Language Modeling

CSE354 - Spring 2021
Task

- Language Modeling (i.e. auto-complete)

how?

- Probabilistic Modeling
  - Probability Theory
  - Logistic Regression
  - Sequence Modeling
Task

- Language Modeling (i.e. auto-complete)

how?

- Probabilistic Modeling
  - Probability Theory
  - Logistic Regression
  - Sequence Modeling

- Eventually: Deep Learning
  - Recurrent Neural Nets
  - Transformer Networks
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”
  (example from Jurafsky, 2017; ..did you say "giraffe ski 2,017"? )
Timeline: *Language Modeling* and *Vector Semantics*

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1913</td>
<td>Markov: Probability that next letter would be vowel or consonant.</td>
</tr>
<tr>
<td>1948</td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td></td>
</tr>
</tbody>
</table>

~logarithmic scale

- **Language Models**
- **Vector Semantics**
- **LMs + Vectors**

GPT3, XLNet, RoBERTA
Markov: Probability that next letter would be vowel or consonant.

These (or similar) are behind almost all state-of-the-art modern NLP systems.
Timeline: **Language Modeling and Vector Semantics**

- **1913**: Markov: Probability that next letter would be vowel or consonant.
- **1948**: Shannon: *A Mathematical Theory of Communication* (first digital language model)
- **1980**: Brown et al.: *Class-based ngram models of natural language*
- **2003**: Blei et al.: *LDA Topic Modeling*
- **2010**: Mikolov: *word2vec*
- **2018**: Collobert and Weston: *A unified architecture for natural language processing: Deep neural networks...*

- **1948**: Osgood: *The Measurement of Meaning*
- **1980**: Deerwater: *Indexing by Latent Semantic Analysis (LSA)*
- **2003**: Switzer: *Vector Space Models*
- **2010**: Bengio: Neural-net based embeddings
- **2018**: XLNet
- **2018**: RoBERTA
- **2018**: GPT3
- **2018**: GPT

**~logarithmic scale**
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 $P(w_5 | w_1, w_2, w_3, w_4) = P(w_n | w_1, w_2, \ldots, w_{n-1})$

: 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

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

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

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

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

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

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

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

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

\[ P(B|A) = P(B, A) / P(A) \iff P(A)P(B|A) = P(B,A) = P(A,B) \]

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

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

\[ P(B|A) = P(B,A) / P(A) \iff P(A)P(B|A) = P(B,A) = P(A,B) \]

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

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

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

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

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

\[ P(B|A) = P(B, A) / P(A) \iff P(A)P(B|A) = P(B,A) = P(A,B) \]

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

The Chain Rule: 
\[ P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|X_1, X_2, \ldots, X_i) \]
\[ P(X_1, X_2, \ldots, X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)\ldots P(X_n|X_1, \ldots, X_{n-1}) \]
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

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

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?

Input: a sequence of natural language

Output: Language Model
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:

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

How to build?

training (fit, learn)

a sequence of natural language
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

Food corpus from Jurafsky (2018). Samples:

- *can you tell me about any good cantonese restaurants close by*
- *mid priced thai food is what i’m looking for*
- *tell me about chez panisse*
- *can you give me a listing of the kinds of food that are available*
- *i’m looking for a good place to eat breakfast*
- *when is caffe venezia open during the day*
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

<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)
## 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

Training (fit, learn)

**Example from (Jurafsky, 2017)**

<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>

**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>
<tr>
<td>want</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>eat</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>chinese</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### 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) 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: \( P(X_i | X_{i-1}) = \frac{\text{count}(X_{i-1} X_i)}{\text{count}(X_{i-1})} \)

<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>

The table above shows the bigram counts for various words. The counts are used to estimate the conditional probabilities of the next word in the sequence given the previous word.
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**

<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>0.002</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>0.00092</td>
<td>0.0037</td>
<td>0</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>

**Bigram model:**

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

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

Training (fit, learn)

\[
X_i \quad \text{first word} \quad X_{i-1} \quad \text{second word} \quad P(X_i \mid X_{i-1})
\]

Example from (Jurafsky, 2017)

Bigram model:

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

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

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

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

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>Xi-1</td>
<td>0.002</td>
<td>0.33</td>
<td>0.66</td>
<td>0.28</td>
<td>0.0017</td>
<td>0.0063</td>
<td>0.0025</td>
<td>0.087</td>
</tr>
<tr>
<td>Xi</td>
<td>0.0022</td>
<td>0.0011</td>
<td>0.0065</td>
<td>0.0065</td>
<td>0.0054</td>
<td>0.0063</td>
<td>0.056</td>
<td>0.0011</td>
</tr>
<tr>
<td>Xi+1</td>
<td>0.00083</td>
<td>0.0017</td>
<td>0.0083</td>
<td>0.0083</td>
<td>0.0025</td>
<td>0.0063</td>
<td>0.0063</td>
<td>0.0011</td>
</tr>
</tbody>
</table>

Example from (Jurafsky, 2017)

- Bigram model: $P(X_i | X_{i-1}) = \frac{\text{count}(X_{i-1} X_i)}{\text{count}(X_{i-1})}$
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?

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: \( P(X_i | X_{i-1}) = \frac{\text{count}(X_{i-1} X_i)}{\text{count}(X_{i-1})} \)
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**

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

**Trigram model:**

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

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

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

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

- a sequence of natural language
- Training Corpus
- Trained Language Model
- 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)**
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)

Test?
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?

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

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

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

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

Perplexity

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

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

What is the next word in the sequence?

Test Corpus

Trained Language Model

Apply Chain Rule:

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

\[ = \sqrt[N]{\frac{1}{P(w_1w_2w_3 \ldots w_N)}} \]

Perplexity
Evaluation

Test Corpus → Trained Language Model

What is the next word in the sequence?

Perplexity

Apply Chain Rule:

Thus,

PP for Bigrams:

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

\[
PP(W) = \left( \frac{1}{P(w_1w_2w_3...w_N)} \right)^{1/N}
\]
Evaluation

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

Thus, PP for Bigrams:

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

Reasoning:
1) Inverse of probability (i.e. minimize perplexity = maximize likelihood)
2) (weighted) average branching factor

Thus, PP for Bigrams:

\[
PP(W) = P(w_1w_2w_3\ldots w_N)^{\frac{1}{N}}
\]

Apply Chain Rule:

\[
PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}
\]
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>
Advantage: models word probability at beginning or end.
### Zeros and Smoothing

Consider the probability distribution $P(X_i | X_{i-1})$ for subsequent words $X_i$ given the previous word $X_{i-1}$. 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>0.002</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>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>
Laplace ("Add one") smoothing: add 1 to all counts

<table>
<thead>
<tr>
<th>first word</th>
<th>_</th>
<th>second word</th>
<th>_</th>
<th>_</th>
<th>_</th>
<th>_</th>
<th>_</th>
<th>_</th>
<th>_</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>6</td>
<td>want</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>to</td>
<td>609</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>eat</td>
<td>5</td>
<td>687</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>7</td>
<td>212</td>
</tr>
<tr>
<td>eat</td>
<td>1</td>
<td>chinese</td>
<td>3</td>
<td>1</td>
<td>17</td>
<td>3</td>
<td>43</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>chinese</td>
<td>2</td>
<td>food</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>83</td>
<td>16</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>food</td>
<td>16</td>
<td>lunch</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>lunch</td>
<td>3</td>
<td>spend</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>spend</td>
<td>2</td>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Unsmoothed probs

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>0.002</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>
Smoothed

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

(vocabulary size)

Example from (Jurafsky, 2017)
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 for now. We will eventually cover, even stronger, deep learning based models.
Why Smoothing?

What about Logistic Regression? Y = next word

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

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? $Y = \text{next word}$

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

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

$P(X_n | X_{n-1}, [X_{n-1} X_{n-2}], [X_{n-1} X_{n-2} X_{n-3}], ...)$
Example how to produce language generator

1. Count unigrams, bigrams, and trigrams
2. Train probabilities for unigram, bigram, and trigram models (over training)
   a. with smoothing
   b. without smoothing
3. Generate language: Given previous word or previous 2 words, take a random draw from what words are most likely to be next.
   Trigram model when good evidence (high counts)
   Backing off to bigram or even unigram
Limitation: Long distance dependencies

The horse which was raced past the barn tripped.
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