Language Modeling

CSE354 - Spring 2020
Task

- Language Modeling (i.e. auto-complete)
- Probabilistic Modeling
  - Probability Theory
  - Logistic Regression
  - Sequence Modeling
Language Modeling

-- assigning a probability to sequences of words.

Version 1: Compute $P(w_1, w_2, w_3, w_4, w_5) = P(W)$

:probability of a sequence of words
Language Modeling

-- assigning a probability to sequences of words.

Version 1: Compute \( P(w_1, w_2, w_3, w_4, w_5) = P(W) \)

:probability of a sequence of words

Version 2: Compute \( P(w_5 | w_1, w_2, w_3, w_4) = P(w_n | w_1, w_2, ..., w_{n-1}) \)

:probability of a next word given history
Language Modeling

Version 1: Compute \( P(w_1, w_2, w_3, w_4, w_5) = P(W) \)
: probability of a sequence of words

\[ P(\text{He ate the cake with the fork}) = ? \]

Version 2: Compute \( P(w_5 | w_1, w_2, w_3, w_4) \)
= \( P(w_n | w_1, w_2, ..., w_{n-1}) \)
: probability of a next word given history

\[ P(\text{fork} | \text{He ate the cake with the}) = ? \]
Language Modeling

Applications:
- Auto-complete: What word is next?
- Machine Translation: Which translation is most likely?
- Spell Correction: Which word is most likely given error?
- Speech Recognition: What did they just say? “eyes aw of an”

(excerpt from Jurafsky, 2017)
Language Modeling

Version 1: Compute \( P(w_1, w_2, w_3, w_4, w_5) = P(W) \)
  :probability of a sequence of words
  \( P(\text{He ate the cake with the fork}) = ? \)

Version 2: Compute \( P(w_5 | w_1, w_2, w_3, w_4) \)
  \[ = P(w_n | w_1, w_2, ..., w_{n-1}) \]
  :probability of a next word given history
  \( P(\text{fork} | \text{He ate the cake with the}) = ? \)
Simple Solution

Version 1: Compute \( P(w_1, w_2, w_3, w_4, w_5) = P(W) \): probability of a sequence of words

\[
P(\text{He ate the cake with the fork}) = \frac{\text{count}(\text{He ate the cake with the fork})}{\text{count}(\text{* * * * * * * * * * * *})}
\]
Simple Solution: The Maximum Likelihood Estimate

Version 1: Compute $P(w_1, w_2, w_3, w_4, w_5) = P(W)$

: probability of a sequence of words

$P(\text{He ate the cake with the fork}) = \frac{\text{count(He ate the cake with the fork)}}{\text{count(* * * * * * * *)}}$

total number of observed 7grams
Simple Solution: The Maximum Likelihood Estimate

\[
P(\text{He ate the cake with the fork}) = \frac{\text{count}(\text{He ate the cake with the fork})}{\text{count}(\text{He ate the cake with the forkt})}
\]

\[
P(\text{fork} \mid \text{He ate the cake with the}) = \frac{\text{count}(\text{He ate the cake with the fork})}{\text{count}(\text{He ate the cake with the forkt})}
\]
Simple Solution: The Maximum Likelihood Estimate

**Problem:** even the Web isn’t large enough to enable good estimates of most phrases.

\[ P(\text{He ate the cake with the fork}) = \]
\[
\frac{\text{count}(\text{He ate the cake with the fork})}{\text{count}(\text{* } \text{* } \text{* } \text{* } \text{* } \text{* } \text{* } \text{* } \text{* } \text{* })}
\]

\[ P(\text{fork} \mid \text{He ate the cake with the}) = \]
\[
\frac{\text{count}(\text{He ate the cake with the fork})}{\text{count}(\text{He ate the cake with the } \text{* })}
\]
Problem: even the Web isn’t large enough to enable good estimates of most phrases.

Solution: Estimate from shorter sequences, use more sophisticated probability theory.
**Problem:** even the Web isn’t large enough to enable good estimates of most phrases.

**Solution:** Estimate from shorter sequences, use more sophisticated probability theory.

\[ P(B|A) = \frac{P(B, A)}{P(A)} \iff P(A)P(B|A) = P(B, A) = P(A, B) \]

Example from (Jurafsky, 2017)
**Problem:** even the Web isn’t large enough to enable good estimates of most phrases.

**Solution:** Estimate from shorter sequences, use more sophisticated probability theory.

\[
P(B|A) = \frac{P(B, A)}{P(A)} \iff P(A)P(B|A) = P(B, A) = P(A, B)
\]

\[
P(A, B, C) = P(A)P(B|A)P(C|A, B)
\]

Example from (Jurafsky, 2017)
**Problem:** even the Web isn’t large enough to enable good estimates of most phrases.

**Solution:** Estimate from shorter sequences, use more sophisticated probability theory.

\[
P(B|A) = \frac{P(B, A)}{P(A)} \iff P(A)P(B|A) = P(B, A) = P(A, B)
\]

\[
P(A, B, C) = P(A)P(B|A)P(C| A, B)
\]

**The Chain Rule:**
\[
P(X_1, X_2, ..., X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)...P(X_n|X_1, ..., X_{n-1})
\]

Example from (Jurafsky, 2017)
Problem: even the Web isn’t large enough to enable good estimates of most phrases.

Solution: Estimate from shorter sequences, use more sophisticated probability theory.

\[
P(B|A) = \frac{P(B, A)}{P(A)} \iff P(A)P(B|A) = P(B, A) = P(A, B)
\]

\[
P(A, B, C) = P(A)P(B|A)P(C| A, B)
\]

The Chain Rule: \[
P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|X_1, X_2, \ldots, X_{i-1})
\]

\[
P(X_1, X_2, \ldots, X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)\ldots P(X_n|X_1, \ldots, X_{n-1})
\]
Solution: Estimate from shorter sequences, use more sophisticated probability theory.

\[
P(B|A) = \frac{P(B, A)}{P(A)} \iff P(A)P(B|A) = P(B, A) = P(A, B)
\]

\[
P(A, B, C) = P(A)P(B|A)P(C|A, B)
\]

**The Chain Rule:**

\[
P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|X_1, X_2, ..., X_i)
\]

\[
P(X_1, X_2, ..., X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)...P(X_n|X_1, ..., X_{n-1})
\]
Solution: Estimate from shorter sequences, use more sophisticated probability theory.

\[ P(B|A) = \frac{P(B, A)}{P(A)} \iff P(A)P(B|A) = P(B, A) = P(A, B) \]

\[ P(A, B, C) = P(A)P(B|A)P(C|A, B) \]

The Chain Rule: \[ P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|X_1, X_2, ..., X_i) \]

\[ P(X_1, X_2, ..., X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)...P(X_n|X_1, ..., X_{n-1}) \]

Markov Assumption: \[ P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|X_{i-k}, X_{i-(k-1)}, ..., X_i) \]

\[ P(X_n|X_1..., X_{n-1}) \approx P(X_n|X_{n-k}, ..., X_{n-1}) \text{ where } k < n \]
**Markov Assumption:**

\[ P(X_n | X_{n-1}, X_{n-2}, X_{n-3}, ...) \]

What about Logistic Regression? \( Y = \text{next word} \)

\[ P(Y|X) = P(X_n | X_{n-1}, X_{n-2}, X_{n-3}, ...) \]

Not a terrible option, but \( X_{n-1} \) through \( X_{n-k} \) would be modeled as independent dimensions. Let’s revisit later.

**The Chain Rule:**

\[ P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_1, \ldots, X_{i-1}) \]
**Unigram Model: k = 0;**

\[
P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i)
\]

**Problem:**

Even the Web isn’t large enough to enable good estimates of most phrases.

**P(B|A) = P(B, A) / P(A) ⇔ P(A)P(B|A) = P(B,A) = P(A,B)**

**P(A, B, C) = P(A)P(B|A)P(C| A, B)**

**The Chain Rule:**

\[
P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | X_1, X_2, ..., X_i)
\]

\[
P(X_1, X_2, ..., X_n) = P(X_1)P(X_2 | X_1)P(X_3 | X_1, X_2)...P(X_n | X_1, ..., X_{n-1})
\]

**Markov Assumption:**

\[
P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | X_{i-k}, X_{i-(k-1)}, ..., X_i)
\]

\[P(X_n | X_1..., X_{n-1}) \approx P(X_n | X_{n-k}, ..., X_{n-1}) \quad \text{where } k < n\]
Markov Assumption: 
\[ P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{i-k}, X_{i-(k-1)}, \ldots, X_i) \]
\[ P(X_n | X_1, \ldots, X_{n-1}) \approx P(X_n | X_{n-k}, \ldots, X_{n-1}) \text{ where } k < n \]

Bigram Model: \( k = 1; \)
\[ P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1}) \]

Example generated sentence:

outside, new, car, parking, lot, of, the, agreement, reached

\[ P(X_1 = \text{“outside”}, X_2 = \text{”new”}, X_3 = \text{“car”}, \ldots) \]
\[ \approx P(X_1 = \text{“outside”}) * P(X_2 = \text{”new”}| X_1 = \text{“outside}) * P(X_3 = \text{”car”} | X_2 = \text{”new”) * ... \]

Example from (Jurafsky, 2017)
Language Modeling

Building a model (or system/API) that can answer the following:

- How common is this sequence?
- What is the next word in the sequence?

*a sequence of natural language* → Language Model → How common is this sequence? → What is the next word in the sequence?
Language Modeling

Building a model (or system / API) that can answer the following:

- How common is this sequence?
- What is the next word in the sequence?
- How to build?
Language Modeling

Building a model (or system / API) that can answer the following:

- A sequence of natural language
- How common is this sequence?
- What is the next word in the sequence?

How to build?

Training Corpus

training (fit, learn)
Language Modeling

Building a model (or system/API) that can answer the following:

- How common is this sequence?
- What is the next word in the sequence?

*a sequence of natural language* 

*Training Corpus* 

Language Model 

training (fit, learn)
### Bigram Counts

<table>
<thead>
<tr>
<th>first word</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>0</td>
<td>82</td>
<td>1</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>0</td>
<td>1</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>

Example from (Jurafsky, 2017)
### Training Corpus

<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>0</td>
<td>82</td>
<td>1</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>0</td>
<td>1</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>

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>2533</td>
<td>927</td>
<td>2417</td>
<td>746</td>
<td>158</td>
<td>1093</td>
<td>341</td>
<td>278</td>
</tr>
</tbody>
</table>

Training (fit, learn)
Language Modeling

Building a model (or system / API) that can answer the following:

- A sequence of natural language
- How common is this sequence?
- What is the next word in the sequence?

Training Corpus

Training (fit, learn)

Example from (Jurafsky, 2017)

Bigram model:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1})$$

Need to estimate: $$P(X_i | X_{i-1}) = \frac{\text{count}(X_{i-1} X_i)}{\text{count}(X_{i-1})}$$

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>0</td>
<td>82</td>
<td>1</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>0</td>
<td>1</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>2533</td>
<td>927</td>
<td>2417</td>
<td>746</td>
<td>158</td>
<td>1093</td>
<td>341</td>
<td>278</td>
</tr>
</tbody>
</table>
Language Modeling

Building a model (or system / API) that can answer the following:

- How common is this sequence?
- What is the next word in the sequence?

Training Corpus

**Bigram model:**

\[ P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1}) \]

Need to estimate: \( P(X_i | X_{i-1}) = \frac{\text{count}(X_{i-1} X_i)}{\text{count}(X_{i-1})} \)
Building a model (or system/API) that can answer the following:

- How common is this sequence?
- What is the next word in the sequence?

**Training Corpus**

*Training (fit, learn)*

First word \( (X_{i-1}) \)

<table>
<thead>
<tr>
<th>first word ( (X_{i-1}) )</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>0.002</td>
<td>0.33</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00079</td>
</tr>
<tr>
<td>want</td>
<td>0.0022</td>
<td>0</td>
<td>0.66</td>
<td>0.0011</td>
<td>0.0065</td>
<td>0.0065</td>
<td>0.0054</td>
<td>0.0011</td>
</tr>
</tbody>
</table>

**Example from (Jurafsky, 2017)**

**Bigram model:**

Need to estimate:

\[
P(X_i | X_{i-1}) = \frac{\text{count}(X_{i-1} X_i)}{\text{count}(X_{i-1})}
\]
Language Modeling

Building a model (or system / API) that can answer the following:

- What is the next word in the sequence?
- How common is this sequence?

Training Corpus

**Training** (fit, learn)

First word ($X_{i-1}$)

<table>
<thead>
<tr>
<th>word</th>
<th>0.002</th>
<th>0.33</th>
<th>0.0022</th>
<th>0.00083</th>
<th>0.0063</th>
<th>0.014</th>
</tr>
</thead>
<tbody>
<tr>
<td>want</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>0.0017</td>
<td>0.66</td>
<td>0.0011</td>
<td>0.28</td>
<td>0.0027</td>
<td>0.014</td>
</tr>
<tr>
<td>eat</td>
<td>0.0036</td>
<td>0.0036</td>
<td>0.00083</td>
<td>0.021</td>
<td>0.0027</td>
<td>0.014</td>
</tr>
<tr>
<td>chinese</td>
<td>0.0065</td>
<td>0.0065</td>
<td>0.0054</td>
<td>0.056</td>
<td>0.0063</td>
<td>0.014</td>
</tr>
<tr>
<td>food</td>
<td>0.00079</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example from (Jurafsky, 2017)**

**Bigram model:**

Need to estimate:

$$P(X_i | X_{i-1}) = \frac{\text{count}(X_{i-1} X_i)}{\text{count}(X_{i-1})}$$
Language Modeling

Building a model (or system / API) that can answer the following:

- How common is this sequence?
- What is the next word in the sequence?

Training Corpus

Bigram model:

\[ P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1}) \]

Need to estimate: \( P(X_i | X_{i-1}) = \frac{\text{count}(X_{i-1} X_i)}{\text{count}(X_{i-1})} \)
Language Modeling

Building a model (or system / API) that can answer the following:

- How common is this sequence?
- What is the next word in the sequence?

**Training Corpus** (fit, learn)

Example from (Jurafsky, 2017)

**Bigram model:**

\[ P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1}) \]

Need to estimate: \( P(X_i | X_{i-1}) = \text{count}(X_{i-1} X_i) / \text{count}(X_{i-1}) \)

**Trigram model:**

\[ P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1}, X_{i-2}) \]

Need to estimate: \( P(X_i | X_{i-1}, X_{i-2}) = \text{count}(X_{i-2} X_{i-1} X_i) / \text{count}(X_{i-2} X_{i-1}) \)
Language Modeling

Building a model (or system / API) that can answer the following:

- Training Corpus
- Trained Language Model
- How common is this sequence?
- What is the next word in the sequence?

a sequence of natural language

training (fit, learn)
Language Modeling

Building a model (or system / API) that can answer the following:

- How common is this sequence?
- What is the next word in the sequence?

Training Corpus → *food* → Trained Language Model → training (fit, learn) → How common is this sequence? → What is the next word in the sequence?
Language Modeling

Building a model (or system / API) that can answer the following:

- a sequence of natural language
- How common is this sequence?
- What is the next word in the sequence?

Training Corpus

Test?

(fit, learn)
Language Modeling

Building a model (or system / API) that can answer the following:

- How common is this sequence?
- What is the next word in the sequence?

Training Corpus

Test Corpus

Test:
Feed the model $X_1...X_{i-1}$ and see how well it predicts $X_i$. 

Trained Language Model

Perplexity
Language Modeling

Building a model (or system / API) that can answer the following:

- How common is this sequence?
- What is the next word in the sequence?

Training Corpus

Test Corpus

Test:
Feed the model $X_1...X_{i-1}$ and see how well it predicts $X_i$.

Perplexity:

$$PP(W) = \left( \frac{1}{P(w_1w_2...w_N)} \right)^\frac{1}{N}$$

$$= \sqrt[N]{\frac{1}{P(w_1w_2...w_N)}}$$
Evaluation

Test Corpus → Trained Language Model → What is the next word in the sequence?

Perplexity

\[ PP(W) = \frac{1}{P(w_1w_2...w_N)^\frac{1}{N}} \]

\[ = \sqrt[N]{\frac{1}{P(w_1w_2...w_N)}} \]
Evaluation

What is the next word in the sequence?

Test Corpus → Trained Language Model

Apply Chain Rule:

\[
PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}
\]

Perplexity

\[
PP(W) = P(w_1w_2 \ldots w_N)^{-\frac{1}{N}}
\]

\[
= \sqrt[N]{\frac{1}{P(w_1w_2 \ldots w_N)}}
\]
Evaluation

Apply Chain Rule:

Thus, PP for Bigrams:

\[
PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}
\]

Thus, PP for Bigrams:

\[
PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}
\]

What is the next word in the sequence?

Perplexity

Apply Chain Rule:

Thus, PP for Bigrams:

\[
PP(W) = \frac{1}{\sqrt[N]{P(w_1w_2 \ldots w_N)}}
\]

Thus, PP for Bigrams:

\[
PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}
\]
Coding Example: Modeling Tweets from POS data

1. Count unigrams, bigrams, and trigrams
2. Train probabilities for unigram, bigram, and trigram models (over training)
3. Generate language
   
   Trigram model when good evidence (high counts)
   
   Backing off to bigram or even unigram
Coding Example: Modeling Tweets from POS data

Practical Considerations:

- Use log probability to keep numbers reasonable and save computation. (uses addition rather than multiplication)

- Out-of-vocabulary (OOV)
  Choose minimum frequency and mark as <OOV>

- Sentence start and end: <s> this is a sentence </s>
Zeros and Smoothing

\[
P(X_i \mid X_{i-1})
\]

<table>
<thead>
<tr>
<th>first word (X_{i-1}) \</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>0.002</td>
<td>0.33</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00079</td>
</tr>
<tr>
<td>want</td>
<td>0.0022</td>
<td>0</td>
<td>0.66</td>
<td>0.0011</td>
<td>0.0065</td>
<td>0.0065</td>
<td>0.0054</td>
<td>0.0011</td>
</tr>
<tr>
<td>to</td>
<td>0.00083</td>
<td>0</td>
<td>0.0017</td>
<td>0.28</td>
<td>0.00083</td>
<td>0</td>
<td>0.0025</td>
<td>0.087</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>0.0027</td>
<td>0</td>
<td>0.021</td>
<td>0.0027</td>
<td>0.056</td>
<td>0</td>
</tr>
<tr>
<td>chinese</td>
<td>0.0063</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.52</td>
<td>0.0063</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>0.014</td>
<td>0</td>
<td>0.014</td>
<td>0</td>
<td>0.00092</td>
<td>0.0037</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>0.0059</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0029</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>0.0036</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Example from (Jurafsky, 2017)
Zeros and Smoothing

Laplace ("Add one") smoothing: add 1 to all counts

<table>
<thead>
<tr>
<th>first word</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>0</td>
<td>82</td>
<td>1</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>
Zeros and Smoothing

Laplace ("Add one") smoothing: add 1 to all counts

<table>
<thead>
<tr>
<th>first word \ second word</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>6</td>
<td>828</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>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>1</td>
<td>7</td>
<td>212</td>
</tr>
<tr>
<td>eat</td>
<td>1</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>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>2</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>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>
<td>1</td>
</tr>
</tbody>
</table>
### Unsmoothed probs

The table presents the unsmoothed probabilities of the second word given the first word, denoted as $P(X_i | X_{i-1})$. The table is an example from (Jurafsky, 2017).
Smoothed

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

Example from (Jurafsky, 2017)

<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>0.0025</td>
<td>0.0025</td>
<td>0.0025</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>
Why Smoothing? Generalizes

Original

With Smoothing

(Example from Jurafsky / Originally Dan Klein)
Why Smoothing? Generalizes

Add-one is blunt: can lead to very large changes.

More Advanced:

Good-Turing Smoothing
Kneser-Nay Smoothing

These are outside scope of course because we will eventually cover, even stronger, deep learning based models.
Why Smoothing?

What about Logistic Regression? \( Y = \text{next word} \)
\[
P(Y|X) = P(X_n | X_{n-1}, X_{n-2}, X_{n-3}, \ldots)
\]

Not a terrible option, but \( X_{n-1} \) through \( X_{n-k} \)
would be modeled as independent dimensions. Let’s revisit later.
What about Logistic Regression? Y = next word

\[
P(Y|X) = P(X_n | X_{n-1}, X_{n-2}, X_{n-3}, \ldots)
\]

Not a terrible option, but \( X_{n-1} \) through \( X_{n-k} \) would be modeled as independent dimensions. Let’s revisit later. Could use:

\[
P(X_n | X_{n-1}, [X_{n-1} X_{n-2}], [X_{n-1} X_{n-2} X_{n-3}], \ldots)
\]
Language Modeling Summary

- Two versions of assigning probability to sequence of words
- Applications
- The Chain Rule, The Markov Assumption: \[ P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{i-k}, X_{i-(k-1)}, \ldots, X_i) \]
- Training a unigram, bigram, trigram model based on counts
- Evaluation: Perplexity
- Zeros, Low Counts, and Generalizability
- Add-one smoothing