Roadmap

- Why do we need language models?
- Definition of language models
- Estimating probability distributions
- Evaluating language models
- Dealing with zeroes
1 What is a Language Model?

2 Evaluating Language Models

3 Estimating Probability Distributions

4 Advanced Zero Avoidance
Language models answer the question: *How likely is a string of English words good English?*

- Autocomplete on phones and websearch
- Creating English-looking documents
- Very common in machine translation systems
  - Help with reordering
    \[ p_{\text{lm}}(\text{the house is small}) > p_{\text{lm}}(\text{small the is house}) \]
  - Help with word choice
    \[ p_{\text{lm}}(\text{I am going home}) > p_{\text{lm}}(\text{I am going house}) \]
N-Gram Language Models

- Given: a string of English words \( W = w_1, w_2, w_3, ..., w_n \)
- Question: what is \( p(W) \)?
- Sparse data: Many good English sentences will not have been seen before

\[ p(w_1, w_2, w_3, ..., w_n) = p(w_1) p(w_2|w_1) p(w_3|w_1, w_2) ... p(w_n|w_1, w_2, ... w_{n-1}) \]

(not much gained yet, \( p(w_n|w_1, w_2, ... w_{n-1}) \) is equally sparse)
Markov Chain

- **Markov assumption:**
  - only previous history matters
  - limited memory: only last $k$ words are included in history
    (older words less relevant)
  -> $k$th order Markov model

- For instance 2-gram language model:

  $$p(w_1, w_2, w_3, ..., w_n) \sim p(w_1) p(w_2|w_1) p(w_3|w_2) ... p(w_n|w_{n-1})$$

- What is conditioned on, here $w_{i-1}$ is called the **history**
Outline

1. What is a Language Model?
2. Evaluating Language Models
3. Estimating Probability Distributions
4. Advanced Zero Avoidance
A good model assigns a text of real English $W$ a high probability. This can be also measured with cross entropy:

$$H(W) = \frac{1}{n} \log p(W^n)$$

Or, **perplexity**

$$\text{perplexity}(W) = 2^{H(W)}$$
## Comparison 1–4-Gram

<table>
<thead>
<tr>
<th>word</th>
<th>unigram</th>
<th>bigram</th>
<th>trigram</th>
<th>4-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>6.684</td>
<td>3.197</td>
<td>3.197</td>
<td>3.197</td>
</tr>
<tr>
<td>would</td>
<td>8.342</td>
<td>2.884</td>
<td>2.791</td>
<td>2.791</td>
</tr>
<tr>
<td>like</td>
<td>9.129</td>
<td>2.026</td>
<td>1.031</td>
<td>1.290</td>
</tr>
<tr>
<td>to</td>
<td>5.081</td>
<td>0.402</td>
<td>0.144</td>
<td>0.113</td>
</tr>
<tr>
<td>commend</td>
<td>15.487</td>
<td>12.335</td>
<td>8.794</td>
<td>8.633</td>
</tr>
<tr>
<td>the</td>
<td>3.885</td>
<td>1.402</td>
<td>1.084</td>
<td>0.880</td>
</tr>
<tr>
<td>rapporteur</td>
<td>10.840</td>
<td>7.319</td>
<td>2.763</td>
<td>2.350</td>
</tr>
<tr>
<td>on</td>
<td>6.765</td>
<td>4.140</td>
<td>4.150</td>
<td>1.862</td>
</tr>
<tr>
<td>his</td>
<td>10.678</td>
<td>7.316</td>
<td>2.367</td>
<td>1.978</td>
</tr>
<tr>
<td>work</td>
<td>9.993</td>
<td>4.816</td>
<td>3.498</td>
<td>2.394</td>
</tr>
<tr>
<td>.</td>
<td>4.896</td>
<td>3.020</td>
<td>1.785</td>
<td>1.510</td>
</tr>
<tr>
<td>&lt;/s&gt;</td>
<td>4.828</td>
<td>0.005</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>average</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>perplexity</td>
<td>265.136</td>
<td>16.817</td>
<td>6.206</td>
<td>4.758</td>
</tr>
</tbody>
</table>
Outline

1. What is a Language Model?
2. Evaluating Language Models
3. Estimating Probability Distributions
4. Advanced Zero Avoidance
How do we estimate a probability?

Suppose we want to estimate $P(w_n = "home" | h = go)$. 
How do we estimate a probability?

Suppose we want to estimate \( P(w_n = \text{"home"} | h = \text{go}) \).

\[
\hat{\theta}_i = \frac{n_{ik}}{n_k} 
\]

Computational Linguistics I: Jordan Boyd-Graber (UMD)
Language Models
September 23, 2013 11 / 28
How do we estimate a probability?

Suppose we want to estimate \( P(w_n = "\text{home}" | h = \text{go}) \).

- Maximum likelihood (ML) estimate of the probability is:

\[
\hat{\theta}_i = \frac{n_i}{\sum_k n_k} \quad (1)
\]
Counts for trigrams and estimated word probabilities

<table>
<thead>
<tr>
<th>word</th>
<th>c.</th>
<th>prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>cross</td>
<td>123</td>
<td>0.547</td>
</tr>
<tr>
<td>tape</td>
<td>31</td>
<td>0.138</td>
</tr>
<tr>
<td>army</td>
<td>9</td>
<td>0.040</td>
</tr>
<tr>
<td>card</td>
<td>7</td>
<td>0.031</td>
</tr>
<tr>
<td>,</td>
<td>5</td>
<td>0.022</td>
</tr>
</tbody>
</table>

- 225 trigrams in the Europarl corpus start with **the red**
- 123 of them end with **cross**

→ maximum likelihood probability is \( \frac{123}{225} = 0.547 \).
Example: 3-Gram

- Counts for trigrams and estimated word probabilities

<table>
<thead>
<tr>
<th>word</th>
<th>c.</th>
<th>prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>cross</td>
<td>123</td>
<td>0.547</td>
</tr>
<tr>
<td>tape</td>
<td>31</td>
<td>0.138</td>
</tr>
<tr>
<td>army</td>
<td>9</td>
<td>0.040</td>
</tr>
<tr>
<td>card</td>
<td>7</td>
<td>0.031</td>
</tr>
<tr>
<td>,</td>
<td>5</td>
<td>0.022</td>
</tr>
</tbody>
</table>

- 225 trigrams in the Europarl corpus start with the red
- 123 of them end with cross
  - maximum likelihood probability is $\frac{123}{225} = 0.547$.

- Is this reasonable?
The problem with maximum likelihood estimates: Zeros

- If there were no occurrences of "bageling" in a history go, we’d get a zero estimate:

\[
\hat{P}(\text{"bageling"}|\text{go}) = \frac{T_{\text{go}, \text{"bageling"}}}{\sum_{w' \in V} T_{\text{go}, w'}} = 0
\]

- We will get \( P(\text{go}|d) = 0 \) for any sentence that contains go bageling!

- Zero probabilities cannot be conditioned away.
How do we estimate a probability?

- In computational linguistics, we often have a prior notion of what our probability distributions are going to look like (for example, non-zero, sparse, uniform, etc.).
- This estimate of a probability distribution is called the maximum a posteriori (MAP) estimate:

\[ \theta_{\text{MAP}} = \arg\max_{\theta} f(x|\theta)g(\theta) \]
Add-One Smoothing

- Equivalent to assuming a \textit{uniform} prior over all possible distributions over the next word (you’ll learn why in CL2)
- But there are many more unseen n-grams than seen n-grams
- Example: Europarl 2-bigrams:
  - 86,700 distinct words
  - $86,700^2 = 7,516,890,000$ possible bigrams
  - but only about 30,000,000 words (and bigrams) in corpus
How do we estimate a probability?

- Assuming a **sparse Dirichlet** prior, \( \alpha < 1 \) to each count

\[
\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k}
\]

(3)

- \( \alpha_i \) is called a smoothing factor, a pseudocount, etc.
How do we estimate a probability?

- Assuming a **sparse Dirichlet** prior, $\alpha < 1$ to each count

  $$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k}$$  \hspace{1cm} (3)

- $\alpha_i$ is called a smoothing factor, a pseudocount, etc.
- When $\alpha_i = 1$ for all $i$, it’s called “Laplace smoothing”
How do we estimate a probability?

- Assuming a **sparse Dirichlet** prior, $\alpha < 1$ to each count

\[ \theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \]  

- $\alpha_i$ is called a smoothing factor, a pseudocount, etc.
- When $\alpha_i = 1$ for all $i$, it’s called “Laplace smoothing”
- What is a good value for $\alpha$?
- Could be optimized on held-out set to find the “best” language model
Example: 2-Grams in Europarl

<table>
<thead>
<tr>
<th>Count</th>
<th>Adjusted count</th>
<th>Test count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>((c + 1)\frac{n}{n+v^2})</td>
<td>((c + \alpha)\frac{n}{n+\alpha v^2})</td>
</tr>
<tr>
<td>0</td>
<td>0.00378</td>
<td>0.00016</td>
</tr>
<tr>
<td>1</td>
<td>0.00755</td>
<td>0.95725</td>
</tr>
<tr>
<td>2</td>
<td>0.01133</td>
<td>1.91433</td>
</tr>
<tr>
<td>3</td>
<td>0.01511</td>
<td>2.87141</td>
</tr>
<tr>
<td>4</td>
<td>0.01888</td>
<td>3.82850</td>
</tr>
<tr>
<td>5</td>
<td>0.02266</td>
<td>4.78558</td>
</tr>
<tr>
<td>6</td>
<td>0.02644</td>
<td>5.74266</td>
</tr>
<tr>
<td>8</td>
<td>0.03399</td>
<td>7.65683</td>
</tr>
<tr>
<td>10</td>
<td>0.04155</td>
<td>9.57100</td>
</tr>
<tr>
<td>20</td>
<td>0.07931</td>
<td>19.14183</td>
</tr>
</tbody>
</table>

Can we do better? In higher-order models, we can learn from similar contexts!
Example: 2-Grams in Europarl

<table>
<thead>
<tr>
<th>Count</th>
<th>Adjusted count</th>
<th>Test count</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>$(c + 1) \frac{n}{n + v^2}$</td>
<td>$(c + \alpha) \frac{n}{n + \alpha v^2}$</td>
</tr>
<tr>
<td>0</td>
<td>0.00378</td>
<td>0.00016</td>
</tr>
<tr>
<td>1</td>
<td>0.00755</td>
<td>0.95725</td>
</tr>
<tr>
<td>2</td>
<td>0.01133</td>
<td>1.91433</td>
</tr>
<tr>
<td>3</td>
<td>0.01511</td>
<td>2.87141</td>
</tr>
</tbody>
</table>

Can we do better?

In higher-order models, we can learn from similar contexts!

<table>
<thead>
<tr>
<th>Count</th>
<th>Adjusted count</th>
<th>Test count</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.03399</td>
<td>7.65683</td>
</tr>
<tr>
<td>10</td>
<td>0.04155</td>
<td>9.57100</td>
</tr>
<tr>
<td>20</td>
<td>0.07931</td>
<td>19.14183</td>
</tr>
</tbody>
</table>
Back-Off

- In given corpus, we may never observe
  - **Scottish beer drinkers**
  - **Scottish beer eaters**
- Both have count 0
  - → our smoothing methods will assign them same probability
- Better: backoff to bigrams:
  - **beer drinkers**
  - **beer eaters**
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

\[
p_i(w_3|w_1, w_2) = \lambda_1 p_1(w_3) \times \lambda_2 p_2(w_3|w_2) \times \lambda_3 p_3(w_3|w_1, w_2)
\]
Back-Off

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

\[
p_{n}^{BO}(w_i|w_{i-n+1}, \ldots, w_{i-1}) =
\begin{cases}
\alpha_n(w_i|w_{i-n+1}, \ldots, w_{i-1}) \\
\quad \text{if } \text{count}_n(w_{i-n+1}, \ldots, w_i) > 0 \\
\quad d_n(w_{i-n+1}, \ldots, w_{i-1}) \ p_{n-1}^{BO}(w_i|w_{i-n+2}, \ldots, w_{i-1}) \\
\quad \text{else}
\end{cases}
\]

- Requires
  - adjusted prediction model \( \alpha_n(w_i|w_{i-n+1}, \ldots, w_{i-1}) \)
  - discounting function \( d_n(w_1, \ldots, w_{n-1}) \)
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 histories for a word

\[ N_{1+}(\bullet w) = |\{ w_i : c(w_i, w) > 0 \}| \]

- Recall: maximum likelihood estimation of unigram language model

\[ p_{ML}(w) = \frac{c(w)}{\sum_i c(w_i)} \]

- In Kneser-Ney smoothing, replace raw counts with count of histories

\[ p_{KN}(w) = \frac{N_{1+}(\bullet w)}{\sum_{w_i} N_{1+}(w_i w)} \]
Interpolated Back-Off

- Back-off models use only highest order n-gram
  - if sparse, not very reliable.
  - two different n-grams with same history occur once → same probability
  - one may be an outlier, the other under-represented in training
- To remedy this, always consider the lower-order back-off models
- Adapting the $\alpha$ function into interpolated $\alpha_I$ function by adding back-off

\[
\alpha_I(w_n|w_1, \ldots, w_{n-1}) = \alpha(w_n|w_1, \ldots, w_{n-1}) \\
+ d(w_1, \ldots, w_{n-1}) p_I(w_n|w_2, \ldots, w_{n-1})
\]

- Note that $d$ function needs to be adapted as well
**Evaluation of smoothing methods:**

Perplexity for language models trained on the Europarl corpus

<table>
<thead>
<tr>
<th>Smoothing method</th>
<th>bigram</th>
<th>trigram</th>
<th>4-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good-Turing</td>
<td>96.2</td>
<td>62.9</td>
<td>59.9</td>
</tr>
<tr>
<td>Witten-Bell</td>
<td>97.1</td>
<td>63.8</td>
<td>60.4</td>
</tr>
<tr>
<td>Modified Kneser-Ney</td>
<td>95.4</td>
<td>61.6</td>
<td>58.6</td>
</tr>
<tr>
<td>Interpolated Modified Kneser-Ney</td>
<td>94.5</td>
<td>59.3</td>
<td>54.0</td>
</tr>
</tbody>
</table>
Reducing Vocabulary Size

- For instance: each number is treated as a separate token
- Replace them with a number token `num`
  - but: we want our language model to prefer
    
    $$p_{lm}(\text{I pay 950.00 in May 2007}) > p_{lm}(\text{I pay 2007 in May 950.00})$$

  - not possible with number token
    
    $$p_{lm}(\text{I pay num in May num}) = p_{lm}(\text{I pay num in May num})$$

- Replace each digit (with unique symbol, e.g., @ or 5), retain some distinctions
  
  $$p_{lm}(\text{I pay 555.55 in May 5555}) > p_{lm}(\text{I pay 5555 in May 555.55})$$
Summary

- Language models: *How likely is a string of English words good English?*
- N-gram models (Markov assumption)
- Perplexity
- Count smoothing
- Interpolation and backoff
In Class . . .

- Any remaining questions for first homework
- Working through Knesser-Ney example
- Discussing homework assignment: building bigram language models