Lecture 2:
n-gram Language Models II

Instructor: Xiang Ren

USC CSCI 444 NLP

2026 Spring

Slides mostly adapted from Dan Jurafsky, some from Mohit Iyyer

Announcements

Semester Project

Students will design and carry out a research project that aims to answer a question in natural language processing.

The focus of the class project can be research-focused or application-focused.

A **research-focused project** will develop models and analyze data of an existing problem in NLP, or formulate a new problem altogether.

An **application-focused project (proof-of-concept demo)** will train (possibly only fine-tuning) and deploy NLP models to new application areas, while not necessarily developing any novel research question to be answered.

Students will leverage tools, concepts, and techniques presented in the class.

Semester Project

- Styled like a research paper
- **Individually**, or Teams of 2 or 3 students
  - We expect to see ~25 groups, based on enrollment
- Grading:
  - 5%: presentation of project pitch (3 mins; 1-2 slide) +
  - 5%: project proposal (1 page) +
  - 5%: presentation of midterm progress (3 mins) +
  - 10%: midterm report (4 pages) +
  - 15%: presentation of main findings (13 mins) +
  - 15%: final report (8 pages)
- **Project pitch on Jan 26**

11	Mar 23	Language Generation (contd.)	J&M, Chap 7.4 + 7.6; Nucleus Sampling;	
	Mar 25	Pre-Training LLMs	J&M, Chap 7; The Llama 3 Herd of Models; OLMo 2.0; Switch Transformers;	Project Midterm Report Due
12	Mar 30	Pre-training and post-training language models; model evaluation	Bloom - BigScience Clip- Radford et al., 2021	
	Apr 1	Paper Presentation and Discussion I		
13	Apr 6	Cutting Edge Topics in NLP		
	Apr 8	Paper Presentation and Discussion II		HW 3 Due
14	Apr 13	Cutting Edge Topics in NLP		
	Apr 15	Paper Presentation and Discussion III		
15	Apr 20	Semester Project Presentations I		Project Presentations due
	Apr 22	Semester Project Presentations II		
16	Apr 27	Semester Project Presentations III		
	Apr 29	Semester Project Presentations IV		
FINAL	May 11			Project Final Report due Serves as final exam

Semester Project

Proposal (5%)

The project proposal (1 page, 10pt font size) should outline the type of project (research-focused or application-focused), and then answer the following questions clearly in a sentence and/or a few paragraphs each:

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares?
- If you are successful, what difference will it make?
- What are the risks? What could go wrong, and how will you pivot early on if that happens?
- How much will it cost? That is, what resources will you need in terms of time and computation? Are these reasonable for a semester and what access you have?
- Identify two milestones along the way to your finished project.

Semester Project

Pitch (5%)

1-2 slides to cover short sentence answers to these questions:

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you are successful, what difference will it make?

Recap: Probabilistic Language Models

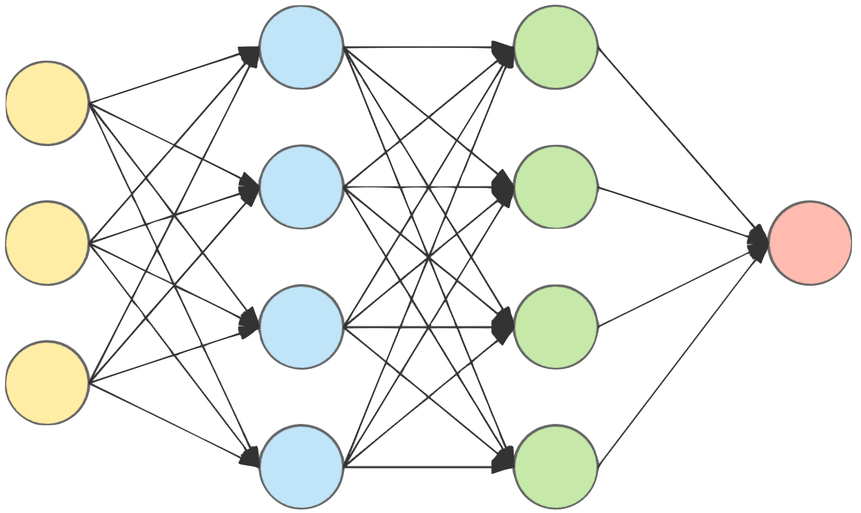
Assign a probability to a sentence

Building a Language Model

What words can follow this?

- Task: Given a sequence of words so far (the context), predict what comes next

You won't break my …



…soul
…spirit
…heart

- We never know for sure what comes next, but we can still make good guesses!

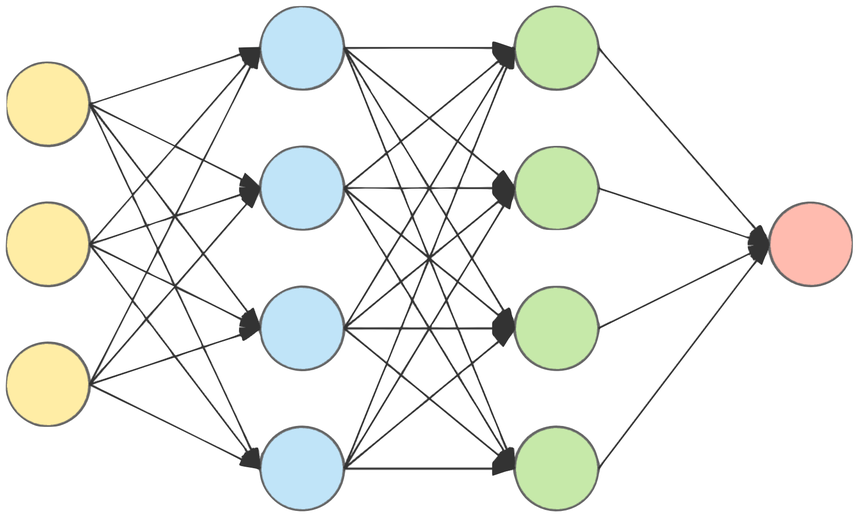
What is common to these words?

Building a Language Model

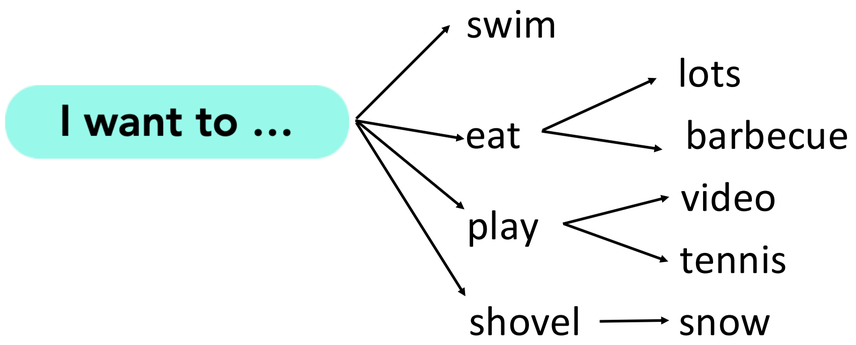
What words can follow this?

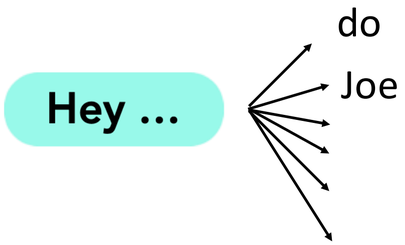
- Task: Given a sequence of words so far the context), predict what comes next
- We never know for sure what comes next, but we can still make good guesses!

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!

The capital of Nebraska is ... Lincoln

Probabilistic Language Modeling

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

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

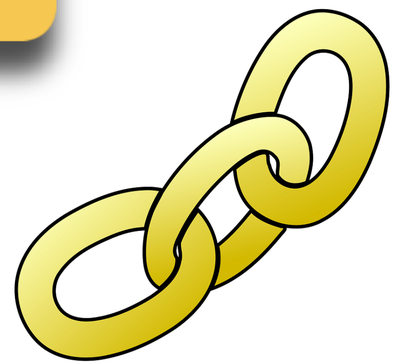
Related task: probability of an upcoming word:

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

A model that assigns probabilities to sequences of words is called a language model

Chain Rule

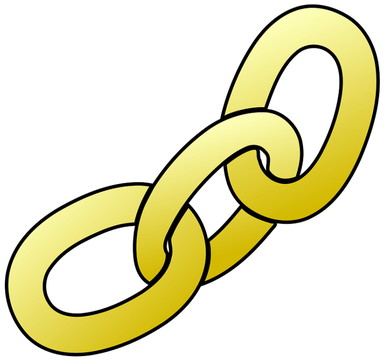
$$P(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} P(w_i | w_{i-1} \ldots w_1)$$



The Chain-Rule

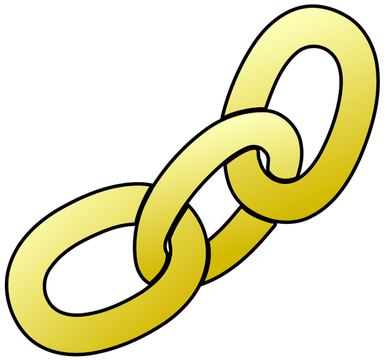
- Recall the definition of conditional probabilities: $P(B|A) = \frac{P(A,B)}{P(A)}$
- Rewriting: $P(A,B) = P(A)P(B|A)$
- More variables: $P(A,B,C,D) = P(A)P(B|A)P(C|B,A)P(D|C,B,A)$
- The Chain Rule in General
- $$P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2) \dots P(x_n|x_1, \dots, x_{n-1})$$

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



The Chain Rule applied to compute joint probability of words in a sentence

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



$$P(\textit{its water is so transparent}) = P(\textit{its}) \times$$
$$P(\textit{water}|\textit{its}) \times$$
$$P(\textit{is}|\textit{its water}) \times$$
$$P(\textit{so}|\textit{its water is}) \times$$
$$P(\textit{transparent}|\textit{its water is so})$$

How to estimate the probability of the next word?

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

Maximum Likelihood Estimate

Machine Learning 101: Maximum Likelihood Estimate

Suppose we have a biased coin that's heads with probability $p = \theta$

- Let θ be the probability of heads - parameter of the model

Suppose we flip the coin four times and see (H, H, H, T)

- Observed data, D = (HHHT)

Machine Learning 101: Maximum Likelihood Estimate

Suppose we have a biased coin that's heads with probability $p = \theta$

- Let θ be the probability of heads - parameter of the model

Suppose we flip the coin four times and see (H, H, H, T)

- Observed data, D = (HHHT)

We don't know what θ is — this is the estimation / learning problem

- Maximum Likelihood Estimate: Choose parameter θ that maximizes likelihood of observed data $D, P(D|\theta)$
  - $\hat{\theta}_{MLE} = \arg\max_{\theta} P(D|\theta)$
  - Data likelihood, $P(D|\theta) = \theta\theta\theta(1-\theta)$ can be maximized by taking the derivative and set it equal to zero and find $\hat{\theta}_{MLE} = 0.75 = \text{count}(\#\text{ heads})/\text{count}(\text{total }\#\text{ of tosses})$

How to estimate the probability of the next word?

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

Could we just count and divide?

- No! Too many possible sentences!
  - We'll never see enough data for estimating these

Markov Assumption

Simplifying assumption:

$$P(\textit{that}|\textit{its water is so transparent}) \approx P(\textit{that}|\textit{transparent})$$

Andrei Markov

Or maybe…

$$P(\textit{that}|\textit{its water is so transparent}) \approx P(\textit{that}|\textit{so transparent})$$

Markov Assumption contd.

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

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

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

k-gram Markov Assumption

n-gram models

Unigram Model

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

Bigram Model

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

k-gram Model

$$P(w_1, w2, \ldots w_n) \approx \prod_i P(w_i | w_{i-k+1} \ldots w_{i-1})$$

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:

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

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

N-gram Models

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

In general this is an insufficient model of language

n-gram Models: Limitations

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."
- "The complex houses married and single soldiers and their families."
- "The horse raced past the barn fell."
- "The old man the boat."

Garden Path Sentences

Language has long-distance dependencies

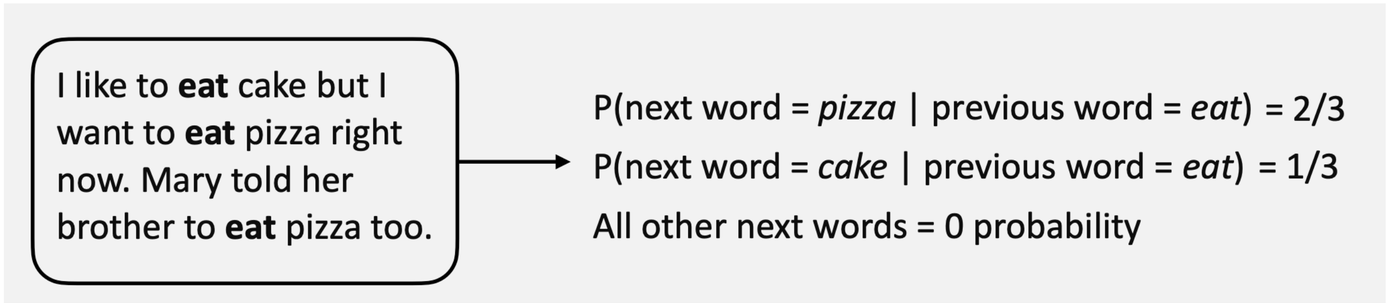
But we can often get away with n-gram models

n-gram Language Models

Simplest probabilistic model

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 English words) instead of 2



Vocabulary

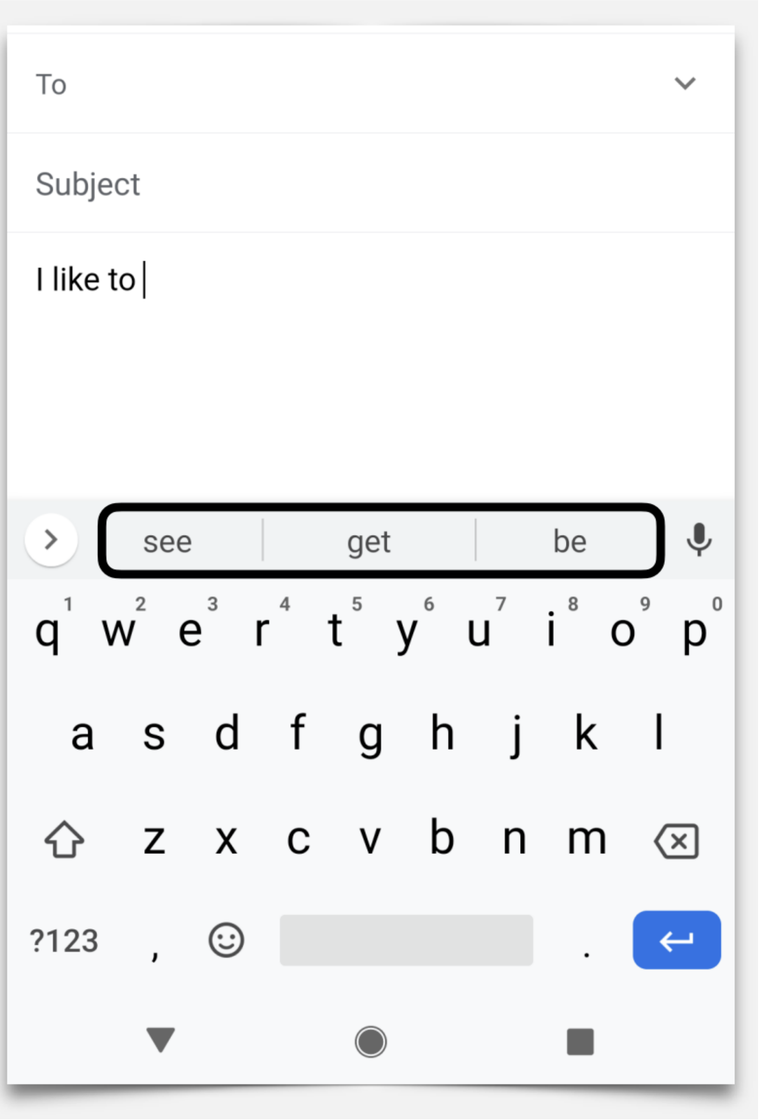
2-gram language models

- 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"

$$P(next\ word = y | previous\ word = x)$$

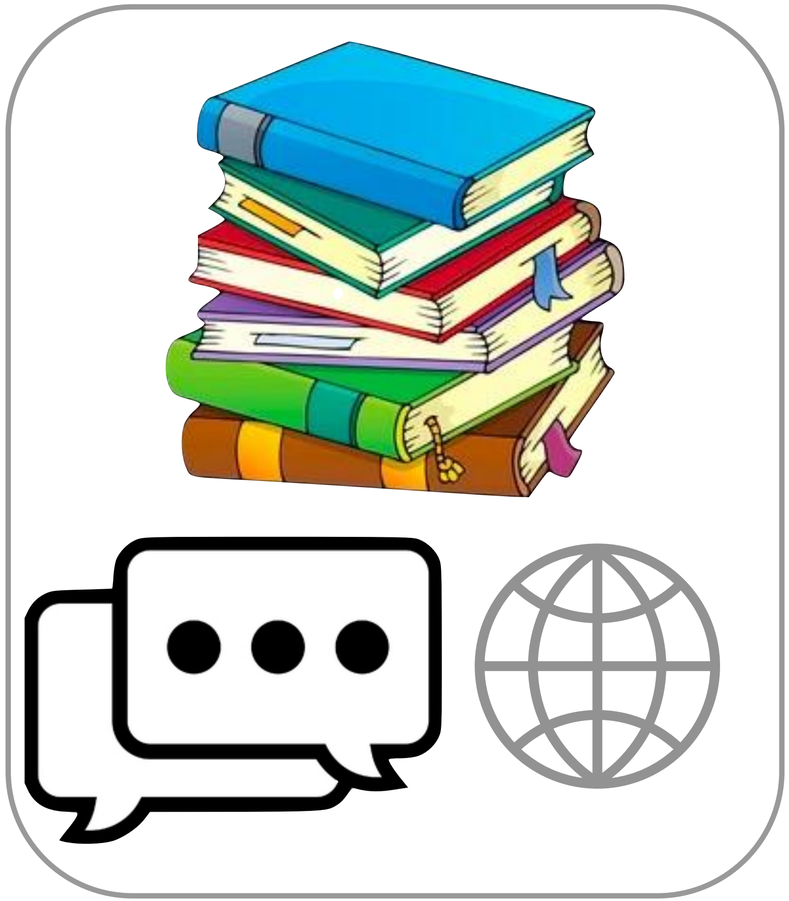
I want to go to_____

$$P(next\ word = was | previous\ word = to) = 0.0$$
$$P(next\ word = LA | previous\ word = to) = 0.2$$
$$P(next\ word = Europe | previous\ word = to) = 0.1$$
$$P(next\ word = Mexico | previous\ word = to) = 0.1$$
$$P(next\ word = eat | previous\ word = to) = 0.1$$

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

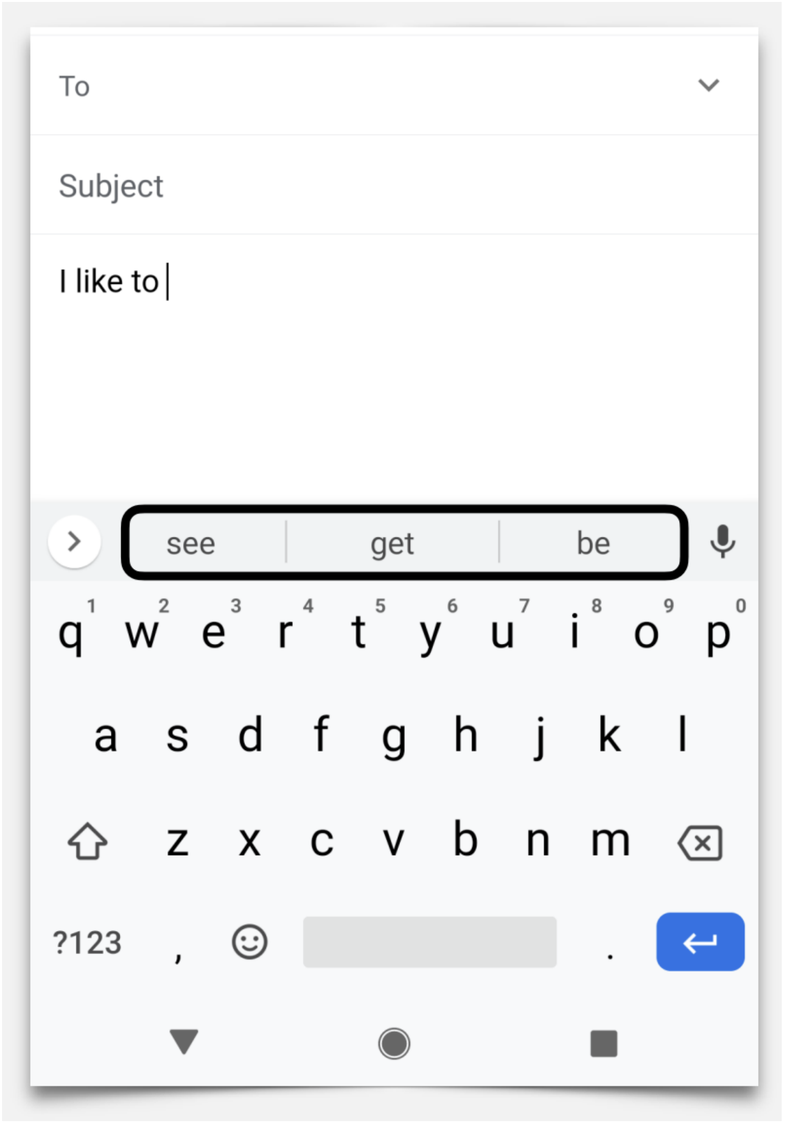
If we see "to", there's a 20% chance the next word is "LA"

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

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

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

What happens when $i = 1$?

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

An example

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

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

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

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

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

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

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

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

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

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

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 (real, common sentences).

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.

How good is a language model?

- Extrinsic vs. Intrinsic Evaluation

  Q: can you think of an extrinsic evaluation?

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

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 $P(sentence)$

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

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

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

Chain rule:

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

Applying Markov's assumption for bigrams:

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

Perplexity Example

Let's suppose a sentence of length 10 consisting of random digits

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

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

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

$$= \left(\frac{1}{10}\right)^{10\,-\frac{1}{10}}$$

$$= 10$$

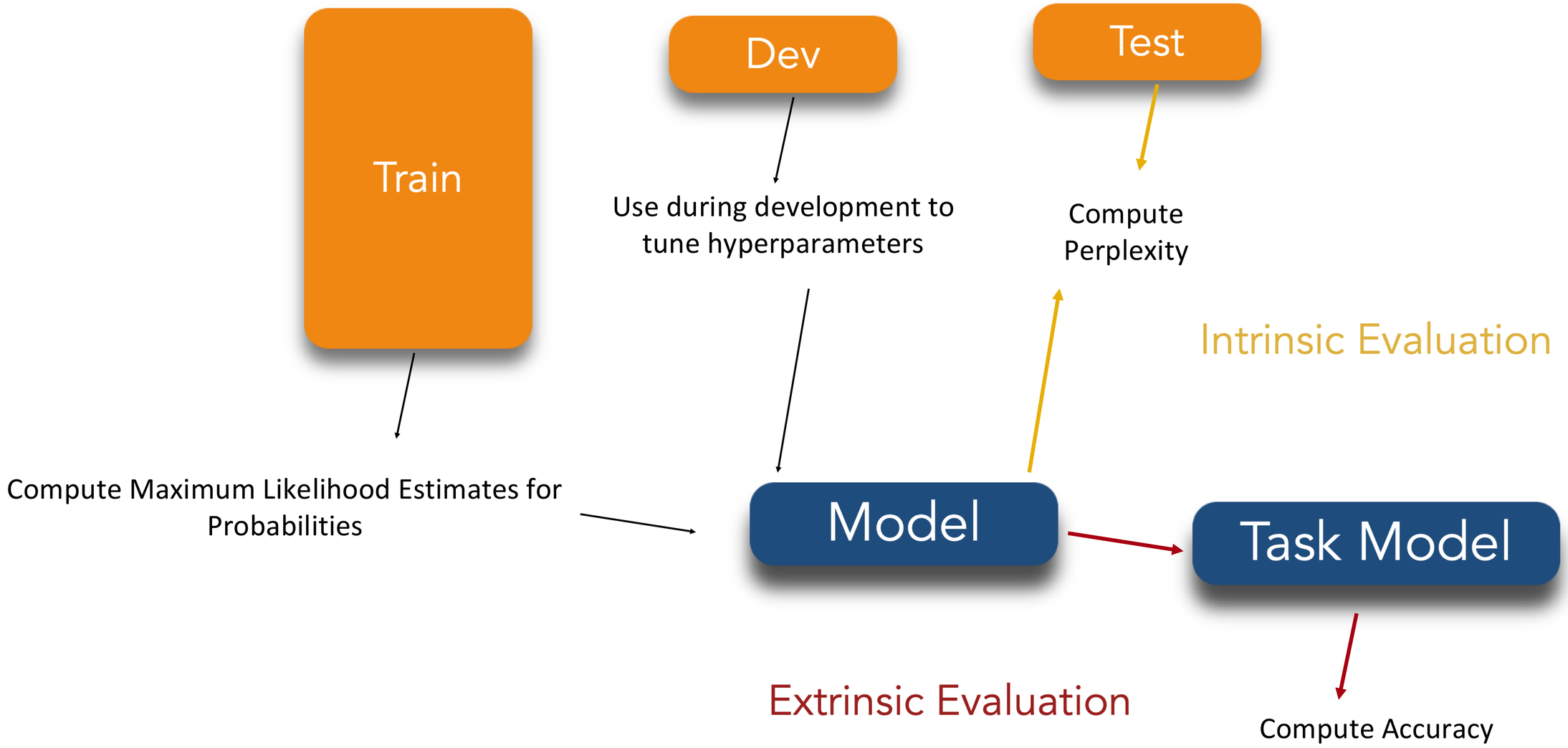
Lower perplexity = better model!

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

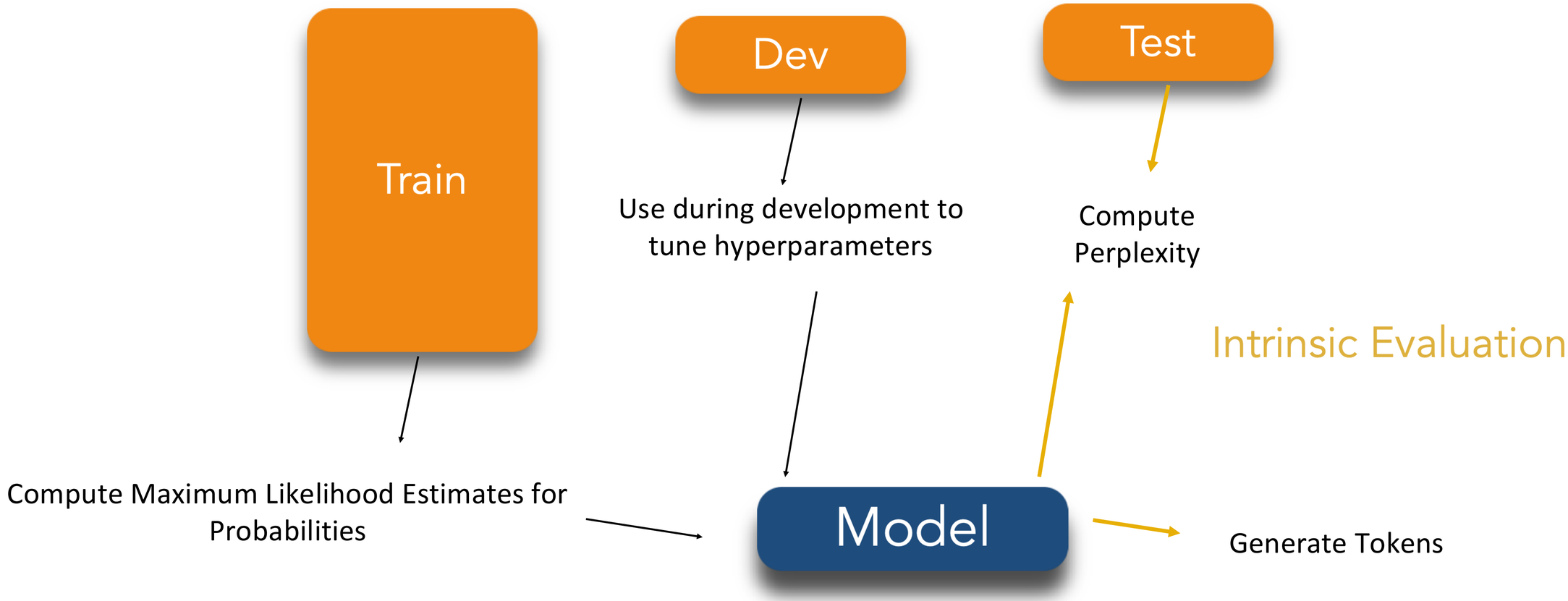
N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

What are the two things that might affect perplexity?

Language Model Development



Language Model Development



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) according to its probability

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.

Shakespeare as a corpus

V=29,066

Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams

- So 99.96% of the possible bigrams were never seen (have zero entries in the table)

4-grams (quadrigrams) are worse: What's coming out looks like Shakespeare because it is Shakespeare!

Most n-grams are never seen!

The WSJ is no Shakespeare!

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

So why not just sample from very high order n-gram models? Do we even need ChatGPT?

Can only produce n-grams from training data!

Shakespearean corpus cannot produce WSJ vocabulary and vice versa

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

The successes we are seeing here are due to a phenomena commonly known as overfitting

Overfitting bad!

N-grams only work well for word prediction if the test corpus looks like the training corpus

- In real life, it often doesn't
- We need to train robust models that generalize!
  - Technical terms for "doing well on the test data" or "doing well on any test data"

Machine Learning 101

- The goal isn't to pound out fake sentences!
- Obviously, generated sentences get "better" as we ***increase the model order.***
  - More precisely: using maximum likelihood estimators, higher order is always better likelihood on training set, but not test set

Machine Learning 101

- At the minimum, we would want to pick the smallest test set that gives us enough statistical power to measure a statistically significant difference between two potential models.

Machine Learning 101

- **LM Test Sets:** Given a large corpus that we want to divide into training and test, test data can either be
    - taken from some continuous sequence of text inside the corpus,
    - or we can remove smaller “stripes” of text from randomly selected parts of our corpus and combine them into a test set.

Overfitting bad!

N-grams only work well for word prediction if the test corpus looks like the training corpus

- In real life, it often doesn't
- We need to train robust models that generalize!
  - Technical terms for "doing well on the test data" or "doing well on any test data"

- One kind of generalization: **Zeros**!
  - Things that don't ever occur in the training set
    - But occur in the test set

N-gram models: Common Issues that need handling

Token | Type | Vocabulary

At test time, we might encounter:

- Token never seen in context (i.e. n-gram with 0 frequency)
- Token never seen (unigram with 0 frequency)
- More severe!
  - Problem: Many words like "***Petrichor***" won't appear in most training sets!
  - These are known as OOV for "out of vocabulary", or unknown tokens

Missing n-grams

Training set:

… denied the allegations

… denied the reports

… denied the claims

… denied the request

Test set

… denied the offer

… denied the loan

$$P(offer|denied\ the) = 0$$

will assign 0 probability to the test set!

And hence we cannot compute perplexity

- No one can divide by 0!

What happens to perplexity??

Missing Unigrams: the <UNK> token

One way to handle OOV tokens is by adding a pseudo-word called <UNK>

Closed Vocabulary: Only allow a list of predetermined tokens, everything else (in the training data) is <UNK>

Closed Vocabulary

We can replace all words that occur fewer than n times in the training set, where n is some small number, by <UNK> and re-estimate the counts and probabilities

Open Vocabulary

When not done carefully, may lead to artificially lower perplexity


Smoothing

Intuition for Smoothing

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

→

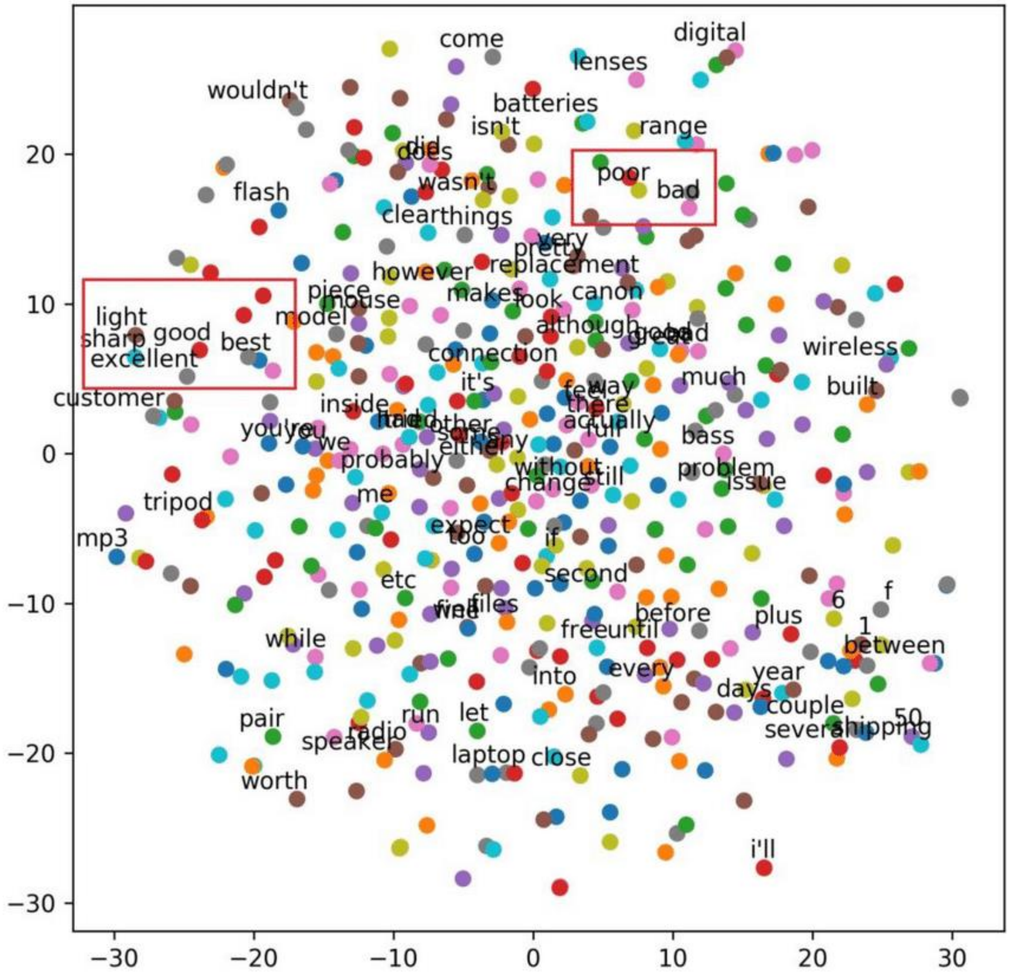
P(next word = *pizza* | previous word = *eat*) = 2/3
P(next word = *cake* | previous word = *eat*) = 1/3
All other next words = 0 probability

- All other vocabulary tokens getting 0 probability just doesn't seem right. We want to assign some probability to other words
- We want to smooth the distribution from our counts

What does a count distribution look like?

Next Class

- Words are more than discrete symbols
- Better feature representations than n-grams
- Parameters!!!



Next Week

- Jan 19 MLK Day – no class!
- HW1 release on Jan 20
- Project pitch on Jan 26 → find your project team ASAP! (if not individual project)
  - Project team finalized by end of Jan → submit your team by Feb 2 and no more adjustment!