Language Modeling and Natural Language Generation
CSCI 1460: Computational Linguistics
Lecture 9

Ellie Pavlick
Fall 2022
Consider the below network. It is a single-layer perceptron using one-hot encodings. I am using a "bag of vectors" approach, i.e., I summed up the word representations element-wise to obtain a representation for a phrase. What will be the predicted next word?

Vocab: \{I, love, you, NLP, ML\}
Input: I love
Consider the one-hot encodings. I am writing a program in Python to learn about how word representations are used to construct a phrase. What will be the output of the program?

```python
import numpy as np

v = np.array([[1, 1, 0, 0, 0]])

w = np.array([[[0, 2, 2, 1, 2],
               [0, 0, 2, 2, 1],
               [0, 1, 0, 0, 1],
               [1, 0, 0, 0, 0],
               [1, 0, 0, 1, 0]]])

print(v.dot(w))
print(v.T.dot(w))
print(np.matmul(v, w))
print(np.matmul(v.T, w))
```

The output should be:

```
[0 2 4 3 3]
[0 2 4 3 3]
[0 2 4 3 3]
[0 2 4 3 3]
```
I am building an MLP classifier to predict whether a review is positive or negative (the same as in question 1). I train my word embedding layer in the process of training the network, and after I've finished training, I cluster words using the trained embeddings. Which of the following should I expect to see?

- words that occur in positive reviews cluster together, and words that occur in negative reviews cluster together (e.g., good/great/awesome vs. bad/awful/terrible)
- nouns cluster together, and verbs cluster together (e.g., food/service/ambiance vs. eat/drink/meet)
- content words cluster together, and stop words cluster together (e.g., food/eat/favorite vs. is/of/and)
Topics

- What is language modeling? When do we use it?
- Ngram language models
- Smoothing
- Perplexity
Topics

- What is language modeling? When do we use it?
- Ngram language models
- Smoothing
- Perplexity
Language Modeling

Definition

• Assign a probability to a sequence of words

OR

• Given a sequence of words, predict the most likely next word

OR

• Generate likely sequences of words
Language Modeling

Applications

• Unconstrained text generation (fun, but not super practical)
• Conditional text generation, e.g.:
  • Machine translation: e.g., find most likely English sentence given Mandarin sentence
  • Speech recognition: e.g., find most likely English sentence given acoustic input
  • Summarization: e.g., find most likely 50 word English document given a 1000 word English document
• …
• Representation learning
Language Modeling
Application: Noisy Channel Speech Recognition Model

Noisy Channel Model

P(sentence|audio wave) x P(sentence)

Noisy Channel Model
Language Modeling
Application: Noisy Channel Speech Recognition Model

\[
\begin{align*}
P(\text{Its easy to recognize speech}) & = 0.5 \\
P(\text{Its easy to wreck a nice beach}) & = 0.5 \\
P(\text{Its easy to recognize speech}) & = 0.1 \\
\end{align*}
\]

Noisy Channel Model

audio wave

Its easy to recognize speech

Its easy to wreck a nice beach
Language Modeling
Application: Noisy Channel Speech Recognition Model

Noisy Channel Model

P(Its easy to recognize speech) = 0.5

P(Its easy to wreck a nice beach) = 0.1

Acoustic Model

P(Its easy to recognize speech | audio wave) = 0.3

P(Its easy to wreck a nice beach | audio wave) = 0.5

Its easy to recognize speech
Language Modeling
Application: Noisy Channel Speech Recognition Model

- $P(\text{Its easy to recognize speech}) = 0.5$
- $P(\text{Its easy to wreck a nice beach}) = 0.1$

Noisy Channel Model
Topics

- What is language modeling? When do we use it?
- **Ngram language models**
- Smoothing
- Perplexity
Ngram Language Models
Directly computing corpus stats

• Simple idea: Just compute the probability $P(w_0, \ldots, w_n)$ directly from a corpus!

• I.e.:

  # occurrences of $w_0, \ldots, w_n$
  # sequences of length $n$
Ngram Language Models
Directly computing corpus stats

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
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

\[ P(\text{tell me about chez panisse}) = \]
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

\[ P(\text{tell me about chez panisse}) = 1 \]
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

\[ P(\text{tell me about chez panisse}) = 1 \]
Ngram Language Models
Directly computing corpus stats

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

P(tell me about chez panisse) = \frac{1}{\text{number of contexts}}
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

\[ P(\text{tell me about chez panisse}) = \frac{1}{\text{number of possible options}} \]
Ngram Language Models
Directly computing corpus stats

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

\[
P(\text{tell me about chez panisse}) = \frac{1}{51}\]
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

\[ P(\text{tell me about chez panisse}) = \frac{1}{51} \]

Problems?
Ngram Language Models
Directly computing corpus stats

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

\[ P(\text{tell me about caffe venezia}) = \frac{0}{51} \]
Ngram Language Models
Directly computing corpus stats

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

\[ P(\text{tell me about caffe venezia}) = \frac{0}{51} \]
Ngram Language Models

Unigram Language Model

\[ P(w_0 \ldots w_n) = P(w_0) \times P(w_1 | w_0) \times P(w_2 | w_0, w_1) \times \ldots \times P(w_n | w_0 \ldots w_{n-1}) \]
Ngram Language Models

Unigram Language Model

\[ P(w_0 \ldots w_n) = P(w_0) \times P(w_1 \mid w_0) \times P(w_2 \mid w_0, w_1) \times \ldots \times P(w_n \mid w_0 \ldots w_{n-1}) \]

Not helpful (yet). Still requires observing \( w_0 \ldots w_n \)
Ngram Language Models

Unigram Language Model

\[ P(w_0 \ldots w_n) = P(w_0) \times P(w_1 | w_0) \times P(w_2 | w_0, w_1) \times \ldots \times P(w_n | w_0 \ldots w_{n-1}) \]

\[ P(w_i | w_0 \ldots w_{i-1}) \approx P(w_i) \]

Independence Assumption
(Just like Naive Bayes)
Ngram Language Models

Unigram Language Model

\[ P(w_0 \ldots w_n) = P(w_0) \times P(w_1 | w_0) \times P(w_2 | w_0, w_1) \times \ldots \times P(w_n | w_0 \ldots w_{n-1}) \]

\[ P(w_i | w_0 \ldots w_{i-1}) \approx P(w_i) \]

\[ P(w_0 \ldots w_n) \approx P(w_0) \times P(w_1) \times P(w_2) \times \ldots \times P(w_n) \]
Ngram Language Models

Unigram Language Model

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

\[ P(w_0 \ldots w_n) \approx P(w_0) \times P(w_1) \times P(w_2) \times \ldots \times P(w_n) \]

\[ P(\text{tell me about caffe venezia}) = P(\text{tell}) \times P(\text{me}) \times P(\text{about}) \times P(\text{caffe}) \times P(\text{venezia}) \]
Ngram Language Models

Unigram Language Model

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

\[
P(w_0 \ldots w_n) \approx P(w_0) \times P(w_1) \times P(w_2) \times \ldots \times P(w_n)
\]

\[
P(\text{tell me about caffe venezia}) = \frac{2}{56} \times \frac{3}{56} \times \frac{3}{56} \times \frac{1}{56} \times \frac{1}{56}
\]
Ngram Language Models

Unigram Language Model

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

\[ P(w_0 \ldots w_n) \approx P(w_0) \times P(w_1) \times P(w_2) \times \ldots \times P(w_n) \]

\[ P(\text{tell me about caffe venezia}) = \frac{2}{56} \times \frac{3}{56} \times \frac{3}{56} \times \frac{1}{56} \times \frac{1}{56} \]

\[ = 3.26 \times 10^{-8} \]
Ngram Language Models

Unigram Language Model

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

\[
P(w_0 \ldots w_n) \approx P(w_0) \times P(w_1) \times P(w_2) \times \ldots \times P(w_n)
\]

\[
P(\text{tell me about caffe venezia})
= \log(2/56) + \log(3/56) + \log(3/56) + \log(1/56) + \log(1/56)
= -13.9
\]
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

P(tell me about caffe venezia)

P(caffe about tell venezia me)

Which is more probable?
Ngram Language Models

Bigram Language Model

\[ P(w_0 \ldots w_n) = P(w_0) \times P(w_1 | w_0) \times P(w_2 | w_0, w_1) \times \ldots \times P(w_n | w_0 \ldots w_{n-1}) \]

\[ P(w_i | w_0 \ldots w_{i-1}) \approx P(w_i | w_{i-1}) \]

Markov Assumption!
Ngram Language Models

Bigram Language Model

\[ P(w_0 \ldots w_n) = P(w_0) \times P(w_1 \mid w_0) \times P(w_2 \mid w_0, w_1) \times \ldots \times P(w_n \mid w_0 \ldots w_{n-1}) \]

\[ P(w_i \mid w_0 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1}) \]

\[ P(w_0 \ldots w_n) \approx P(w_0 \mid <s>) \times P(w_1 \mid w_0) \times P(w_2 \mid w_1) \times \ldots \times P(w_n \mid w_{n-1}) \]
Ngram Language Models

Bigram Language Model

\[
P(w_0 \ldots w_n) \approx P(w_0 \mid \langle s \rangle) \times P(w_1 \mid w_0) \times P(w_2 \mid w_1) \times \ldots \times P(w_n \mid w_{n-1})
\]

\[
P(\text{tell me about caffe venezia}) = P(\text{tell} \mid \langle s \rangle) \times P(\text{me} \mid \text{tell}) \times P(\text{about} \mid \text{me}) \times P(\text{caffe} \mid \text{about}) \times P(\text{venezia} \mid \text{caffe}) \times P(\langle s \rangle \mid \text{venezia})
\]
Ngram Language Models

Unigram Language Model

\[ P(w_0 \ldots w_n) \approx P(w_0) \times P(w_1) \times P(w_2) \times \ldots \times P(w_n) \]

Bigram Language Model

\[ P(w_0 \ldots w_n) \approx P(w_0 | <s>) \times P(w_1 | w_0) \times P(w_2 | w_1) \times \ldots \times P(w_n | w_{n-1}) \]

n-gram Language Model

\[ P(w_0 \ldots w_n) \approx \prod_{i=0}^{n} P(w_i | w_{i-(n-1)} \ldots w_{i-1}) \]
Ngram Language Models

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have

Unigram
Ngram Language Models

To him swallowed both. Which. Of are ay device and

Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow

Unigram

Bigram
Ngram Language Models

To him swallowed both. Which. Of life have are ay device and rote.

Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow

Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done

Unigram

Bigram

Trigram
Ngram Language Models

Unigram

To him swallowed confess hear both. Which. Of are ay device and

Bigram

Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry.

Trigram

Fly, and will rid me these news of price. The parting.

King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in;
Topics

• What is language modeling? When do we use it?
• Ngram language models
• **Smoothing**
• Perplexity
Ngram Language Models

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

\[
P(w_0 \ldots w_n) \approx P(w_0 | <s>) \times P(w_1 | w_0) \times P(w_2 | w_1) \times \ldots \times P(w_n | w_{n-1})
\]

\[
P(\text{tell me about caffe venezia}) = P(\text{tell}|<s>) \times P(\text{me}|\text{tell}) \times P(\text{about}|\text{me}) \times P(\text{caffe}|\text{about}) \times P(\text{venezia}|\text{caffe}) \times P(</s>|\text{venezia})
\]
Ngram Language Models

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

\[
P(w_0 \ldots w_n) \approx P(w_0 | <s>) \times P(w_1 | w_0) \times P(w_2 | w_1) \times \ldots \times P(w_n | w_{n-1})
\]

\[
P(\text{tell me about caffe venezia}) = P(\text{tell}|<s>) \times P(\text{me}|\text{tell}) \times P(\text{about}|\text{me}) \times P(\text{caffe}|\text{about}) \times P(\text{venezia}|\text{caffe}) \times P(</s>|\text{venezia})\]
Ngram Language Models

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

\[ P(w_0...w_n) \approx P(w_0 | <s>) \times P(w_1 | w_0) \times P(w_2 | w_1) \times ... \times P(w_n | w_{n-1}) \]

\[
P(\text{tell me about caffe venezia}) = P(\text{tell}|<s>) \times P(\text{me}|\text{tell}) \times P(\text{about}|\text{me}) \times P(\text{caffe}|\text{about}) \times P(\text{venezia}|\text{caffe}) \times P(</s>|\text{venezia})
\]

Zero counts!
Generalization in LMs
Smoothing Strategies

• Laplace Smoothing (i.e., “Add-One” smoothing)
• Backoff
• Kneser-Ney Smoothing
Generalization in LMs

Smoothing Strategies

- Laplace Smoothing (i.e., “Add-One” smoothing)
- Backoff
- Kneser-Ney Smoothing
### Smoothing

#### Laplace ("Add One")

<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>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>want</td>
<td>2</td>
<td>608</td>
<td>1</td>
<td>6</td>
<td></td>
<td>6</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td></td>
<td>4</td>
<td>686</td>
<td></td>
<td>2</td>
<td>6</td>
<td>211</td>
</tr>
<tr>
<td>eat</td>
<td></td>
<td>2</td>
<td></td>
<td>16</td>
<td>2</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>chinese</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>82</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>food</td>
<td>15</td>
<td>15</td>
<td></td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lunch</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spend</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Smoothing

### Laplace (“Add One”)

The Laplace (“Add One”) smoothing technique is a method used to address the problem of zero probabilities in a probability distribution. When estimating the probability of a word sequence, adding one to each count helps to avoid zero probabilities, which can lead to poor performance in natural language processing tasks.

The table below represents the smoothed probability distribution $P(w_{	ext{want}} | w_{i})$ for the word sequence "I want to eat Chinese lunch spend". Each cell contains the count of the corresponding word pair plus one, following the Laplace smoothing approach.

<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>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>want</td>
<td>2</td>
<td>608</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td>4</td>
<td>686</td>
<td>2</td>
<td>6</td>
<td>211</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eat</td>
<td></td>
<td></td>
<td>2</td>
<td>16</td>
<td>2</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chinese</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>82</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>food</td>
<td>15</td>
<td>15</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lunch</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spend</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Smoothing**

**Laplace ("Add One")**

\[
P(w_n | w_{n-1}) = \frac{\#(w_{n-1}w_n)}{\#w_{n-1}}
\]

<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>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>want</td>
<td>2</td>
<td>608</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td>4</td>
<td>686</td>
<td>2</td>
<td></td>
<td>6</td>
<td>211</td>
<td></td>
</tr>
<tr>
<td>eat</td>
<td></td>
<td>2</td>
<td>16</td>
<td>2</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chinese</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>82</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>food</td>
<td>15</td>
<td>15</td>
<td></td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lunch</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spend</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Smoothing**

**Laplace (“Add One”)**

<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>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>

**Simple Idea:** Just add 1 to everything!
Smoothing
Laplace (“Add One”)

Need to renormalize to keep it a probability distribution

<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>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>1</td>
<td>83</td>
<td>2</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>2</td>
<td>1</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>
Smoothing
Laplace (“Add One”)

\[ P(w_n | w_{n-1}) = \frac{\#(w_{n-1}w_n) + 1}{\sum_w (\#(w_{n-1}w) + 1)} \]

<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>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>1</td>
<td>83</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>
Smoothing
Laplace ("Add One")

\[ P(w_n | w_{n-1}) = \frac{\#(w_{n-1}w_n) + 1}{\#w_{n-1} + V} \]

<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>6</td>
<td>828</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</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>
Smoothing
Laplace ("Add One")

\[ P(w_n | w_{n-1}) = \frac{\#(w_{n-1}w_n) + 1}{\#w_{n-1} + V} \]

<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>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>
</tbody>
</table>

Often interpreted as "discounting".

I.e., we borrow probability mass from high-count words in order to make room for unseen words.
Generalization in LMs

Smoothing Strategies

- Laplace Smoothing (i.e., “Add-One” smoothing)
- Backoff
- Kneser-Ney Smoothing
Generalization in LMs
Smoothing Strategies

• Laplace Smoothing (i.e., “Add-One” smoothing)
• Backoff/Interpolation
• Kneser-Ney Smoothing
Smoothing
Backoff

- Intuition: We can estimate the probability of a longer sequence from the probabilities of its subsequences
- If an ngram of length $n$ is not observed, use the corresponding length $n-1$ ngram instead

$$P(\text{“tell me about caffe venezia”}) \approx P(\text{“me about caffe venezia”})$$
$$\approx P(\text{“about caffe venezia”}) \approx P(\text{“caffe venezia”}) \approx P(\text{“venezia”})$$
Smoothing
Interpolation

• All counts are estimated using a weighted combination of smaller ngrams

• $P (\text{"tell me about caffe venezia"}) = \lambda_1 P (\text{"tell me about caffe venezia"}) \times \lambda_2 P (\text{"me about caffe venezia"}) \times \lambda_3 P (\text{"about caffe venezia"}) \times \lambda_4 P (\text{"caffe venezia"}) \times \lambda_5 P (\text{"venezia"})$

• Requires some renormalization (like in Laplace Smoothing)
Generalization in LMs
Smoothing Strategies

• Laplace Smoothing (i.e., “Add-One” smoothing)
• Backoff
• Kneser-Ney Smoothing
Smoothing
Kneser-Ney

- State-of-the-art smoothing algorithm that combines several ideas:
  1. Absolute discounting (estimated from data)
  2. Replace ngram probabilities with *continuation* probabilities
Smoothing
Kneser-Ney

- State-of-the-art smoothing algorithm that combines several ideas:
  1. Absolute discounting (estimated from data)
  2. Replace ngram probabilities with *continuation* probabilities

\[
P_{\text{AbsoluteDiscounting}}(w_i | w_{i-1}) = \frac{C(w_{i-1}w_i) - d}{\sum_v C(w_{i-1}v)} + \lambda(w_{i-1})P(w_i)
\]
Smoothing
Kneser-Ney

• State-of-the-art smoothing algorithm that combines several ideas:
  1. Absolute discounting (estimated from data)
  2. Replace ngram probabilities with \textit{continuation} probabilities

\[ P_{\text{Absolute Discounting}}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) - d}{\sum_v C(w_{i-1}v)} + \lambda(w_{i-1})P(w_i) \]

Similar to Laplace, except we discount factor doesn't necessarily correspond to adding 1...
Smoothing
Kneser-Ney

• State-of-the-art smoothing algorithm that combines several ideas:
  1. Absolute discounting (estimated from data)
  2. Replace ngram probabilities with continuation probabilities

Instead, we estimate it from data!

\[ P_{\text{Absolute Discounting}}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) - d}{\sum_v C(w_{i-1}v)} + \lambda(w_{i-1})P(w_i) \]
Smoothing
Kneser-Ney

• State-of-the-art smoothing algorithm that combines several ideas:
  1. Absolute discounting (estimated from data)
  2. Replace n-gram probabilities with continuation probabilities

\[
P_{\text{AbsoluteDiscounting}}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) - d}{\sum_v C(w_{i-1}v)} + \lambda(w_{i-1})P(w_i)
\]

Can be fixed (e.g., 0.75) or a function of \(n\)
Smoothing
Kneser-Ney

• State-of-the-art smoothing algorithm that combines several ideas:
  1. Absolute discounting (estimated from data)
  2. Replace ngram probabilities with continuation probabilities

\[
P_{\text{AbsoluteDiscounting}}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) - d}{\sum_v C(w_{i-1}v)} + \lambda(w_{i-1})P(w_i)
\]

Interpolated with observed unigram probability
Smoothing
Kneser-Ney

- State-of-the-art smoothing algorithm that combines several ideas:
  1. Absolute discounting (estimated from data)
  2. Replace n-gram probabilities with *continuation* probabilities

Consider: I can’t see without my reading ____ .

In corpus:
- $P(\text{reading glasses}) = P(\text{reading Kong}) = 0$
- $P(\text{Kong}) > P(\text{glasses})!$
- What to do?
Smoothing
Kneser-Ney

• State-of-the-art smoothing algorithm that combines several ideas:
  1. Absolute discounting (estimated from data)
  2. Replace ngram probabilities with *continuation* probabilities

  # unique contexts for “glasses” > # unique contexts for “Kong”
  so we assume:
  \[
P(\text{glasses}|\text{as-yet-unseen-ctx}) > P(\text{Kong}|\text{as-yet-unseen-ctx})
  \]
Smoothing
Kneser-Ney

- State-of-the-art smoothing algorithm that combines several ideas:
  1. Absolute discounting (estimated from data)
  2. Replace ngram probabilities with *continuation* probabilities

\[
P_{KN}(w_i|w_{i-1}) = \frac{\max(C(w_{i-1}w_i)-d,0)}{C(w_{i-1})} + \lambda(w_{i-1})P_{\text{CONTINUATION}}(w_i)
\]
Topics

• What is language modeling? When do we use it?
• Ngram language models
• Smoothing
• Perplexity
Perplexity

• How do we decide if a language model is “good”?
• A good language model should assign high probability to sentences that actually appear
• Instead of using probability directly, we use a metric called “perplexity”
  • Inverse probability of test set, normalized by # of words
Perplexity

\[
\text{ppl}(W) = \sqrt{\prod_{i=1}^{n} \frac{1}{P(w_1 \ldots w_n)}}
\]
Perplexity

\[ \text{ppl}(W) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_1...w_n)}} = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_i|w_{i-1})}} \]

(for bigram model)
Perplexity

Intuition

\[ \text{ppl}(W) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_1 \ldots w_n)}} = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_i | w_{i-1})}} \]

language with digits 0-9, all equally probable

\( W = \text{sequence of } n \text{ digits} \)
Perplexity

Intuition

$$\text{ppl}(W) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_1 \ldots w_n)}} = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_i | w_{i-1})}}$$

Language with digits 0-9, all equally probable

$W = \text{sequence of } n \text{ digits}$

$$\text{ppl}(W) = (10^n)^\frac{1}{n} = 10$$
Perplexity

Intuition

• “Weighted average branching factor”, i.e., how many next words can follow any given word?

• In PPL, lower is better! A model with lower PPL is less “surprised” by new data

• I.e., a model with lower PPL has more certainty about true sequences. It considers branching factors to be lower, because it has a good sense of what should come next
Perplexity

Intuition

• In natural language, distributions are highly non-uniform, so branching factors are (relatively) low

• PPL will never be zero! Natural language has inherent uncertainty

• PPL is *not* comparable across different datasets! Some datasets/languages/corpora are “easier”/lower uncertainty than others
Perplexity
Intuition

• Higher-order n-grams lead to lower ppl in general, but:
  • More likely to overfit to training data
  • Require more memory
  • Result in many more zero-counts