Lecture 2:
Markov Language Modeling

Language Modeling

- *Language model*: a probabilistic mechanism for generating text
  - Or equivalently, a probabilistic model that defines a distribution of text

- Claude Shannon (“Prediction and entropy of printed English”, 1951) first investigated the problem of the extent to which the next word is predictable

- Shannon’s Game: predict the next letter, given the former ones:

```
(1) THE ROOM WAS NOT VERY LIGHT A SMALL OBLONG
(2) ----ROO------NOT-V-----I------SM----OBL----
(1) READING LAMP ON THE DESK SHED GLOW ON
(2) REA----------O-----D----SHED-GLO--O--
(1) POLISHED WOOD BUT LESS ON THE SHABBY RED CARPET
(2) P-L-S-----O---BU--L-S--O------SH-----RE--C------
```

Language Modeling

- Shannon's game allows estimating the unpredictability, or the information in each letter and word
- Can computers do the same?
  - This could be an interesting benchmark for language technology
- If so, what is language modeling useful for?
- Is there psychological evidence that humans are sensitive to the predictability of the next word?
  - There is

The Language Modeling Problem

- **Goal:** given a sequence of words, what's the distribution of the next word

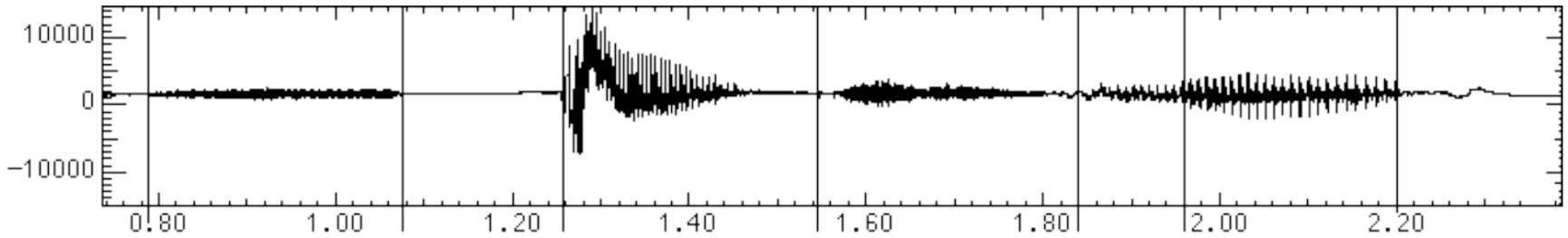
- More formally:
  - Setup: assume a finite set of words (vocabulary) *V*
  - Data: a training set of examples
  - Problem: estimate the probability distribution over all sequences over V

$$\sum_{x \in \mathcal{V}^*} p(x) = 1$$

Application: Speech Recognition

- Audio in, text out
- SOTA: 0.3% error for digit strings, 5% dictation, 50%+ TV

- “Wreck a nice beach?”
  - “Recognize speech”
- “Eye eight uh Jerry?”
  - “I ate a cherry”



The Noisy Channel Model

- Goal: predict sentence given acoustics

$$w^* = \arg\max_w P(w|a)$$

- The noisy channel approach:

$$w^* = \arg\max_w P(w|a)$$
$$= \arg\max_w P(a|w)P(w)/P(a)$$
$$= \arg\max_w P(a|w)P(w)$$

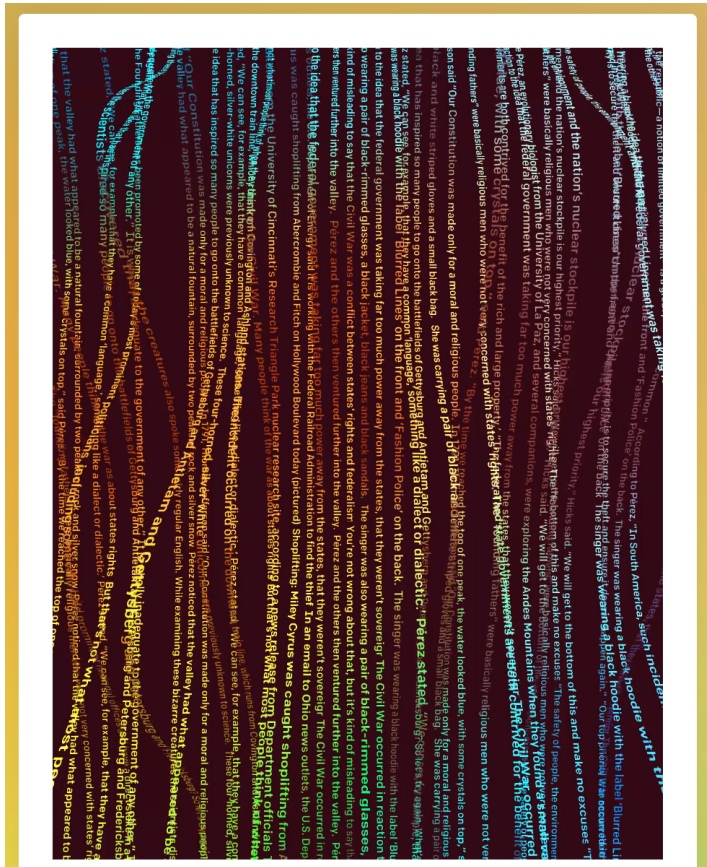
Acoustic Model

Language Model

Acoustically Scored Hypotheses

the station signs are in deep in english	-14732
the stations signs are in deep in english	-14735
the station signs are in deep into english	-14739
the station 's signs are in deep in english	-14740
the station signs are in deep in the english	-14741
the station signs are indeed in english	-14757
the station 's signs are indeed in english	-14760
the station signs are indians in english	-14790
the station signs are indian in english	-14799
the stations signs are indians in english	-14807
the stations signs are indians and english	-14815

Language Models for Generating



Better Language Models and Their Implications

We've trained a large-scale unsupervised language model which generates coherent paragraphs of text, achieves state-of-the-art performance on many language modeling benchmarks, and performs rudimentary reading comprehension, machine translation, question answering, and summarization—all without task-specific training.

FEBRUARY 14, 2019
24 MINUTE READ

https://openai.com/blog/better-language-models/

Language Models for Generating

SYSTEM PROMPT (HUMAN-WRITTEN)

A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)

The incident occurred on the downtown train line, which runs from Covington and Ashland stations.

In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

“The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,” said Tom Hicks, the U.S. Energy Secretary, in a statement. “Our top priority is to secure the theft and ensure it doesn’t happen again.”

The stolen material was taken from the University of Cincinnati’s Research Triangle Park nuclear research site, according to a news release from Department officials.

The Nuclear Regulatory Commission did not immediately release any information.

According to the release, the U.S. Department of Energy’s Office of Nuclear Material Safety and Security is leading that team’s investigation.

“The safety of people, the environment and the nation’s nuclear stockpile is our highest priority,” Hicks said. “We will get to the bottom of this and make no excuses.

THE NEW YORKER

nyer.cm/5H11CgF

Learning Language Models

- Goal: Assign useful probabilities *P(x)* to sentences x
- Input: many observations of training sentences *x*
- Output: system capable of computing *P(x)*

- Probabilities should broadly indicate plausibility of sentences:
  - *P(I saw a van) >> P(eyes awe of an)*
  - Not only grammaticality: *P(artichokes intimidate zippers) ≈ 0*
  - In principle, plausibility depends on the domain, context, speaker

Trivial Language Model

- Simplest option: empirical distribution over training sentences…

$$p(x_1 \dots x_n) = \frac{c(x_1 \dots x_n)}{N} \text{ for sentence } x = x_1 \dots x_n$$

- Problem: does not generalize (at all)
- We need to assign non-zero probability to previously unseen sentences!

Unigram Model

- Generative process: pick a word, pick a word, pick a word … until you pick STOP

$$p(x_1 \dots x_n) = \prod_{i=1}^{n} p(x_i)$$

  - Problem: these models disregard context completely

Bigram Models

- Condition on previous single word
- Generative process:
  - pick START, pick a word conditioned on previous one, repeat until STOP
- The model can be factored thusly:

$$p(x_1 \dots x_n) = \prod_{i=1}^{n} p(x_i|x_{i-1})$$

- This is equivalent to assuming the words of the sentence are (homogeneous, 1st order) Markov chains, namely that:

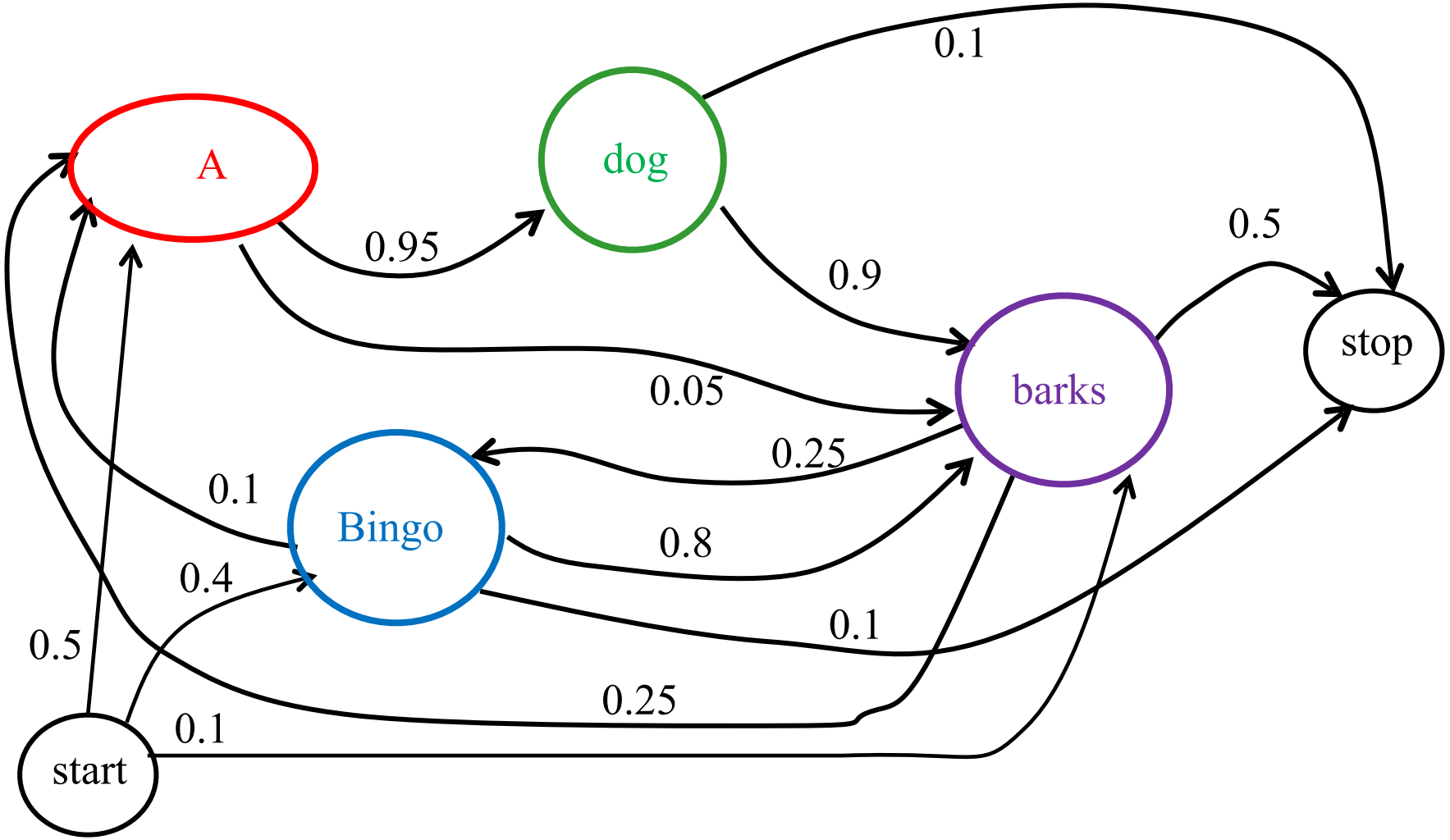
$$p(x_i|x_{i-1} \dots x_0) = p(x_i|x_{i-1})$$

Markov Model / Markov Chain

- A finite state automaton with probabilistic state transitions

- Makes Markov assumption that next state only depends on the current state and independent of previous history **given the current state**

Sample Markov Model for LM



Formal Representation of Markov LMs

- The parameters of a Markov LM can be represented as a stochastic, positive-valued matrix (*transition matrix*)
  - Sometimes the distribution of the first symbol is represented separately, but we can think about the sequences as always starting with *n* instances of a special START symbol

$$A_{ij} = P(x_m = w_j | x_{m-1} = w_i)$$

$$w_i, w_j \in WORDS \cup \{START, STOP\}$$

- Example:

START FUNNY LITTLE THINGS STOP

START	$\begin{pmatrix} 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0.8 & 0.1 & 0.1 & 0 \\ 0 & 0.1 & 0.1 & 0.8 & 0 \\ 0 & 0 & 0 & 0.1 & 0.9 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$
FUNNY	
LITTLE	
THINGS	
STOP	

Higher-order Markov Models can Help

198015222 the first
194623024 the same
168504105 the following
158562063 the world
…
14112454 the door

23135851162 the *

197302 close the window
191125 close the door
152500 close the gap
116451 close the thread
87298 close the deal

3785230 close the *

3380 please close the door
1601 please close the window
1164 please close the new
1159 please close the gate
…
0 please close the first

13951 please close the *

Higher-order Markov Models can Help

- Indeed a bigram model will probably assign a similar probability to the sentences:

  The birds I saw last night **sing** beautifully

  The birds I saw last night **sings** beautifully

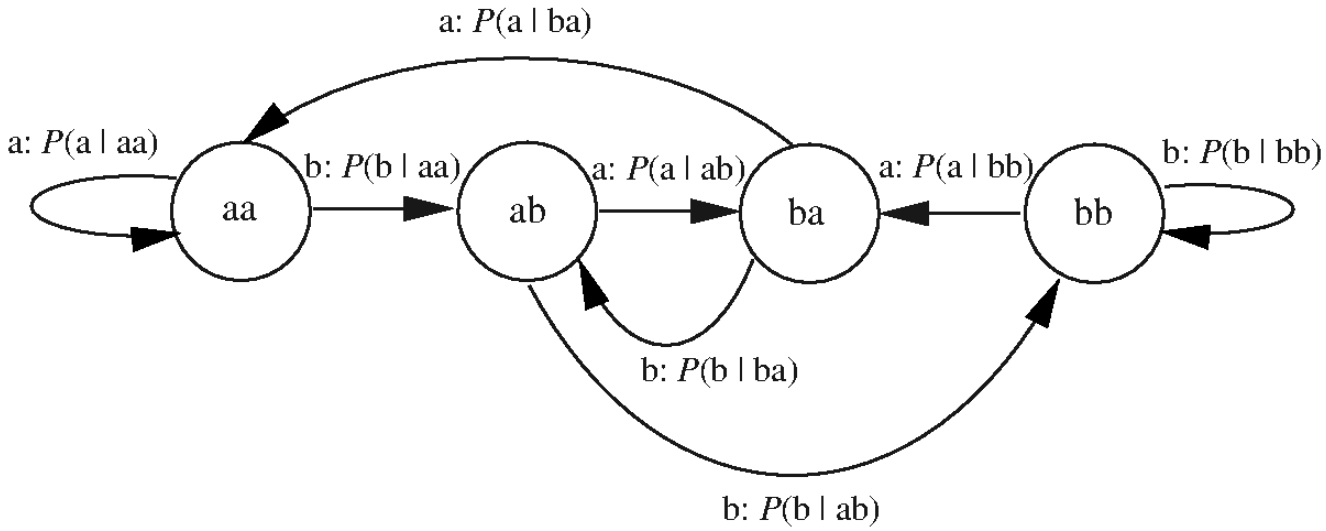
Or:

  He saw himself in the mirror

  He saw herself in the mirror

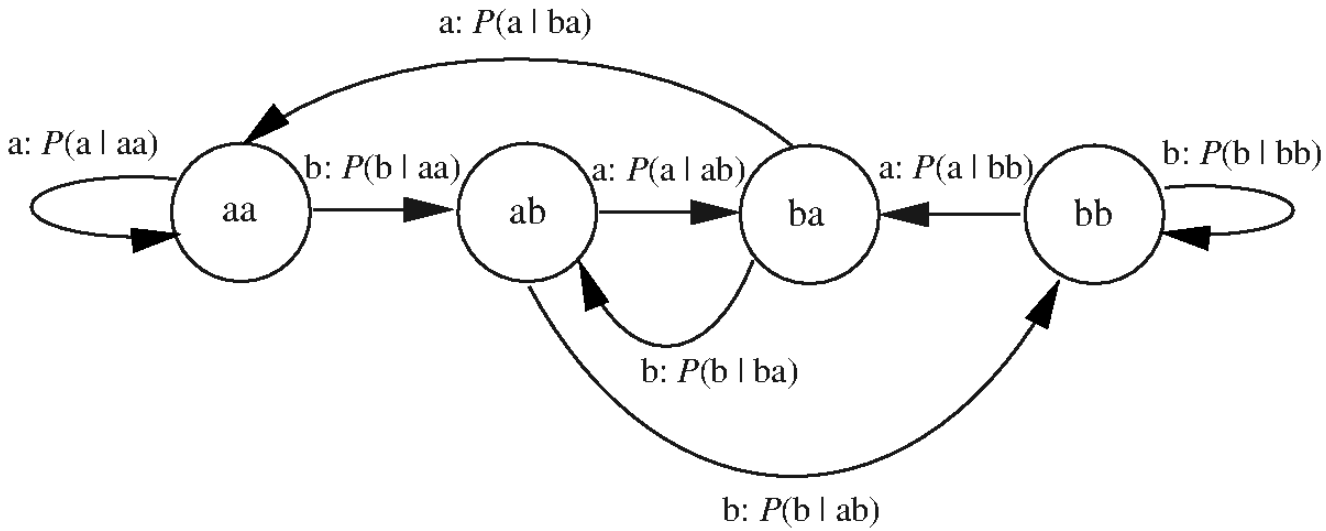
Higher-order Markov Language Models

- k-gram models are equivalent to bi-gram models where the state space (the words) are k-tuples of the bi-gram state space
- So we think of these models as modeling the transition between a k-tuple of states to a k-tuple of states, where the first *k-1* coordinates of x_n must be equal to last *k-1* coordinates of x_{n-1}
- Diagram for k=2 (over *{a,b}*):



Higher-order Markov Language Models

- The difficulty: the number of parameters increases exponentially with *k*.
- So a k-gram language model would have about $|\text{Words}|^k$ entries in its transition matrix



Learning Bigram Language Models

- The parameters of a Markov LM can be represented as a stochastic, positive-valued matrix (*transition matrix*)
  - Sometimes the distribution of the first symbol is represented separately, but we can think about the sequences as always starting with *n* instances of a special START symbol

$$A_{ij} = P(x_m = w_j | x_{m-1} = w_i)$$

- A maximum likelihood estimator for the transition matrix is obtained by counting:

$$q_{ML}(x_m = w_j | x_{m-1} = w_i) = \frac{count(w_i, w_j)}{\sum_{j'} count(w_i, w_{j'})}$$

https://www.stat.cmu.edu/~cshalizi/462/lectures/06/markov-mle.pdf

Learning Higher-order Language Models

- A maximum likelihood estimator for a k[th] order language model is:

$$q_{ML}(x_m = w_j | x_{m-1} = w_{i1}, x_{m-2} = w_{i2}, \ldots, x_{m-k} = w_{ik}) = \frac{count(w_{ik}, \ldots, w_{i1}, w_j)}{\sum_{j'} count(w_{ik}, \ldots, w_{i1}, w_{j'})}$$

Zero Counts

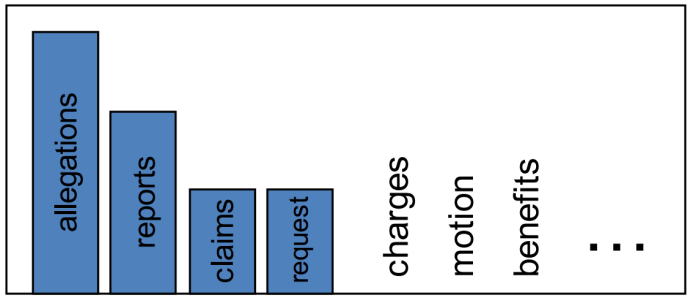
- Training set:
  - … denied the allegations
  - … denied the reports
  - … denied the claims
  - … denied the request
- Test set:
  - … denied the offer
  - … denied the loan

$$q_{ML}(offer|the, denied)$$

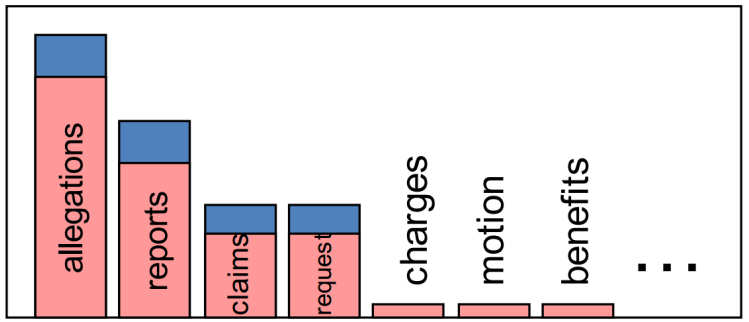
Zero Counts

- But, zero transition probabilities mean that the whole sentence receives a 0 probability!
- Therefore, zero probability estimates are generally avoided
- Methods for assigning probability mass to rare events are often called *smoothing methods*
  - Some have statistical motivation, others don't

P($offer|the, denied$)
3 allegations
2 reports
1 claims
1 request
7 total



P($offer|the, denied$)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



Add-δ (Laplace) Smoothing

- Classic solution: add counts
  - If *q(w)* is some distribution over words *w* then the smoothed $q_{add\text{-}\delta}$ *(w)* is

$$q_{add-\delta}(w) = \frac{c(w) + \delta}{\sum_{w'}(c(w') + \delta)} = \frac{c(w) + \delta}{c() + \delta|\mathcal{V}|}$$

- Problem: doesn't work well in practice for LMs

n-gram Smoothing

- The problem is not only zero counts
  - In general, where the event conditioned on has a low probability, the maximum likelihood estimator is unreliable
- Estimating the probability of an n-gram is a fundamental task in NLP
  - Not only in language models
  - For instance, knowing the common n-grams a word participates in can be telling in terms of its meaning
- The problem becomes worse as n increases
  - The denominator grows exponentially with n

Back-off Models

- **The idea:** assume we want to estimate a trigram distribution for a word x_m given its two previous words x_{m-1} and x_{m-2}:

$$q_{ML}(x_m = w_j | x_{m-1} = w_i, w_{m-2} = w_l) = \frac{count(w_l, w_i, w_j)}{\sum_{j'} count(w_l, w_i, w_{j'})}$$

- If the bigram (x_{m-1}, x_{m-2}) didn't appear enough times, we back-off to conditioning only on x_{m-1}:

$$q_{ML}(x_m = w_j | x_{m-1} = w_i) = \frac{count(w_i, w_j)}{\sum_{j'} count(w_i, w_{j'})}$$

- If w_{i-1} is not frequent enough, we back-off to w_i's frequency:

$$q_{ML}(x_m = w_j) = \frac{count(w_j)}{\sum_{j'} count(w_{j'})}$$

Notation

- For brevity, we will assume that the transitions are not dependent on *m* ("homogeneous Markov models") and use the notation

$$P(w_j|w_i, w_l)$$

instead of

$$P(x_m = w_j|x_{m-1} = w_i, w_{m-2} = w_l)$$

Back-off: Linear Interpolation

- Interpolate the trigram, bigram and unigram distributions by a convex combination:

$$
\begin{aligned}
q(w_i \mid w_{i-2}, w_{i-1}) = \quad & \lambda_1 \times q_{\mathsf{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\
& + \lambda_2 \times q_{\mathsf{ML}}(w_i \mid w_{i-1}) \\
& + \lambda_3 \times q_{\mathsf{ML}}(w_i)
\end{aligned}
$$

$$
\text{where } \lambda_1 + \lambda_2 + \lambda_3 = 1, \text{ and } \lambda_i \geq 0 \text{ for all } i.
$$

- In order to estimate the λ_i values, we hold out part of the training data, compute the N-gram probabilities q_{ML} on the rest of the training data, and search for the λ_i that yield the highest probability on the held-out set.

Back-off: Linear Interpolation

- That is, maximize *L,* the log-likelihood, where $c'(w_1,w_2,w_3)$ is the number of times the trigram (w_1,w_2,w_3) appeared in the validation set

$$L(\lambda_1, \lambda_2, \lambda_3) = log \left(\prod_{(w_1,w_2,w_3) \in V^3} q(w_3|w_1, w_2)^{c'(w_1,w_2,w_3)} \right)$$

- Which yields this (concave) optimization problem:

$$L(\lambda_1, \lambda_2, \lambda_3) = \sum_{w_1,w_2,w_3} c'(w_1, w_2, w_3) \log q(w_3 \mid w_1, w_2)$$

such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all i, and where

$$\begin{aligned} q(w_i \mid w_{i-2}, w_{i-1}) = \quad & \lambda_1 \times q_{\mathsf{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\ & + \lambda_2 \times q_{\mathsf{ML}}(w_i \mid w_{i-1}) \\ & + \lambda_3 \times q_{\mathsf{ML}}(w_i) \end{aligned}$$

Back-off: Linear Interpolation

- It is common to take more ls to get a better fit
- Here is one example of how to do that. Define:

$$\Pi(w_{i-2}, w_{i-1}) = \begin{cases} 1 & \text{If Count}(w_{i-1}, w_{i-2}) = 0 \\ 2 & \text{If } 1 \leq \text{Count}(w_{i-1}, w_{i-2}) \leq 2 \\ 3 & \text{If } 3 \leq \text{Count}(w_{i-1}, w_{i-2}) \leq 5 \\ 4 & \text{Otherwise} \end{cases}$$

- And maximize:

$$\begin{aligned} q(w_i \mid w_{i-2}, w_{i-1}) = \quad & \lambda_1^{\Pi(w_{i-2},w_{i-1})} \times q_{\text{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\ & + \lambda_2^{\Pi(w_{i-2},w_{i-1})} \times q_{\text{ML}}(w_i \mid w_{i-1}) \\ & + \lambda_3^{\Pi(w_{i-2},w_{i-1})} \times q_{\text{ML}}(w_i) \end{aligned}$$

where $\lambda_1^{\Pi(w_{i-2},w_{i-1})} + \lambda_2^{\Pi(w_{i-2},w_{i-1})} + \lambda_3^{\Pi(w_{i-2},w_{i-1})} = 1$,
and $\lambda_i^{\Pi(w_{i-2},w_{i-1})} \geq 0$ for all i.

Discounting Methods

- Maximum likelihood estimates tend to be too high for low count items:

x	Count(x)	$q_{\text{ML}}(w_i \mid w_{i-1})$
x	48	
x , dog	15	15/48
x , woman	11	11/48
x , man	10	10/48
x , park	5	5/48
x , job	2	2/48
x , telescope	1	1/48
x , manual	1	1/48
x , afternoon	1	1/48
x , country	1	1/48
x , street	1	1/48

Discounting Methods

- One simple way to handle this is to discount all counts by a constant term: (say, 0.5)
- But what do we do with the missing probability mass?

x	Count(x)	Count$^*(x)$	$\frac{\text{Count}^*(x)}{\text{Count(the)}}$
X	48		
X, dog	15	14.5	14.5/48
X, woman	11	10.5	10.5/48
X, man	10	9.5	9.5/48
X, park	5	4.5	4.5/48
X, job	2	1.5	1.5/48
X, telescope	1	0.5	0.5/48
X, manual	1	0.5	0.5/48
X, afternoon	1	0.5	0.5/48
X, country	1	0.5	0.5/48
X, street	1	0.5	0.5/48

Discounting Methods

- In a bigram model, the missing probability mass for a preceding word w_{i-1} is:

$$\lambda(w_{i-1}) = 1 - \sum_{w} \frac{c^*(w_{i-1}, w)}{c(w_{i-1})}$$

- In our example: $\lambda(w_{i-1}) = 10 \bullet 0.5/48$
- We will distribute *this* mass (also) to bigrams with probability 0
  - In fact, $P(w_i)$ will be non-zero for every word in the training data
- But how?

Discounting + Unigram Interpolation

- First guess: smooth by reducing some fixed quantity from the bi-gram count, and interpolate with the unigram estimate:

discounted bigram Interpolation weight

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

unigram

Kneser-Ney Smoothing

- Better estimate for the probabilities of unigrams:
  - What's the next word: *I can't see without my reading*_____?
  - "Francisco" is more common than "glasses"
  - … but "Francisco" always follows "San"
- So we want to interpolate with a measure of "How likely is *w* to appear as a novel continuation?"
  - For each word, count the number of bigram types it completes
  - Every bigram type was a novel continuation the first time it was seen

$$P_{CONTINUATION}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Kneser-Ney Smoothing

- Properly normalized:

$$P_{CONTINUATION}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|}$$

- A frequent word (“Francisco”) occurring in only one context (“San”) will have a low continuation probability

Kneser-Ney Smoothing

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

- λ is a normalizing constant; assume d≥ 1.
- The probability mass we've discounted:

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} \left| \{w : c(w_{i-1}, w) > 0\} \right|$$

Generalization in LMs

- Much of the criticism against Markov LMs (and about statistical methods to Computational Linguistics) has focused on the difficulty of such models in generalizing beyond the observed data
- Smoothing is a simple example for such a generalization
- But what about?

"Colorless green ideas sleep furiously"

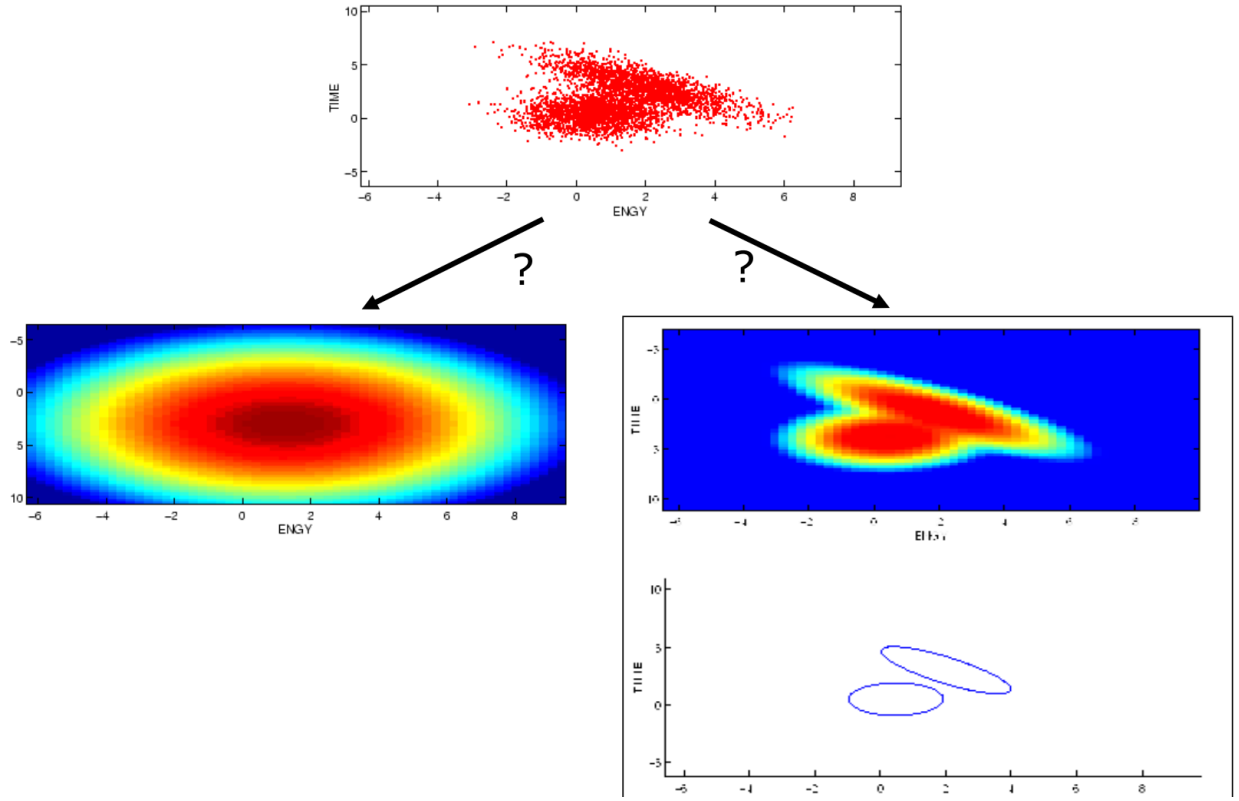
vs.

"Furiously sleep ideas green colorless"

Chomsky (1957)

Latent Variables

- We first need to understand the concept of a *latent variable model*
- A latent variable model is a statistical model, which includes a random variable that is never observed
  - However, the assumption of the model helps explain the data
  - For instance, if we assume the data was sampled from two Gaussians, we can effectively estimate the probability distribution



Pereira's Solution

- Now assume that when a word appears, it is accompanied by a category, but that this category is never observed.
- The identity of the next word is determined based on this category
  - So words share a category if they share a distribution of the next word
- Examples:
  - The words "the" and "a" may share a category, because they are both followed by nouns and verbs
  - Transitive verbs may share a category because they tend to precede "the" and "a"

Formal grammar and information theory: together again? F. Pereira, Phil. Trans. R. Soc. Lond. A (2000)

Pereira's Solution

- In probabilistic terms we will say that each word w_i is accompanied by a latent category c_i such that

$$Pr(w_i|w_{i-1}) = \sum_{c \in CATS} Pr(w_i, c|w_{i-1}) = \sum_{c \in CATS} Pr(c|w_{i-1}) \cdot Pr(w_i|c, w_{i-1}) =$$

$$\sum_{c \in CATS} Pr(c|w_{i-1}) \cdot Pr(w_i|c)$$

  - The last transition assumes that w_{i-1} and w_i are conditionally independent given c

- The model is now free to assign words to categories, such that knowing the category of a word is the best predictor of the following word

Pereira's Solution

- Using 16 categories and newspaper corpora to train the model (using EM), Pereira found that:

$$\frac{\Pr(Colorless\ green\ ideas\ sleep\ furiously)}{\Pr(Furiously\ sleep\ ideas\ green\ colorless)} \approx 200000$$

- Today's LMs use similar technology to generalize to novel word sequences, by creating a multi-dimensional representation of each word

LM Evaluation

- Assume we have *m* sentences $s_1, s_2, \ldots, s_m$
- The common way to evaluate language models is by the probability they assign to held-out data
- More exactly, we use *perplexity*:

$$Perplexity = e^{-l} \qquad l = \frac{1}{M} \sum_{i=1}^{m} logP(s_i) = \frac{1}{M} \sum_{i=1}^{m} \sum_{j} logP(x_j^{(i)} | x_1^{(i)}, \ldots, x_{j-1}^{(i)})$$

M is the total number of tokens (words) in the test data

m is the number of sentences in the test data

Some Perplexity Values

- Perplexity measures the effective number of possibilities for the next word (on average)
- If our model predicts the probability of a word to be 1/N, regardless of the context (a uniform model), the perplexity will be *N*
- Perplexity values would change considerably between corpora, but they often vary between 50 and 500

- **Extrinsic evaluation** (by applying it to a downstream application) is usually conducted in parallel to perplexity evaluation

Predictability and Psycholinguistics

- Language modeling as a statistical process has been much studied in the context of psycholinguistics
- Examples:
  - Reading time of a word has been shown to be logarithmic in its predictability (Smith and Levy, 2013)
  - The length of words has been shown to be related to their predictability (Piantadosi et al., 2010)
    - Related to the “Uniform Information Density” hypothesis, which states that information is evenly spread out through a sentence
  - Event related potentials (ERPs) have been found where prediction is violated (see here)

Questions?

We’ll also talk about Neural Network-based LM later on