? (in add-$\lambda$ smoothing)
• Try many $\lambda$ values & report the one that gets best results?

• How to measure whether a particular $\lambda$ gets good results?
• Is it fair to measure that on test data (for setting $\lambda$)?
  • Moral: Selective reporting on test data can make a method look artificially good.  
    So it is unethical.
  • Rule: Test data cannot influence system development. No peeking! Use it only to evaluate the final system(s). Report all results on it.
Setting Smoothing Parameters

• How to pick best value for \( \lambda \)? (in add-\( \lambda \) smoothing)
• Try many \( \lambda \) values & report the one that gets best results?

```
Training                                Test
```

• How to measure whether a particular \( \lambda \) gets good results?
• Is it fair to measure that on test data (for setting \( \lambda \))?  
  • Moral: Selective reporting on test data can make a method look artificially good. So it is unethical.
  • Rule: Test data cannot influence system development. No peeking! Use it only to evaluate the final system(s). Report all results on it.

```
23
```

Pick \( \lambda \) that gets best results on this 20% …

```
Dev. Training Training Training
```

… when we collect counts from this 80% and smooth them using add-\( \lambda \) smoothing.

```
Training Training Training
```

Now use that \( \lambda \) to get smoothed counts from all 100% …

```
Training Training Training
```

… and report results of that final model on test data.

```
24
```
Large or small Dev set?

• Here we held out 20% of our training set (yellow) for development.
• Would like to use > 20% yellow:
  • 20% not enough to reliably assess $\lambda$
• Would like to use > 80% blue:
  • Best $\lambda$ for smoothing 80% $\neq$ best $\lambda$ for smoothing 100%

Cross-Validation

• Try 5 training/dev splits as below
  • Pick $\lambda$ that gets best average performance

  🤖 Tests on all 100% as yellow, so we can more reliably assess $\lambda$.
  🙁 Still picks a $\lambda$ that’s good at smoothing the 80% size, not 100%.
  • But now we can grow that 80% without trouble
N-fold Cross-Validation ("Leave One Out")

- Test each sentence with smoothed model from other N-1 sentences
- Still tests on all 100% as yellow, so we can reliably assess $\lambda$
- Trains on nearly 100% blue data ($(N-1)/N$) to measure whether $\lambda$ is good for smoothing that

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

<table>
<thead>
<tr>
<th></th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>6</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>want</td>
<td>3</td>
<td>1</td>
<td>609</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>687</td>
<td>3</td>
<td>7</td>
<td>212</td>
</tr>
<tr>
<td>eat</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>17</td>
<td>3</td>
<td>43</td>
<td>1</td>
</tr>
<tr>
<td>chinese</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>83</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>food</td>
<td>16</td>
<td>1</td>
<td>16</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>lunch</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>spend</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Laplace-smoothed bigrams

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0.0015</td>
<td>0.21</td>
<td>0.00025</td>
<td>0.0025</td>
<td>0.00025</td>
<td>0.00025</td>
<td>0.00025</td>
<td>0.00075</td>
</tr>
<tr>
<td>want</td>
<td>0.0013</td>
<td>0.00042</td>
<td>0.26</td>
<td>0.00084</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0025</td>
<td>0.00084</td>
</tr>
<tr>
<td>to</td>
<td>0.00078</td>
<td>0.00026</td>
<td>0.0013</td>
<td>0.18</td>
<td>0.00078</td>
<td>0.00026</td>
<td>0.0018</td>
<td>0.055</td>
</tr>
<tr>
<td>eat</td>
<td>0.00046</td>
<td>0.00046</td>
<td>0.0014</td>
<td>0.00046</td>
<td>0.0078</td>
<td>0.0014</td>
<td>0.02</td>
<td>0.00046</td>
</tr>
<tr>
<td>chinese</td>
<td>0.0012</td>
<td>0.00062</td>
<td>0.00062</td>
<td>0.00062</td>
<td>0.00062</td>
<td>0.052</td>
<td>0.0012</td>
<td>0.00062</td>
</tr>
<tr>
<td>food</td>
<td>0.0063</td>
<td>0.00039</td>
<td>0.0063</td>
<td>0.00039</td>
<td>0.00079</td>
<td>0.002</td>
<td>0.00039</td>
<td>0.00039</td>
</tr>
<tr>
<td>lunch</td>
<td>0.0017</td>
<td>0.00056</td>
<td>0.00056</td>
<td>0.00056</td>
<td>0.00056</td>
<td>0.0011</td>
<td>0.00056</td>
<td>0.00056</td>
</tr>
<tr>
<td>spend</td>
<td>0.0012</td>
<td>0.00058</td>
<td>0.0012</td>
<td>0.00058</td>
<td>0.00058</td>
<td>0.00058</td>
<td>0.00058</td>
<td>0.00058</td>
</tr>
</tbody>
</table>

### Reconstituted counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n) + 1] \times C(w_{n-1})}{C(w_{n-1}) + V}$$

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>3.8</td>
<td>527</td>
<td>0.64</td>
<td>6.4</td>
<td>0.64</td>
<td>0.64</td>
<td>0.64</td>
<td>1.9</td>
</tr>
<tr>
<td>want</td>
<td>1.2</td>
<td>0.39</td>
<td>238</td>
<td>0.78</td>
<td>2.7</td>
<td>2.7</td>
<td>2.3</td>
<td>0.78</td>
</tr>
<tr>
<td>to</td>
<td>1.9</td>
<td>0.63</td>
<td>3.1</td>
<td>430</td>
<td>1.9</td>
<td>0.63</td>
<td>4.4</td>
<td>133</td>
</tr>
<tr>
<td>eat</td>
<td>0.34</td>
<td>0.34</td>
<td>1</td>
<td>0.34</td>
<td>5.8</td>
<td>1</td>
<td>15</td>
<td>0.34</td>
</tr>
<tr>
<td>chinese</td>
<td>0.2</td>
<td>0.098</td>
<td>0.098</td>
<td>0.098</td>
<td>0.098</td>
<td>8.2</td>
<td>0.2</td>
<td>0.098</td>
</tr>
<tr>
<td>food</td>
<td>6.9</td>
<td>0.43</td>
<td>6.9</td>
<td>0.43</td>
<td>0.86</td>
<td>2.2</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>lunch</td>
<td>0.57</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.38</td>
<td>0.19</td>
<td>0.19</td>
</tr>
<tr>
<td>spend</td>
<td>0.32</td>
<td>0.16</td>
<td>0.32</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
</tr>
</tbody>
</table>
## Compare with raw bigram counts

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>5</td>
<td>827</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>want</td>
<td>2</td>
<td>0</td>
<td>608</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>686</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>211</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>16</td>
<td>2</td>
<td>42</td>
<td>0</td>
</tr>
<tr>
<td>chinese</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>3.8</td>
<td>527</td>
<td>0.64</td>
<td>6.4</td>
<td>0.64</td>
<td>0.64</td>
<td>0.64</td>
<td>1.9</td>
</tr>
<tr>
<td>want</td>
<td>1.2</td>
<td>0.39</td>
<td>238</td>
<td>0.78</td>
<td>2.7</td>
<td>2.7</td>
<td>2.7</td>
<td>0.78</td>
</tr>
<tr>
<td>to</td>
<td>1.9</td>
<td>0.63</td>
<td>3.1</td>
<td>430</td>
<td>1.9</td>
<td>0.63</td>
<td>4.4</td>
<td>133</td>
</tr>
<tr>
<td>eat</td>
<td>0.34</td>
<td>0.34</td>
<td>1</td>
<td>0.34</td>
<td>5.8</td>
<td>1</td>
<td>15</td>
<td>0.34</td>
</tr>
<tr>
<td>chinese</td>
<td>0.2</td>
<td>0.098</td>
<td>0.098</td>
<td>0.098</td>
<td>0.098</td>
<td>8.2</td>
<td>0.2</td>
<td>0.098</td>
</tr>
<tr>
<td>food</td>
<td>6.9</td>
<td>0.43</td>
<td>6.9</td>
<td>0.43</td>
<td>0.86</td>
<td>2.2</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>lunch</td>
<td>0.57</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.38</td>
<td>0.19</td>
<td>0.19</td>
</tr>
<tr>
<td>spend</td>
<td>0.32</td>
<td>0.16</td>
<td>0.32</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
</tr>
</tbody>
</table>

## Add-1 estimation is a blunt instrument

- So add-1 isn’t used for N-grams:
  - We’ll see better methods
- But add-1 is used to smooth other NLP models
  - In domains where the number of zeros isn’t so huge.
Unigram Smoothing Example

- Tiny Corpus, V=4; N=20

\[ P_n(w) = \frac{c_i + 1}{N + V} \]

<table>
<thead>
<tr>
<th>Word</th>
<th>True Ct</th>
<th>Unigram Prob</th>
<th>New Ct</th>
<th>Adjusted Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>eat</td>
<td>10</td>
<td>.5</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>British</td>
<td>4</td>
<td>.2</td>
<td>5</td>
<td>.21</td>
</tr>
<tr>
<td>food</td>
<td>6</td>
<td>.3</td>
<td>7</td>
<td>.29</td>
</tr>
<tr>
<td>happily</td>
<td>0</td>
<td>.0</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1.0</td>
<td>~20</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Backoff and Interpolation

- Sometimes it helps to use less context
  - Condition on less context for contexts you haven’t learned much about
- Backoff:
  - use trigram if you have good evidence,
  - otherwise bigram, otherwise unigram
- Interpolation:
  - mix unigram, bigram, trigram
- Interpolation works better

---

Backoff and interpolation

- \( p(\text{zombie} \mid \text{see the}) \) vs. \( p(\text{baby} \mid \text{see the}) \)
  - What if \( \text{count(see the ngram)} = \text{count(see the baby)} = 0 \)?
  - baby beats ngram as a unigram
  - the baby beats the ngram as a bigram
  - \( \therefore \) see the baby beats see the ngram?
    (even if both have the same count, such as 0)
Class-Based Backoff

• Back off to the class rather than the word
  • Particularly useful for proper nouns (e.g., names)
  • Use count for the number of names in place of the particular name
  • E.g. `< N | friendly >` instead of `< dog | friendly >`

Linear Interpolation

• Simple interpolation
  \[
  \hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n) \quad \sum_i \lambda_i = 1
  \]

• Lambdas conditional on context:
  \[
  \hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 (w_{n-2}^{n-1})P(w_n|w_{n-2}w_{n-1}) + \lambda_2 (w_{n-2}^{n-1})P(w_n|w_{n-1}) + \lambda_3 (w_{n-2}^{n-1})P(w_n)
  \]
How to set the lambdas?

- Use a **held-out** corpus

| Training Data | Held-Out Data | Test Data |

- Choose $\lambda$s to maximize the probability of held-out data:
  - Fix the N-gram probabilities (on the training data)
  - Then search for $\lambda$s that give largest probability to held-out set:

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced
  - Vocabulary $V$ is fixed
  - Closed vocabulary task
- Often we don’t know this
  - **Out Of Vocabulary** = OOV words
  - Open vocabulary task
- Instead: create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon $L$ of size $V$
    - At text normalization phase, any training word not in $L$ changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - If text input: Use UNK probabilities for any word not in training
Huge web-scale n-grams

- How to deal with, e.g., Google N-gram corpus
- Pruning
  - E.g., only store N-grams with count > threshold.
    - Remove singletons of higher-order n-grams
- Efficient data structures, etc.

N-gram Smoothing Summary

- Add-1 smoothing:
  - OK for some tasks, but not for language modeling
- See text for
  - The most commonly used method:
    - Extended Interpolated Kneser-Ney
  - For very large N-grams like the Web:
    - Stupid backoff
Other Applications

• N-grams are not only for words
  • Characters
  • Sentences

More examples

• Yoav’s blog post:
  [http://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139](http://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139)
• 10-gram character-level LM:

  First Citizen: Nay, then, that was hers, It speaks against your other service: But since the youth of the circumstance be spoken: Your uncle and one Baptista's daughter.

  SEBASTIAN: Do I stand till the break off.

  BIRON:
  Hide thy head.
Example: Language ID

- “Horses and Lukasiewicz are on the curriculum.”
  - Is this English or Polish or ??

- Let’s use n-gram models …
- Space of outcomes will be character sequences \((x_1, x_2, x_3, \ldots)\)

Language ID: Problem Formulation

- Let \(p(X) = \text{probability of text } X \text{ in English}\)
- Let \(q(X) = \text{probability of text } X \text{ in Polish}\)
- Which probability is higher?
  - (we’d also like bias toward English since it’s more likely \(a \text{ priori}\) – ignore that for now)

“Horses and Lukasiewicz are on the curriculum.”
\[p(x_1=h, x_2=o, x_3=r, x_4=s, x_5=e, x_6=s, \ldots)\]
Apply the Chain Rule

\[ p(x_1=\text{h}, x_2=\text{o}, x_3=\text{r}, x_4=\text{s}, x_5=\text{e}, x_6=\text{s}, \ldots) \]
\[ = p(x_1=\text{h}) \]
\[ \times p(x_2=\text{o} \mid x_1=\text{h}) \]
\[ \times p(x_3=\text{r} \mid x_1=\text{h}, x_2=\text{o}) \]
\[ \times p(x_4=\text{s} \mid x_2=\text{o}, x_3=\text{r}) \]
\[ \times p(x_5=\text{e} \mid x_3=\text{r}, x_4=\text{s}) \]
\[ \times p(x_6=\text{s} \mid x_4=\text{s}, x_5=\text{e}) \]
\[ \times \ldots = 0 \]

Use Bigrams

\[ p(x_1=\text{h}, x_2=\text{o}, x_3=\text{r}, x_4=\text{s}, x_5=\text{e}, x_6=\text{s}, \ldots) \]
\[ \approx p(x_1=\text{h}) \]
\[ \times p(x_2=\text{o} \mid x_1=\text{h}) \]
\[ \times p(x_3=\text{r} \mid x_1=\text{h}, x_2=\text{o}) \]
\[ \times p(x_4=\text{s} \mid x_2=\text{o}, x_3=\text{r}) \]
\[ \times p(x_5=\text{e} \mid x_3=\text{r}, x_4=\text{s}) \]
\[ \times p(x_6=\text{s} \mid x_4=\text{s}, x_5=\text{e}) \]
\[ \times \ldots = 7.3\text{e}-10 \times \ldots \]
English vs. Polish?

English

Polish

N-gram Model

compute $p(X)$

compute $q(X)$

Compare!

Chapter Summary

• N-gram probabilities can be used to *estimate* the likelihood
  • Of a word occurring in a context (N-1)
  • Of a sentence occurring at all
• Perplexity can be used to evaluate the goodness of fit of a LM
• Smoothing techniques and backoff models deal with problems of unseen words in corpus
  • Improvement via algorithm versus big data