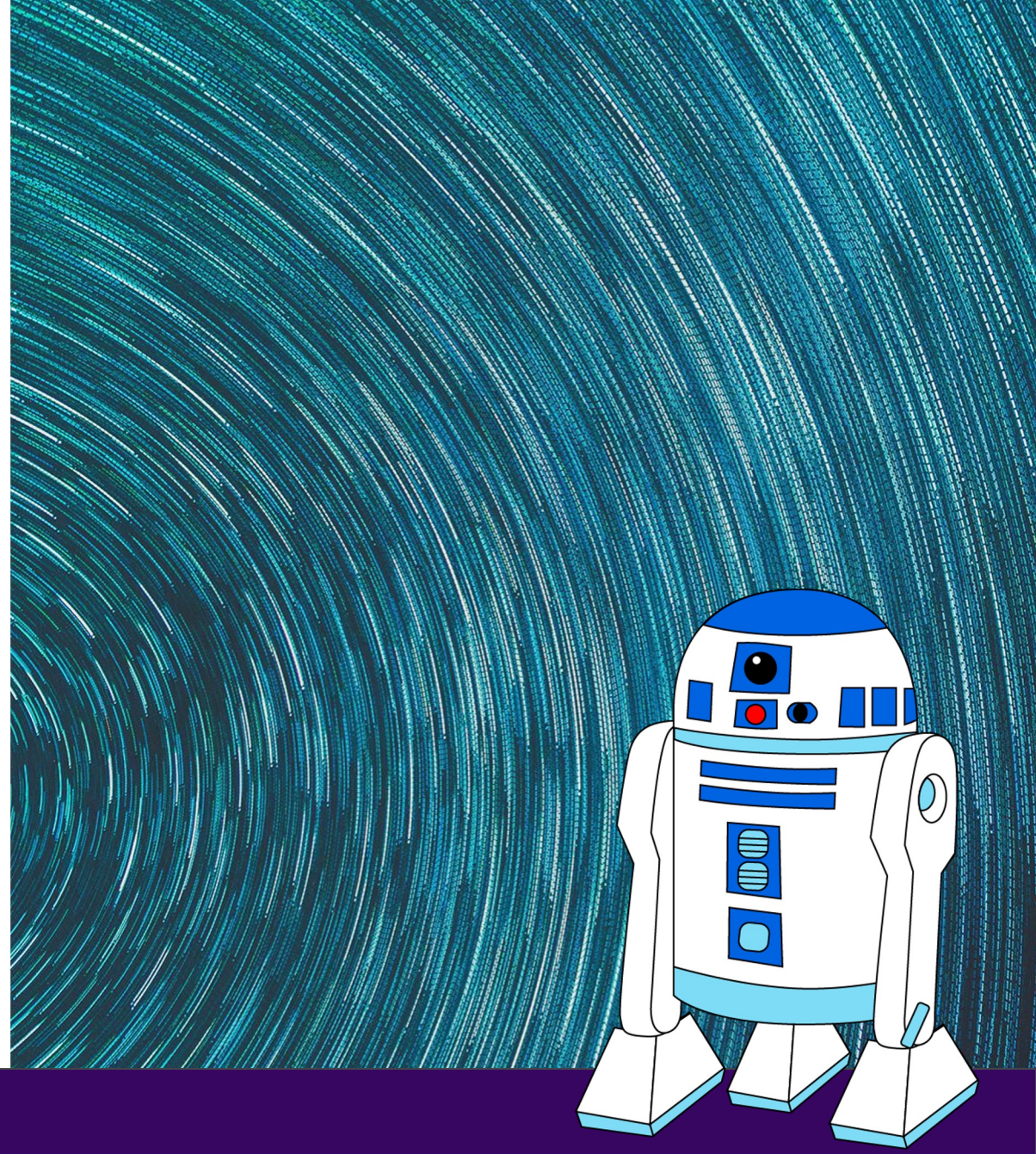
CIS 4210/5210:
ARTIFICIAL INTELLIGENCE

Probabilities and Language Models



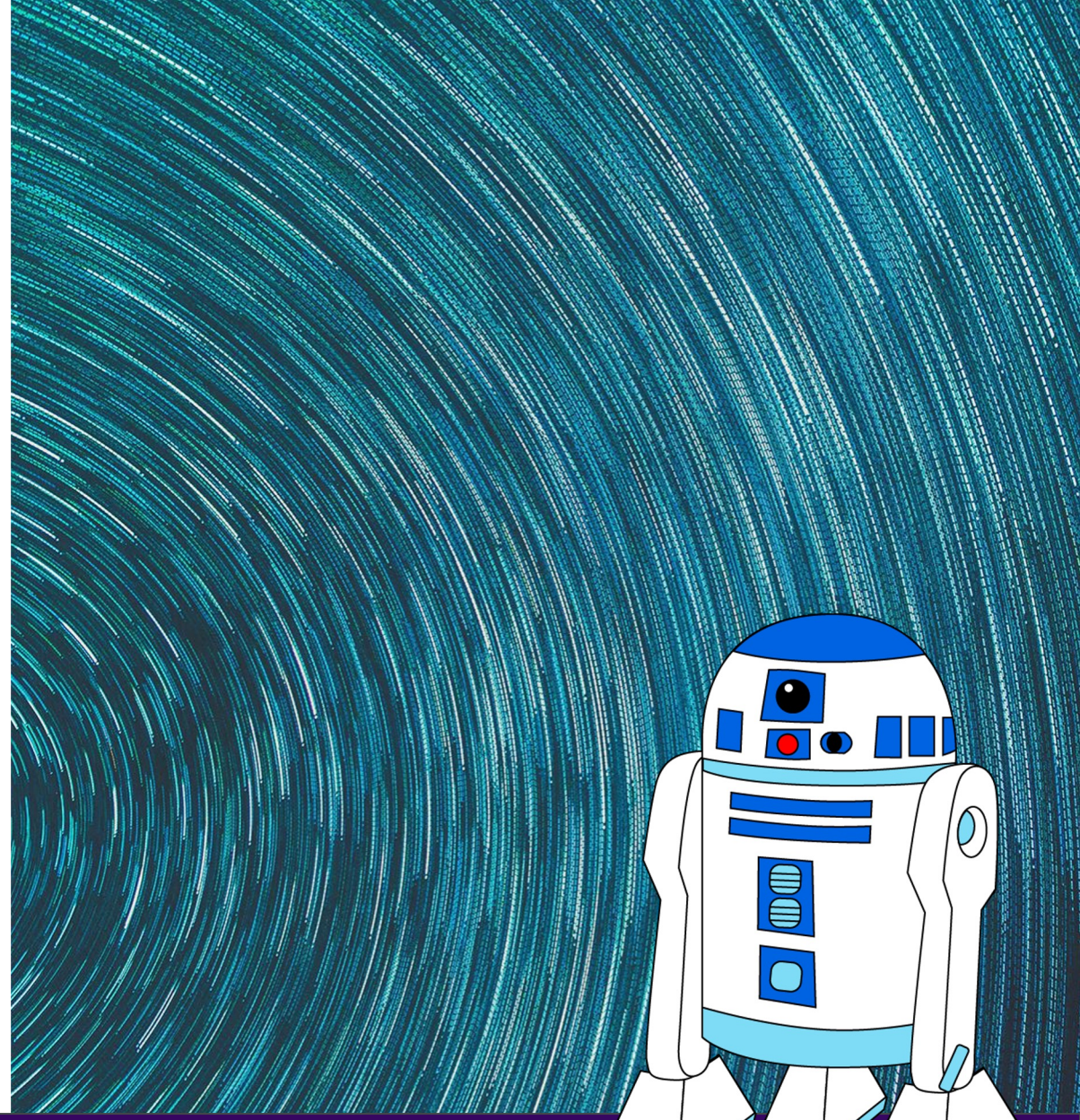
Probabilities and Language Models

- Review the basics of probability theory
  - Axioms of probability
  - What an event is
  - What a probabilistic model is
  - Interactions between joint probabilities, conditional probabilities, marginal distributions
- The chain rule
- Bayes rule
- Performing probabilistic inference
- Probabilistic language models

Using a new textbook for this section.

CIS 4210/5210:
ARTIFICIAL INTELLIGENCE

Review of Probabilities



Penn Engineering
University of Pennsylvania

Uncertainty

General situation:

- **Observed variables (evidence)**: Agent knows certain things about the state of the world (e.g., sensor readings or symptoms)
- **Unobserved variables (states)**: Agent needs to reason about other aspects (e.g. where an object is or what disease is present)
- **Model**: Agent knows something about how the known variables relate to the unknown variables

Probabilistic reasoning gives us a framework for managing our beliefs and knowledge

What Probabilities Are About

Like logical assertions, probabilities are about **possible worlds**. Instead of strictly ruling out possibilities (where a logical assertion is false), probabilities quantify **how likely** a particular possible world is.

In probability theory, the possible worlds are called the **sample space**, and they **mutually exclusive** and **exhaustive.**

A fully specified probability model associates a probability **P(*w*)** with each possible world ***w***.

Random Variables

A random variable is some aspect of the world about which we (may) have uncertainty

- R = Is it raining?
- U = Is the professor carrying an umbrella?

We denote random variables with capital letters

Axioms of Probability

The probability of any possible world is between 0 and 1.

$$0 \leq P(w) \leq 1 \text{ for every w}$$

The total probability of the set of all possible worlds is 1:

$$\sum_{w \in \Omega} P(w) = 1$$

Probability Distributions

Unobserved random variables have distributions

$P(T)$

T	P
hot	0.5
cold	0.5

$P(W)$

W	P
sun	0.6
rain	0.1
fog	0.3

A distribution is a TABLE of probabilities of values

A probability (lower case value) is a single number

Shorthand notation:

$$P(hot) = P(T = hot),$$
$$P(cold) = P(T = cold),$$
$$P(rain) = P(W = rain),$$
$$\dots$$

OK *if* all domain entries are unique

Must have: $P(W = rain) = 0.1$ and

$$\forall x \;\; P(X = x) \geq 0$$

$$\sum_x P(X = x) = 1$$

Joint Distributions

A *joint distribution* over a set of random variables: $X_1, X_2, \ldots X_n$ specifies a real number for each assignment (or *outcome*):

$$P(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n)$$

$$P(x_1, x_2, \ldots x_n)$$

- Must obey:

$$P(x_1, x_2, \ldots x_n) \geq 0$$

$$\sum_{(x_1, x_2, \ldots x_n)} P(x_1, x_2, \ldots x_n) = 1$$

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

Size of distribution if n variables with domain sizes d?

- For all but the smallest distributions, impractical to write out!

Probabilistic Models

A probabilistic model is a joint distribution over a set of random variables

Probabilistic models:

- (Random) variables with domains
- Assignments are called *outcomes*
- Joint distributions: say whether assignments (outcomes) are likely
- *Normalized:* sum to 1.0
- Ideally: only certain variables directly interact

Distribution over T,W

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

Events

An *event* is a set E of outcomes

$$P(E) = \sum_{(x_1 \ldots x_n) \in E} P(x_1 \ldots x_n)$$

From a joint distribution, we can calculate the probability of any event

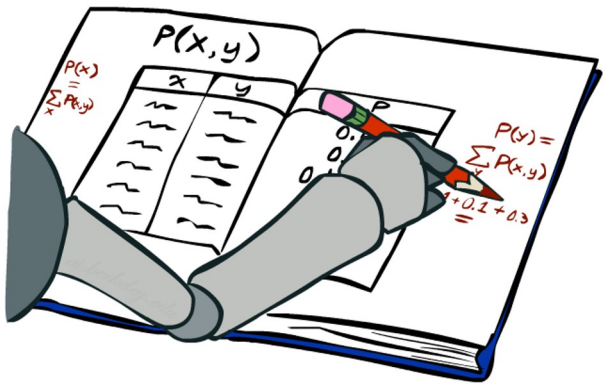
- Probability that it's hot AND sunny?
- Probability that it's hot?
- Probability that it's hot OR sunny?

Typically, the events we care about are *partial assignments*, like P(T=hot)

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

Marginal Distributions



Marginal distributions are sub-tables which eliminate variables

Marginalization (summing out): Combine collapsed rows by adding

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

$$P(t) = \sum_s P(t, s)$$

$P(T)$

T	P
hot	0.5
cold	0.5

$$P(s) = \sum_t P(t, s)$$

$P(W)$

W	P
sun	0.6
rain	0.4

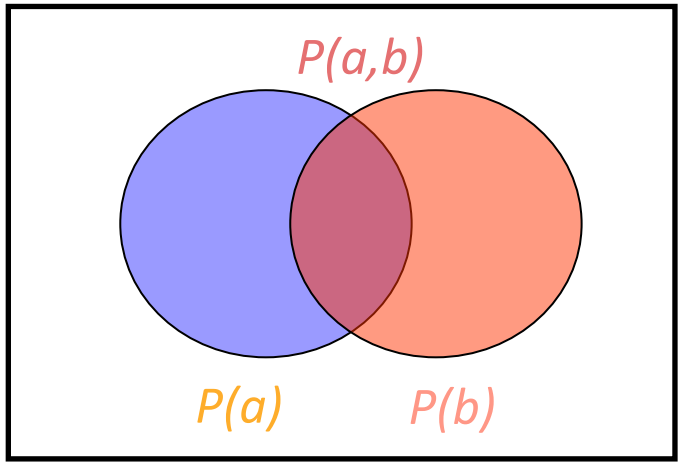
$$P(X_1 = x_1) = \sum_{x_2} P(X_1 = x_1, X_2 = x_2)$$

Conditional Probabilities

A simple relation between joint and conditional probabilities

- In fact, this is taken as the *definition* of a conditional probability

$$P(a|b) = \frac{P(a,b)}{P(b)}$$



$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

$$P(W = s|T = c) = \frac{P(W = s, T = c)}{P(T = c)} = \frac{0.2}{0.5} = 0.4$$

$$= P(W = s, T = c) + P(W = r, T = c)$$
$$= 0.2 + 0.3 = 0.5$$

Conditional Distributions

Conditional distributions are probability distributions over some variables given fixed values of others

Conditional Distributions

$P(W|T)$

$P(W|T = hot)$

W	P
sun	0.8
rain	0.2

$P(W|T = cold)$

W	P
sun	0.4
rain	0.6

Joint Distribution

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

Probabilistic Inference

Probabilistic inference: compute a desired probability from other known probabilities (e.g. conditional from joint)

We generally compute conditional probabilities

- P(on time | no reported accidents) = 0.90
- These represent the agent's *beliefs* given the evidence

Probabilities change with new evidence:

- P(on time | no accidents, 5 a.m.) = 0.95
- P(on time | no accidents, 5 a.m., raining) = 0.80
- Observing new evidence causes *beliefs to be updat*

** Works fine with multiple query variables, too*

Inference by Enumeration

General case:

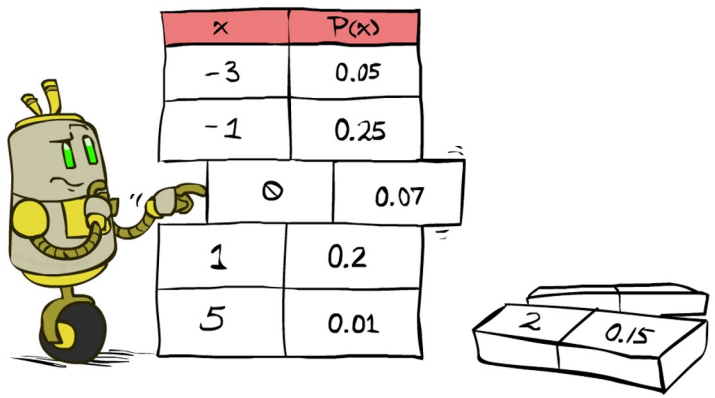
- Evidence variables: $E_1 \dots E_k = e_1 \dots e_k$
- Query* variable: Q
- Hidden variables: $H_1 \dots H_r$

$\left.\right\} X_1, X_2, \dots X_n$ *All variables*

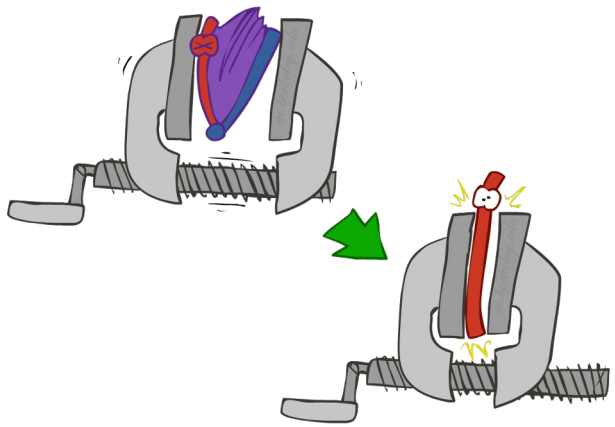
- We want:

$$P(Q|e_1 \dots e_k)$$

- Step 1: Select the entries consistent with the evidence



- Step 2: Sum out H to get joint of Query and evidence



$$P(Q, e_1 \dots e_k) = \sum_{h_1 \dots h_r} P(\underbrace{Q, h_1 \dots h_r, e_1 \dots e_k}_{X_1, X_2, \dots X_n})$$

- Step 3: Normalize

$$\times \frac{1}{Z}$$

$$Z = \sum_q P(Q, e_1 \cdots e_k)$$

$$P(Q|e_1 \cdots e_k) = \frac{1}{Z} P(Q, e_1 \cdots e_k)$$

The Product Rule

$$P(y)P(x|y) = P(x,y)$$

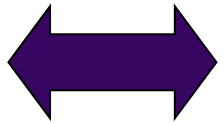
Example:

$P(W)$

R	P
sun	0.8
rain	0.2

$P(D|W)$

D	W	P
wet	sun	0.1
dry	sun	0.9
wet	rain	0.7
dry	rain	0.3



$P(D,W)$

D	W	P
wet	sun	
dry	sun	
wet	rain	
dry	rain	

The Chain Rule

More generally, can always write any joint distribution as an incremental product of conditional distributions

$$P(x_1, x_2, x_3) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)$$

$$P(x_1, x_2, \ldots x_n) = \prod_i P(x_i|x_1 \ldots x_{i-1})$$

Why is this always true?

Bayes' Rule

Two ways to factor a joint distribution over two variables:

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

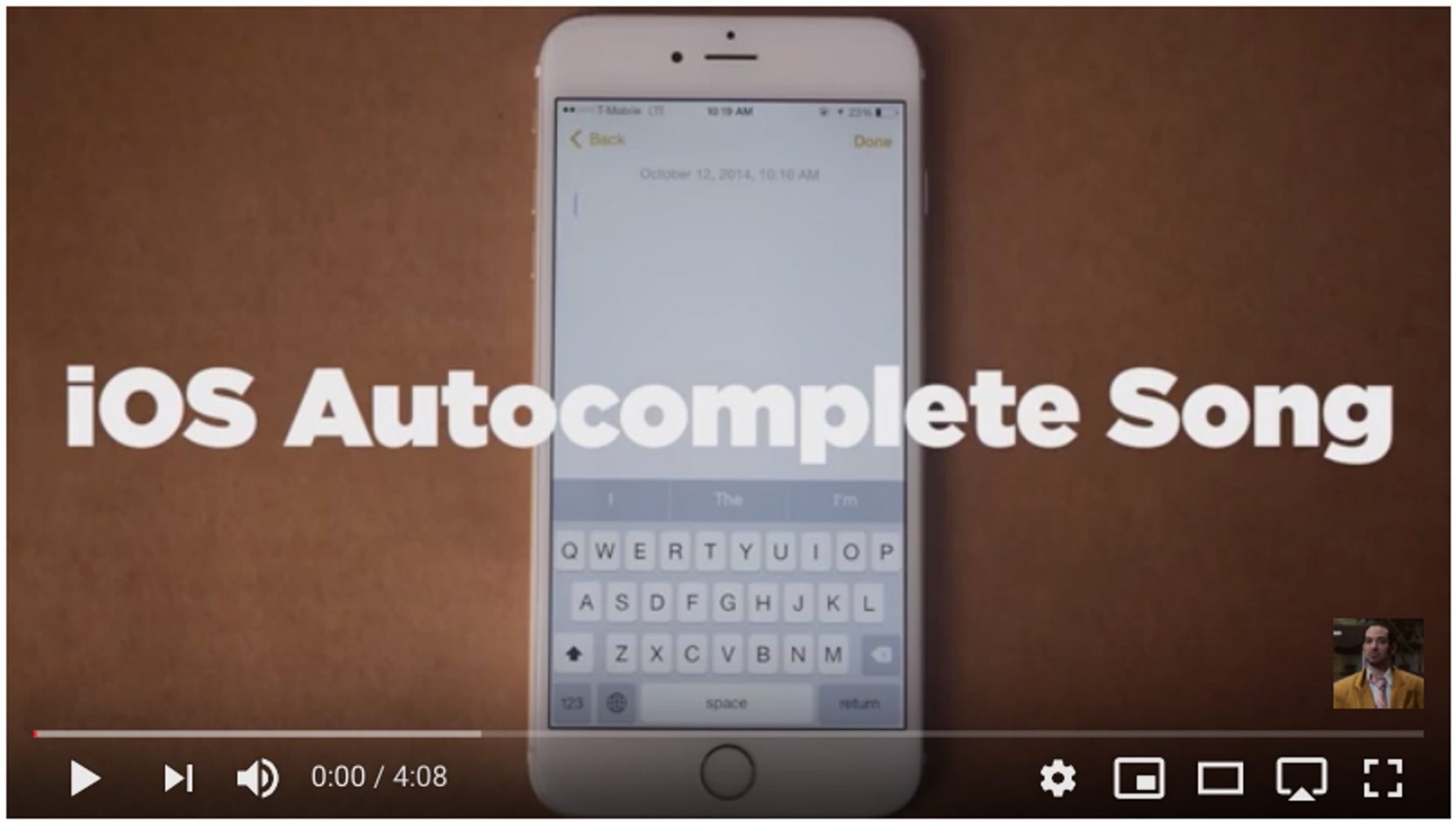
Dividing, we get:

$$P(x|y) = \frac{P(y|x)}{P(y)}P(x)$$

Why is this at all helpful?

- Lets us build one conditional from its reverse
- Often one conditional is tricky but the other one is simple
- Foundation of many systems we'll see later (e.g. ASR, MT)

In the running for most important AI equation!



♫ iOS Autocomplete Song | Song A Day #2110

https://www.youtube.com/watch?v=M8MJFrdfGe0

Probabilistic Language Models

Probabilistic Language Models

One goal: assign a probability to a sentence

- Autocomplete for texting
- Machine Translation
- Spelling Correction
- Speech Recognition

Other Natural Language Generation tasks: summarization, question-answering, dialog systems

Probabilistic Language Modeling

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

Related task: probability of an upcoming word

A model that computes either of these is called a **language model** or **LM**

Probabilistic Language Modeling

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

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

Related task: probability of an upcoming word

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

A model that computes either of these

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

Better: **the grammar** But **language models** are standard

How to compute P(W)

How to compute this joint probability:

P(the, underdog, Philadelphia, Phillies, won)

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

The Chain Rule

The Chain Rule

Recall the definition of conditional probabilities

$p(B|A) = P(A,B)/P(A)$ Rewriting: **$P(A,B) = P(A)P(B|A)$**

More variables:

$$P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$

The Chain Rule in General

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

Joint probability of words in sentence

Joint probability of words in sentence

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

P("the underdog Philadelphia Phillies won") =

P(the) ×
P(underdog|the) ×
P(Philadelphia|the underdog) ×
P(Phillies|the underdog Philadelphia) x
P(won|the underdog Philadelphia Phillies)

How to estimate these probabilities

Could we just count and divide?

How to estimate these probabilities

Could we just count and divide? Maximum likelihood estimation (MLE)

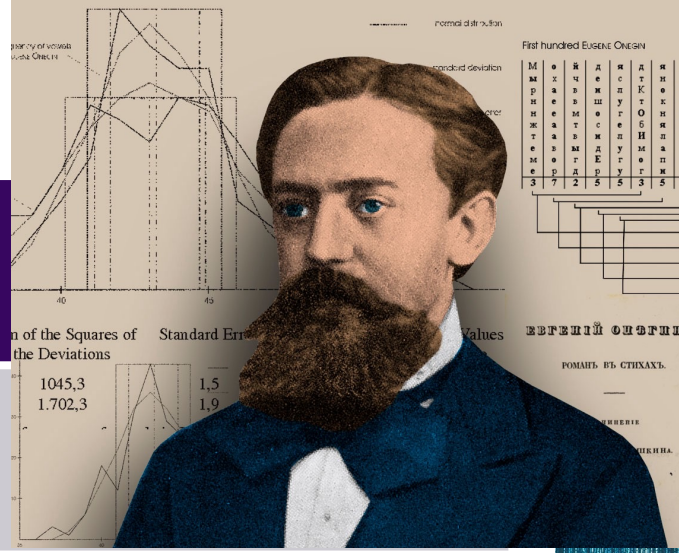
$$P(\text{won} \mid \text{the underdog team}) = \frac{\text{Count}(\text{the underdog team won})}{\text{Count}(\text{the underdog team})}$$

Why doesn't this work? Why is it not practical?

Simplifying Assumption = Markov Assumption

Historical Notes

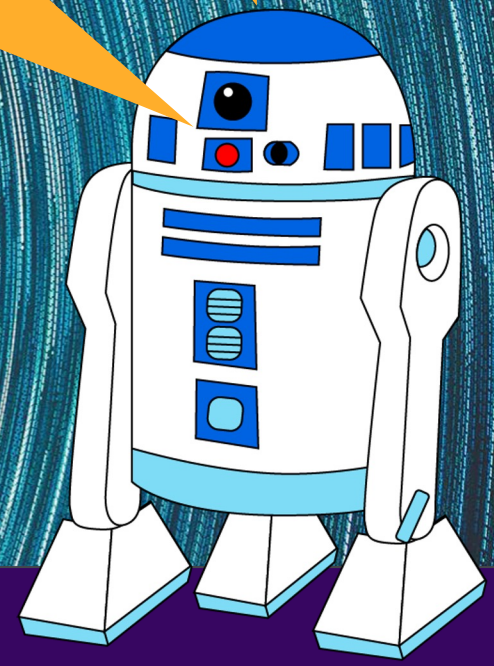
1913	Andrei Markov counts 20k letters in *Eugene Onegin*
1948	Claude Shannon uses n-grams to approximate English
1956	Noam Chomsky decries finite-state Markov Models
1980s	Fred Jelinek at IBM TJ Watson uses n-grams for ASR, think about 2 other ideas for models: (1) MT, (2) stock market prediction
1993	Jelinek and team develops statistical machine translation argmax_e p(e|f) = p(e) p(f|e)
	Jelinek left IBM to found CLSP at JHU Peter Brown and Robert Mercer move to Renaissance Technology



CIS 4210/5210:
ARTIFICIAL INTELLIGENCE

Halloween candy (sour patch kids) by the podium. Please take some!

No lecture on Tuesday Nov 8. I'm canceling it to give you time to vote.



Language Modeling

Estimating N-gram Probabilities

Joint probability of words in sentence

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

P("the underdog Philadelphia Phillies won") =

P(the) ×
P(underdog|the) ×
P(Philadelphia|the underdog) ×
P(Phillies|the underdog Philadelphia) x
P(won|the underdog Philadelphia Phillies)

How to estimate these probabilities

Could we just count and divide? Maximum likelihood estimation (MLE)

$$P(\text{won} \mid \text{the underdog team}) = \frac{\text{Count}(\text{the underdog team won})}{\text{Count}(\text{the underdog team})}$$

Why doesn't this work? Why is it not practical?

Simplifying Assumption = Markov Assumption

$P(\text{won} \mid \text{the underdog team}) \approx P(\text{won} \mid \text{team})$

Only depends on the previous k words, not the whole context

$\approx P(\text{won} \mid \text{underdog team})$

$\approx P(w_i \mid w_{i-2}\, w_{i-1})$

$$P(w_1 w_2 w_3 w_4 \ldots w_n) \approx \prod_i^n P(w_i \mid w_{i-k} \ldots w_{i-1})$$

K is the number of context words that we take into account

N-gram models use limited history

Model	History
unigram	**no history**
bigram	1 word as history
trigram	2 words as history
4-gram	3 words as history

Estimating bigram probabilities

The Maximum Likelihood Estimate

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

A Worked Example

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

<s> I am Sam </s>

<s> Sam I am </s>

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

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

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

N-gram models use limited history

unigram	**no history**	$\prod_{i}^{n} p(w_i)$	$p(w_i) = \frac{count(w_i)}{all\ words}$
bigram	1 word as history	$\prod_{i}^{n} p(w_i \mid w_{i-1})$	$p(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$
trigram	2 words as history	$\prod_{i}^{n} p(w_i \mid w_{i-2}, w_{i-1})$	$p(w_i \mid w_{i-2}, w_{i-1}) = \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})}$
4-gram	3 words as history	$\prod_{i}^{n} p(w_i \mid w_{i-3}, w_{i-2}, w_{i-1})$	$p(w_i \mid w_{i-3}, w_{i-2}, w_{i-1}) = \frac{count(w_{i-3}, w_{i-2}, w_{i-1}, w_i)}{count(w_{i-3}, w_{i-2}, w_{i-1})}$

N-gram models versus long distance dependencies

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

In general, this is an insufficient model of language
Because languages have **long distance dependencies**

*The **pictures are** beautiful.*

*The **pictures** of the old man **are** beautiful.*

*The **pictures** of the old man holding his grandchild **are** beautiful.*

Problems for MLE

Zeros

Train	Test
denied the allegations	denied the memo
denied the reports	
denied the claims	
denied the requests	

P(memo|denied the) = 0

And we also assign 0 probability to all sentences containing it!

Problems for MLE

Out of vocabulary items (OOV)

<unk> to deal with OOVs

Fixed lexicon L of size V

Normalize training data by replacing any word not in L with <unk>

Avoid zeros with smoothing

Practical Issues

We do everything in log space

- Avoid underflow
- (also adding is faster than multiplying)

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

Google N-Gram Release, August 2006

AUG
3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

...

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

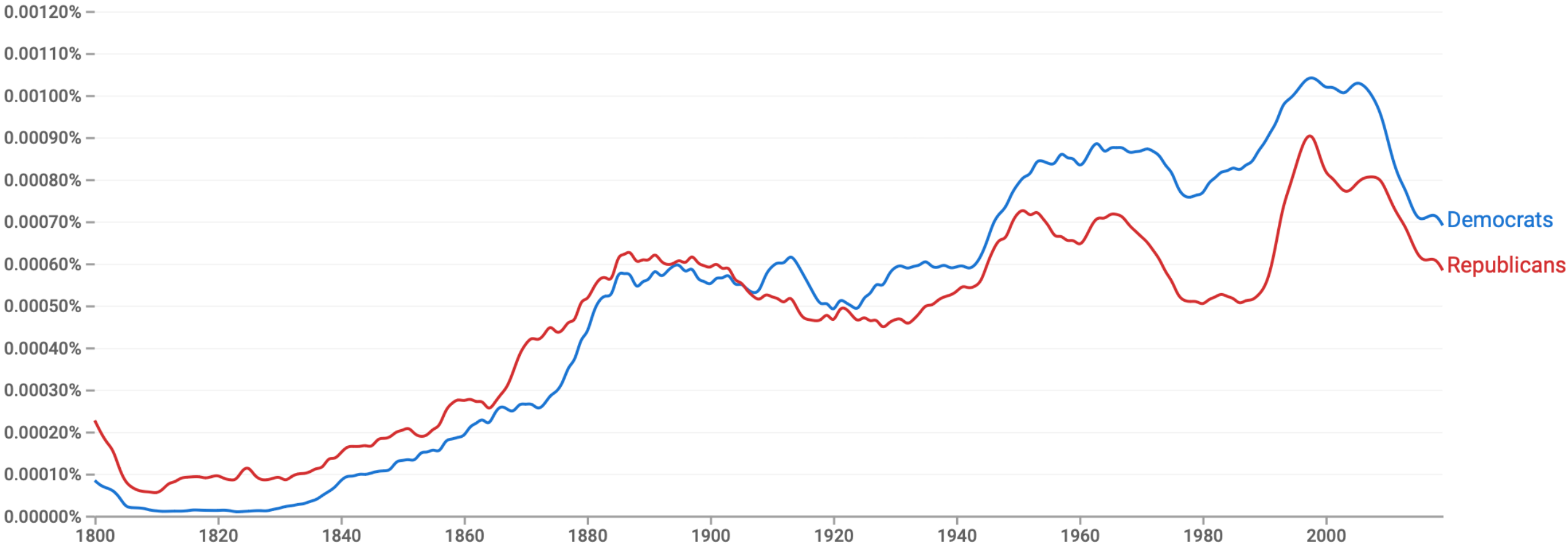
Google N-Gram Release

```
serve as the incoming 92
serve as the incubator 99
serve as the independent 794
serve as the index 223
serve as the indication 72
serve as the indicator 120
serve as the indicators 45
serve as the indispensable 111
serve as the indispensible 40
serve as the individual 234
```

https://ai.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html

Google Book N-Grams

https://books.google.com/ngrams



Evaluation and Perplexity

Evaluation: How Good is Our Model

Does our language model prefer good sentences to bad ones?

- Assign higher probability to “real” or “frequently observed” sentences
  - Than “ungrammatical” or “rarely observed” sentences?

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.

Training on the 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”

Bad science!

And violates the honor code

Extrinsic Evaluation of Language Models

Difficulty of extrinsic (task-based) evaluation of LMs

Extrinsic evaluation

- Time-consuming; can take days or weeks

So

- Sometimes use **intrinsic** evaluation: **perplexity**
- Bad approximation
  - unless the test data looks **just** like the training data
  - So **generally only useful in pilot experiments**
- But is helpful to think about

Intuition of Perplexity

The Shannon Game:

- How well can we predict the next word?

Intuition of Perplexity

The Shannon Game:

- How well can we predict the next word?

I always order pizza with cheese and ____

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

….

and 1e-100

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 ____

- Unigram models are terrible at this game. (Why?)

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 ____

- Unigram models are terrible at this game. (Why?)

A better model of a text

is one that 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

$$PP(W) = P(w_1 w_2 \cdots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \cdots w_N)}}$$

Chain rule $$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1, w_2 \cdots w_{i-1})}}$$

For bigrams $$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

Perplexity as branching factor

Let’s suppose a sentence consisting of random digits

What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

Perplexity as branching factor

Let's suppose a sentence consisting of random digits

What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 \cdots w_N)^{-\frac{1}{N}} \\
&= \left(\frac{1}{10}\right)^{N^{-\frac{1}{N}}} \\
&= \frac{1}{10}^{-1} \\
&= 10
\end{aligned}
$$

Lower perplexity = better model

Minimizing perplexity is the same as maximizing probability

Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Generation

The Shannon Visualization Method

Choose a random bigram (<s>, w) according to its probability

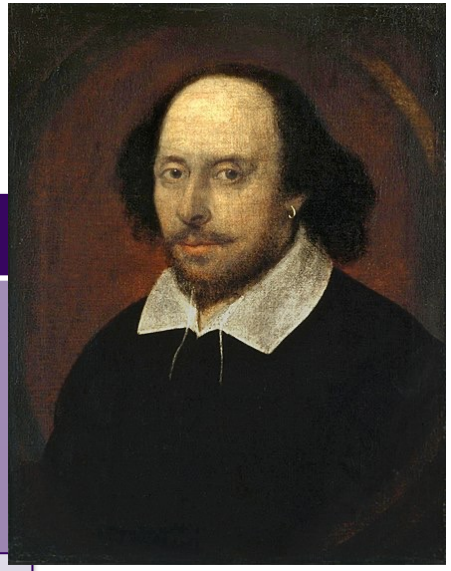
Now choose a random bigram (w, x) according to its probability

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
```

Approximating Shakespeare



Model	Generated Output
Unigram	–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
Bigram	–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.
Trigram	–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 corpus

N=884,647 tokens, V=29,066 types

Shakespeare as corpus

N=884,647 tokens, V=29,066 types

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 worse: What's coming out looks like Shakespeare because it ***is*** Shakespeare

Can you guess the author of these random 3-gram sentences?

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 gram Brazil on market conditions

This shall forbid it should be branded, if renown made it empty.

“You are uniformly charming!” cried he, with a smile of associating and now and then I bowed and they perceived a chaise and four to wish for.

The perils of overfitting

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

Smoothing

Zero probability bigrams

Bigrams with zero probability

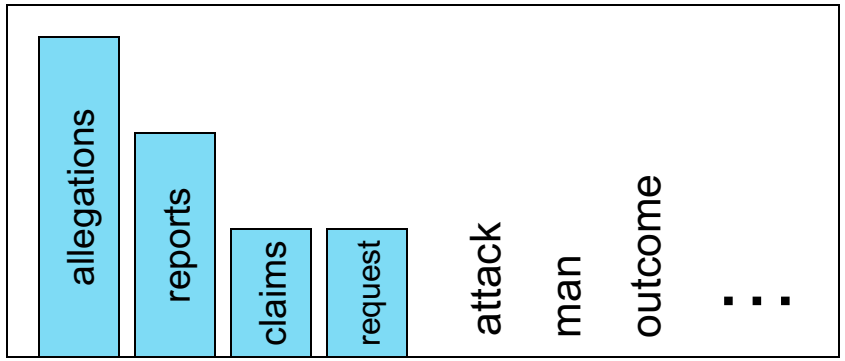
- mean that we will assign 0 probability to the test set!

And hence we cannot compute perplexity (can't divide by 0)!

The intuition of smoothing

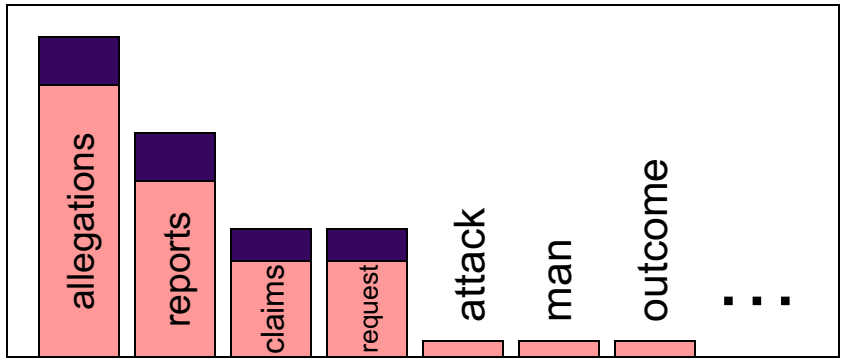
When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



Steal probability mass to generalize better

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



Add-one estimation

Also called Laplace smoothing

Pretend we saw each word one more time than we did

Just add one to all the counts!

MLE estimate:

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

Add-1 estimate:

$$P_{Add-1}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$