NLP1 2023/24

Language Modelling

Lecturer: Wilker Aziz
(week 1, lecture b)

Where are we at?

- → What makes NLP hard
- → Text classification
- → **Language modelling**

Self-study: tabular CPDs

Yay

46

Nay

10

👍 6

👤 56

Ask me anything about tabular CPDs

2 questions
3 upvotes

Outline

- → **Language model**
- → NGram LM
- → Tabular parameterisation
- → Evaluation

What is a language model (LM)?

25 responses

A model that is able to produce language

Probability distribution over words

Models the conditional probability of the next token given the previous ones

Formal description of language

A model of language, with all its norms, rules and intricacies

A model that is capable to represent a language

model that generate probabilities of a series of words

LM is a model that can generate a sequence of words as output

Formal description of language

👍 3

👤 23

What is a language model (LM)?

25 responses

A model trained on a text dataset able to generate new text

probabilistic description of language

GPT

A model that can understand/classify language

a model that assigns probability to words or sentences

Probability distribution over tokens

Chatgpt says. A language model ai understands

a model which is able to produce language?(are we expected to know this already?)

A language model is a type of artificial intelligence system that is designed to understand and generate human language. Language models are trained on large datasets of text.

👍 3

👤 23

What is a language model (LM)?

25 responses

grammatics

Are we supposed to already know what a language model is?

Formal mathematical model of a language.

a model that turn language into a quantitative mode and back into language

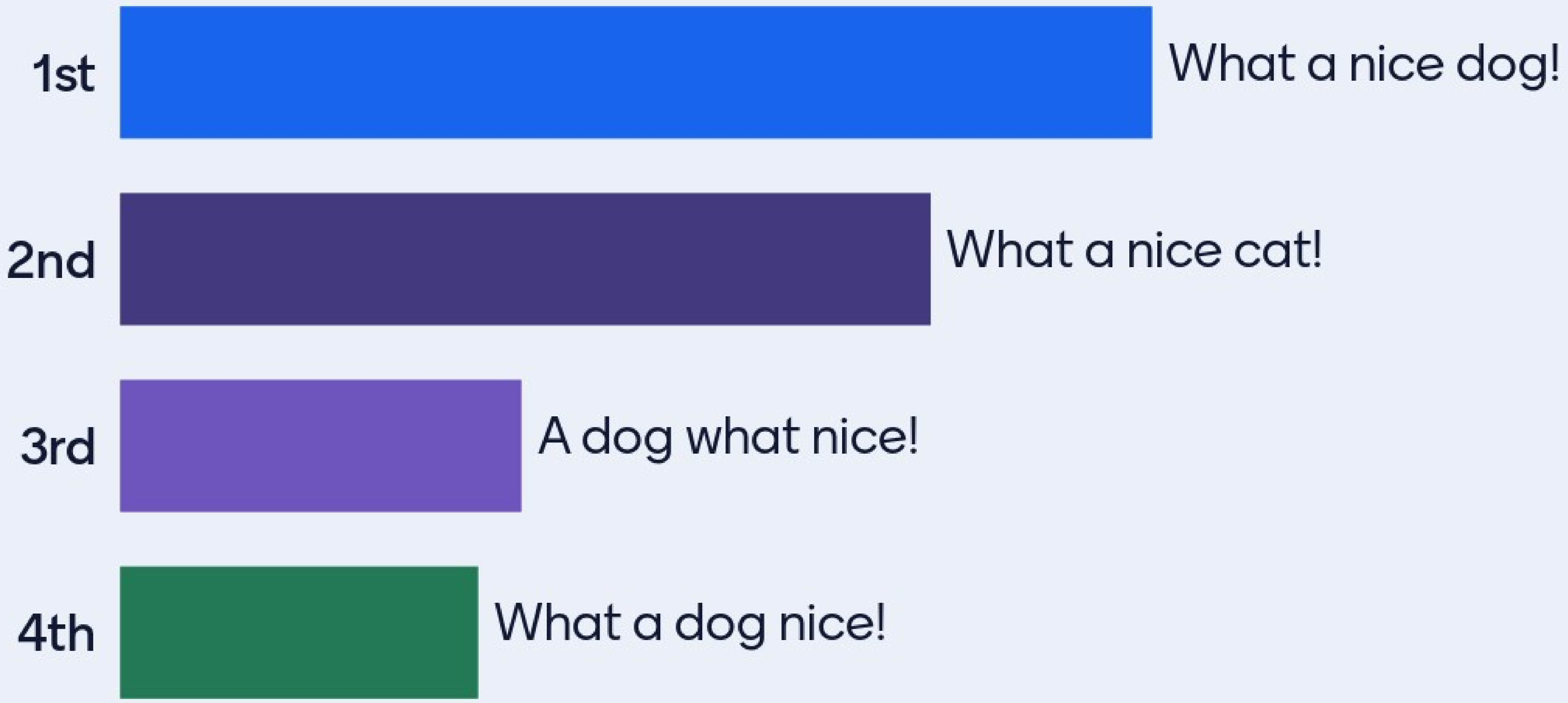
explicative and generative formalisation of natural language

A model that simulates language

A model that is able to parse language in a structured way

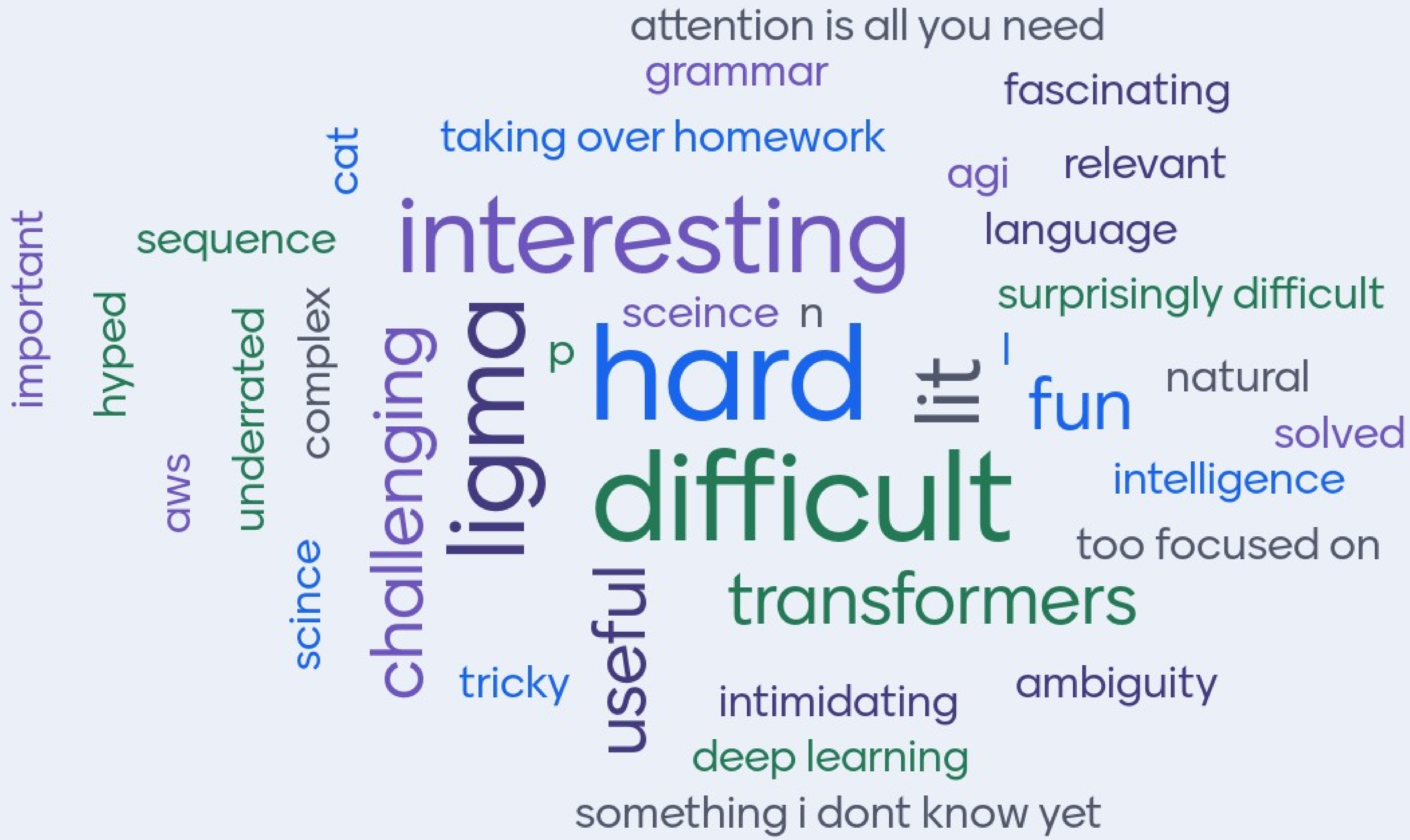
3 👍 23 👤

Order these sentences from most to least plausible



Natural language processing is ___

61 responses



attention is all you need
grammar
fascinating
taking over homework
cat
agi
relevant
important
sequence
interesting
language
hyped
underrated
complex
challenging
ligma
sceince
n
surprisingly difficult
p
hard
lit
l
fun
natural
aws
solved
intelligence
difficult
too focused on
scince
useful
transformers
tricky
intimidating
ambiguity
deep learning
something i dont know yet

8

41

Those are some of the things an LM can do

Language Model

A language model is a *probability distribution* over the set of all strings in a language.

Being a probability distribution means that 1) an LM can assign probability to text, and 2) you can generate text by drawing samples from the LM.

Applications

- → Order alternative sentences (e.g., in speech recognition)
- → Generate text in context (e.g., autocomplete)
- → Backbone of various NLP systems: translation, summarisation, chatbots

Goals for this class: to learn

→ what LMs *are*

→ how to *prescribe* LMs

→ how to *estimate* LMs

→ how to *evaluate* LMs

A good LM is

a probability distribution whose samples *resemble* observed text.

Examples: if the typical sentence has 30 words, sampling from a good LM will reproduce that pattern; if the typical sentence has SUBJ VERB OBJ (in this order), samples from a good LM will exhibit that pattern too.

What a nice dog!

30 responses

1%

90%

0.00000000001

0.00000001

We don't know the dataset, so we can't estimate the probability

100%

0.0004

.00000000000001

it depends

2

27

What a nice dog!

30 responses

100 in this context

100%

100

0

1e-4

100%

99

0.000000000000001

0.95

👍 2

👤 27

What a nice dog!

30 responses

We can't determine this given the information. We don't know the vocab size.

Don't know the language?

depends on the embeddings

We don't know the query

P(what)P(a)p(nice)p(dog)

a measure?

0.69

no context

we do not know. not that small since what and a come more often.

2

27

What a nice dog!

30 responses

P(what)*P(a)*P(nice)...

Dogs are pets, so many people have dogs.the choice of "nice" instead of "good" makes it unlikelyWe do not know the context in which this sentence is saidGrammatically it's correct making it likely

1/((1/26)^15)

2

27

Think about it...

A text may be more or less plausible, more or less frequent, more or less expected depending on where we see it (e.g., news, fiction, classroom), who produces it, who it's meant for, etc.

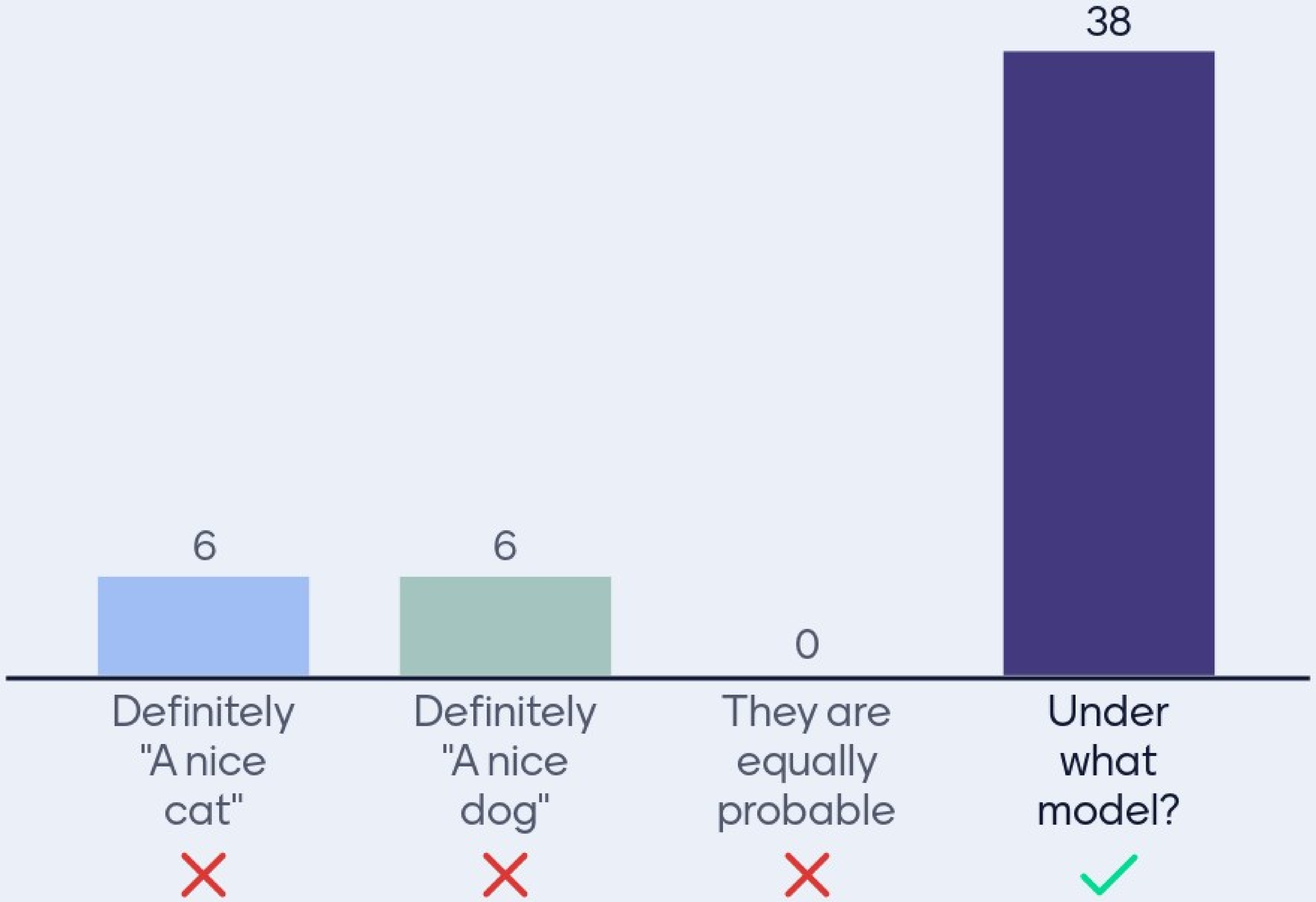
There's no unique notion of THE probability of a piece of text.

Fun fact. De Finetti's seminal *Theory of Probability* starts with a provocative claim: PROBABILITY DOES NOT EXIST. Not as an inherent property of events, rather, probability exists as a construct within models.

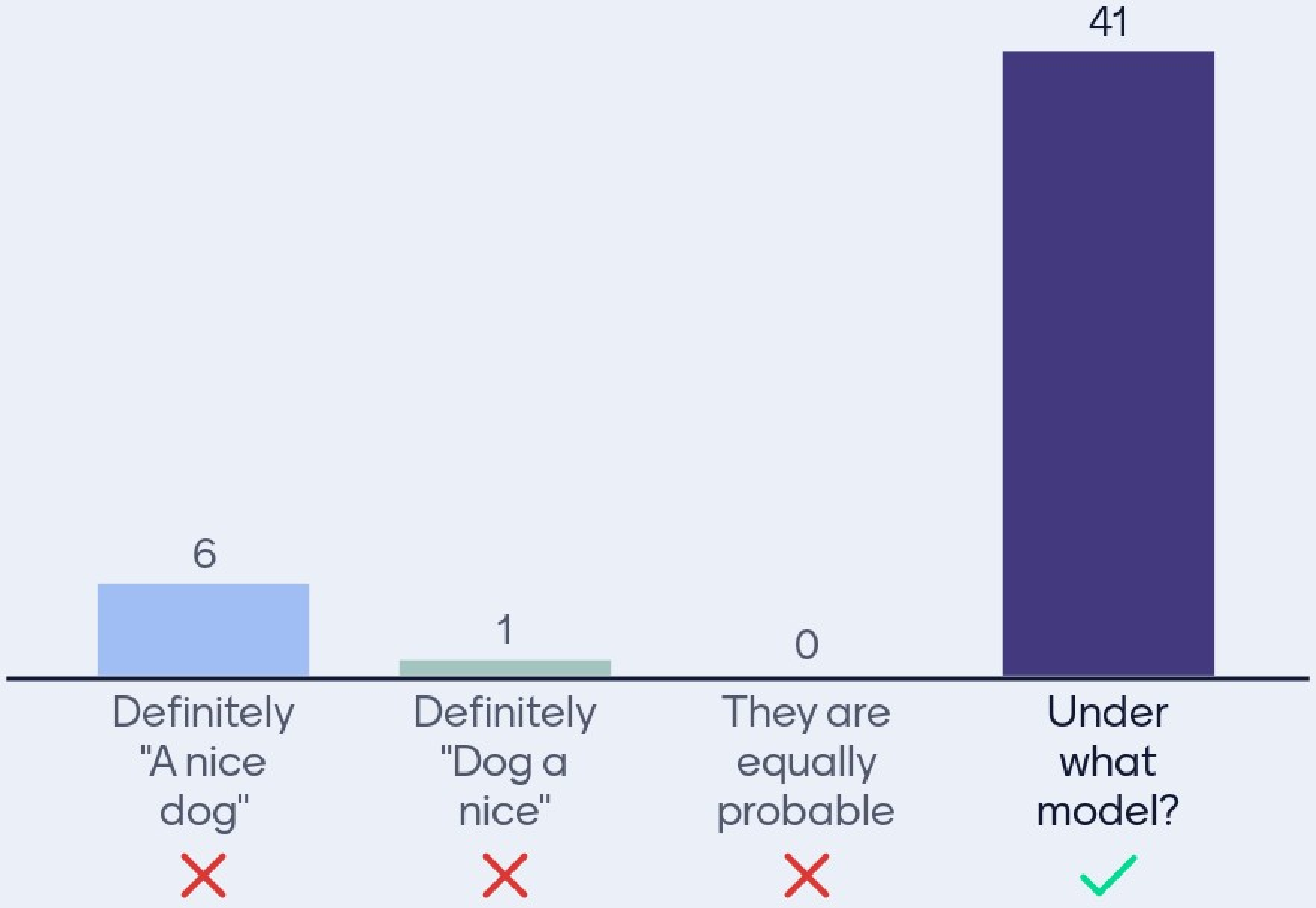
Text does not have probability as an inherent property

As probabilities aren't observable, we cannot really learn LMs by regressing from text to target probabilities, instead we learn to assign probability to observed text by designing a statistical parametric model of how texts come about, and optimise parameters using observed data and a statistical criterion.

What is more probable? "A nice cat" or "A nice dog".



What is more probable? "A nice dog" or "Dog a nice".



Prescribing LMs

Sample space. To prescribe a probability distribution, we start by specifying the set of outcomes that the LM can generate and assign probability to.

Probability mass function (pmf). We then prescribe a mechanism to assign probability to each and every outcome in the sample space.

Text as a Data Type

Text is a linguistic structure which we view as a finite sequence.

We regard any one piece of text as a finite-length sequence of symbols (or tokens), each coming from the same **countably finite** set of known symbols (i.e., the vocabulary). Example: (what, a, nice, dog, !).

Digital text is a stream of characters, hence we need a tokenisation algorithm.

Sample space

The sample space of a language model is the set of all finite-length sequences made of symbols from a finite vocabulary of symbols $\mathcal{W}$. We call this set $\mathcal{W}^*$.

Example: if $\mathcal{W}$ is a good approximation to the vocabulary of English words, then $\mathcal{W}^*$ will include most valid pieces of English-written text (e.g., "the cat is going around"). It also includes many sequences that aren't valid English (e.g., "cat the going around is").

Formalisation

W is a random *word*. An outcome w is a symbol in a vocabulary $\mathcal{W}$ of size V.

$X = \langle W_1, \ldots, W_L \rangle$ is a random *sequence* of L words. We can also denote it $W_{1:L}$. An outcome $\langle w_1, \ldots, w_l \rangle$ is a sequence of l symbols from $\mathcal{W}$.

A language model is a mechanism to assign probability $P_X(\langle w_1, \ldots, w_l \rangle)$ to **any** outcome $w_{1:l} \in \mathcal{W}^*$.

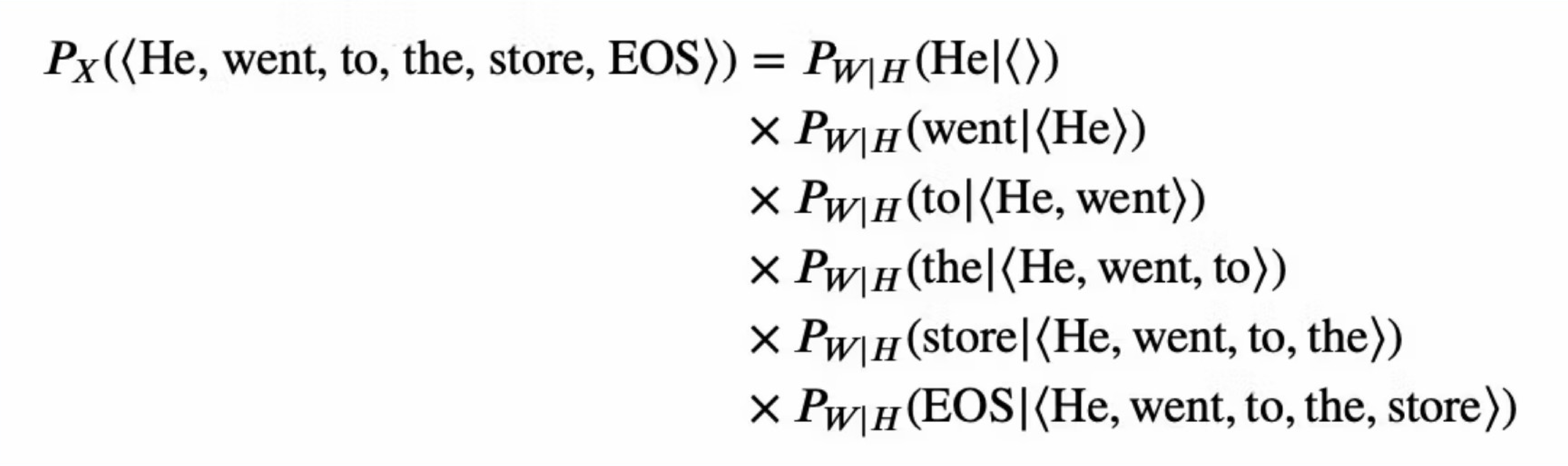
Challenge

P_X is a distribution over a countably infinite space of variable-length sequences.

There is no standard statistical distribution over such a sample space. Hence, we need to develop one.

Key Idea

Re-express the probability of a sequence (an outcome from a countably **infinite** space) using the probabilities of the "steps" needed to generate it. Each "step" being an outcome from a countably **finite** space.



$$
\begin{aligned}
P_X(\langle \text{He, went, to, the, store, EOS}\rangle) = \; & P_{W|H}(\text{He}|\langle\rangle) \\
& \times P_{W|H}(\text{went}|\langle \text{He}\rangle) \\
& \times P_{W|H}(\text{to}|\langle \text{He, went}\rangle) \\
& \times P_{W|H}(\text{the}|\langle \text{He, went, to}\rangle) \\
& \times P_{W|H}(\text{store}|\langle \text{He, went, to, the}\rangle) \\
& \times P_{W|H}(\text{EOS}|\langle \text{He, went, to, the, store}\rangle)
\end{aligned}
$$

Example. W is a random word, H is a random *history* of words, EOS is a special word marking the end of a piece of text.

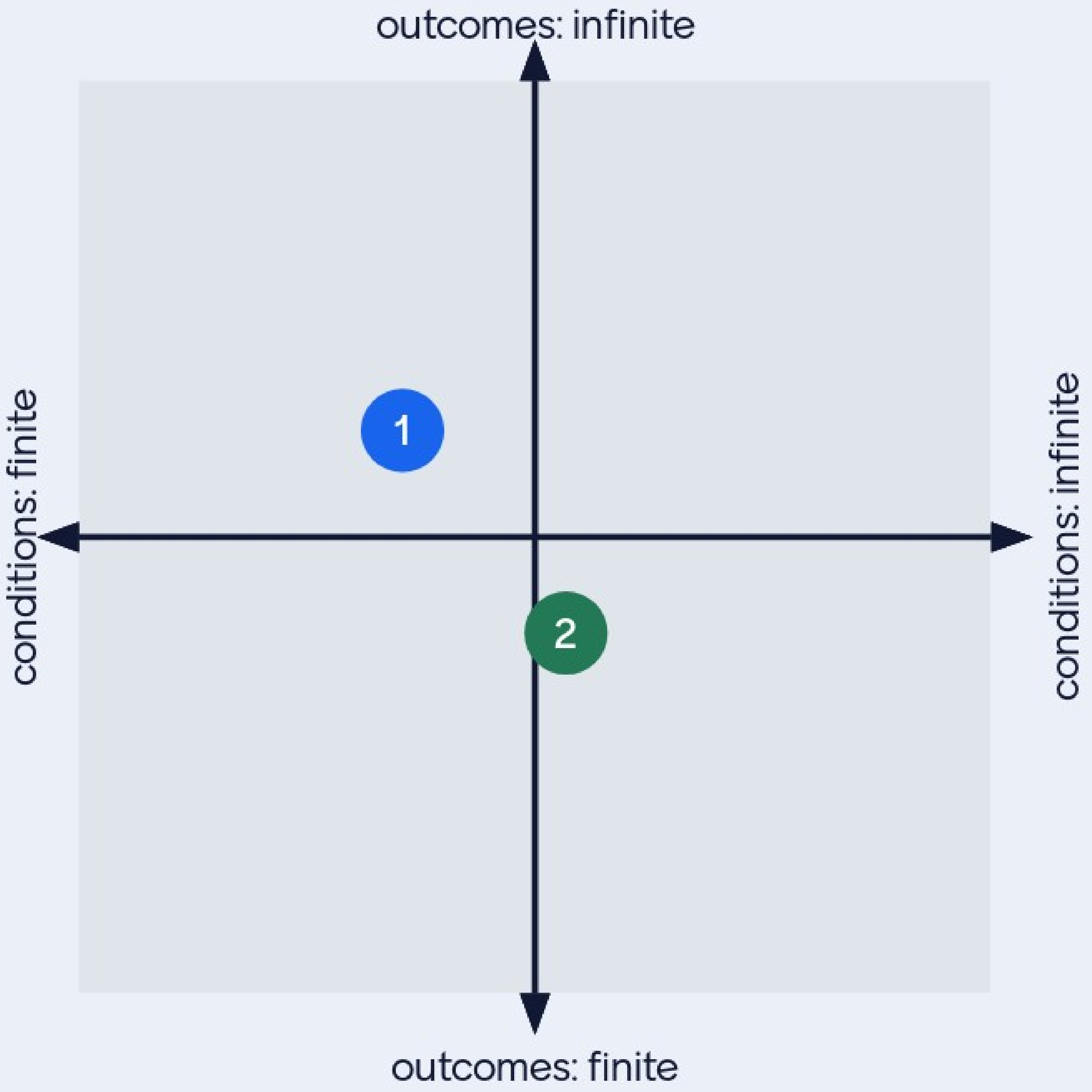
Chain rule

We re-express the probability our LM assigns to $w_{1:l}$ by applying chain rule:

$$P_X(w_{1:l}) = \prod_{i=1}^{l} P_{W|H}(w_i | w_{<i})$$

Here H is a random *history* of previous tokens. An outcome $w_{<i} = \langle w_1, \ldots, w_{i-1} \rangle$ is a sequence of $i - 1$ symbols from $\mathcal{W}$. When $i \leq 1$, $w_{<i}$ is the empty sequence.

What chain rule achieves



1 P(X)

2 P(W|H)

History-conditioned word distributions

For any given history h (e.g., "what a nice"), we need to be able to assign probability $P_{W|H}(w|h)$ to any symbol w in the vocabulary.

That is, we need to assign probability to $P_{W|H}(\mathrm{dog}|h)$, $P_{W|H}(\mathrm{thing}|h)$, $P_{W|H}(\mathrm{what}|h)$, etc., for each and every single one of the V words we know.

Categorical distribution

$$O \sim \text{Categorical}(0.1, 0.2, 0.7)$$

Pronounced: O is distributed as a 3-way Categorical random variable with parameter vector (0.1, 0.2, 0.7), whose coordinates prescribe the probabilities of the outcomes.

Categorical distribution

We may store the Categorical parameter in a variable, like $\pi_{1:3} = (0.1, 0.2, 0.7)$.

Then we write: $O \sim \text{Categorical}(\pi_{1:3})$.

The pmf of this model is $p(k; \boldsymbol{\pi}) = \pi_k$

(For outcomes that are symbols, like "dog" we first create a one-to-one correspondence between words and numerical identifiers)

Categorical distribution

Sometimes, our parameter is specific to a context c, and we store it in a variable like $\pi_{1:3}^{(c)} = (0.2, 0.3, 0.5)$. In a programming language like python you can represent $\pi_{1:3}^{(c)}$ as something like:
pi = dict(); pi[c]=(0.2, 0.3, 0.5)

Then we write: $O|C = c \sim \text{Categorical}(\pi_{1:3}^{(c)})$.
Pronounced: **given** $C = c$, the distribution of O is Categorical with parameter $\pi_{1:3}^{(c)}$.

The pmf of this model is $p(k|c; \boldsymbol{\pi}) = \pi_k^{(c)}$.

Assigning Probability or Generating Outcomes?

Our procedure to assign probability to an outcome also prescribes a *sampler* (or *simulator*), that is, an algorithm to *generate* outcomes from the LM distribution P_X.

This procedure is also known as a *generative story*.

Generative story

1. Start with an empty history $H = \langle \rangle$. Set $i = 1$.

2. Condition on the available history $w_{<i}$ and draw a word w_i with probability $P_{W|H}(w_i|w_{<i})$ extending the history with it.

3. If w_i is a special end-of-sequence symbol (EOS), terminate, else increment i and repeat (2).

This specifies a **factorisation** of P_X in terms of elementary factors of the kind $P_{W|H}$.

Summary

- An LM is a distribution P_X over text
- Rather than working with P_X directly, we re-express it via chain rule
- For any given history h, an LM predicts a distribution $P_{W|H=h}$ over the vocabulary
- The vocabulary is finite, so $P_{W|H=h}$ is a tractable V-dimensional object.
- But there's no limit to the set of possible histories

Outline

- → Language model
- → **NGram LM**
- → Tabular parameterisation
- → Evaluation

How can we deal with or circumvent the need to deal with a growing history?

12 responses

Markov

increase the probability of EOS as the size of the history grows?

sliding window

only track the last n words

context window

iid

Markov models

NB assumption

We need to identify how long the time dependencies need to be, not all words in the history will be equally useful.

How can we deal with or circumvent the need to deal with a growing history?

12 responses

local history

Fixed size of tokens per sentence

divide text in chunks

12

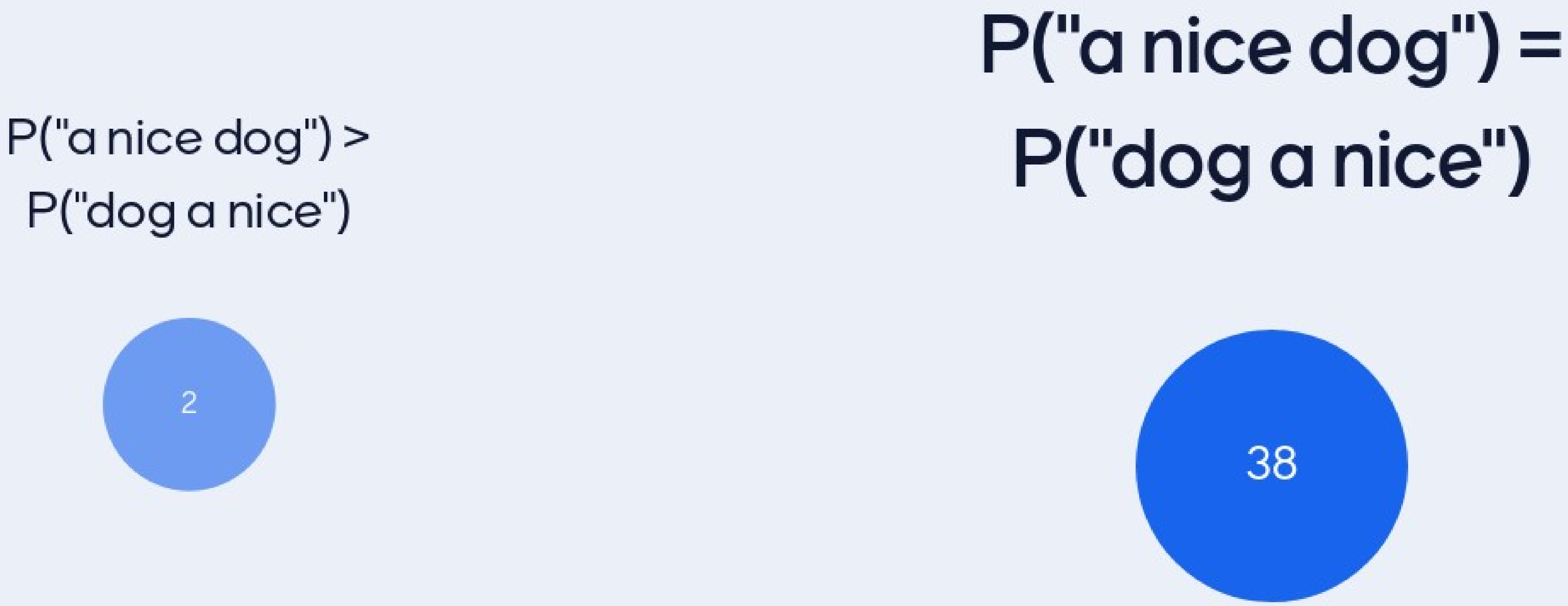
Unigram LM

We could say words are independent of one another.

$$P_X(\langle \text{He}, \text{went}, \text{to}, \text{the}, \text{store}, \text{EOS} \rangle) \overset{\text{ind.}}{=}$$
$$\begin{aligned} & P_W(\text{He}) \\ & \times P_W(\text{went}) \\ & \times P_W(\text{to}) \\ & \times P_W(\text{the}) \\ & \times P_W(\text{store}) \\ & \times P_W(\text{EOS}) \end{aligned}$$

Though, is it realistic?

Which statement is true under a unigram LM?



P("a nice dog") >
P("dog a nice")

P("a nice dog") =
P("dog a nice")

7

40

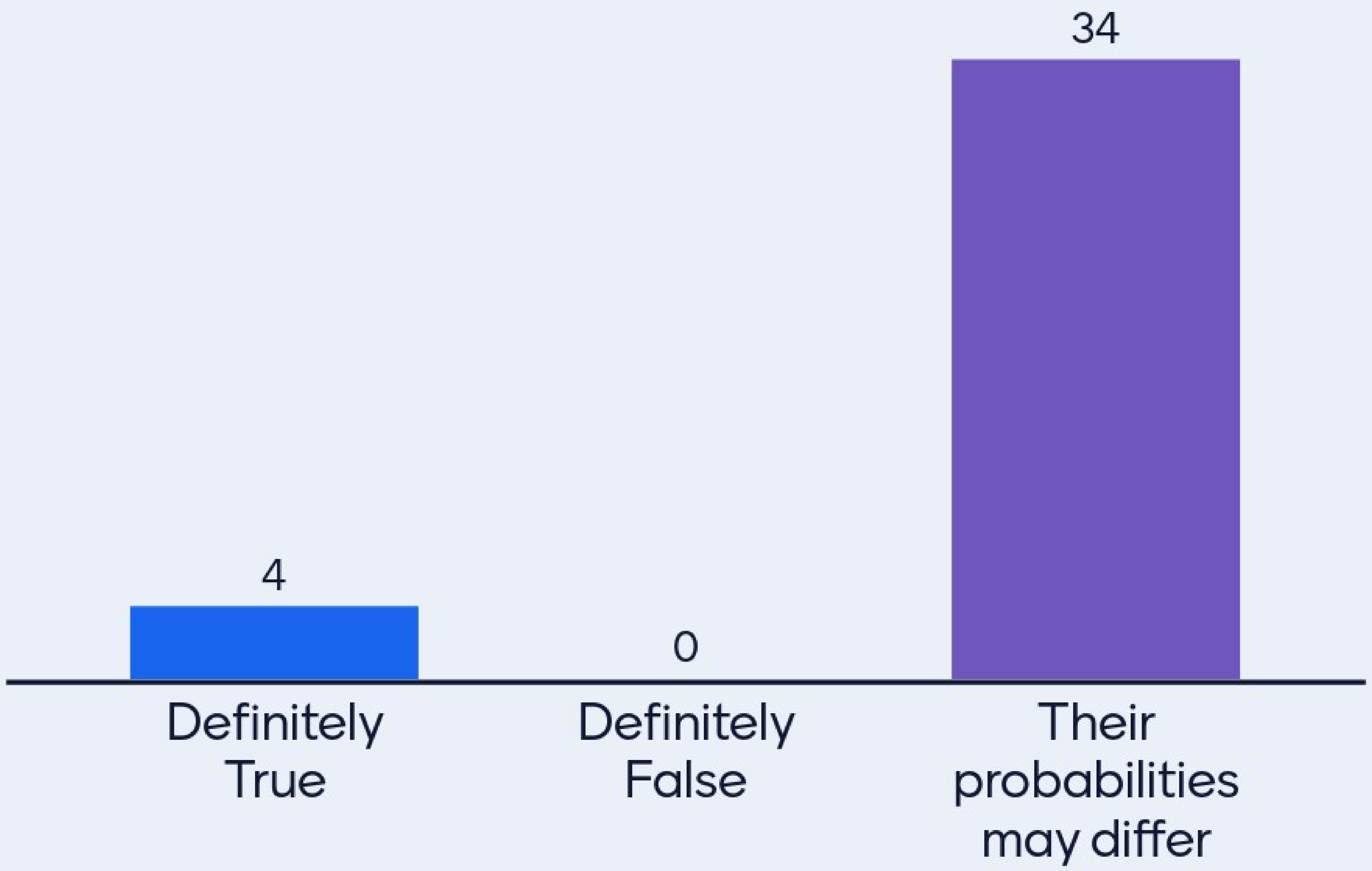
Bigram LM

Maybe we say a word is independent of every other word except the previous one?

$$
\begin{aligned}
&P_X(\langle \text{He}, \text{went}, \text{to}, \text{the}, \text{store}, \text{EOS}\rangle) \overset{\text{ind.}}{=} \\
&P_{W|H}(\text{He}|\langle \text{BOS}\rangle) \\
&\times P_{W|H}(\text{went}|\langle \text{He}\rangle) \\
&\times P_{W|H}(\text{to}|\langle \text{went}\rangle) \\
&\times P_{W|H}(\text{the}|\langle \text{to}\rangle) \\
&\times P_{W|H}(\text{store}|\langle \text{the}\rangle) \\
&\times P_{W|H}(\text{EOS}|\langle \text{store}\rangle)
\end{aligned}
$$

5

Under the bigram LM, "a nice dog" is more probable than "dog a nice".



4

38

Trigram LM

Maybe it should depend on the previous 2 words?

$$
\begin{aligned}
&P_X(\langle \text{He, went, to, the, store, EOS} \rangle) \overset{\text{ind.}}{=} \\
&P_{W|H}(\text{He} | \langle \text{BOS}, \text{BOS} \rangle) \\
&\times P_{W|H}(\text{went} | \langle \text{BOS}, \text{He} \rangle) \\
&\times P_{W|H}(\text{to} | \langle \text{He}, \text{went} \rangle) \\
&\times P_{W|H}(\text{the} | \langle \text{went}, \text{to} \rangle) \\
&\times P_{W|H}(\text{store} | \langle \text{to}, \text{the} \rangle) \\
&\times P_{W|H}(\text{EOS} | \langle \text{the}, \text{store} \rangle)
\end{aligned}
$$

Markov assumption

The key idea in the NGram LM is to make a conditional independence assumption:

$$P_X(w_{1:l}) \overset{\text{ind.}}{=} \prod_{i=1}^{l} P_{W|H}(w_i | \langle w_{i-n+1}, \ldots, w_{i-1} \rangle)$$

a word is independent on all but the recent history of $n-1$ words. This is the so-called **Markov assumption of order** $n-1$.

Outline

- → Language model
- → NGram LM
- → **Tabular parameterisation**
- → Evaluation

Tabular CPDs

Each history-conditioned word distribution is Categorical:

$$W|H = h \sim \text{Categorical}(\pi_{1:V}^{(h)})$$

Where $\pi_{1:V}^{(h)}$ is V-dimensional, $0 \leq \pi_k^{(h)} \leq 1$ for any k, and $\sum_{k=1}^{V} \pi_k^{(h)} = 1$.

We store the parameters $\pi_{1:V}^{(h)}$, for each and every h we know, in a table.

Probability mass function (pmf)

The NGram LM is built upon history-conditioned word distributions of the form

$$W|H = h \sim \mathrm{Categorical}(\pi_{1:V}^{(h)})$$

where a history has size $n-1$.

The pmf of the model is

$$p(w_{1:l}; \boldsymbol{\pi}) = \prod_{i=1}^{l} \pi_{w_i}^{(h_i)}$$

where $h_i = \langle w_{i-n+1}, \ldots, w_{i-1} \rangle$ is the $n-1$ words before w_i.

Parameter Estimation

For each unique history h, we have a cpd:

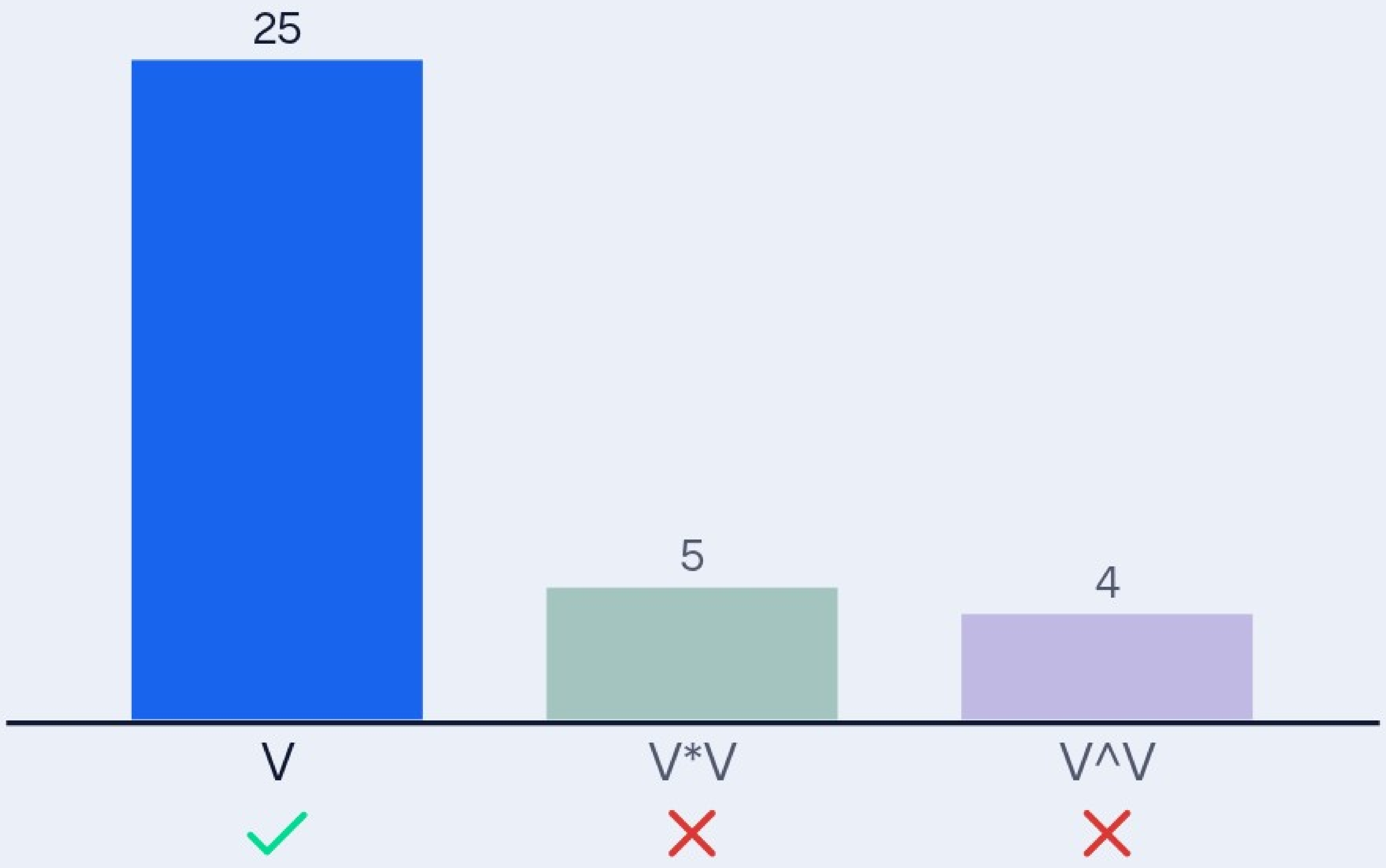
$$W|H = h \sim \text{Categorical}(\pi_{1:V}^{(h)})$$

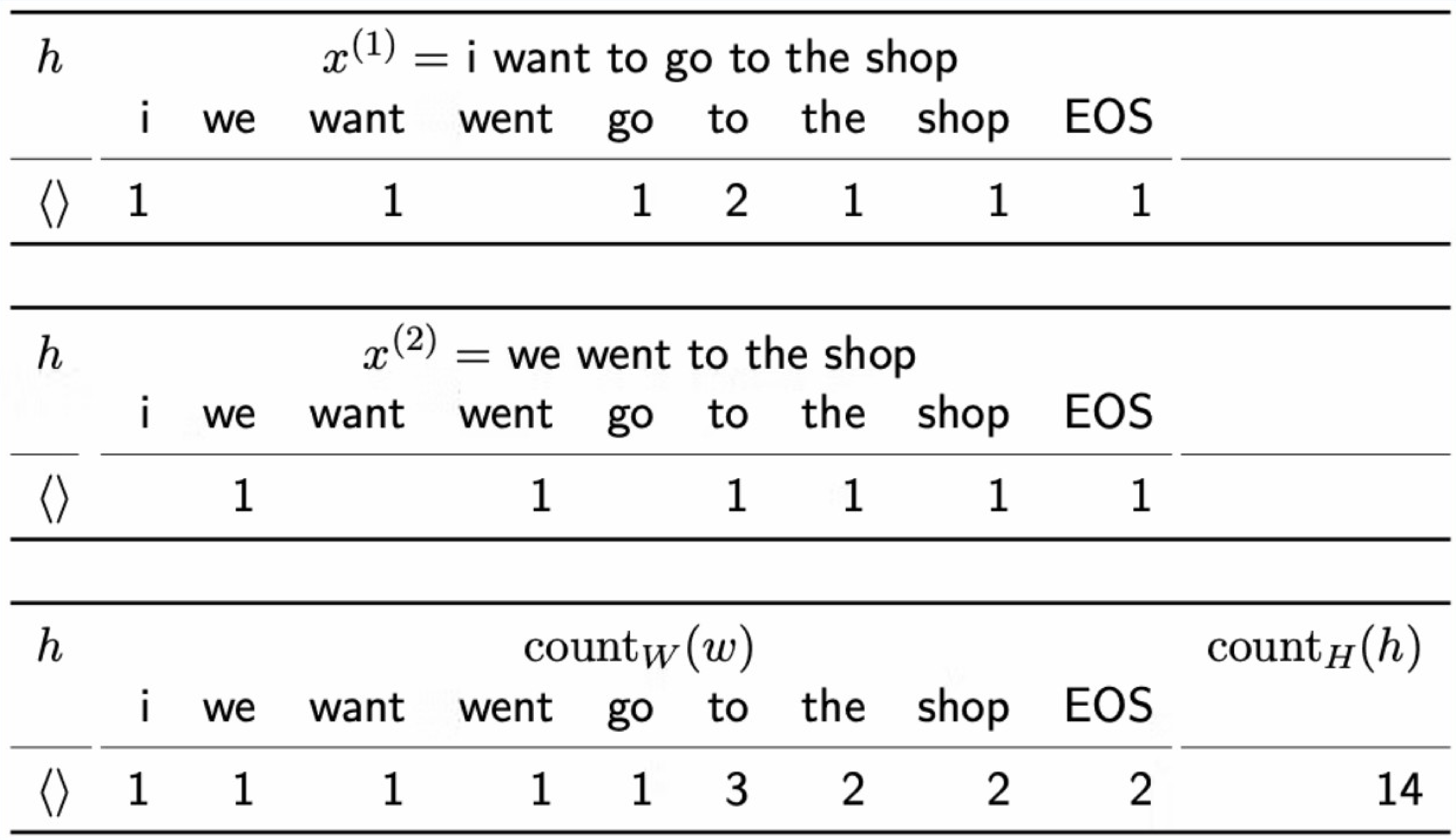
Given a dataset of observed texts, the maximum likelihood estimate of the conditional probability $\pi_w^{(h)}$ of w given h is:

$$\pi_w^{(h)} = \frac{\text{count}_{HW}(h, w)}{\sum_{o=1}^{V} \text{count}_{HW}(h, o)} = \frac{\text{count}_{HW}(h, w)}{\text{count}_{H}(h)}$$

Hint. Conceptually we store all parameters in tables, in practice, we use sparse data structures and store only the non-zero parameters.

Unigram LM: size of tabular representation of P(W|H)





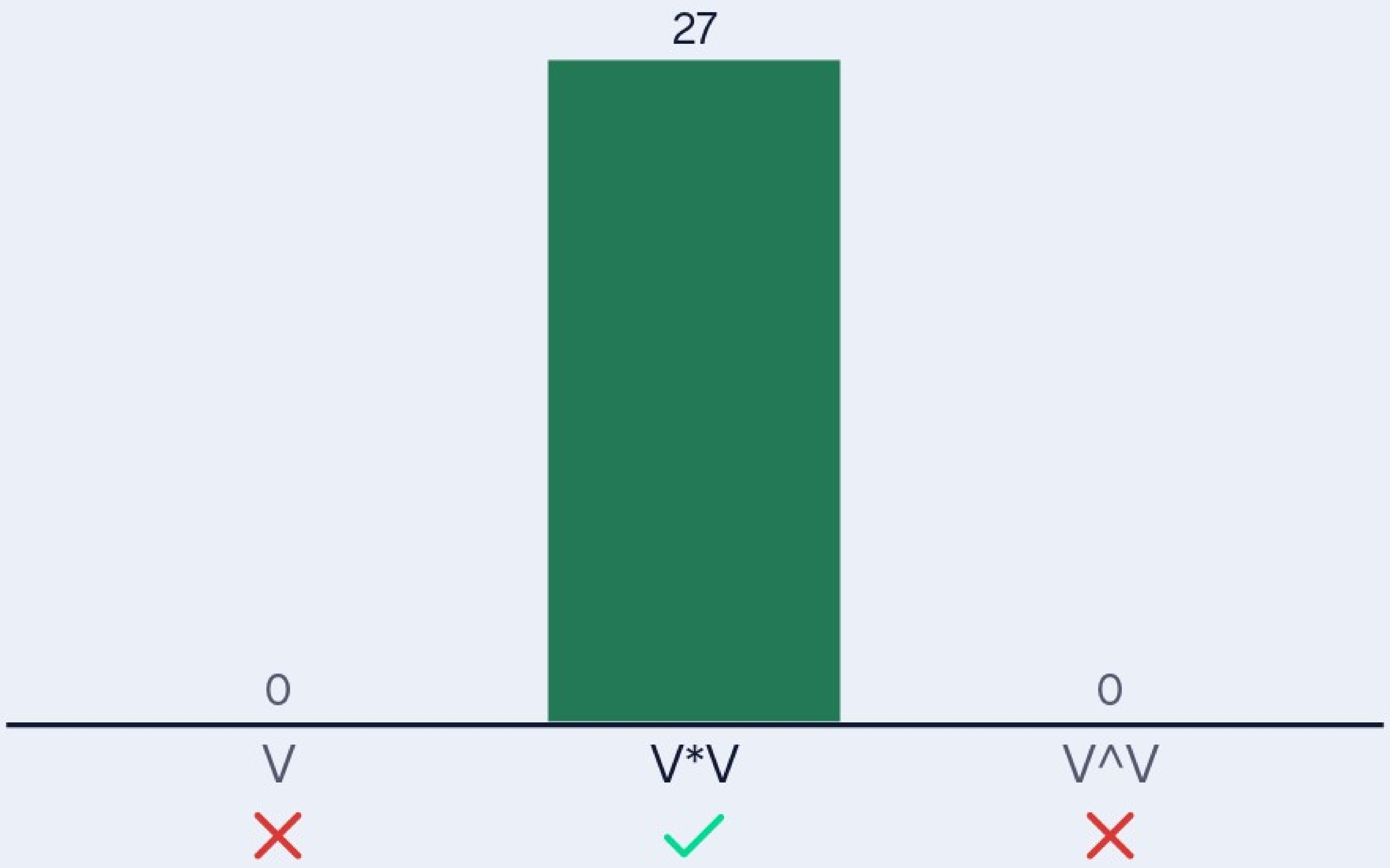
h	$x^{(1)} =$ i want to go to the shop								
	i	we	want	went	go	to	the	shop	EOS
$\langle\rangle$	1		1		1	2	1	1	1

h	$x^{(2)} =$ we went to the shop								
	i	we	want	went	go	to	the	shop	EOS
$\langle\rangle$		1		1		1	1	1	1

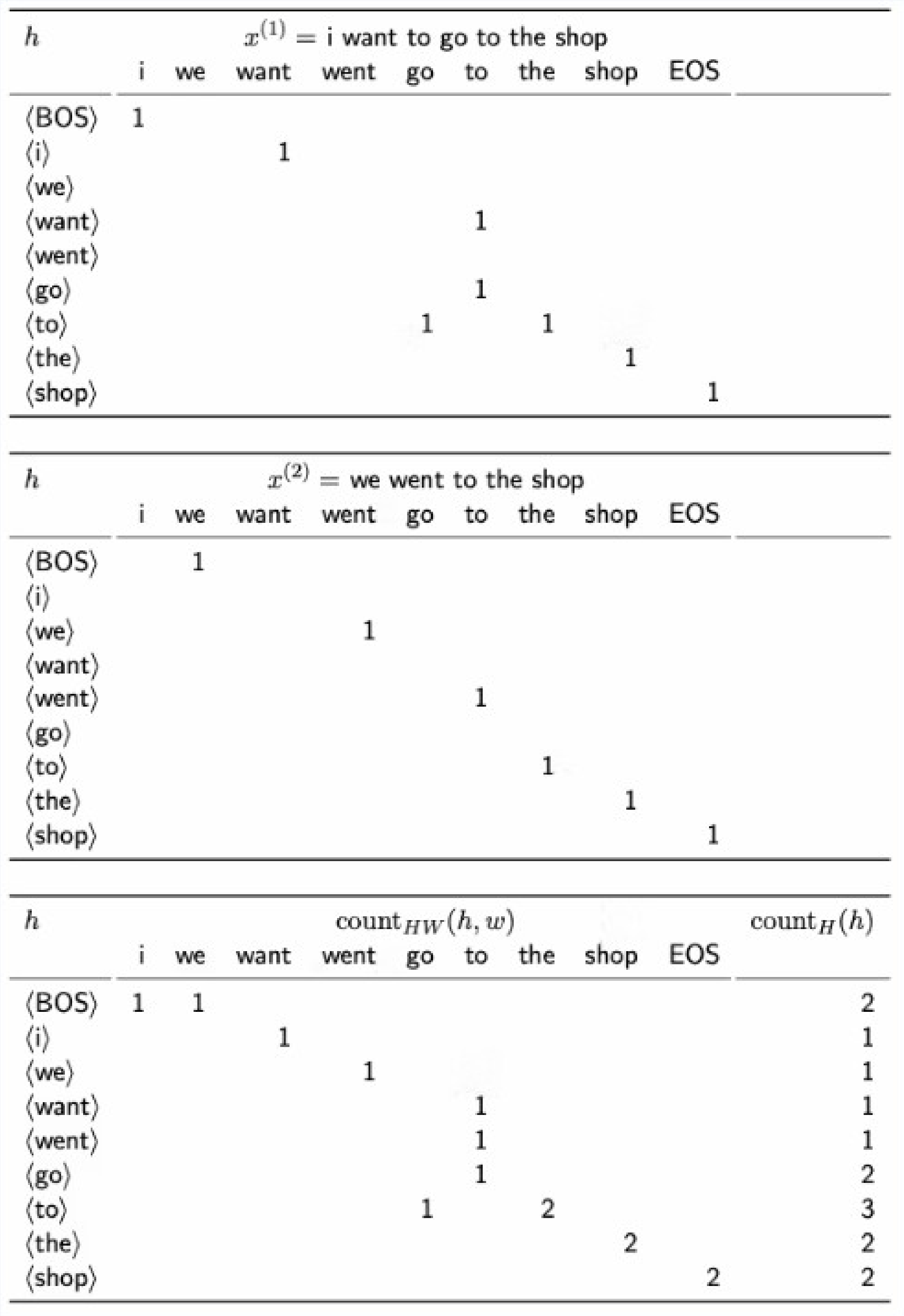
h	$\mathrm{count}_W(w)$									$\mathrm{count}_H(h)$
	i	we	want	went	go	to	the	shop	EOS	
$\langle\rangle$	1	1	1	1	1	3	2	2	2	14

Unigrams from a toy dataset

Bigram LM: size of tabular representation of P(W|H)



27



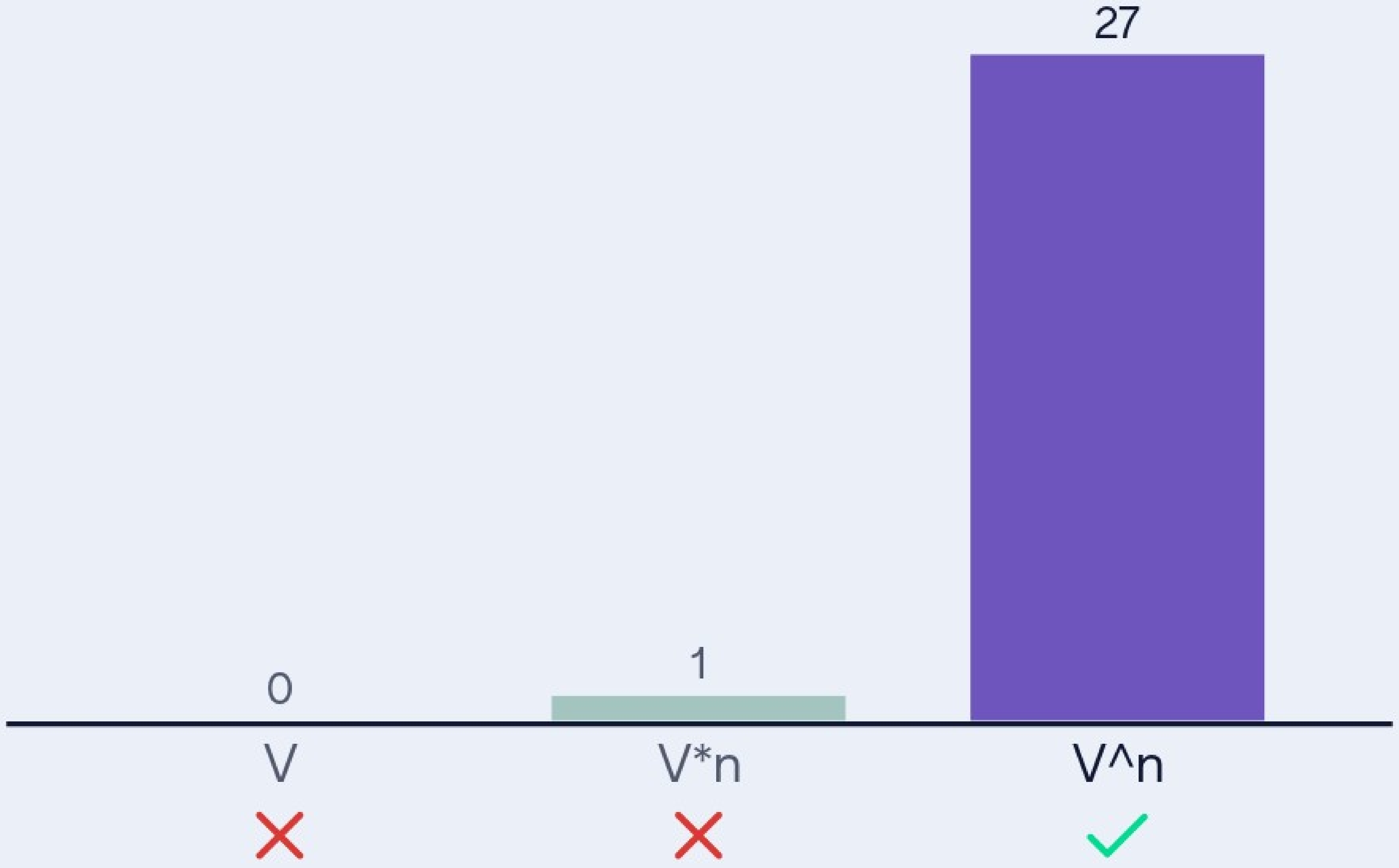
h	$x^{(1)} =$ i want to go to the shop								
	i	we	want	went	go	to	the	shop	EOS
⟨BOS⟩	1								
⟨i⟩			1						
⟨we⟩									
⟨want⟩						1			
⟨went⟩									
⟨go⟩						1			
⟨to⟩					1		1		
⟨the⟩								1	
⟨shop⟩									1

h	$x^{(2)} =$ we went to the shop								
	i	we	want	went	go	to	the	shop	EOS
⟨BOS⟩		1							
⟨i⟩									
⟨we⟩				1					
⟨want⟩									
⟨went⟩						1			
⟨go⟩									
⟨to⟩							1		
⟨the⟩								1	
⟨shop⟩									1

h	$\text{count}_{HW}(h, w)$									$\text{count}_H(h)$
	i	we	want	went	go	to	the	shop	EOS	
⟨BOS⟩	1	1								2
⟨i⟩			1							1
⟨we⟩				1						1
⟨want⟩						1				1
⟨went⟩						1				1
⟨go⟩						1				2
⟨to⟩					1		2			3
⟨the⟩								2		2
⟨shop⟩									2	2

Bigrams from a toy dataset

NGram LM: size of tabular representation of P(W|H)



Memorise phrases

The NGram LM makes up a sentence by gluing phrases, which it memorises in its tabular cpds along with their probabilities.

An increase in the order has an exponential cost: $V^n \rightarrow V^{n+1}$

Parameter Estimation

Given a dataset of observed texts, the maximum likelihood estimate of the conditional probability $\pi_w^{(h)}$ of w given h is:

$$\pi_w^{(h)} = \frac{\text{count}_{HW}(h, w)}{\text{count}_H(h)}$$

Can you foresee a problem with this?

Data sparsity

The longer the history, the less likely it is that we have seen it.

Most of the possible history-word pairs will never be seen.

Tricks: smoothing, interpolation, backoff, etc.
Optional: section 3.5 of textbook.

Smoothing

Reserve probability for unseen NGrams:

$$\begin{aligned}\pi_w^{(h)} &= \frac{\text{count}_{HW}(h,w)+\alpha(h)}{\sum_{o\in\mathcal{W}}(\text{count}_{HW}(h,o)+\alpha(h))}\\ &= \frac{\text{count}_{HW}(h,w)+\alpha(h)}{V\alpha(h)+\text{count}_H(h)}\end{aligned}$$

Often $\alpha(h)$ is a constant.

Example, $\alpha(h)=1$, $\text{count}_H(\langle\text{a, nice}\rangle)=100$, $\text{rabbit}\in\mathcal{W}$, but $\text{count}_{HW}(\langle\text{a, nice}\rangle,\text{rabbit})=0$. Then,

$$\pi_{\text{rabbit}}^{(\langle\text{a,nice}\rangle)} = \frac{0+1}{V\times 1+100}$$

Tip. When implementing smoothed models, it's easier to store *counts* (rather than parameters), because counts are sparse (many 0s) but parameters aren't.

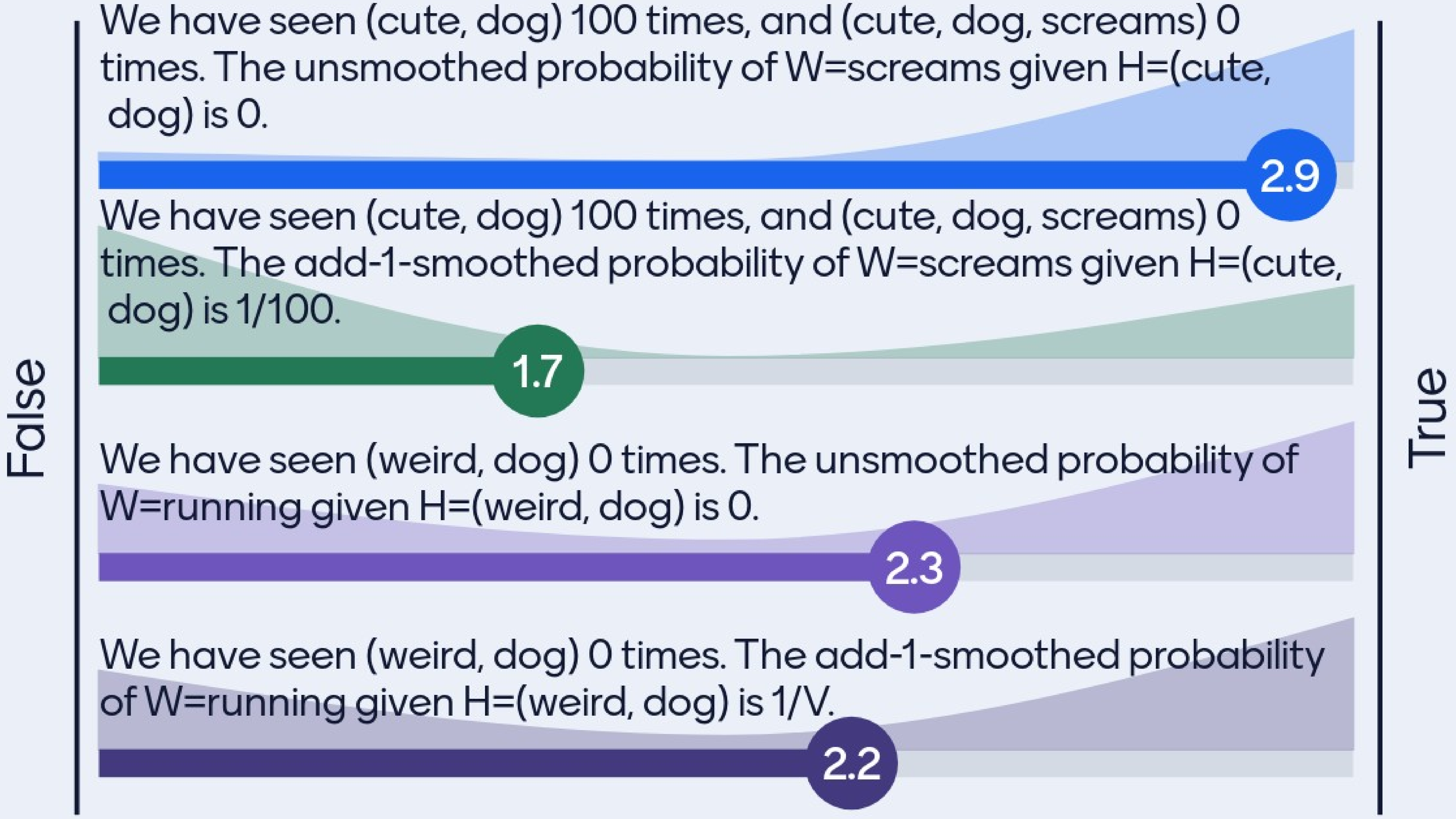
Unknown Words

Here the situation is a little different.
We want to deal with a symbol that's not at all in the vocabulary.

Idea: augment the vocabulary with a placeholder symbol such as UNK, whenever you encounter an unknown symbol in the future (e.g., "hare") treat it as UNK.

In combination with smoothing, this should help. See section 3.4.1 of textbook.

In a trigram LM estimated via MLE,



We have seen (cute, dog) 100 times, and (cute, dog, screams) 0 times. The unsmoothed probability of W=screams given H=(cute, dog) is 0.

We have seen (cute, dog) 100 times, and (cute, dog, screams) 0 times. The add-1-smoothed probability of W=screams given H=(cute, dog) is 1/100.

We have seen (weird, dog) 0 times. The unsmoothed probability of W=running given H=(weird, dog) is 0.

We have seen (weird, dog) 0 times. The add-1-smoothed probability of W=running given H=(weird, dog) is 1/V.

Data Sparsity Strikes Back

In an unsmoothed model, if $\mathrm{count}_H(h)$ is 0, we do not really have an MLE for $P_{W|H=h}$.

In a *smoothed* model, we get a uniform distribution. In some sense, uniform is better than undefined, but both options are unsatisfying.

Interpolation

Maybe we have never seen *(ridiculously, kind, cat)*, but we've seen *(kind, cat)*, or at least *(cat)*.

If we use these related observations, we might define a 4-gram LM for which $P_{W|H}(\text{jumps}|\langle\text{ridiculously, kind, cat}\rangle)$ is more useful than undefined or $1/V$.

Interpolation

We let the probability of some $W = w$ given some $H = (a, b, c)$ be a *convex* combination of the probability different models assign to it.

Example: a unigram LM assigns prob p_1 for w in isolation, a bigram LM assigns prob p_2 for w given c, a trigram LM assigns prob p_3 for w given (b, c) and a 4-gram LM assigns prob p_4 for w given (a, b, c).

We then use: $P_{W|H}(w|\langle a, b, c\rangle) = \sum_{k=1}^{4} \omega_k p_k$ with

coefficiencs $0 \leq \omega_k \leq 1$ and $\sum_{k=1}^{4} \omega_k = 1$.

Limitations

Different symbols (or sequences of symbols) are treated as unrelated, when in reality they often aren't: *cats* and *dogs* are *animals*, *dogs* is the plural form of *dog*, a *jump* and a *leap* are semantically similar, *I gave you a book* and *I gave a book to you* are syntactically different but mean the same.

Neural parameterisations of CPDs address this limitation.

Limitations

Our Markov assumptions are motivated by convenience alone, long range dependency is a very common thing in natural languages.

With a neural parameterisation of a CPD, we can design models that **do not** require Markov assumptions. They're called *autoregressive* models.

Outline

- → Language model
- → NGram LM
- → Tabular parameterisation
- → **Evaluation**

Intrinsic

We assess the surprisal (negative log probability) that our model assigns to unseen texts, but we typically re-express it in terms of *perplexity per token* (a measure of average confusion).

Required: section 3.8 of textbook.

Extrinsic

Plug the LM in a task (e.g., autocomplete)
and measure the performance in that task.

Statistical

Compare how the statistics of **generated** text distribute in relation to statistics of **observed** text.

Examples:
Meister and Cotterell, 2021
and Giulianelli et al, 2023

Be Critical

Your statistical model is as good as your statistical assumptions, your estimation procedure, and the data you use to fit it.

Most assumptions are wrong or insufficient. Any dataset (however large) is at best a snippet of language production by some groups of speakers: not good enough to represent a whole world of speakers, not good enough to represent any one specific group of speakers.

Models are not trained to comply with human values, they are not trained to produce factually correct text, they are trained such that their samples **look like** they could have been found in the training data.

What did you learn today?

Waiting for responses ...

Summary

- LMs are distributions over sentences
- Chain rule is the key to an LM
- Markov assumption (cond. indep.) leads to a viable model
- Classic approach uses inefficient tabular cpds
- Modern approaches parameterise cpds efficiently using NNs

What next?

- Check the links in the class (esp wrt evaluation)
- Next class: sequence labelling
  - self-study: logistic CPD, complete example, code