Lecture 2:
n-gram Language Models

Instructor: Swabha Swayamdipta
USC CSCI 499 LMs in NLP
Jan 17, Spring 2024

Slides mostly adapted from Dan Jurafsky, some from Greg Durrett + Mohit Iyyer

Lecture Outline

1. Announcements + Recap
2. Probabilistic Language Models
3. n-gram Language Models
4. Evaluation and Perplexity
5. Generating from an n-gram Language Model
    i) Zeros

Announcements

+

Recap

Prerequisites

Students are required to have taken

- CSCI-270 Introduction to Algorithms and Theory of Computing, and
- CSCI-360 Introduction to AI / CSCI-467 Introduction to Machine Learning / equivalent experience

Classes will contain a lot of **probability theory** and some knowledge of **linear algebra**.

Fluency with python programming is recommended!

Please email me for special circumstances or specific clarifications

Logistics

- HW1 Released Today (Due 1/31)
- Next Wed (1/24): Project Pitches! *Please do not miss class!*
    - 3 minute project pitch (5% of your grade)
        - What is the problem? Why should we care about it?
        - How is this connected to language models?
        - What would the inputs and outputs look like? Examples!
        - Come up with a good name for your project so it's memorable for voting
    - Slides encouraged
    - Everyone votes on teams. We will release all votes and it's up to you to form teams of 3
    - It's natural for the ideas to morph between the project pitch and the project proposal
- TA Office Hours Location: RTH 420 / Friday 10-11am
    - Next week extra OH by TA: Thu and Fri 10-11am

USC Viterbi

Projects from Fall Version of the Class

Available here: https://swabhs.com/fall23-csci499-lm4nlp/details/class-projects/

Also linked from the class Website:

report) is due on the specified date by 11:59 PM PT .

Assignments

There will be three components to course grades, see more details.

- **Homeworks (40%).**
- **Quizzes + Class Participation (20%).**
- **Class Project (40%).**

Students are allowed a maximum of 6 late days **total** for all assignments (*NO LATE DAYS ALLOWED FOR* quizzes), with a maximum of 3 late days per deliverable.

Note: Please familiarize yourself with the academic policies and read the note about student well-being.

Pre-Requisites

Students are required to have taken CSCI-270 Introduction to Algorithms and Theory of Computing (4.0 units) as well as one of (CSCI-360 Introduction to AI, CSCI-467 Introduction to Machine Learning or equivalent experience). Fluency with python programming is recommended. Please email the instructor for special circumstances or specific clarifications.

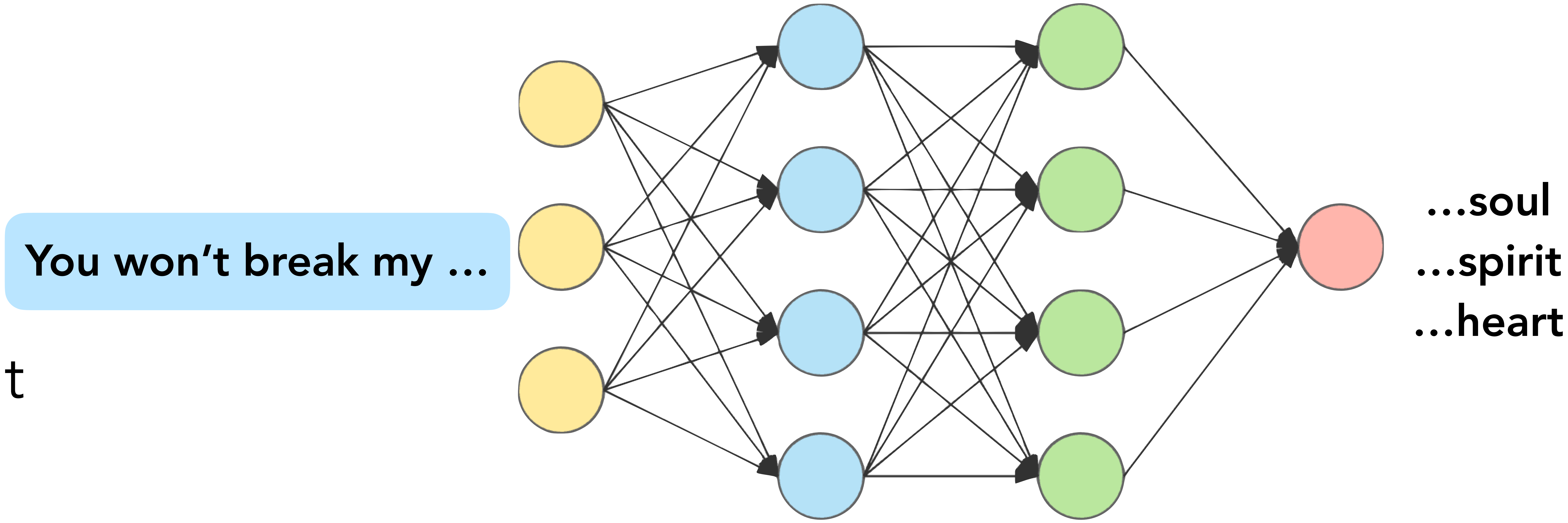
Previous Iterations

- Fall 2023
  - See previous class projects here.

Building a Language Model

What words can follow this?

- Task: Given a sequence of words so far (the **context**), predict what comes next
- We never know for sure what comes next, but we can still make good guesses!



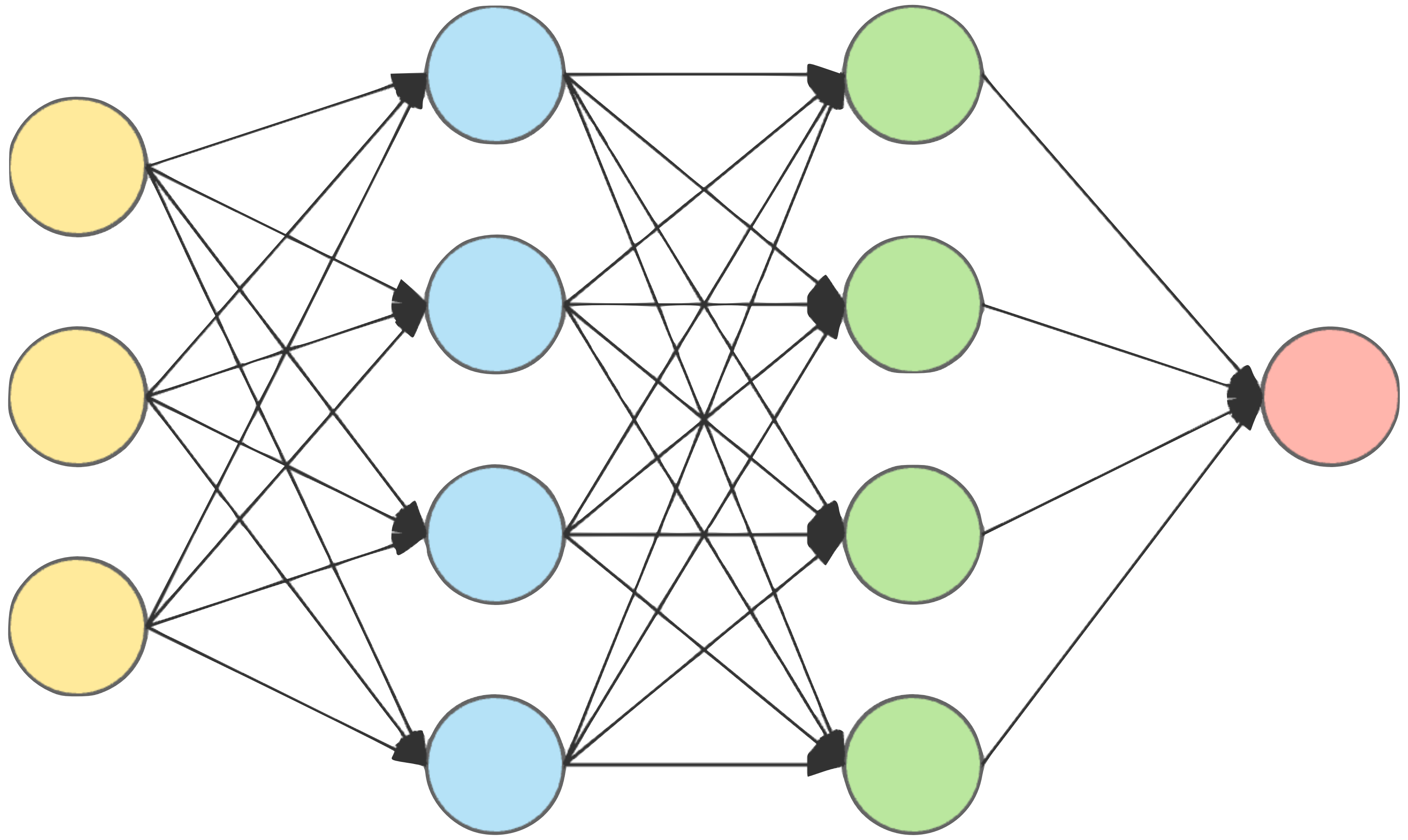
What is common to these words?

Building a Language Model

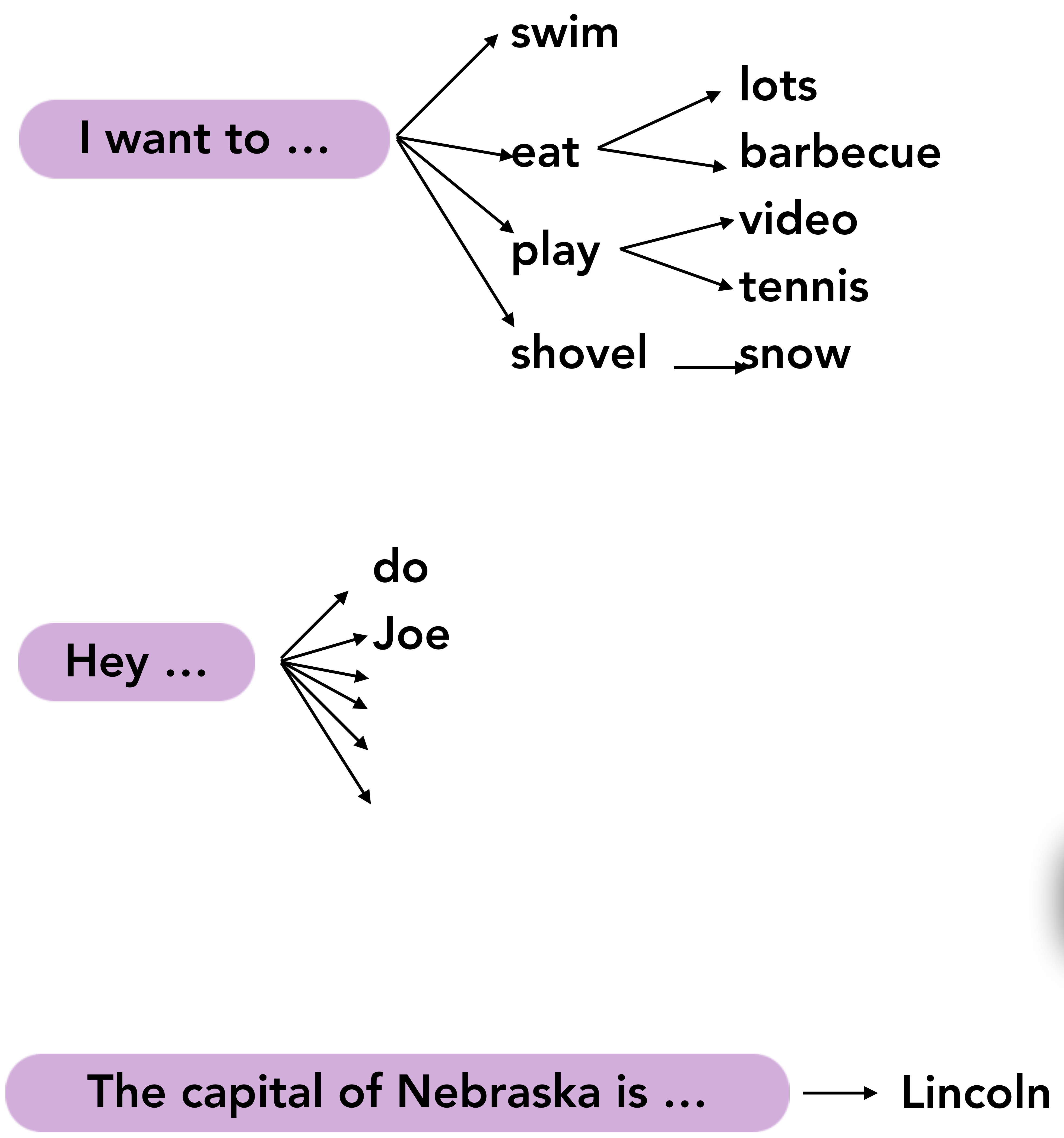
What words can follow this?

- Task: Given a sequence of words so far the context), predict what comes next
- We never know for sure what comes next, but we can still make good guesses!

The 44th President of United States was ...



...Barack Obama



Certain sentence constructions are more likely than others, due to grammaticality, obscurity or commonness

Sentences have different probabilities!

Probabilistic Language Models!

Assign a probability to a sentence

Probabilistic Language Modeling

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

$$P(\mathbf{w}) = P(w_1, w_2, w_3, w_4, w_5, \ldots w_n)$$

Difference

Related task: probability of an upcoming word: $P(w_n \mid w_1, w_2, w_3, w_4, \ldots w_{n-1})$

A model that assigns probabilities to sequences of words (e.g., either of these: $P(\mathbf{w})$ or $P(w_n \mid w_1, w_2, \ldots w_{n-1})$) is called a language model

How to compute $P(W)$?

"its water is so transparent that you can see the bottom"

$P($"its water is so transparent that you can see the bottom"$)$

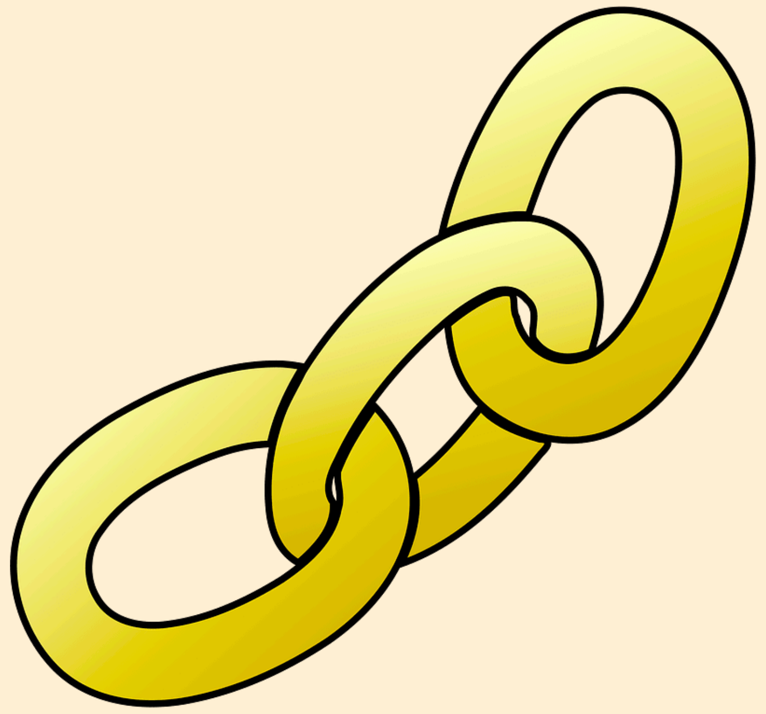
$P($its, water, is, so, transparent, that, you, can, see, the, bottom$)$

How to compute this joint probability, $P(\mathbf{w}) = P(w_1, w_2, w_3, w_4, w_5, \dots w_n)$?

e.g. $P($its, water, is, so, transparent, that$)$

Intuition: let's rely on the Chain Rule of Probability

The Chain-Rule



Recall the definition of conditional probabilities: $P(B \mid A) = \dfrac{P(A, B)}{P(A)}$

Rewriting: $P(A, B) = P(A)P(B \mid A)$

More variables: $P(A, B, C, D) = P(A)P(B \mid A)P(C \mid B, A)P(D \mid C, B, A)$

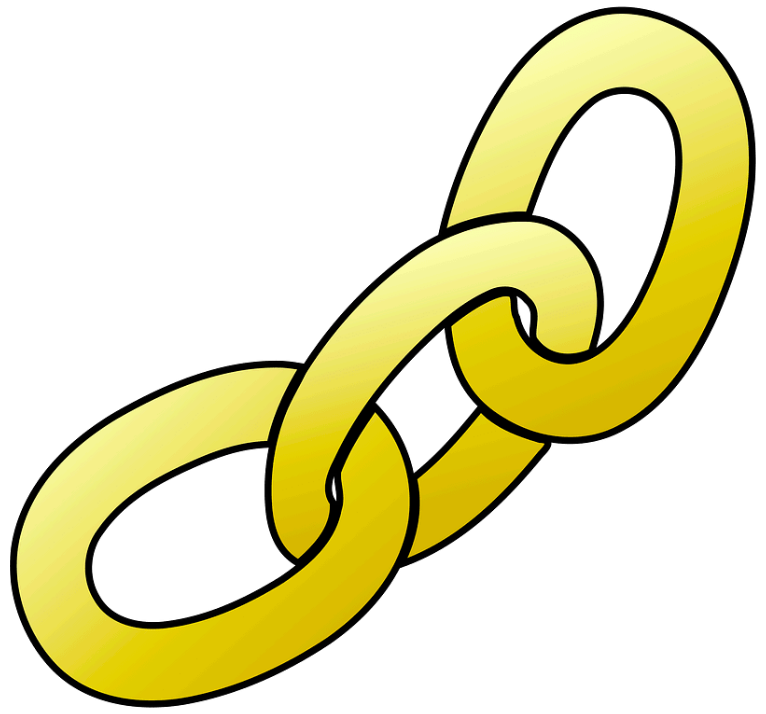
The Chain Rule in General

$$P(x_1, x_2, x_3, \ldots, x_n) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2)\ldots P(x_n \mid x_1, \ldots, x_{n-1})$$

$$= \prod_{i=1}^{n} P(x_i \mid x_1 \ldots x_{i-1})$$

Chain Rule for words in a sentence

$$P(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1} \ldots w_1)$$



$$\begin{aligned} P(\text{its water is so transparent}) = \; & P(\text{its}) \times \\ & P(\text{water} \mid \text{its}) \times \\ & P(\text{is} \mid \text{its water}) \times \\ & P(\text{so} \mid \text{its water is}) \times \\ & P(\text{transparent} \mid \text{its water is so}) \end{aligned}$$

Ordering matters in language!

Why Probabilistic Models?

Why would you want to predict upcoming words, or assign probabilities to sentences?

- Probabilities are essential for language generation
- Any task in which we have to identify words in noisy, ambiguous input, like speech recognition
- For writing tools like spelling correction or grammatical error correction

I will be back soonish

I will be bassoon dish

Your so silly

You’re so silly

Everything has improve

Everything has improved

Probabilistic Language Models

Machine Translation:

- P(high winds tonight) $>$ P(large winds tonight)

Spell Correction:

- P(I'm about fifteen minuets away) $<$ P(I'm about fifteen minutes away)

Speech Recognition:

- P(I saw a van) $>>$ P(eyes awe of an)

Summarization, question-answering, etc., etc.!!

But how to learn these probabilities?

Probability Estimation via Statistical Modeling

Suppose we have a biased coin that's heads with probability p.

As we know, $0 \leq p \leq 1$, and for a normal coin, $p = 0.5$ (equal probability of heads or tails)

Suppose we flip the coin four times and see (H, H, H, T). What is p?

We don't know what p is — could be 0.5! But $p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

maximum likelihood estimate

The probability of the data is $ppp(1-p)$: if you take the derivative and set it equal to zero and find $p = 0.75$

n-gram Language Model

The decision for what words occur after a word w is exactly the same as the biased coin, but with ***many*** possible outcomes (as many as all the words) instead of 2

I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

Vocabulary

How to estimate the probability of the next word?

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

Could we just count and divide?

No! Too many possible sentences!

We'll never see enough data for estimating these

Markov Assumption

Andrei Markov

Simplifying Assumption:

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

Or maybe…

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

Markov Assumption contd.

$$P(w_1, w_2, \ldots w_n) = \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

In other words, we approximate each component in the product such that it is only conditioned on the previous k elements

$$P(w_i \mid w_1, w_2, \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

$$P(\mathbf{w}) = P(w_1, w_2, w_3, w_4, w_5, \ldots w_n)$$

Difference 

$$P(w_n \mid w_1, w_2, w_3, w_4, \ldots w_{n-1})$$

Recap: Probabilistic Modeling

- What is a probabilistic language model?
- How do we simplify it?
- How do we estimate it?
- Why would we need one?
- Next: a simple probabilistic language model

Simplest Case: Unigram model

$$P(w_1, w2, \ldots w_n) \approx \prod_i P(w_i)$$

Some automatically generated sentences from a unigram model

fifth, an, of, futures, the, an, incorporated, a, a, the, inflation, most, dollars, quarter, in, is, mass
thrift, did, eighty, said, hard, 'm, july, bullish
that, or, limited, the

Bigram Model

Condition on the previous word:

$$P(w_i \mid w_1, w2, \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

Some automatically generated sentences from a bigram model

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen
outside, new, car, parking, lot, of, the, agreement, reached
this, would, be, a, record, november

USC Viterbi

n-gram Language Models

We can extend to trigrams, 4-grams, 5-grams, …

In general this is an insufficient model of language

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

Long-distance / Long-range dependencies

But we can often get away with N-gram models

Lecture Outline

1. Announcements + Recap
2. Probabilistic Language Models
3. n-gram Language Models
4. Evaluation and Perplexity
5. Generating from an n-gram Language Model
    i) Zeros

n-gram Language Models

simplest probabilistic model

2-gram language models

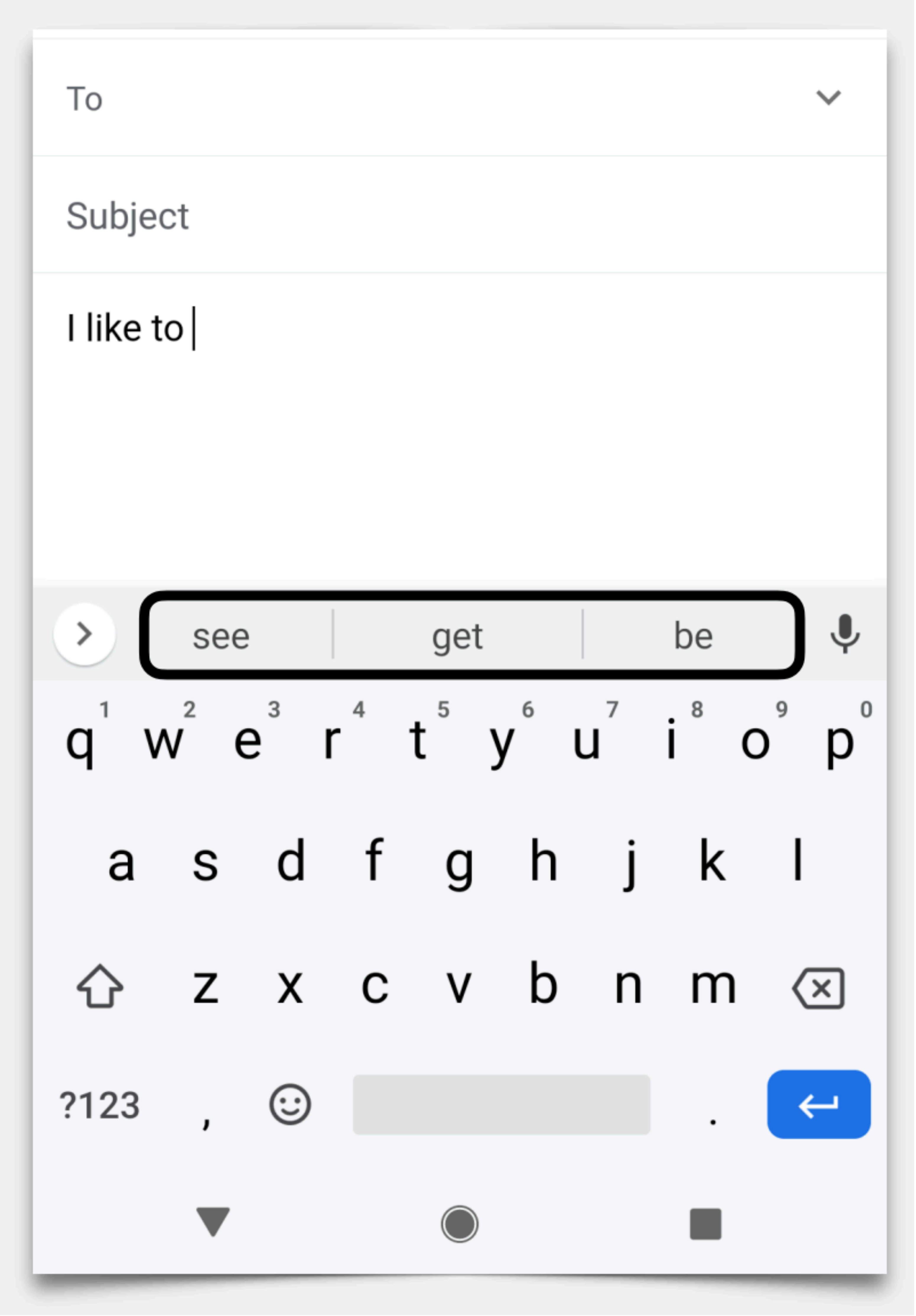
If we have these probabilities, we can build a predictive text system:

P(next word = _ | previous word = to)

Check all the possible words from that list, pick the ones with the highest probability (most likely next words)

Where do these probabilities come from?

- We're going to **learn them** from a bunch of text data we see



2-gram language models

Based on a **conditional probability distribution**:

"the probability of the next word is y given that the previous word is x"

$$P(\text{next word} = y \mid \text{previous word} = x)$$

I want to go to______

$$P(\text{next word} = \text{was} \mid \text{previous word} = \text{to}) = 0.0$$
$$P(\text{next word} = \text{LA} \mid \text{previous word} = \text{to}) = 0.2$$
$$P(\text{next word} = \text{Europe} \mid \text{previous word} = \text{to}) = 0.1$$
$$P(\text{next word} = \text{Mexico} \mid \text{previous word} = \text{to}) = 0.1$$
$$P(\text{next word} = \text{eat} \mid \text{previous word} = \text{to}) = 0.1$$

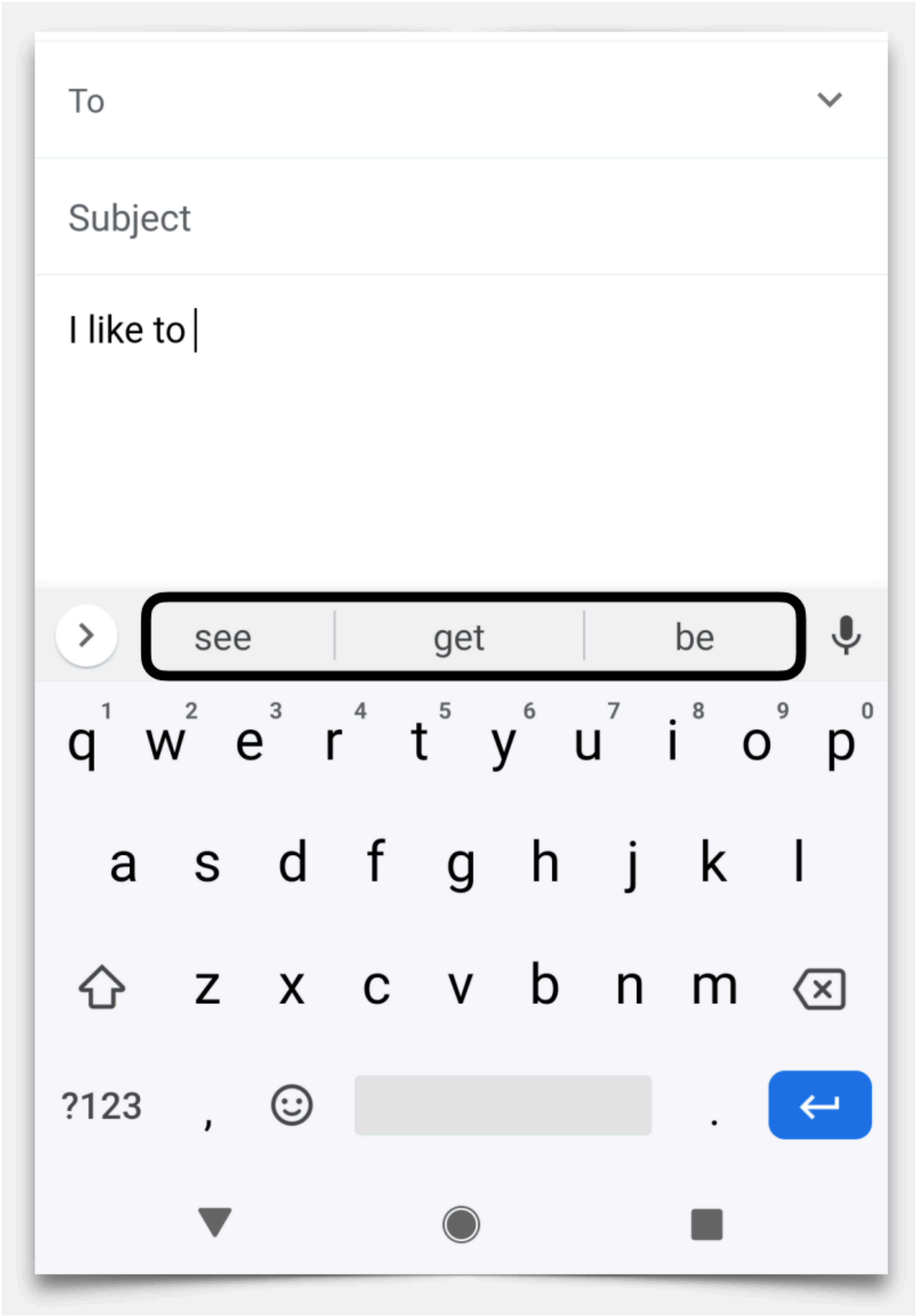
These have to add up to 1 over the vocabulary (every possible word y could be)

If we see "to", there's a 20% chance the next word is "LA"

Assume a **fixed vocabulary** of ~30,000 words

Lots and lots of text data

2-gram LM probabilities



Estimating bigram probabilities

The maximum likelihood estimate

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

What happens when $i = 1$?

Special edge case tokens: <s> and </s> for beginning of sentence and end of sentence, respectively

An example

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67$ $P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33$ $P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

Larger Example:
Berkeley Restaurant Project (BRP)

- 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

Total: 9222 similar sentences

USC Viterbi

BRP: Raw Counts

Out of 9222 sentences

Unigrams

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

Bigrams

Next Word

History

	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

BRP: Bigram Probabilities

Bigram Probabilities: Raw bigram counts normalized by unigram counts

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

w_i

w_{i-1}

	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

What kinds of knowledge?

P(english|want) = .0011

P(chinese|want) = .0065

P(to|want) = .66

P(eat | to) = .28

P(food | to) = 0

P(want | spend) = 0

P (i | <s>) = .25

USC Viterbi

Bigram estimates of sentence probabilities

P(<s> I want english food </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

Quite low…

Underflow Issues

We do everything in log space

- Avoid underflow
- Adding is faster than multiplying

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Lecture Outline

1. Announcements + Recap
2. Probabilistic Language Models
3. n-gram Language Models
4. Evaluation and Perplexity
5. Generating from an n-gram Language Model
   i) Zeros

Evaluation and Perplexity

How good is a language model?

Does our language model prefer good sentences to bad ones?

- Key Idea: Assign higher probability to "real" or "frequently observed" sentences than "ungrammatical" or "rarely observed" sentences?
  - In practice we don't explicitly need to do the latter!

Intrinsic Evaluation

We train parameters of our model on a **training set**.

We test the model's performance on data we haven't seen.

- A **test set** is an unseen dataset that is different from our training set, totally unused.
- An **evaluation metric** tells us how well our model does on the test set.

Extrinsic evaluation of N-gram models

Best evaluation for comparing models A and B

1. Put each model in a task
   - spelling corrector, speech recognizer, MT system
2. Run the task, get an accuracy for A and for B
   - How many misspelled words corrected properly
   - How many words translated correctly
3. Compare accuracy for A and B

Downsides??

Text Generation: Intrinsic or Extrinsic Evaluation?

Machine Learning 101

Train Set vs Test Set:

- We can't allow test sentences into the training set
- We will assign it an artificially high probability when we set it in the test set
- "Training on the test set" is bad science! And violates the honor code

Another risk of cheating:

- using a particular test set so often that we implicitly tune to its characteristics.
- So how to evaluate while developing a model? Use a fresh test set that is truly unseen: **development set**!

In practice, we often just divide our data into 80% training, 10% development, and 10% test.

How best to evaluate an LM?

- Extrinsic evaluation can be time-consuming; hard to design
  - Which is the best task? How many tasks to try?
- Therefore, we often use intrinsic evaluation:
  - Bad approximation
    - unless the test data looks just like the training data
  - Generally only useful in pilot experiments

Perplexity

USC Viterbi

Intuition of Perplexity

The **Shannon Game**: How well can we predict the next word?

I always order pizza with cheese and ____

The 33rd President of the US was ____

I saw a ____

mushrooms 0.1
pepperoni 0.1
anchovies 0.01
….
fried rice 0.0001
….
and 1e-100

Unigrams are terrible at this game!

A better model of a text is one which assigns a higher probability to the word that actually occurs

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

Perplexity is the inverse probability of the test set, normalized by the number of words

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

Minimizing perplexity is the same as maximizing probability

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$= \sqrt[N]{\frac{1}{\prod_i P(w_i \mid w_1 \ldots w_{i-1})}}$$

Applying Markov's assumption for bigrams:

$$= \sqrt[N]{\frac{1}{\prod_i P(w_i \mid w_{i-1})}}$$

Perplexity Example

Let's suppose a sentence of length 50 consisting of random digits

$$P(w) = \frac{1}{10}$$

What is the perplexity of this sentence according to a model that assigns uniform probability to each digit?

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{\frac{-1}{N}}$$

$$= (\frac{1}{10}^{50})^{-\frac{1}{50}}$$

$$= 10$$

Lower perplexity = better model!

Training 38 million words, test 1.5 million words, from the Wall Street Journal

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

What are the two things that might affect perplexity?

Lecture Outline

1. Announcements + Recap
2. Probabilistic Language Models
3. n-gram Language Models
4. Evaluation and Perplexity
5. Generating from an n-gram Language Model
   i) Zeros

Generating from an n-gram model and Zeros

Recall: BRP

P(english|want) = .0011

P(chinese|want) = .0065

P(to|want) = .66

P(eat | to) = .28

P(food | to) = 0

P(want | spend) = 0

P (i | <s>) = .25

How can we generate sentences from this bigram model?

Generating from a bigram model

Choose a random bigram (<s>, w) according to its probability
Now choose a random bigram (w, x)
And so on until we choose </s>
Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food </s>
I want to eat Chinese food
```

Shakespearean n-grams

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in; –It cannot be but so.