

---

# Language Modeling, N-Gram Models

using examples from the text Jurafsky and Martin,  
and from slides by Dan Jurafsky

# Language Models

---

- The goal of a Language Model is to assign a probability that a sentence (or phrase) will occur in natural uses of the language
- Why?
  - Machine Translation:
    - $P(\text{high winds tonite}) > P(\text{large winds tonite})$
  - Spell Correction
    - The office is about fifteen **minuets** from my house
      - »  $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$
  - Speech Recognition
    - $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$
  - + Summarization, question-answering, and many other NLP applications

# Language Models

---

- Goal: compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, w_4, w_5 \dots w_n)$$

- Related task: probability of an upcoming word:

$$P(w_5 | w_1, w_2, w_3, w_4)$$

conditional probability that  $w_5$  occurs, given that we know that  $w_1, w_2, w_3, w_4$  already occurred.

- A model that computes either of these:

$P(W)$  or  $P(w_n | w_1, w_2 \dots w_{n-1})$  is called a **language model**.

- We might call this a grammar because it predicts the (word-level) structure of the language, but language model is the standard terminology.

# Chain Rule Applied

- Compute the probability of a sentence by computing the joint probability of all the words conditioned by the previous words

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

Example:  $P(\text{"its water is so transparent"}) =$   
 $P(\text{its}) \times P(\text{water} | \text{its}) \times P(\text{is} | \text{its water})$   
 $\times P(\text{so} | \text{its water is}) \times P(\text{transparent} | \text{its water is so})$

- With our corpus, we can just compute the probability that something occurred by counting its occurrences and dividing by the total number

$$P(\text{the} | \text{its water is so transparent that}) =$$
$$\frac{\text{Count}(\text{its water is so transparent that the})}{\text{Count}(\text{its water is so transparent that})}$$

- But there are way too many unique English sentences in any realistic corpus for this to work! We'll never see enough data. 4

# Markov Assumption

---



Andrei Markov

- Instead we make the simplifying Markov assumption that we can predict the next word based on only one word previous:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- Or perhaps two words previous:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

# N-gram models

---

- Unigram Model: (word frequencies)  
The simplest case is that we predict a sentence probability just based on the probabilities of the words with no preceding words

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

- **Bigram Model:** (two word frequencies)  
Prediction based on one previous word:

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-1})$$

# N-gram Models

---

- We can extend to trigrams, 4-grams, 5-grams
  - Each higher number will get a more accurate model, but will be harder to find examples of the longer word sequences in the corpus
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:

“The computer which I had just put into the machine room on the fifth floor crashed.”

    - the last word *crashed* is not very likely to follow the word *floor*, but it is likely to be the main verb of the word *computer*
  - But we can often get away with N-gram models

# N-Gram probabilities

---

- For N-Grams, we need the **conditional probability**:

$P(\langle \text{next word} \rangle \mid \langle \text{preceding word sequence of length } n \rangle)$

e.g.  $P(\textit{the} \mid \textit{They picnicked by})$

- We define this as
  - the observed frequency (count) of the whole sequence divided by
  - the observed frequency of the preceding, or initial, sequence (sometimes called the **maximum likelihood estimation (MLE)**):

$$\begin{aligned} &P(\langle \text{next word} \rangle \mid \langle \text{preceding word sequence of length } n \rangle) \\ &= \frac{\text{Count}(\langle \text{preceding word sequence} \rangle \langle \text{next word} \rangle)}{\text{Count}(\langle \text{preceding word sequence} \rangle)} \end{aligned}$$

- Example:  $\text{Count}(\textit{They picnicked by the}) / \text{Count}(\textit{They picnicked by})$



# Example of Bigram probabilities

---

- Divide the count of the bigram by the count of the first word: 
$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$
- Example mini-corpus of three sentences, where we have sentence detection and we include the sentence tags in order to represent the beginning and end of the sentence.
  - <S> I am Sam </S>
  - <S> Sam I am </S>
  - <S> I do not like green eggs and ham </S>
- Bigram probabilities:
  - $P(I | <S>) = 2/3 = .67$  (probability that I follows <S>)
  - $P(</S> | Sam) = 1/2 = .5$
  - $P(Sam | <S>) = 1/3 = .33$
  - $P(Sam | am) = 1/2 = .5$
  - $P(am | I) = 2/3 = .67$

## Example using bigram probabilities to compute the probabilities of sentences:

---

- Berkeley Restaurant Project sentences
  - 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

# Raw Bigram Counts from the corpus

- Out of 9222 sentences, showing counts that the word on the left is followed by the word on the top

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

# Bigram probabilities

- Unigram counts:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Resulting bigram probability  
(bigram counts / unigram counts):

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

# Using N-Grams for sentences

---

- For a bigram grammar  $\prod_{k=1}^n P(w_k | w_{k-1})$ 
  - P(sentence) can be approximated by multiplying all the bigram probabilities in the sequence
- Example of using bigram probabilities to compute the probability of a sentence:

$$\begin{aligned} P(\text{I want to eat Chinese food}) = \\ P(\text{I} | \langle S \rangle) P(\text{want} | \text{I}) P(\text{to} | \text{want}) P(\text{eat} | \text{to}) \\ P(\text{Chinese} | \text{eat}) P(\text{food} | \text{Chinese}) \end{aligned}$$

# More Bigrams from the restaurant corpus

Eat on	.16	Eat Thai	.03
Eat some	.06	Eat breakfast	.03
Eat lunch	.06	Eat in	.02
Eat dinner	.05	Eat Chinese	.02
Eat at	.04	Eat Mexican	.02
Eat a	.04	Eat tomorrow	.01
Eat Indian	.04	Eat dessert	.007
Eat today	.03	Eat British	.001

Examples due to Rada Mihalcea

# Additional Bigrams

---

<S> I	.25	Want some	.04
<S> I'd	.06	Want Thai	.01
<S> Tell	.04	To eat	.26
<S> I'm	.02	To have	.14
I want	.32	To spend	.09
I would	.29	To be	.02
I don't	.08	British food	.60
I have	.04	British restaurant	.15
Want to	.65	British cuisine	.01
Want a	.05	British lunch	.01

# Computing Sentence Probabilities

---

- $P(\text{I want to eat British food}) = P(\text{I}|\langle S \rangle) P(\text{want}|\text{I}) P(\text{to}|\text{want}) P(\text{eat}|\text{to}) P(\text{British}|\text{eat}) P(\text{food}|\text{British})$   
 $= .25 \times .32 \times .65 \times .26 \times .001 \times .60 = .000080$
- vs.
- $P(\text{I want to eat Chinese food}) = .00015$
- Probabilities seem to capture “syntactic” facts, “world knowledge”
  - eat is often followed by a noun
  - British food is not too popular
- N-gram models can be trained by counting and normalization



## Using n-gram probabilities in a language model: Why do we need smoothing?

---

- Every N-gram training matrix is sparse, even for very large corpora ( remember Zipf' s law)
  - There are words that don't occur in the training corpus that may occur in future text
  - These are known as the **unseen words**
- Whenever a probability is 0, it will multiply the entire sequence to be 0
- Solution: estimate the likelihood of unseen N-grams and include a small probability for unseen words

# Intuition of smoothing (from Dan Klein)

- When we have sparse statistics:

$P(w \mid \text{denied the})$

3 allegations

2 reports

1 claims

1 request

7 total



- Steal probability mass to generalize better

$P(w \mid \text{denied the})$

2.5 allegations

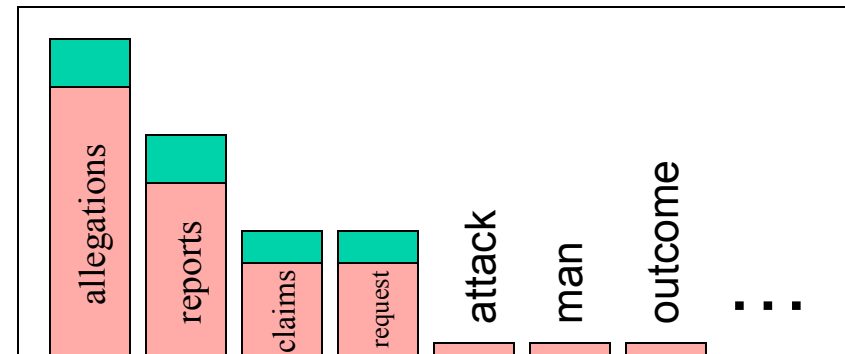
1.5 reports

0.5 claims

0.5 request

2 other

7 total



# Smoothing

---

- Add-one smoothing
  - Given:  $P(w_n|w_{n-1}) = C(w_{n-1}w_n)/C(w_{n-1})$
  - Add 1 to each count:  $P(w_n|w_{n-1}) = [C(w_{n-1}w_n) + 1] / [C(w_{n-1}) + V]$
- Backoff Smoothing for higher-order N-grams
  - Notice that:
    - N-grams are more precise than (N-1)grams
    - But also, N-grams are more sparse than (N-1) grams
  - How to combine things?
    - Attempt N-grams and back-off to (N-1) if counts are not available
    - E.g. attempt prediction using 4-grams, and back-off to trigrams (or bigrams, or unigrams) if counts are not available
- More complicated techniques exist: in practice, NLP LM use Knesser-Ney smoothing

# Language Modeling Toolkit

---

- Language Modeling tools help you define your language models by making those tables, calculating the probabilities.
- SRI Language Modeling:
  - <http://www.speech.sri.com/projects/srilm/>
  - <http://www.speech.sri.com/projects/srilm/papers/icslp2002-srilm.pdf> -- a conference paper that gives an overview of the toolkit