Bug Bounty has kicked off!

Thank you to everyone who is already submitting issues!!!

Report any mistake in lectures/labs/coursework/etc as an issue on GitHub:

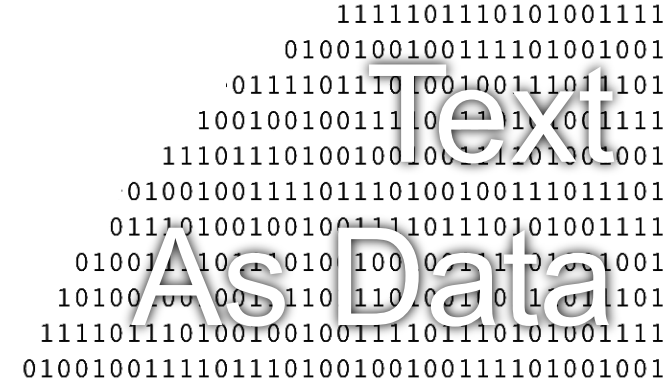
https://github.com/jakelever/textasdata_bugbounty

Prizes at end of semester for most bugs reported: 1st (£30), 2nd (£20) and 3rd (£10).

Link is also on the Moodle.

Language Modelling

Jake Lever & Sean MacAvaney



University of Glasgow | School of Computing Science

Welcome!

Today's lecture will introduce

- Motivation for Language modelling
- Probability background
- Language modelling
- Smoothing
- Links to information theory
- Text generation

Language modelling exercise

Two of the three words make sense:

[Denied the _____]

allegations

claim

sock

Science says so!

Neuroscience suggests that **humans predict** the next word when listening

In this lecture, we'll use word statistics to do the same.

Goldstein, Ariel, et al. "Shared computational principles for language processing in humans and deep language models." Nature Neuroscience 25.3 (2022): 369-380. https://www.nature.com/articles/s41593-022-01026-4

What really is **Language modelling** ?

- **Task** of building a **predictive model of language**.
- **Previous** lectures have primarily dealt with **unordered (bags) of word** representations, today we'll start on modelling sequences of words.
- **Goal:**

Predict the probability of:

1. Next word in a sequence

2. A sentence of words

What really is **Language modelling** ?

- **Task** of building a **predictive model of language**.
- **Previous** lectures have primarily dealt with **unordered (bags) of word** representations, today we'll start on modelling sequences of words.
- **Goal:**

What does this even mean?

Predict the probability of:

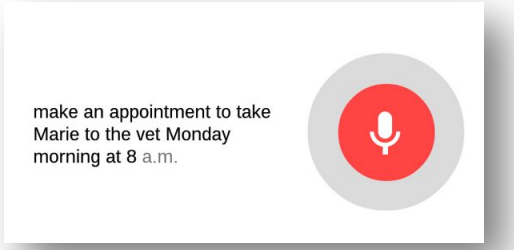
1. Next word in a sequence

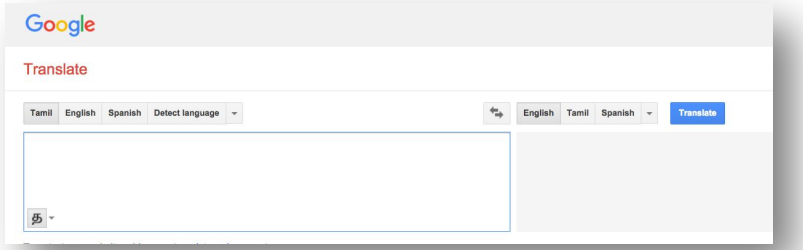
2. A sentence of words

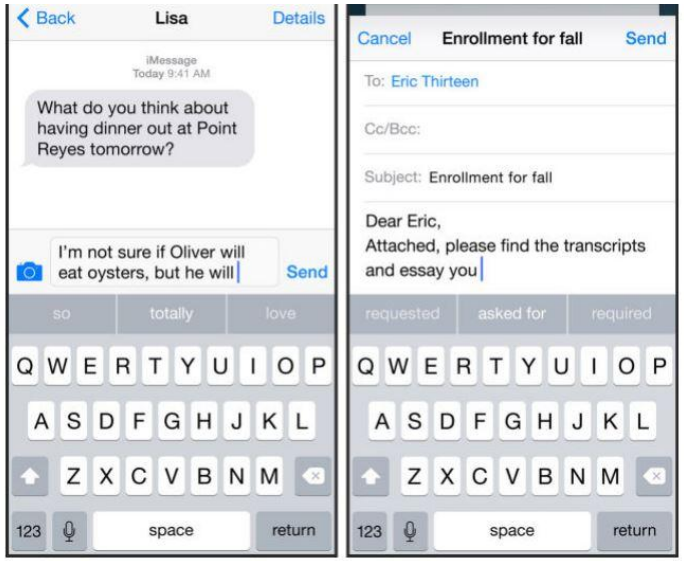
Language modelling applications

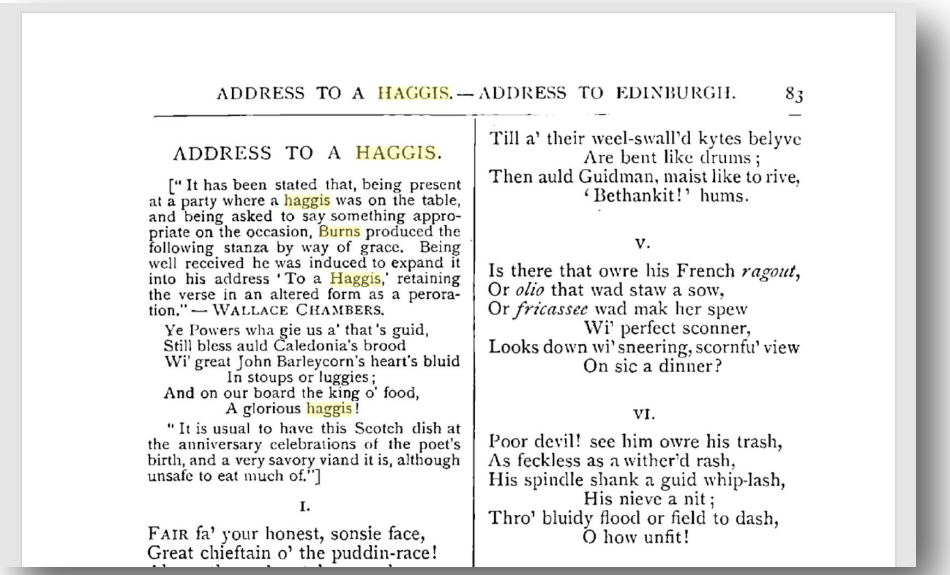
Applications:

- Speech recognition
- Optical Character Rec.
- Spelling correction
- Machine translation
- Authorship detection
- Auto-completion
- … many more!









ADDRESS TO A HAGGIS.—ADDRESS TO EDINBURGH. 83

ADDRESS TO A HAGGIS.

["It has been stated that, being present at a party where a haggis was on the table, and being asked to say something appropriate on the occasion, Burns produced the following stanza by way of grace. Being well received he was induced to expand it into his address 'To a Haggis,' retaining the verse in an altered form as a peroration." — WALLACE CHAMBERS.

Ye Powers wha gie us a' that 's guid,
Still bless auld Caledonia's brood
Wi' great John Barleycorn's heart's bluid
In stoups or luggies;
And on our board the king o' food,
A glorious haggis!

"It is usual to have this Scotch dish at the anniversary celebrations of the poet's birth, and a very savory viand it is, although unsafe to eat much of."]

I.

FAIR fa' your honest, sonsie face,
Great chieftain o' the puddin-race!

Till a' their weel-swall'd kytes belyve
Are bent like drums;
Then auld Guidman, maist like to rive,
'Bethankit!' hums.

V.

Is there that owre his French *ragout*,
Or *olio* that wad staw a sow,
Or *fricassee* wad mak her spew
Wi' perfect sconner,
Looks down wi' sneering, scornfu' view
On sic a dinner?

VI.

Poor devil! see him owre his trash,
As feckless as a wither'd rash,
His spindle shank a guid whip-lash,
His nieve a nit;
Thro' bluidy flood or field to dash,
O how unfit!

Language modelling applications

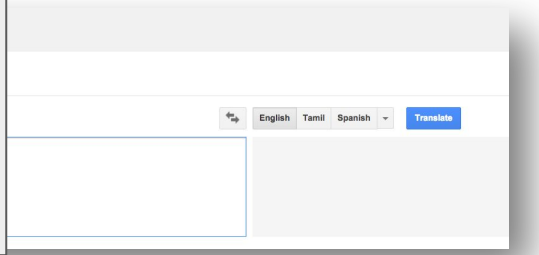
Applications:

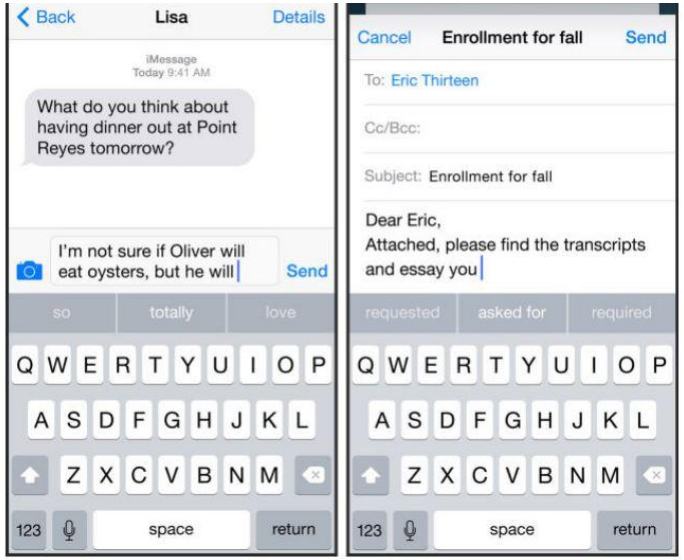
- Speech recognition
- Optical Character Rec.
- Spelling correction
- Machine translation
- Authorship detection
- Auto-completion
- … many more!

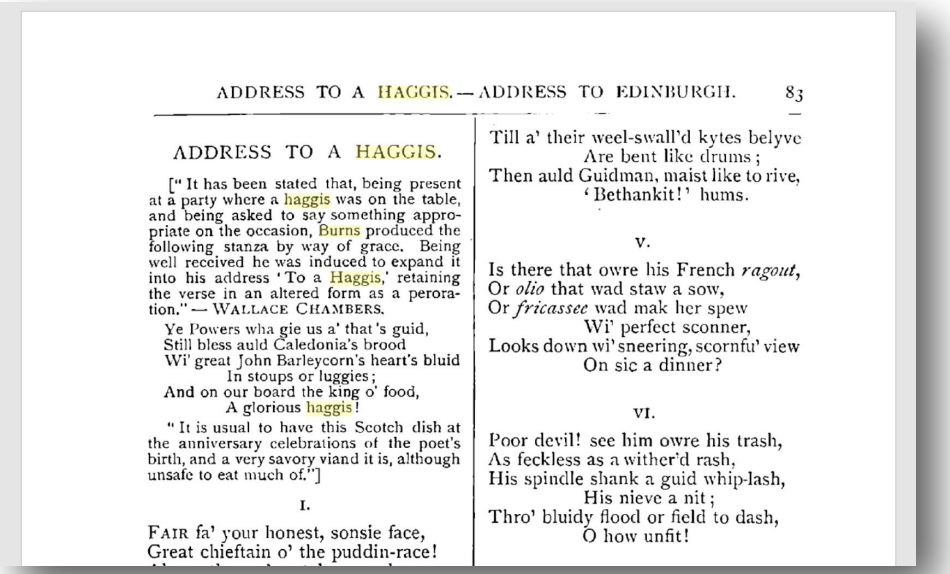
"When I bake, I usually use **butter** instead of margarine."

vs

"When I bake, I usually use **but her** instead of margarine."







ADDRESS TO A HAGGIS.—ADDRESS TO EDINBURGH. 83

ADDRESS TO A HAGGIS.

["It has been stated that, being present at a party where a haggis was on the table, and being asked to say something appropriate on the occasion, Burns produced the following stanza by way of grace. Being well received he was induced to expand it into his address 'To a Haggis,' retaining the verse in an altered form as a peroration." — WALLACE CHAMBERS.

Ye Powers wha gie us a' that 's guid,
Still bless auld Caledonia's brood
Wi' great John Barleycorn's heart's bluid
In stoups or luggies;
And on our board the king o' food,
A glorious haggis!

"It is usual to have this Scotch dish at the anniversary celebrations of the poet's birth, and a very savory viand it is, although unsafe to eat much of."]

I.

FAIR fa' your honest, sonsie face,
Great chieftain o' the puddin-race!

Till a' their weel-swall'd kytes belyve
Are bent like drums;
Then auld Guidman, maist like to rive,
'Bethankit!' hums.

V.

Is there that owre his French *ragout*,
Or *olio* that wad staw a sow,
Or *fricassee* wad mak her spew
Wi' perfect sconner,
Looks down wi' sneering, scornfu' view
On sic a dinner?

VI.

Poor devil! see him owre his trash,
As feckless as a wither'd rash,
His spindle shank a guid whip-lash,
His nieve a nit;
Thro' bluidy flood or field to dash,
O how unfit!

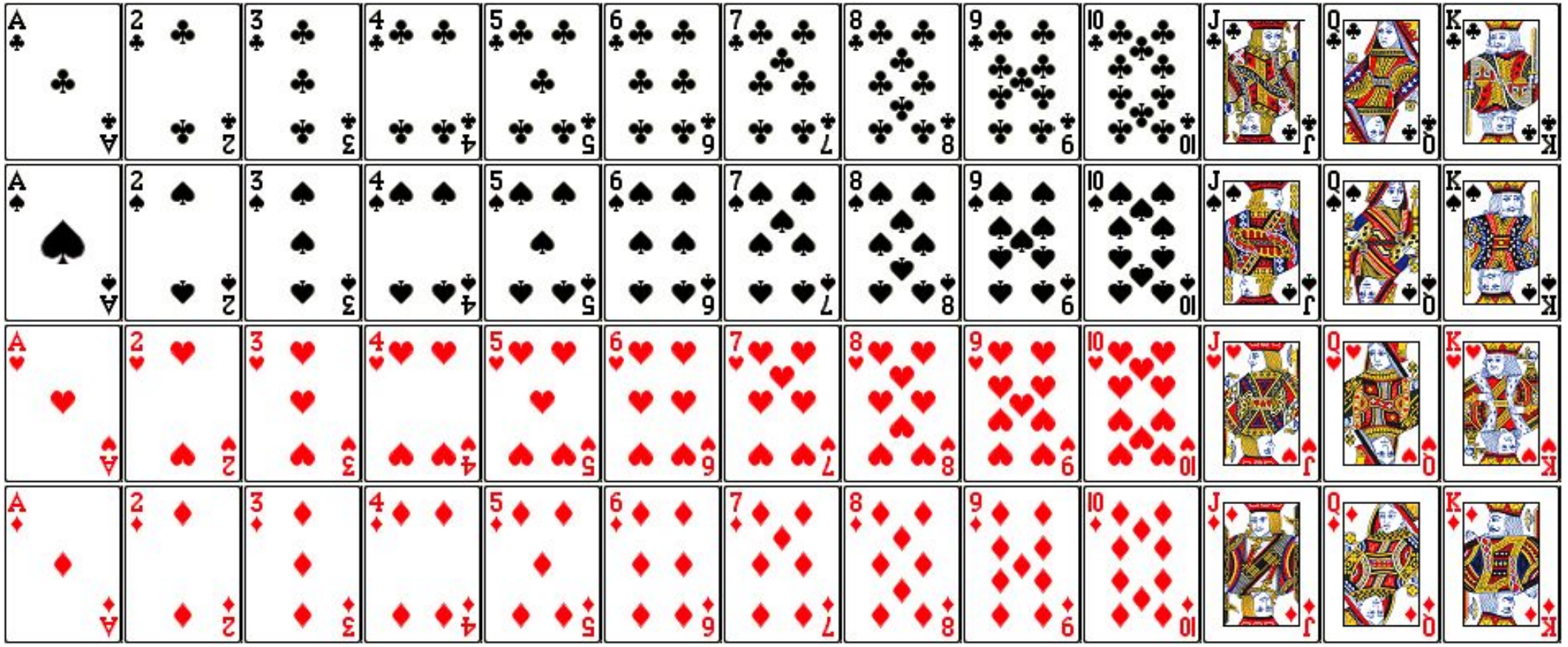
Modelling Language

- **Historically:** Using language statistics
  Simple, surprisingly effective.
  Still useful in many settings!
  We'll dive into this today.

- **Recently:** Using neural networks
  Very expensive, but highly effective.
  The technique behind large language models.
  (Covered later in this course.)

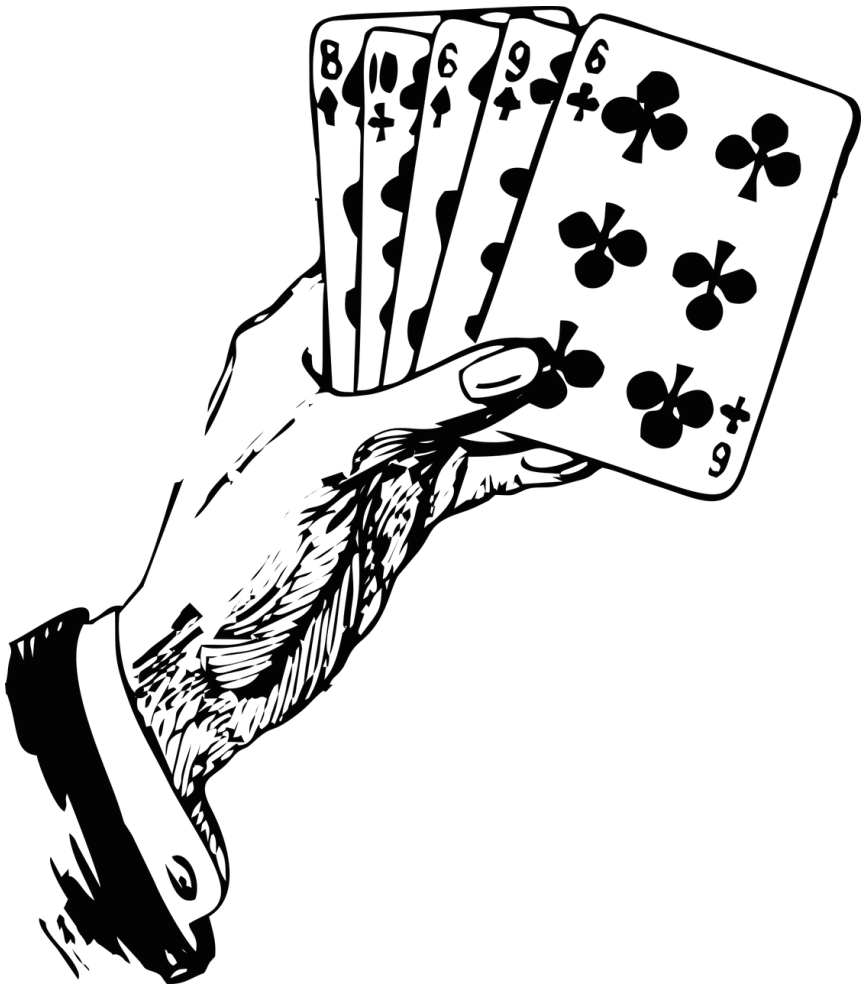
Probability Review

Let’s use a deck of cards to explore probability



4 suits, 13 cards in each suit, 52 cards total

Probabilities and Distributions: A quick revision



A probability must satisfy:

$0 <= P(x) <= 1$ and $\Sigma_x P(x) = 1$

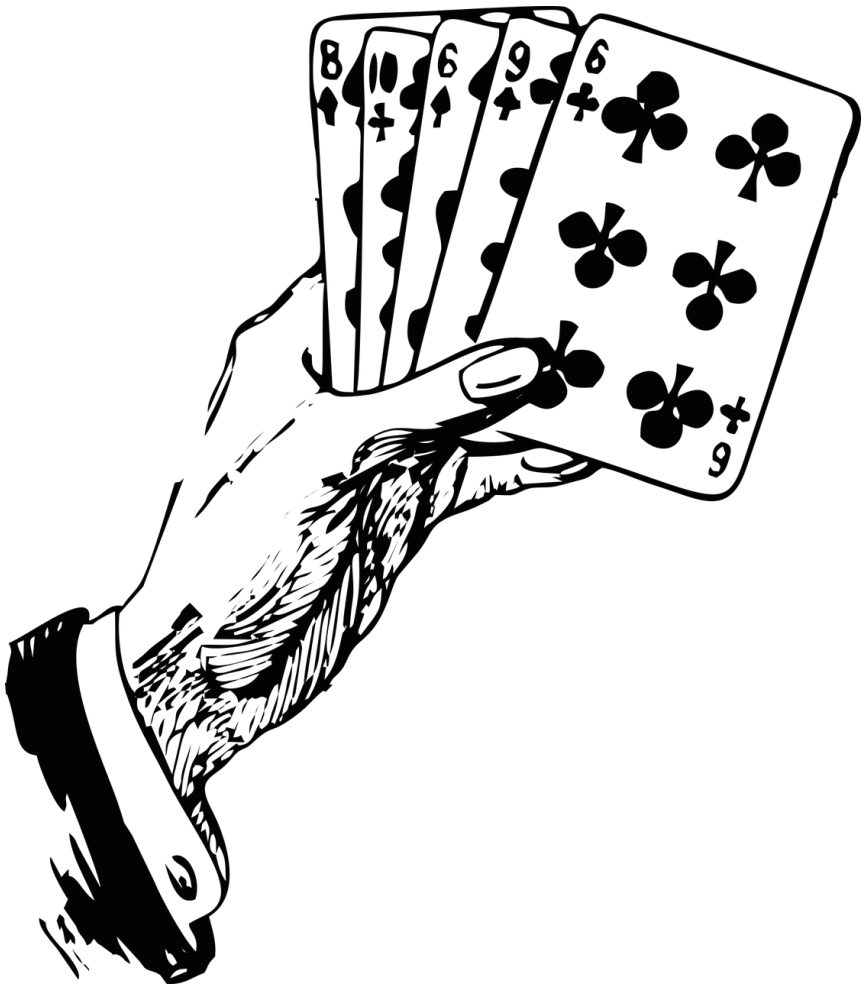
E.g. Σ_x P(Card drawn is x) = 1

What are these probabilities?

P(Card drawn is 8) =

P(Card drawn is red) =

Probabilities and Distributions: A quick revision



A probability must satisfy:

$0 <= P(x) <= 1$ and $\Sigma_x P(x) = 1$

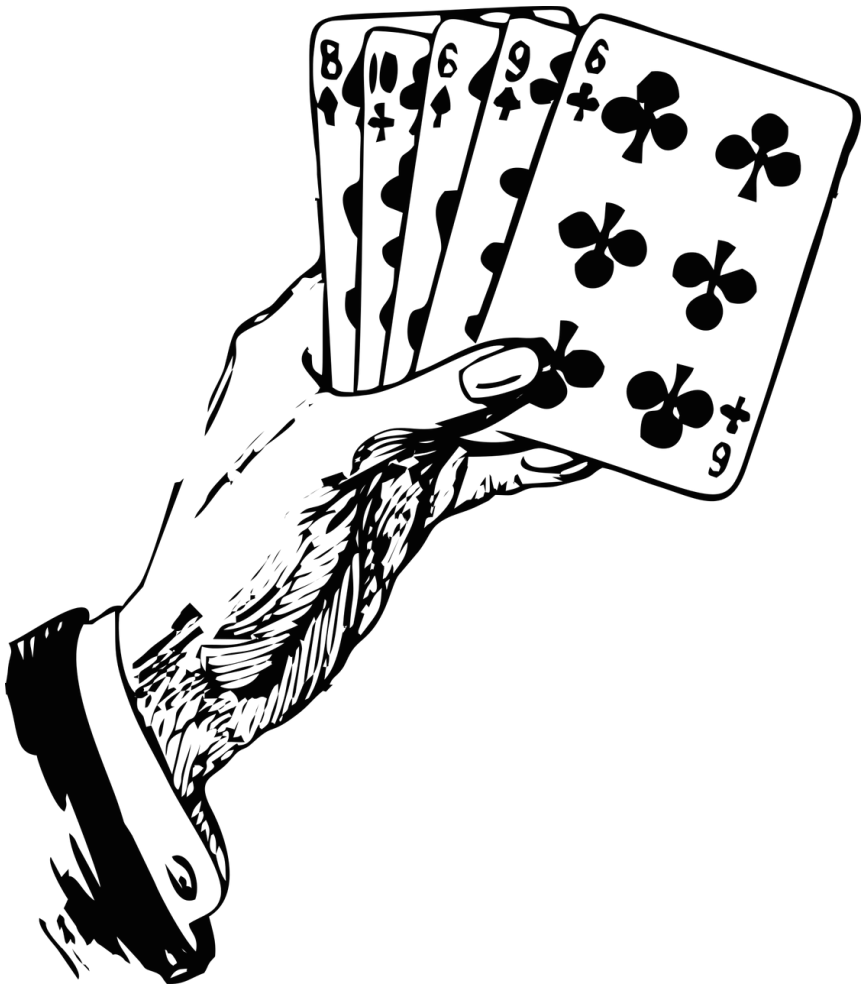
E.g. Σ_x P(Card drawn is x) = 1

What are these probabilities?

P(Card drawn is 8) = 4 / 52

P(Card drawn is red) =

Probabilities and Distributions: A quick revision



A probability must satisfy:

$0 <= P(x) <= 1$ and $\Sigma_x P(x) = 1$

E.g. Σ_x P(Card drawn is x) = 1

What are these probabilities?

P(Card drawn is 8) = 4 / 52

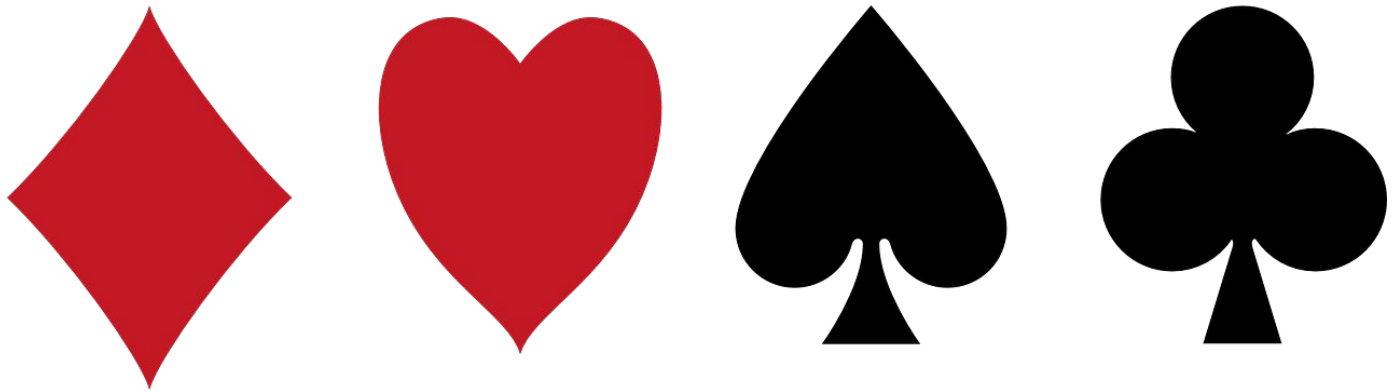
P(Card drawn is red) = 26 / 52

Joint Probability - P(A, B)

The chance of a set of events happening together

What is this probability?

P(A drawn card is 8 AND is red) = P(card is 8, card is red) =

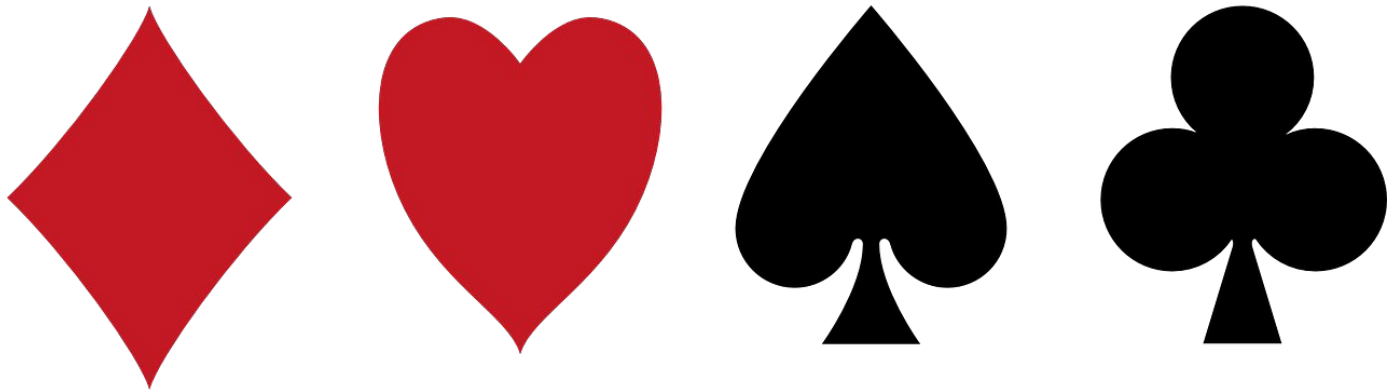


Joint Probability - P(A, B)

The chance of a set of events happening together

What is this probability?

P(A drawn card is 8 AND is red) = P(card is 8, card is red) = 2 / 52

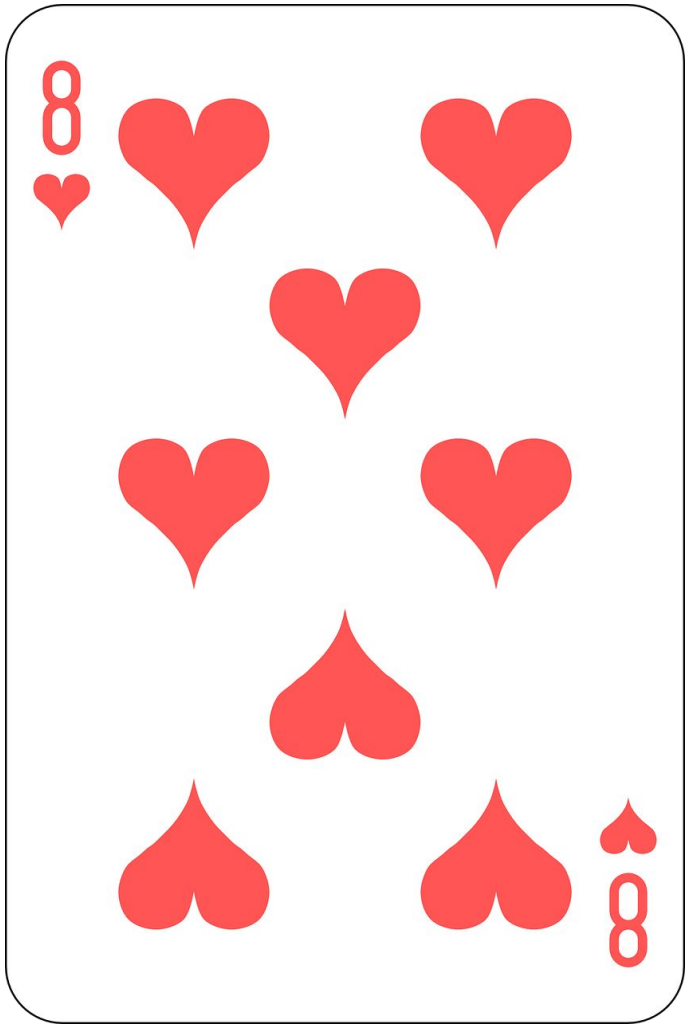


Conditional Probability - P(A | B)

Chance of an event (A) conditioned on another event (B)

What is this probability?

P(Card is 8 | Card drawn is red) =
P(Card is red | Card drawn is 8) =

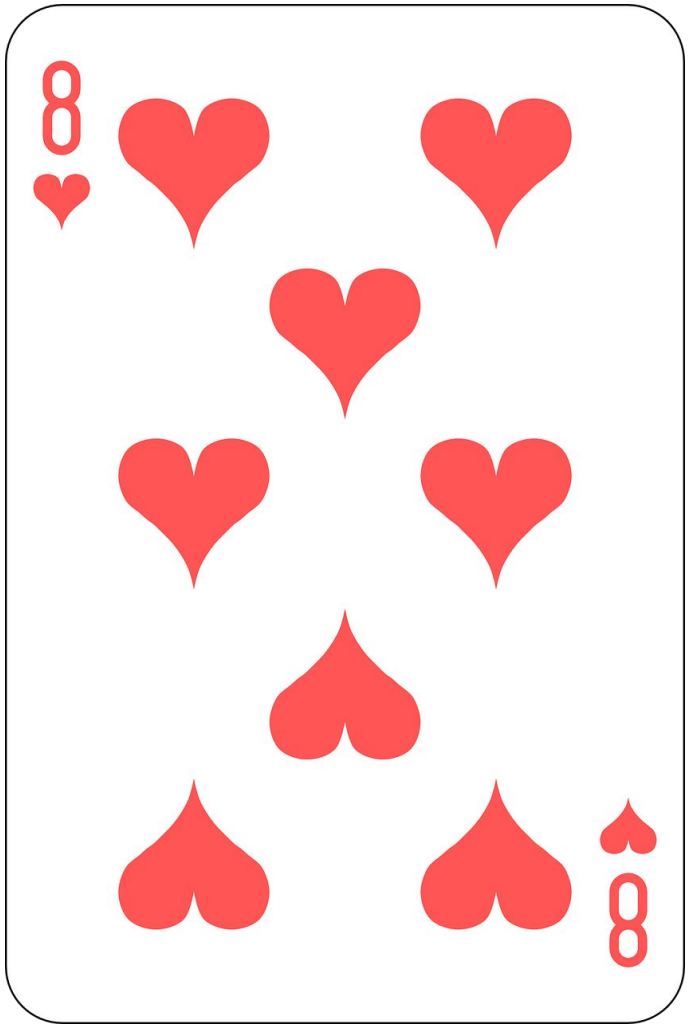


Conditional Probability - P(A | B)

Chance of an event (A) conditioned on another event (B)

What is this probability?

P(Card is 8 | Card drawn is red) = 2 / 26
P(Card is red | Card drawn is 8) = 2 / 4

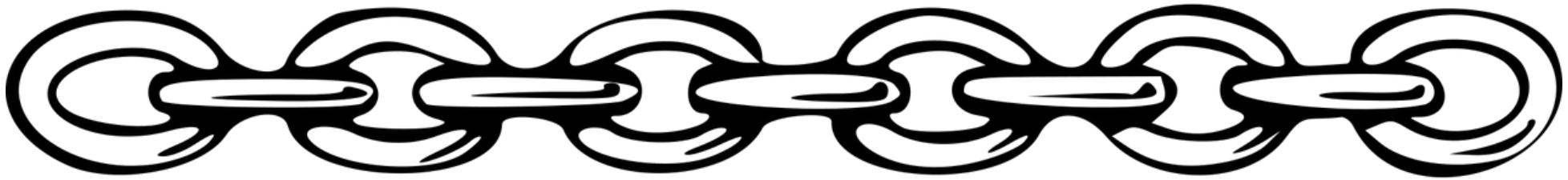


Bayes Theorem

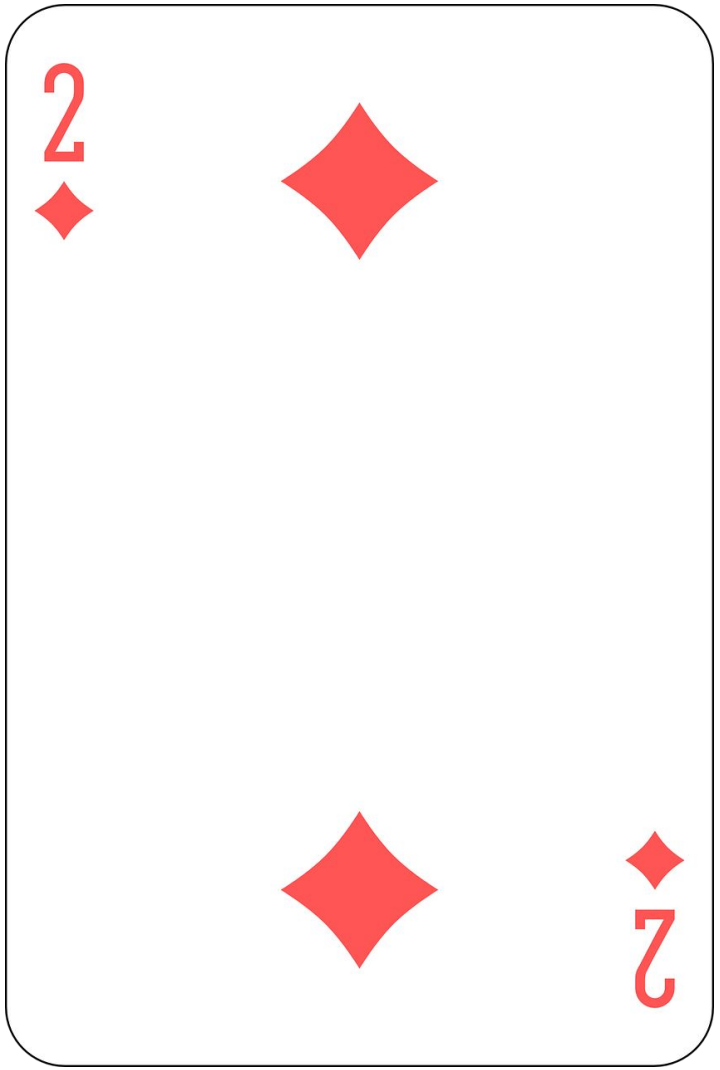
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Chain Rule

$$
\begin{aligned}
P(A, B, C, D) &= P(A \mid B, C, D) * P(B, C, D) \\
&= P(A \mid B, C, D) * P(B \mid C, D) * P(C, D) \\
&= P(A \mid B, C, D) * P(B \mid C, D) * P(C \mid D) * P(D)
\end{aligned}
$$



Independence



A and B are independent if

$P(A \mid B) = P(A)$ // B has no influence on A.

$P(B \mid A) = P(B)$ // A has no influence on B.

And:

$P(A,B) = P(A \mid B) * P(B)$ // Using chain rule

$= P(A) * P(B)$ // From above

We can use this to check if events are independent

Are: Card is 8 and Card is red independent?

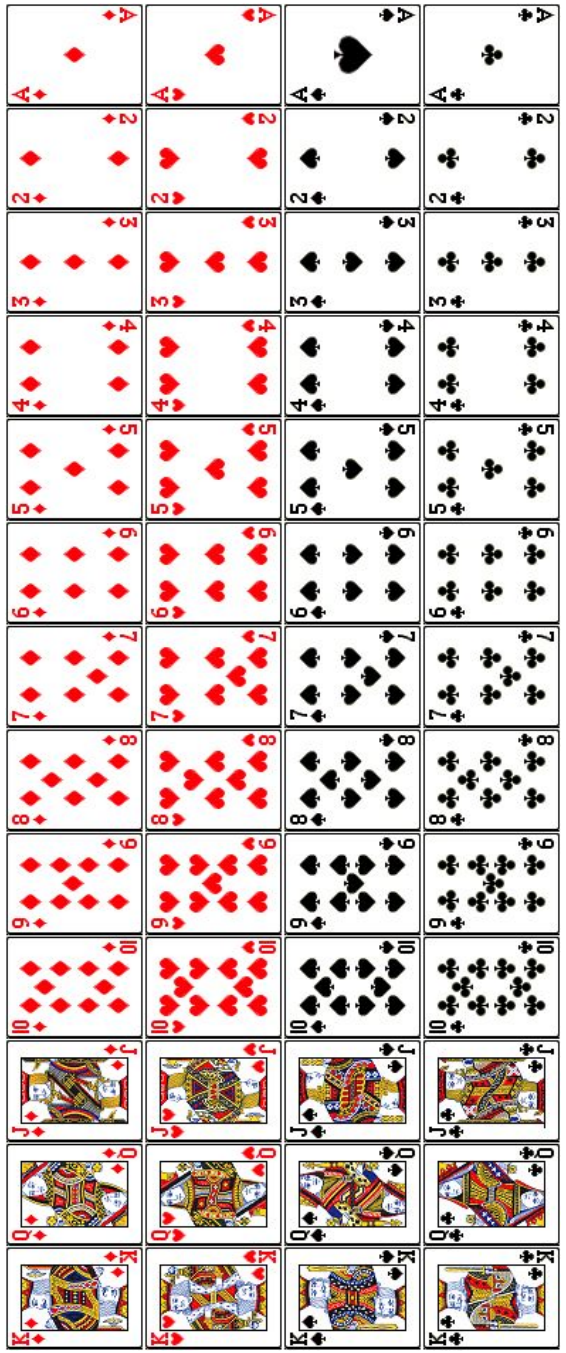
A = card is 8
B = card is red

P(A,B) = P(card is red and an 8)
= 2 / 52

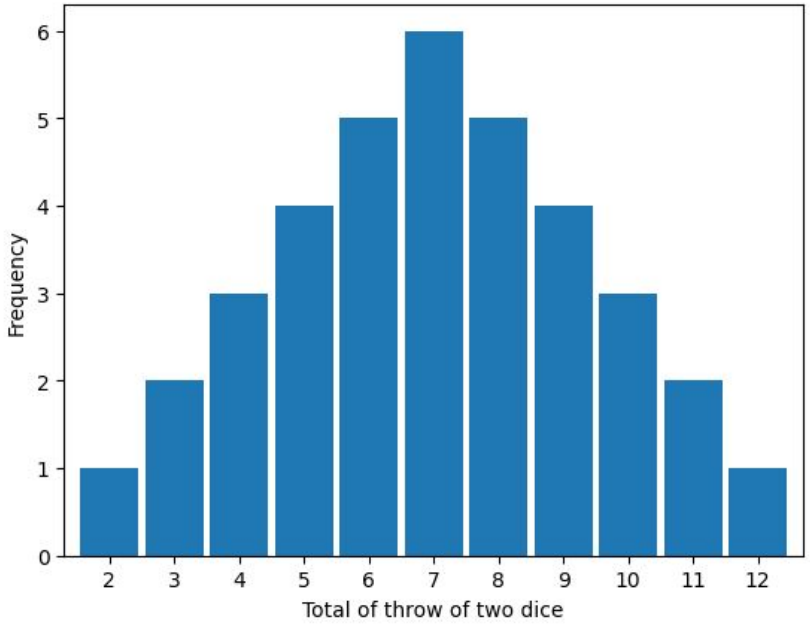
P(A) * P(B) = P(card is red) * P(card is an 8)
= 2/4 * 1/13
= 2 / 52

In this case, P(A,B) = P(A) * P(B) so A and B are independent

Imagine a deck of cards which has lost the **8 of clubs ♣**. Are A and B independent now?



Probability Distributions



- A probability distribution specifies the probabilities for **all possible outcomes** of an experiment (or set of experiments).
- Bernoulli/Binomial (for experiments with only two outcomes)
  - e.g. flipping unfair coins, cards being red, etc
- Multinomial (for >2 outcomes)
  - e.g. number on a die, total of multiple dice, picking coloured balls from an urn with red, green and blue balls
  - **Or probability of observing a word?**

Guessing the Next Token

Language models are trained to predict the next word

The meaning of life is…

Examples of an alien language:

Examples of an alien language:

What is the next emoji?

✈🎁😀🌈✈…

Examples of an alien language:

What is the next emoji?

✈🎁😀🌈✈…

menti.com
use code:
2454 2063

What techniques did you use to guess?

- Counting emojis just before ✈?
  - 🎁: 6, 💀: 4, 🦋: 4, 🔥: 2, ☕: 1
- Or before 🌈✈?
  - 🦋: 3, 💀: 2, 🎁: 1
  - Using 2+ emojis may be more specific (but fewer matches)
- Or perhaps use more than the prior emojis?
  - With enough data, you could maybe spot more elaborate patterns

What is the next emoji?

✈🎁😀🌈✈…

It was English all along…

bed: 💀
beside: 🌈
boy: 🎁
bread: ☕
eats: 👍
fruit: 🦋
is: 😀
on: 🚀
table: 🔥
the: ✈

✈🎁😀🌈✈… = the boy is beside the…

✈🎁👍🦋=the boy eats fruit

🦋😀🚀✈🔥=fruit is on the table

✈🎁😀🚀✈💀=the boy is on the bed

✈🦋😀🌈✈💀=the fruit is beside the bed

✈🎁🌈✈🦋👍🦋=the boy beside the fruit eats fruit

✈🎁👍☕=the boy eats bread

✈☕🌈✈🦋😀🌈✈💀=the bread beside the fruit is beside the bed

✈🎁👍🦋🚀☕=the boy eats fruit on bread

☕🌈✈🦋😀🚀✈🔥=bread beside the fruit is on the table

✈💀😀🌈✈🎁=the bed is beside the boy

Guessing the next token can implicitly capture grammar and more!

- Using one emoji before ✈ gave: 🎁: 6, 💀: 4, 🦋: 4, 🔥: 2, ☕: 1

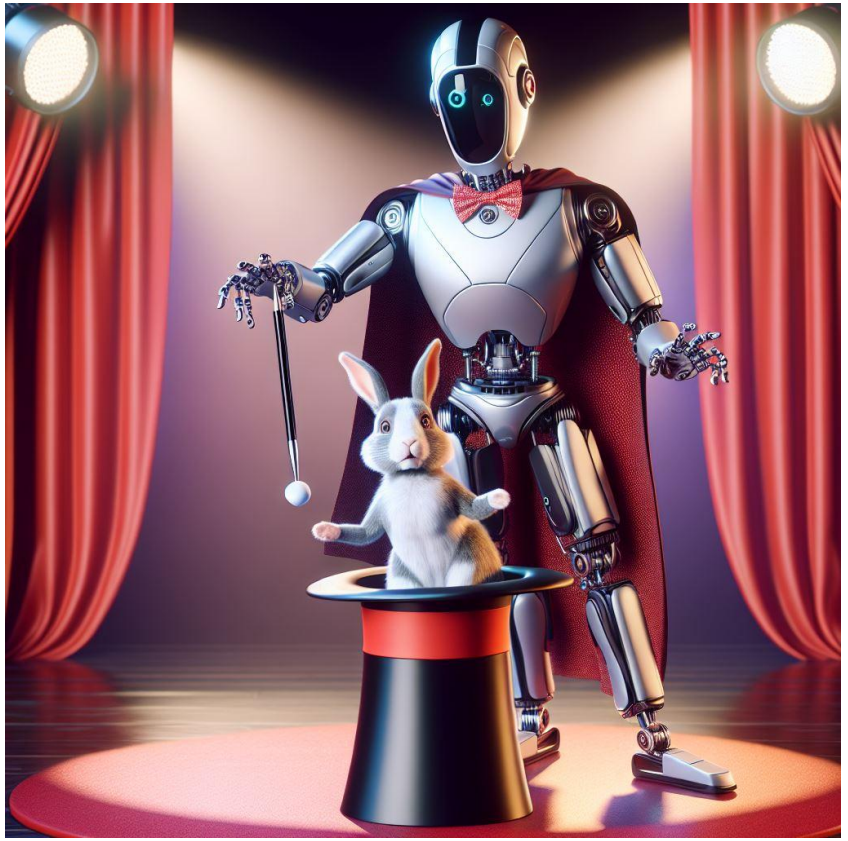
nouns!

bed: 💀
beside: 🌈
boy: 🎁
bread: ☕
eats: 👍
fruit: 🦋
is: 😀
on: 🚀
table: 🔥
the: ✈

LLMs don’t know what they’re saying

- Language models (e.g. ChatGPT) may appear to “know” things and “reason” about things
- Did you “know” anything about ✈🎁😀🌈✈?

Language model’s “skills” are often an illusion



created with Copilot and DALL-E 3

Summary of guessing the next word

- Guessing the next word is the core function of a language model
- Methods that we tried with our alien language
  - The prior emoji
  - Two prior emojis
  - Other complex patterns?
- Language modeling can implicitly capture things like grammar
  - But they don’t have any “understanding”

Language Modelling

Language Modelling: Formal Problem Definition

A language model is something that specifies the following two quantities, for all words in the vocabulary.

1. Probability of a **whole** sentence or sequence

   $P(w_1, w_2, \ldots, w_n)$

P("When I bake, I usually use butter instead of margarine")
>
P("When I bake, I usually use but her instead of margarine")

Language Modelling: Formal Problem Definition

A language model is something that specifies the following two quantities, for all words in the vocabulary.

1. Probability of a **whole** sentence or sequence

   $P(w_1, w_2, \ldots, w_n)$

P("When I bake, I usually use butter instead of margarine")
>
P("When I bake, I usually use but her instead of margarine")

2. Probability of the **next** word in a sequence

   $P(w_{k+1} \mid w_1, \ldots, w_k)$

P(**"margarine"** | "When I bake, I usually use butter instead of")
>
P(**"puppets"** | "When I bake, I usually use butter instead of")

Note on notation:

$P(w_1, w_2, \ldots, w_n)$ is short for $P(W_1 = w_1, W_2 = w_2, \ldots, W_n = w_n)$
Multinomial random variable W_1 taking on value w_1 and so on.
e.g., $P(I, love, fish) = P(W_1 = I, W_2 = love, W_3 = fish)$

Language Modelling: Formal Problem Definition

A language model is something that specifies the following two quantities, for all words in the vocabulary.

1. Probability of a **whole** sentence or sequence

   $P(w_1, w_2, \ldots, w_n)$

P("When I bake, I usually use butter instead of margarine")
>
P("When I bake, I usually use but her instead of margarine")

2. Probability of the **next** word in a sequence

   $P(w_{k+1} \mid w_1, \ldots, w_k)$

P(**"margarine"** | "When I bake, I usually use butter instead of")
>
P(**"puppets"** | "When I bake, I usually use butter instead of")

Note on notation:

$P(w_1, w_2, \ldots, w_n)$ is short for $P(W_1 = w_1, W_2 =$

Multinomial random variable W_1 taking on v

e.g., $P(I, love, fish) = P(W_1 = I, W_2 = love, W_3 =$ fish)

Important: Language modelling captures both aspects of:
- **Syntax/Grammar:** Structure of the language
- **Semantics:** Meaning / real-world "knowledge"

How to model language?

- Count! (and normalize).
- Need source text (also known as a corpora)
  - We used examples of our alien emoji language
  - Real-world corpora include all the text from Wikipedia (and more!)

Counting

One way to estimate the probability is the fraction of times you see it:

$$P(w_1, w_2, \ldots, w_n) = \text{Count}(w_1, w_2, \ldots, w_n) / N$$

where

- Count = frequency of sequence $w_1 \ldots w_n$
- N is the count of the total number of n-length sequences

Issues with using counts

A probability of a text sequence can be estimated using the number of times it appears in a large set of documents

E.g., Count(“When I bake, I usually use butter instead of margarine”) / N

Think Pair Share:

- What text might this work well for?
- What text will this not work for?

https://padlet.com/jakelever/tad2025_4

Issues with estimating the probability using observed counts

$$P(I, like, chocolate, Canada, and, basket, weaving) = \#(w_1, w_2, \ldots, w_n) / N$$

How often do people talk about their love of these three exact items?

- Conditioning on 4 or more tokens is hard because of sparsity
- Estimating probabilities from sparse observations is unreliable
- Also, no solution for a new sequence

Decomposing the probabilities

We can use chain rule to decompose joint probability into a product of conditional probabilities:

P(“the weather is snowy outside”) = P(the)
* P(weather | the)
* P(is | the weather)
* P(snowy | the weather is)
* P(outside | the weather is snowy)

Estimating conditional probabilities with long contexts is difficult! (And inefficient)

We’ll need to simplify the problem by making some assumptions.

Markov Assumption

$$P(w_{k+1} \mid \underbrace{w_{i-k}, w_{i-k+1}, \ldots, w_{k}}_{\text{Context window}})$$

Next event in a sequence depends only on its immediate past (**context**)

Instead of an arbitrarily long context, we rely on the last “n” words

n-gram models

A model is often **described by its 'order':** the size of the context window +1 / n-gram length

Unigram	$P(w_{k+1})$
Bigram	$P(w_{k+1} \mid w_k)$
Trigram	$P(w_{k+1} \mid w_{k-1}, w_k)$
4-gram	$P(w_{k+1} \mid w_{k-2}, w_{k-1}, w_k)$

Unigram models

- Next event in a sequence is *independent* of the past.

$$P(W) = \prod_i P(w_i)$$

- An extreme assumption, but can be useful nonetheless.
  - Used in **bag-of-words model and similarity**
- Not very good at modelling a sequence
  - Nonsensical phrases or sentences can get high probability.
    - P(the a an the a an the an a) > P(The dog barks)

Bigrams models

- Bi-grams: Next word is dependent on the previous word alone.
  - This model is fast, efficient, and surprisingly effective.

$$P(W) = \prod_{i} P(w_i|w_i\_1)$$

Language model exercise

Given a sequence, [the dog barks], what's the conditional probability of the last word:

- P(barks | the, dog)

- "the dog barks" occurs 56 times in a text collection
- "the dog" occurs 1190 times in a text collection
- What is the 'order' of this model?

Language model exercise

Given a sequence, [the dog barks], what's the conditional probability of the last word:

- P(barks | the, dog)

- "the dog barks" occurs 56 times in a text collection
- "the dog" occurs 1190 times in a text collection
- What is the 'order' of this model?

Answer:

- P(barks | the, dog) =
  #(the, dog, barks) / # (the, dog)
- P(barks | the, dog) = 56 / 1190
- P(barks | the, dog) = 0.047…
- Trigram model (order 3)

Practical LM Issues: Start & End Tokens

We introduce special START and END tokens at the beginning of a sequence. This allows us to model the likelihood of beginning or ending of a sentence with a word.

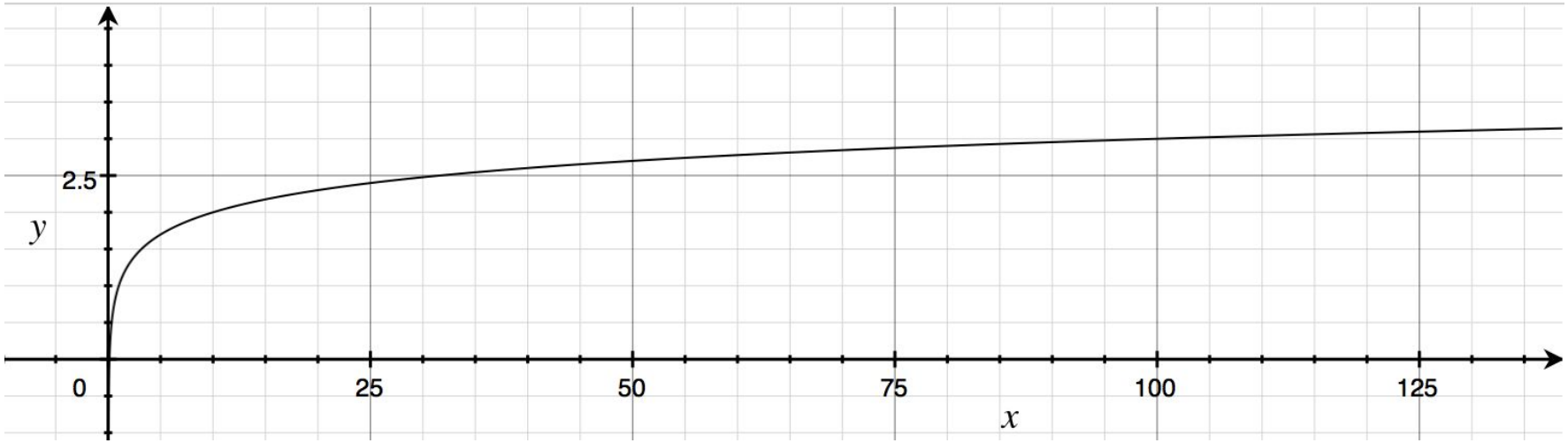
- Similar to the HTML tags and
- Implicitly the first word for a bigram model is: P(w|<start>), sometimes P(w)
- Allows identification of the end of a sequence, e.g., P(<end> | “I like pie.”)

<start>…….<end>

Practical LM Issues: Log Space

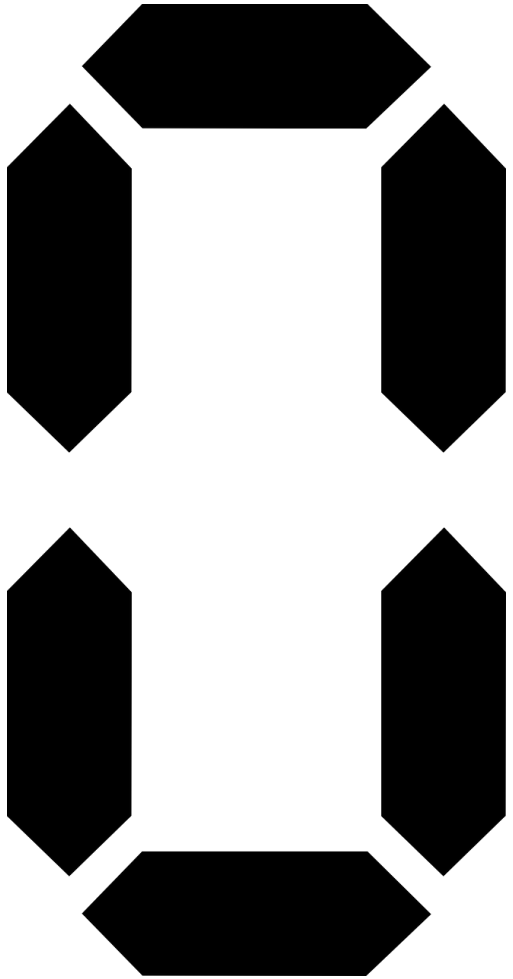
We do probability computations in log space

- To avoid underflow of very small numbers
- Adding is faster than multiplying
- Sometimes easier to interpret
- $\log (p_1 \times p_2 \ldots \times p_n) = \log(p_1) + \log(p_2) + \ldots + \log(p_n)$
- The base of the log is arbitrary



Problem!?

- What about a sequence with a new word:
  P(language models are fascinating #ilovetad)
- Model has never seen #ilovetad before.
  - P(string) = 0
- Problem: What do we do about never before observed words (or sequences)?
  - How do we avoid assigning 0 probability to this event?



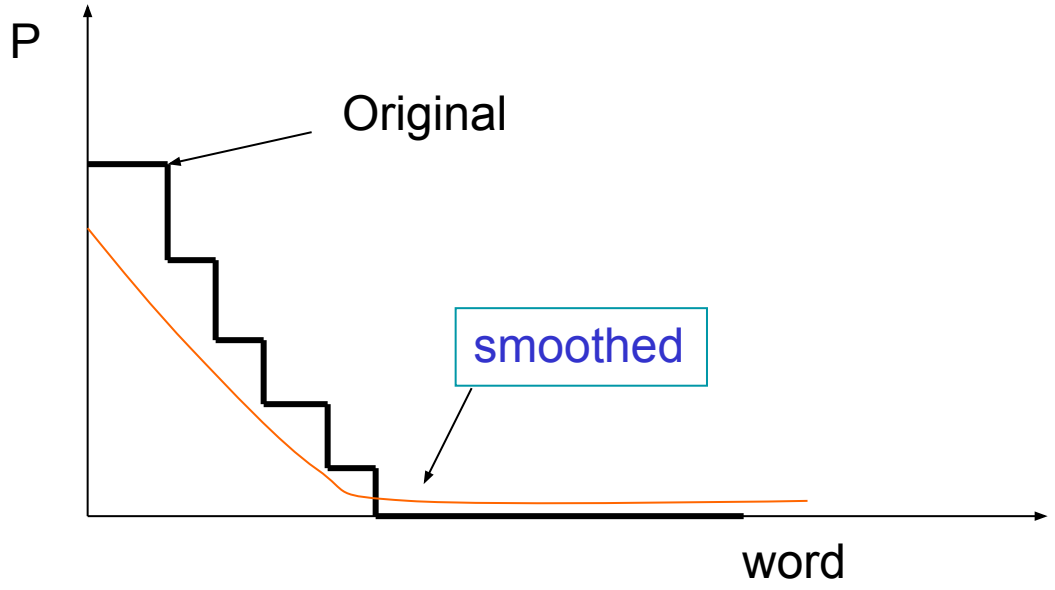
Smoothing!

- **Goal:** assign a low (non-zero) probability to words or n-grams not observed in the text collection (*training data*)

- Texts are just a sample from the language model
  - Missing words **should not have zero probability** of occurring!
  - A non-occurring term is possible (even though it didn't occur in the document or training examples)

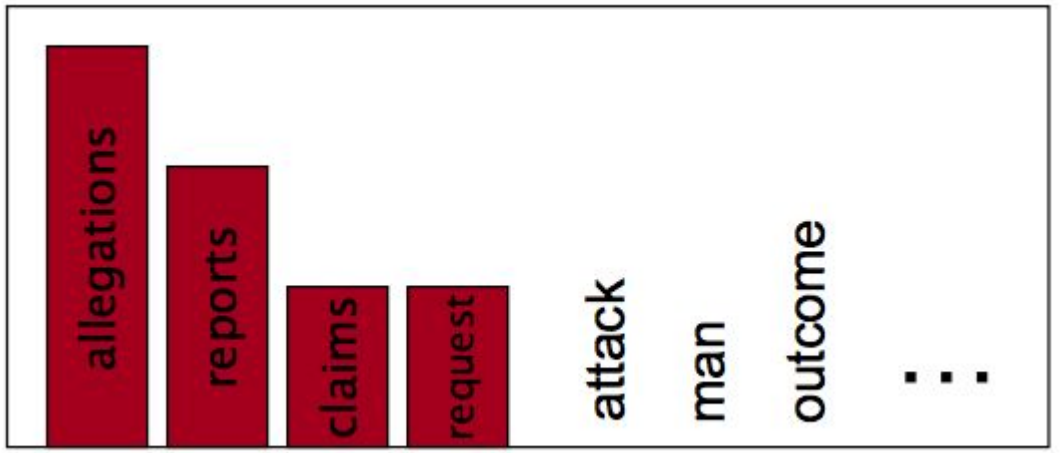
- Smoothing is a technique for estimating **probabilities for missing** (or unseen) **words**
  - Lower (discount) the probability estimates for words that are seen in the document text
  - Assign the "left-over" probability to the estimates for the words that are not seen in the text

Discounted Smoothing

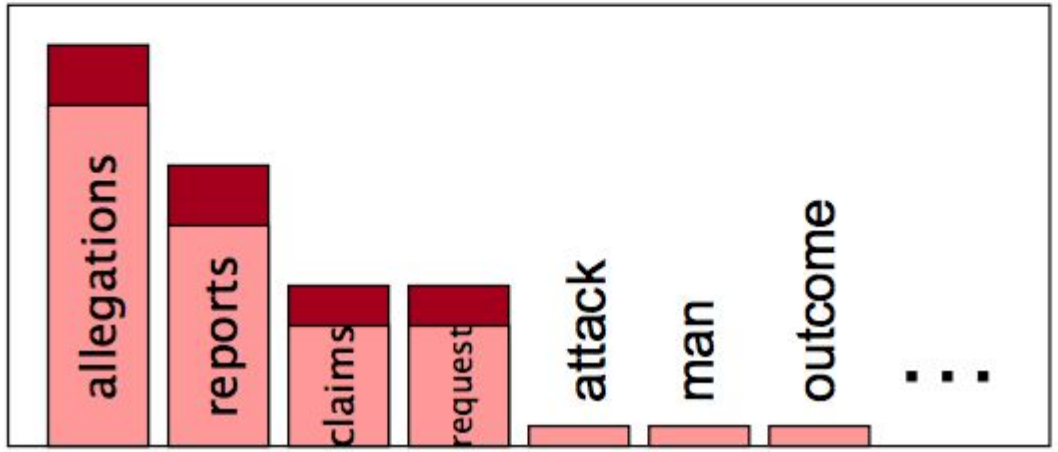


Take some probability of the non-zero tokens and give it to the tokens with zero probability

P(w | denied, the)



Original distribution



Discounted

Add-K / Laplace Smoothing

A type of discounted smoothing.

Assume that there were some additional documents in the corpus, where every possible sequence of words was seen exactly k times

$$P_{add-k}(t|\theta) = \frac{count(t,\theta) + k}{\sum_{t'} count(t',\theta) + k|V|}$$

where count(t|θ) is the count of term t in corpus θ and |V| is the number of unique words in the vocabulary

- Add-one smoothing can be used with k=1, but often bad as too much probability is taken from seen events
- k can be a fraction

Interpolated smoothing

- Combines a higher order (more sparse) model with a lower order (less sparse) model (e.g. trigram -> bigram -> unigram)
- Example for a trigram model:

$$
\begin{aligned}
P_{interpolated}(t_i|t_{i-1}, t_{i-2}) &= \lambda_1 P(t_i|t_{i-1}, t_{i-2}) \\
&= \lambda_2 P(t_i|t_{i-1}) \\
&= \lambda_3 P(t_i)
\end{aligned}
$$

$$
\lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0 \text{ and } \lambda_1 + \lambda_2 + \lambda_3 = 1
$$

- How do you pick values for λ?
  - Experiment!

Jelinek-Mercer smoothing

- Interpolate between the frequency of a token in the document ($P(t|\theta_d)$) and the frequency of a token in the whole corpus ($P(t|\theta_c)$)

$$P_{jelinek-mercer}(t|\theta_d, \theta_c) = \lambda P(t|\theta_d) + (1-\lambda)P(t|\theta_c)$$

Probability based on tokens in **this document**

Probability based on tokens across **whole corpus**

Summary of document smoothing

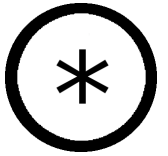
$$P_{add-k}(t|\theta) = \frac{count(t,\theta) + k}{\sum_{t'} count(t',\theta) + k|V|}$$

$$P_{jelinek-mercer}(t|\theta_d,\theta_c) = \lambda P(t|\theta_d) + (1-\lambda)P(t|\theta_c)$$

$$\begin{aligned} P_{interpolated}(t_i|t_{i-1},t_{i-2}) &= \lambda_1 P(t_i|t_{i-1},t_{i-2}) \\ &= \lambda_2 P(t_i|t_{i-1}) \\ &= \lambda_3 P(t_i) \end{aligned}$$

More advanced smoothing

- There are many(!) other smoothing models for text.
  - Kneser-Ney - a ‘state-of-the-art’ model
- But many have complicated parameters that require tuning
- When in doubt, start simple!
- ‘Stupid’ backoff used at Google
  - Computing large n-gram probabilities (up to length 5)
  - If P() > 0 then use it; if 0, then use α = 0.4 * next lower-order model
  - Example: if trigram is 0, use 0.4* bigram probability
  - **Not a probability!!**



Break!

Up next: Evaluating language models

Evaluation

What is a good language model? An ideal perspective!

- To be a bit recursive, a good language model should model the language well.
- Well formed English sentences should be more probable.

What is a good language model? An ideal perspective!

Grammatically/syntactically – match the "rules" of the language

P("like i tad") < P("i like tad")

Semantically – conveys actual meaning

P("I saw an elephant at the circus") > P("I saw a supernova at the circus")

P("I saw a supernova at the planetarium") > P("I saw an elephant at the planetarium")

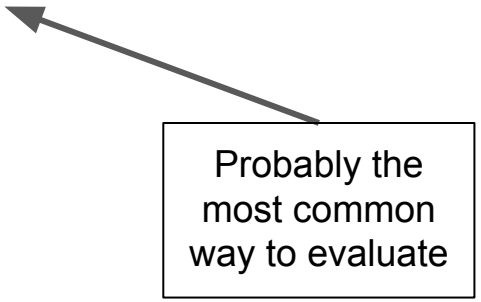
Not all syntactically-sound sentences make sense though, e.g., "colorless green ideas sleep furiously" – Noam Chomsky

made with Midjourney

Evaluating language models

- A trigram model captures more context than a bigram model, so it should be a “better” model, but how do we measure it?

- **Extrinsic**: measure performance on downstream application
  - Ex: Spelling correction, speech recognition, translation, etc…



- **Intrinsic**: design a measure inherent to the current task
  - Challenging for language modelling
  - What does your language model make of “real text”?

Calculating the probability of “real unseen text”

A good language model should give a higher probability for plausible / real text, e.g. the works of Charles Dickens

Token	Probability of token from LM
It	0.6
was	0.3
the	0.2
best	0.8
of	0.3
times	0.1

$P(w_i \mid w_1, w_2, w_3, \ldots)$

Calculating the probability of “real unseen text”

A good language model should give a higher probability for plausible / real text, e.g. the works of Charles Dickens

Token	Probability of token from LM
It	0.6
was	0.3
the	0.2
best	0.8
of	0.3
times	0.1

Probability of text	0.000864

Calculate probability of entire sequence with:

$$\prod P(w_i \mid w_1, w_2, w_3, \ldots)$$

i.e. multiply all the probabilities up

But, the probability quickly gets very very small

Calculating the probability of “real unseen text”

A good language model should give a higher probability for plausible / real text, e.g. the works of Charles Dickens

Token	Probability of token from LM
It	0.6
was	0.3
the	0.2
best	0.8
of	0.3
times	0.1

Probability of text	0.000864
Per-token probability	0.309

To deal with the tiny probabilities of a whole sequence, we work with per-token probabilities.

For a sequence with N token:

Per-token P = Nth root of P(sequence)
= $P(\text{sequence})^{1/N}$

This is taking the **geometric mean** of the probabilities

Calculating the probability of “real unseen text”

A good language model should give a higher probability for plausible / real text, e.g. the works of Charles Dickens

Token	Probability of token from LM
It	0.6
was	0.3
the	0.2
best	0.8
of	0.3
times	0.1

Probability of text	0.000864
Per-token probability	0.309
Perplexity	3.240

Lastly, we take the reciprocal (1 / per-token probability) to get the perplexity.

$$\text{Perplexity} = \frac{1}{\sqrt[n]{\prod P(w_i \mid w_1, w_2, \ldots, w_{i-1})}}$$

The equation for perplexity

$$\text{Perplexity} = \frac{1}{\sqrt[n]{\prod_{i=1}^{n} P(w_i \mid w_1, w_2, \ldots, w_{i-1})}}$$

Token probability from language model (could be unigram, bigram, or other)

where n = the length of the sequence

- **Low** perplexity means the language model gave **high** probabilities for the text
  - so wasn't surprised by it
- **High** perplexity means the language model gave **low** probabilities for the text
  - so was surprised by it
- Perplexity range from 1 to the size of vocabulary (as we shall see)

Perplexity of a random language model

What if each token was just randomly picked from a vocabulary of 10,000 words?

Token	Probability of token from LM
It	0.0001
was	0.0001
the	0.0001
best	0.0001
of	0.0001
times	0.0001

Probability of text	1e-24
Per-word probability	0.0001
Perplexity	10000

Perplexity of a random language model equals the size of the vocabulary

$$\text{Perplexity} = \frac{1}{\sqrt[n]{\prod P(w_i \mid w_1, w_2, \ldots, w_{i-1})}}$$

Perplexity of a perfect language model

What if each token was guessed perfectly?

Token	Probability of token from LM
It	1
was	1
the	1
best	1
of	1
times	1

Probability of text	1
Per-word probability	1
Perplexity	1

Perplexity of a perfect language model is 1

$$\text{Perplexity} = \frac{1}{\sqrt[n]{\prod P(w_i \mid w_1, w_2, \ldots, w_{i-1})}}$$

Example of perplexity results from recent research papers

- Big corpora of text (e.g. from Wikipedia) often used to calculate perplexity
- Perplexity (ppl) used to compare different approaches to language modelling

Rank	Model	Test perplexity	↓ Validation perplexity	BPB	Number of params	Extra Training Data	Paper	Code	Result	Year	Tags
1	**Megatron-LM**	10.81			8300M	✓	Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism			2019	Transformer
2	**GLM-XXLarge** (bidirectional)	11.33			10000M	✓	GLM: General Language Model Pretraining with Autoregressive Blank Infilling			2021	
3	**GLM-XXLarge** (unidirectional)	12.22			10000M	✓	GLM: General Language Model Pretraining with Autoregressive Blank Infilling			2021	
4	**kNN-LM**	15.79	15.81		247M	×	Generalization through Memorization: Nearest Neighbor Language Models			2019	
5	**Routing Transformer**	15.8				×	Efficient Content-Based Sparse Attention with Routing Transformers			2020	Transformer
6	**Transformer-XL** (RMS dynamic eval)	16.4	15.8		257M	×	Dynamic Evaluation of Transformer Language Models			2019	Transformer

https://paperswithcode.com/sota/language-modelling-on-wikitext-103

Practically calculating perplexity

- Recall: multiplying probabilities directly is a bad idea!
  - Numbers get very tiny and hard to compute
- We work with log-transformed probabilities instead
  - Multiplication becomes addition!
- How does this affect calculating perplexity?
  - Log-transformed per-word probability gives an estimate for **cross-entropy**
  - This has direct links to perplexity

```
  0.032473218972138
x 0.321684321351642
x 0.987312654123324
x 0.426545433489721
x 0.897324231982315
x 0.165643216843212
x 0.567423136542131
x 0.879238602321546
x 0.065432852138621
x 0.16841280564213 4
x 0.789123681241238
x 0.213432165423184
x 0.572618465132987
x 0.546723135448973
x 0.370917652048341
x 0.232654853282831
x 0.218281238386153
x 0.024272176096324
x 0.245823244659824
```

Cross-entropy

- A measure of the difference between the true probability distribution and the language model's probability distribution.
- We treat some "real" text as a sample from the true probability distribution.
  - In reality, we should use lots of text so our cross-entropy estimate is more accurate

Token	**Probability of token from LM**	**Log2 probability**
It	0.6	-0.73697
was	0.3	-1.73697
the	0.2	-2.32193
best	0.8	-0.32193
of	0.3	-1.73697
times	0.1	-3.32193

Average log probability	-1.696
Cross-entropy	1.696

Cross entropy equals the negative of the average log probability (base 2) of each word

Perplexity from cross-entropy

$$\text{Perplexity} = 2^{\text{cross-entropy}}$$

Token	Probability of token from LM	Log2 probability
It	0.6	-0.73697
was	0.3	-1.73697
the	0.2	-2.32193
best	0.8	-0.32193
of	0.3	-1.73697
times	0.1	-3.32193

Average log probability	-1.696
Cross-entropy	1.696
Perplexity	3.240

It's the same number as before!

The equation for perplexity uses a power of 2, because we used log base 2 probabilities.

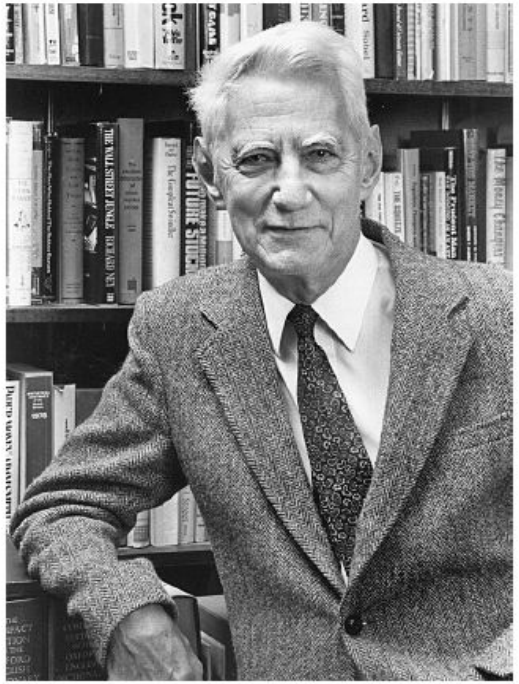
The link to information theory

Link to information theory

The **intrinsic measures (perplexity and cross-entropy)** have their root in “Information Theory”

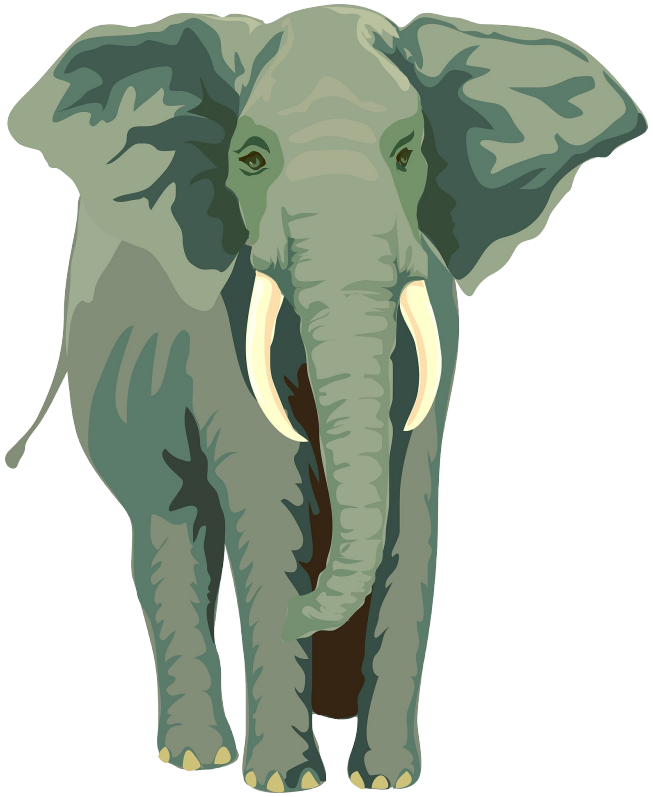
Most notably **Claude Shannon** worked on how to best **encode information**.

His work lead to the foundation of the “Information Theory” field



Claude Shannon
1916–2001

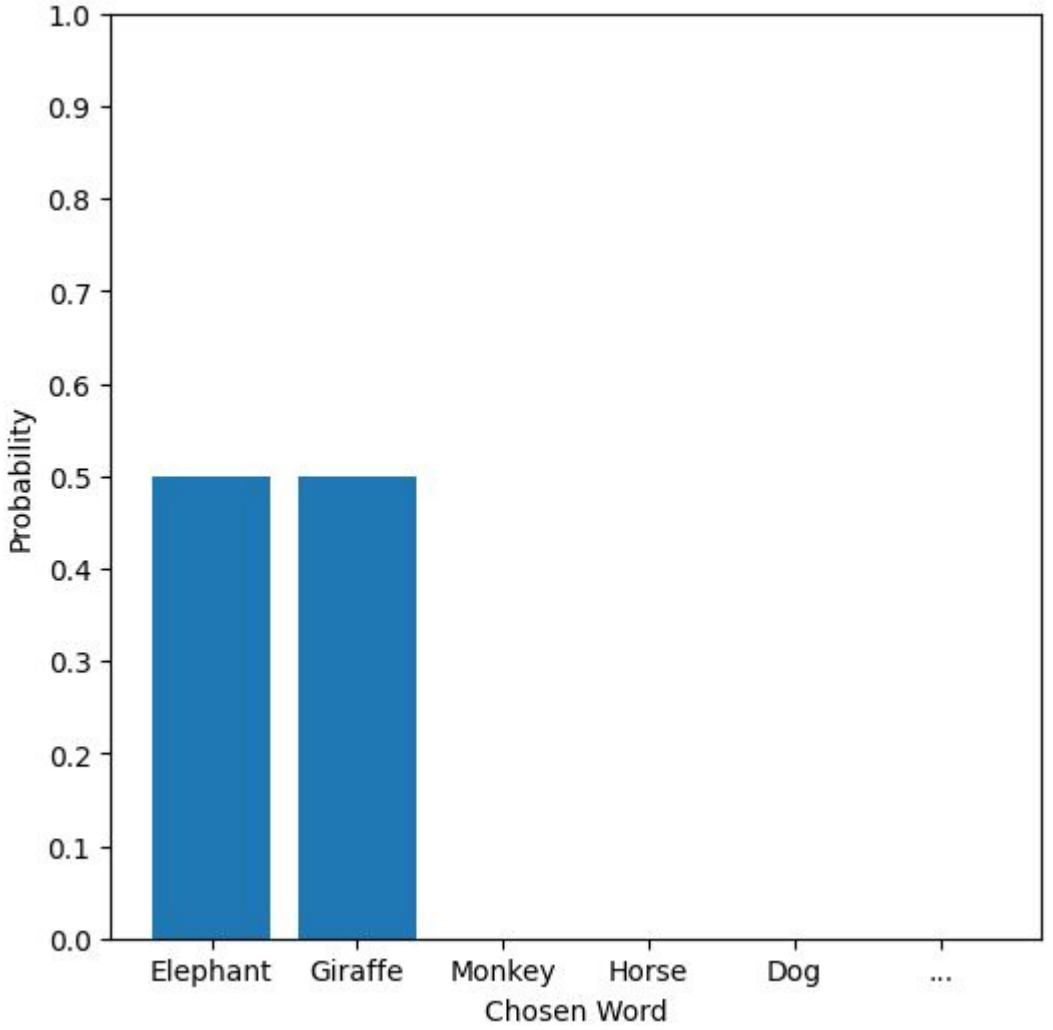
Let’s play 20 questions



- You can ask Yes/No questions to guess what I’m thinking
- I’m not very good at it and always think of one of two animals
- What questions should you ask?

Treating my choice as a probability distribution

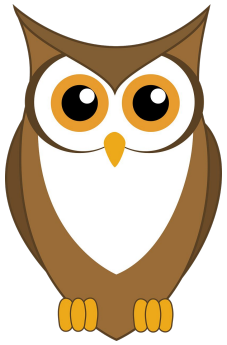
- Only two possible choices
- Let’s say they have equal possibility

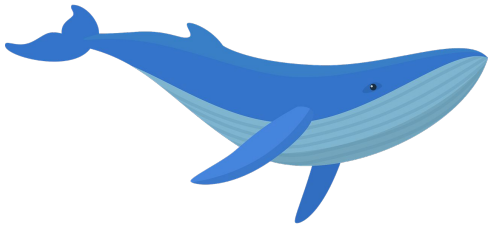


My friend is better, but always picks from 8 things







How many Yes/No questions would you need to get there?
(if you are playing optimally)

My friend is better, but always picks from 8 things

- Assuming equal likelihood:
  - Three questions: Does it fly? Is it man-made? Is it X?

Number of questions is a useful metric for information content

Few required questions = low information content

Lots of questions = high information content

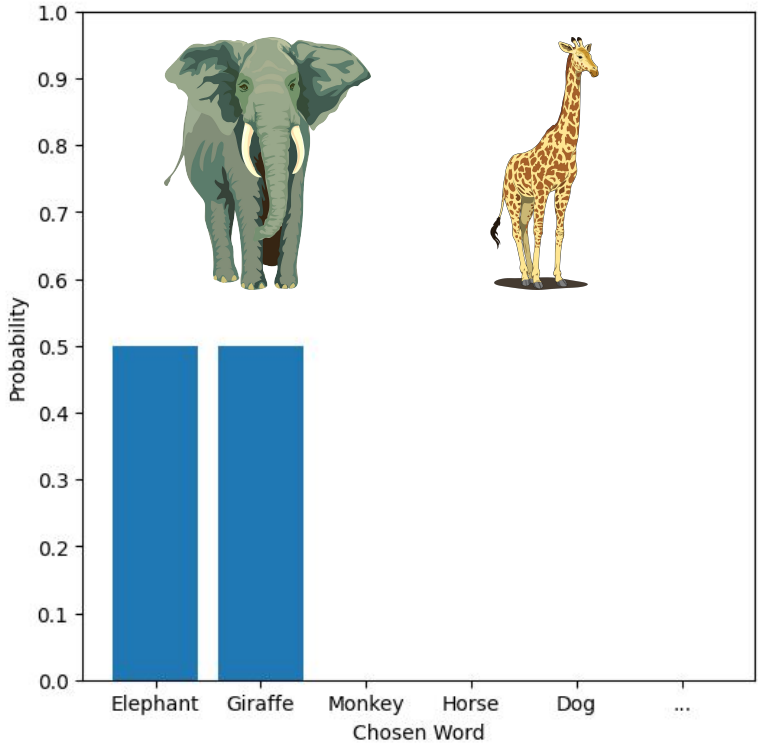
Uses bits instead of “questions”

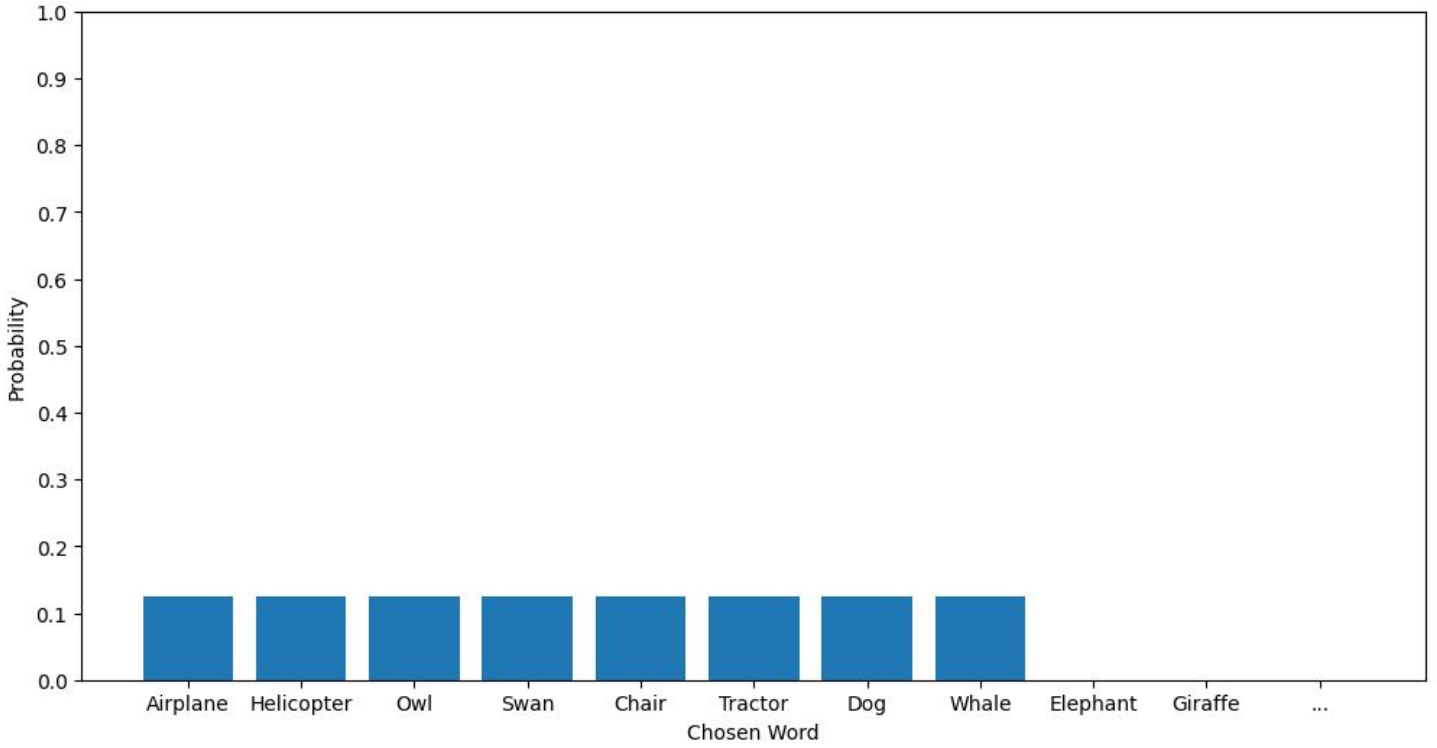
1 = Yes 0 = No

Few bits = low information content
Lots of bits = high information content

Breather

Probability distributions for which word is chosen





Requires

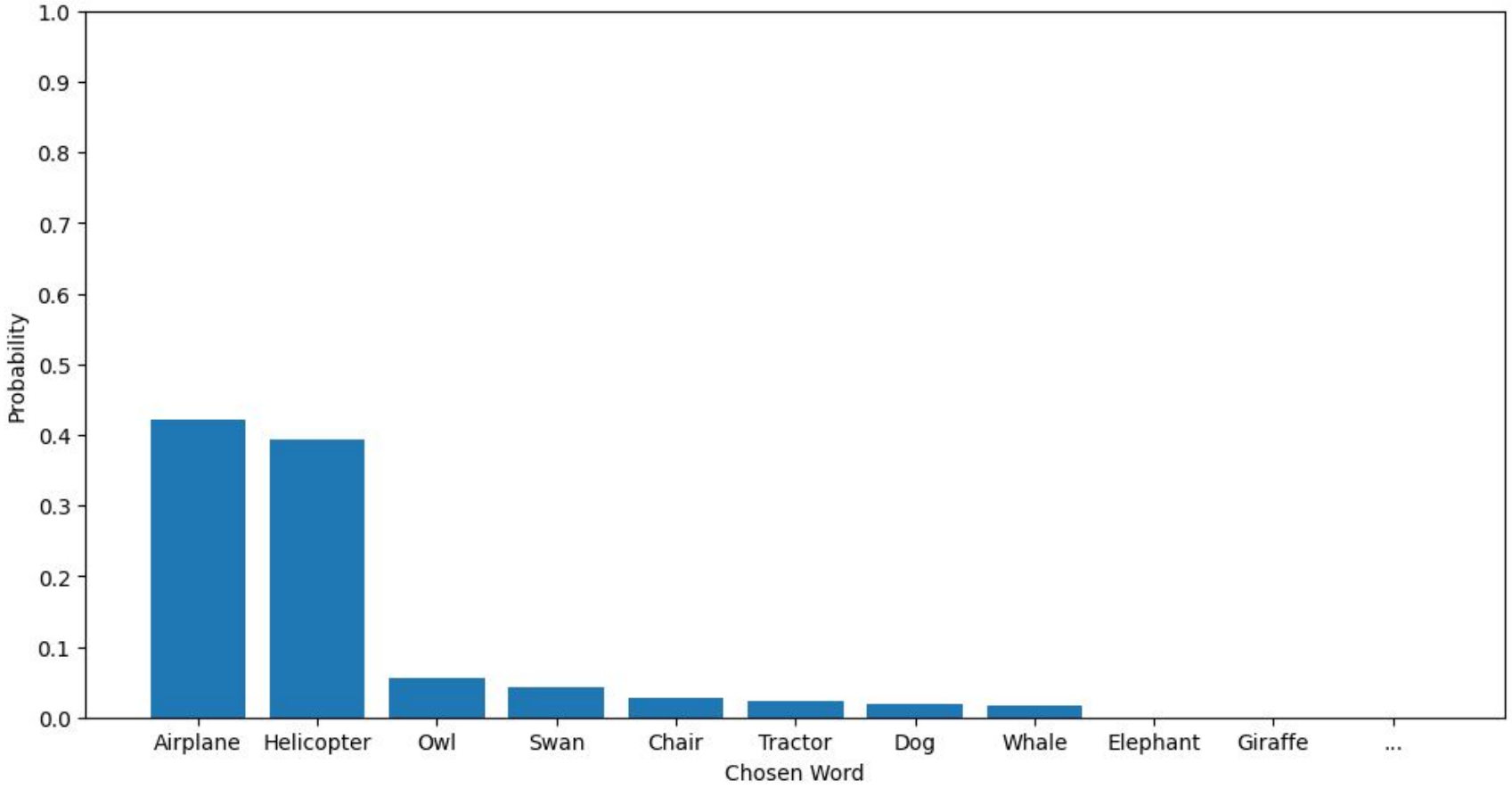
- 1 question
- 1 bit of information content

Requires

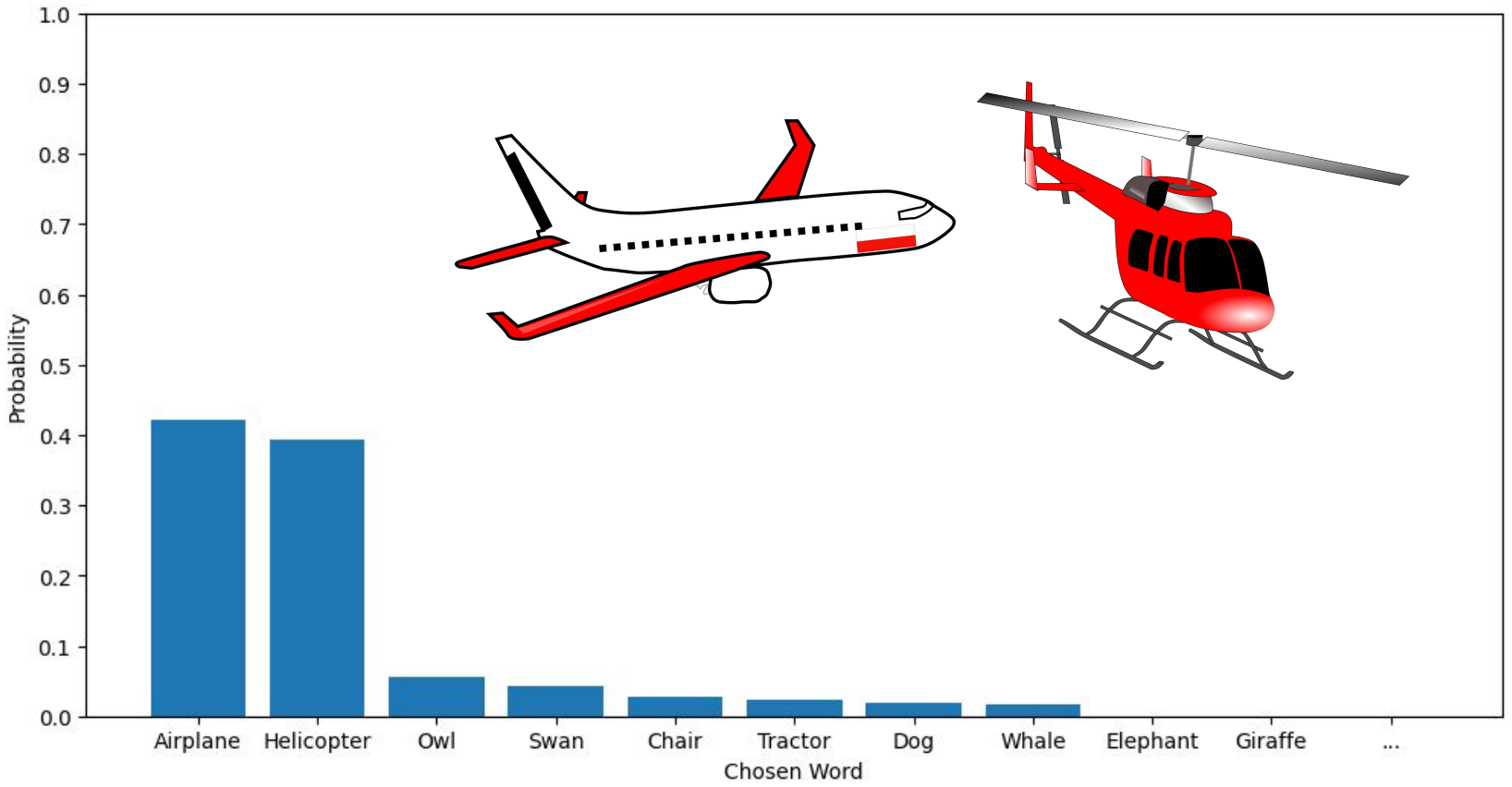
- 3 questions
- 3 bits of information content

What about a non-uniform distribution?

- I’m very likely to chose Airplane or Helicopter
- What questions to ask?
- Would we need greater or fewer questions?



Different strategy for non-uniform distribution



- Asking “is it an airplane or helicopter” first may be optimal strategy

Entropy of a probability distribution

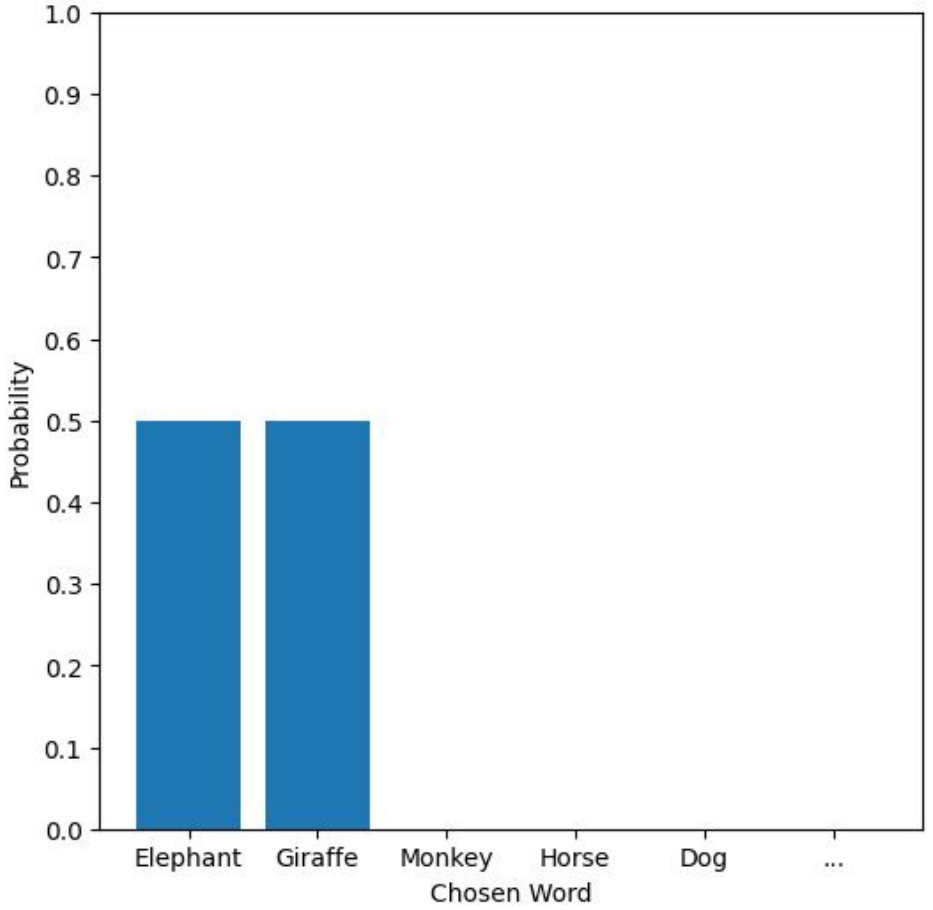
- Entropy is the measurement of the average uncertainty of information in a probability distribution
  - Expected number of bits needed to encode a message.
- Informally, the total “surprise”
  - Adding up the surprise of each possible event, weighted by its likelihood

$$H(X) = -\sum_{x} \underbrace{P(x)}_{\text{Weighting by the likelihood}} \underbrace{log_2(P(x))}_{\text{The number of bits to encode this event (the surprise)}}$$

Let's calculate entropy: elephant or giraffe?

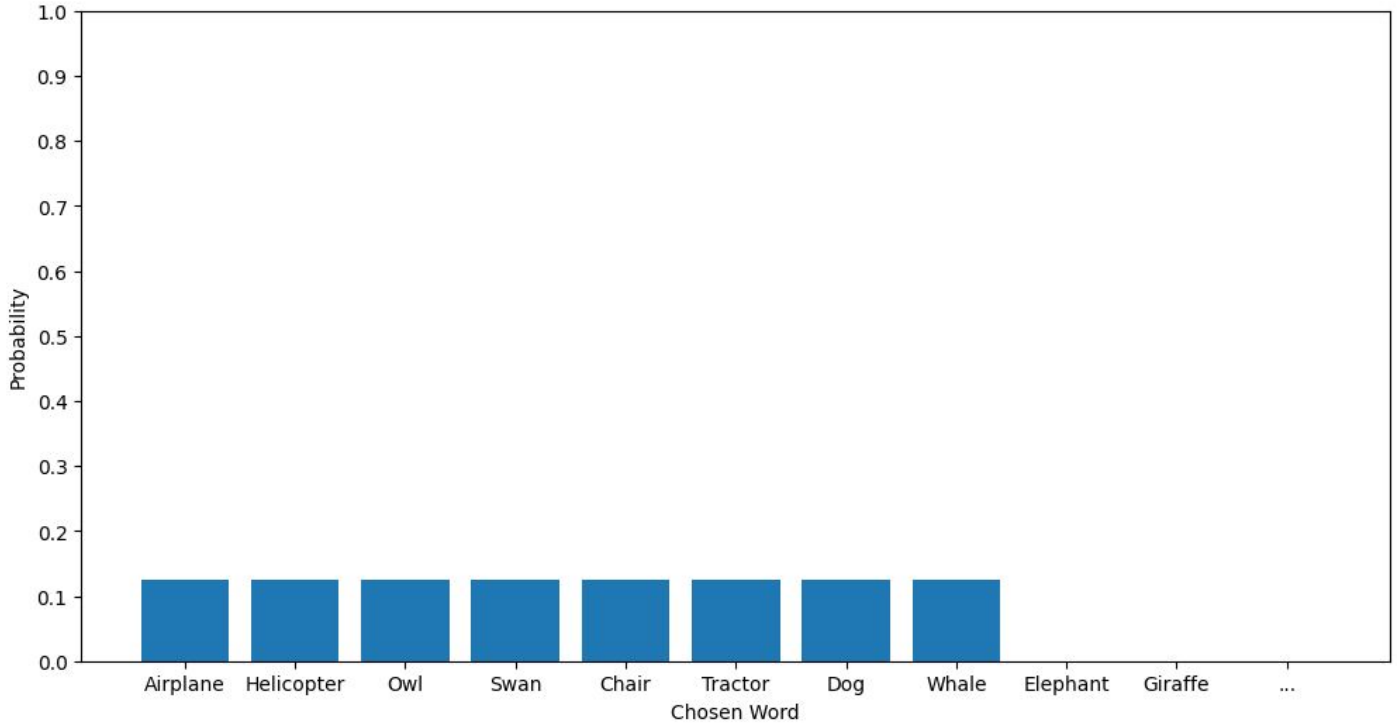
$$H(X) = -\sum_{x} P(x) log_2(P(x))$$
$$= -(0.5 log_2(0.5) + 0.5 log_2(0.5))$$
$$= -(-0.5 + -0.5)$$
$$= 1$$

1 question!!!



Let's calculate entropy: one of eight items

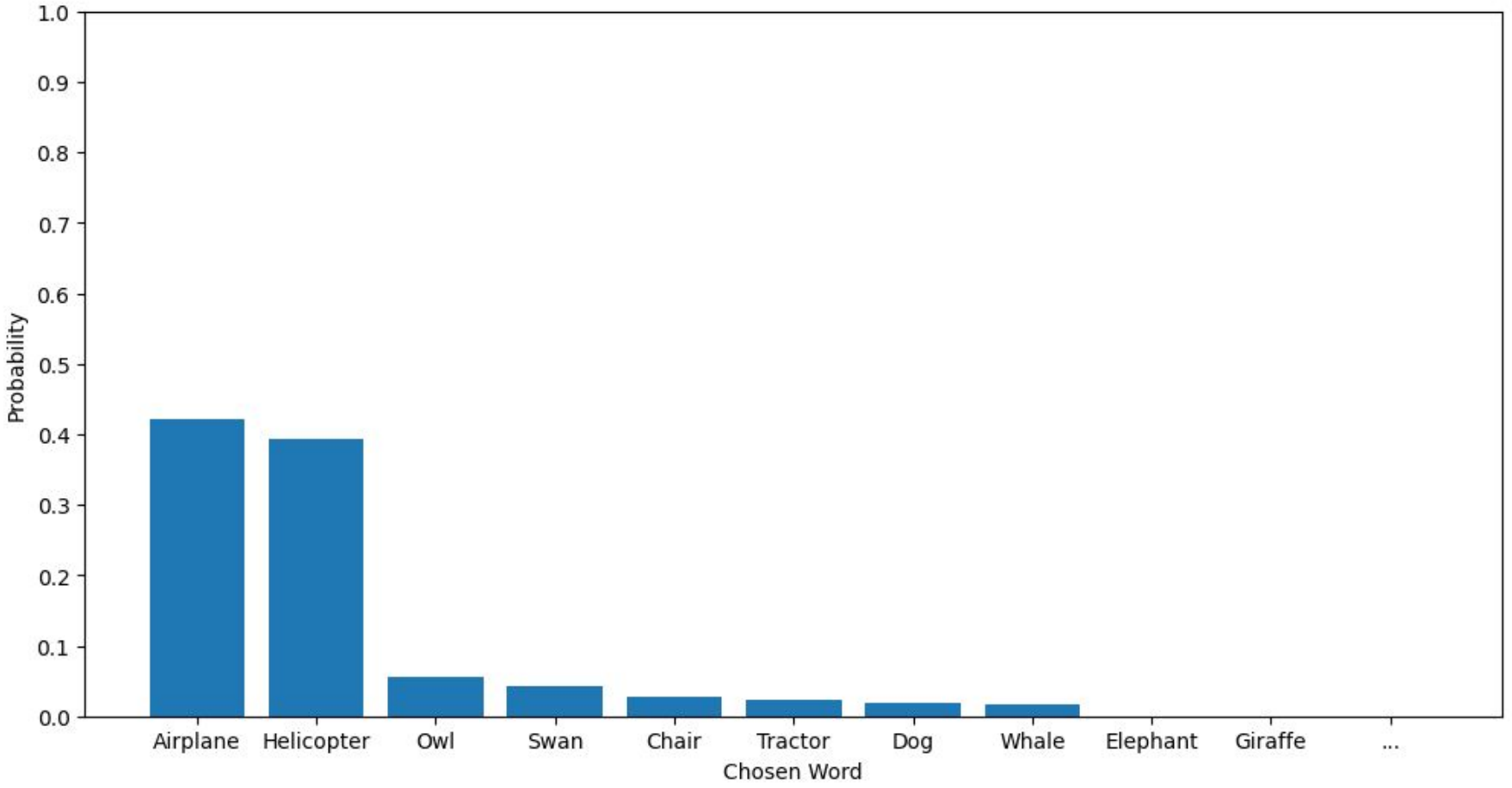
$$
\begin{aligned}
H(X) &= -\sum_{x} P(x) log_2(P(x)) \\
&= -\sum_{x} P(0.125) log_2(P(0.125)) \\
&= -\sum_{x} -0.375 \\
&= -(-3) \\
&= 3
\end{aligned}
$$



3 questions!!!

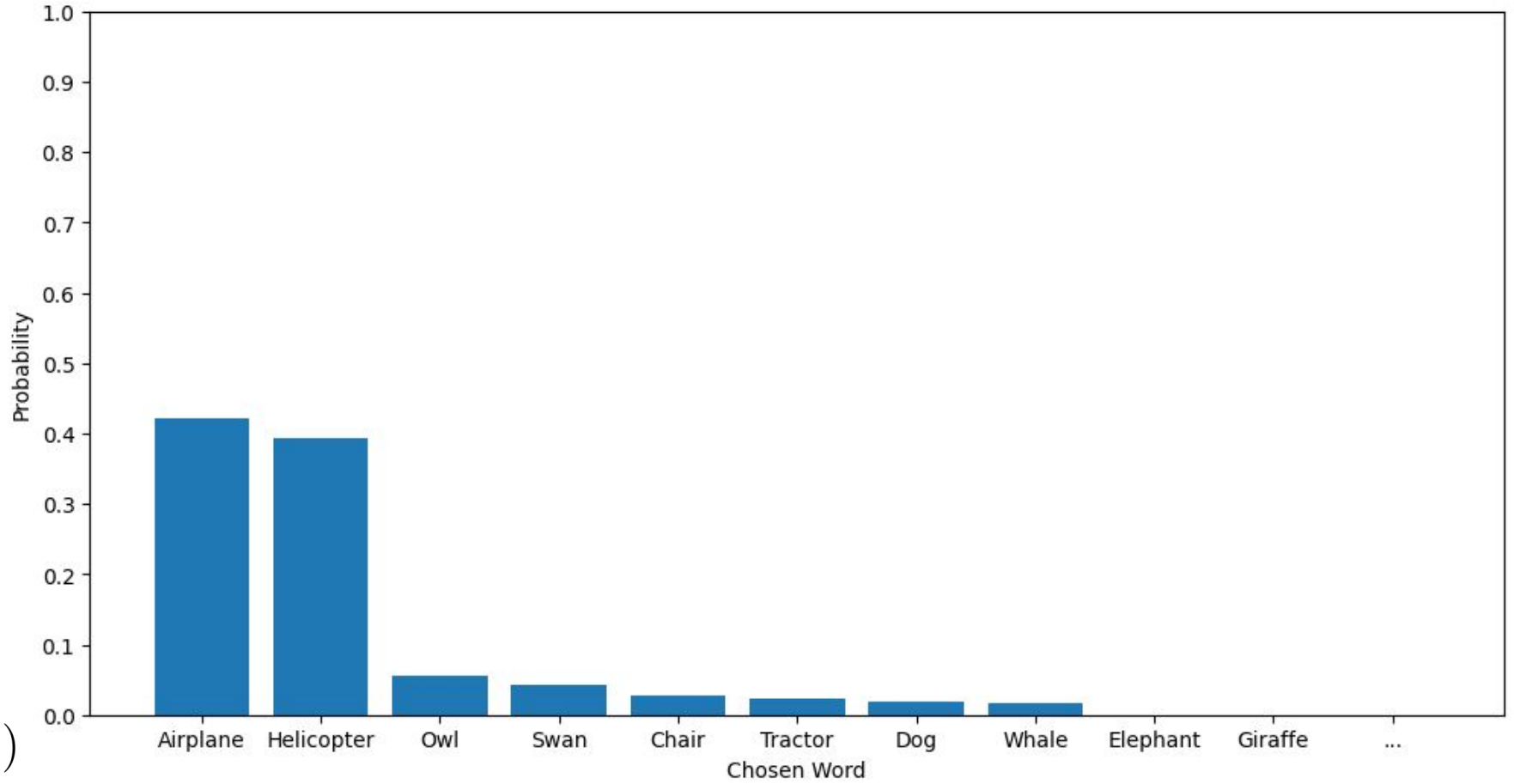
What about the non-uniform distribution?

x	P(x)	$P(x)\log_2(P(x))$
Airplane	0.421	-0.525
Helicopter	0.393	-0.530
Owl	0.056	-0.233
Fly	0.042	-0.193
Swan	0.028	-0.145
Tractor	0.022	-0.123
Dog	0.020	-0.111
Whale	0.017	-0.099



What about the non-uniform distribution?

x	P(x)	$P(x)\log_2(P(x))$
Airplane	0.421	-0.525
Helicopter	0.393	-0.530
Owl	0.056	-0.233
Fly	0.042	-0.193
Swan	0.028	-0.145
Tractor	0.022	-0.123
Dog	0.020	-0.111
Whale	0.017	-0.099



$$H(X) = -\sum_x P(x)log_2(P(x))$$

$$= -(-1.959)$$

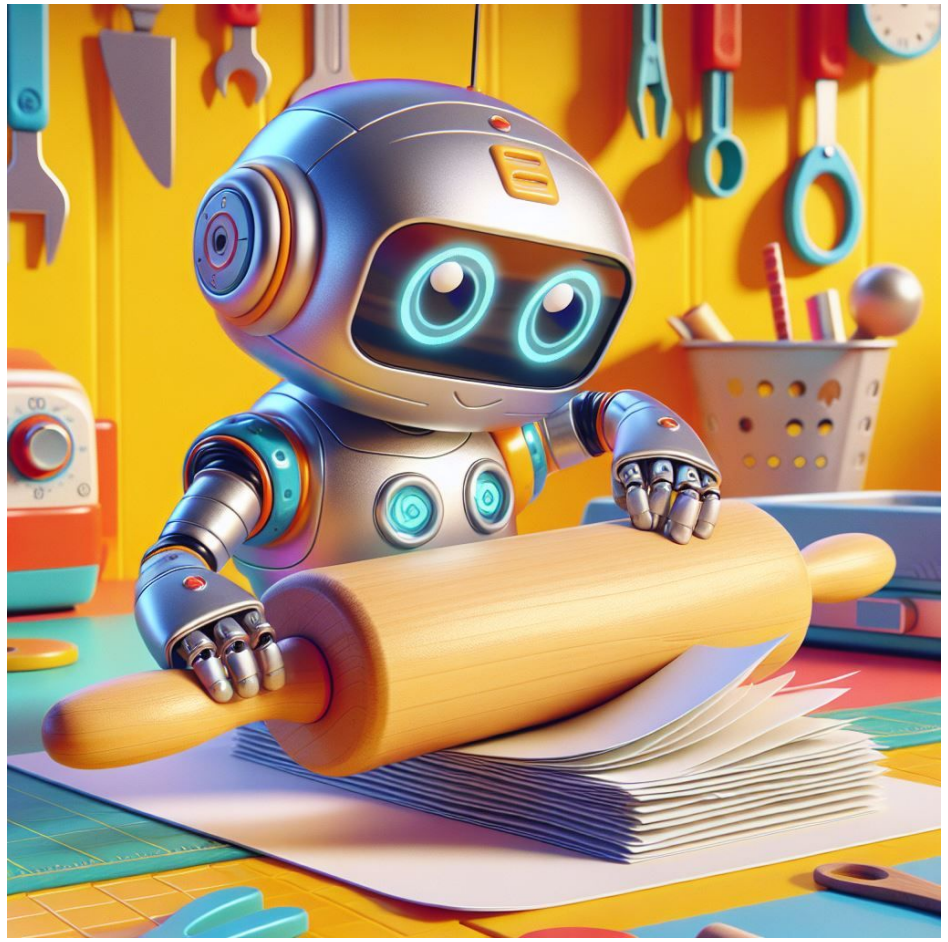
$$= 1.959$$

~2 questions (less than 3 for the uniform)

Entropy is a key idea in data compression

Data with fewer possible values, or non-uniform distribution

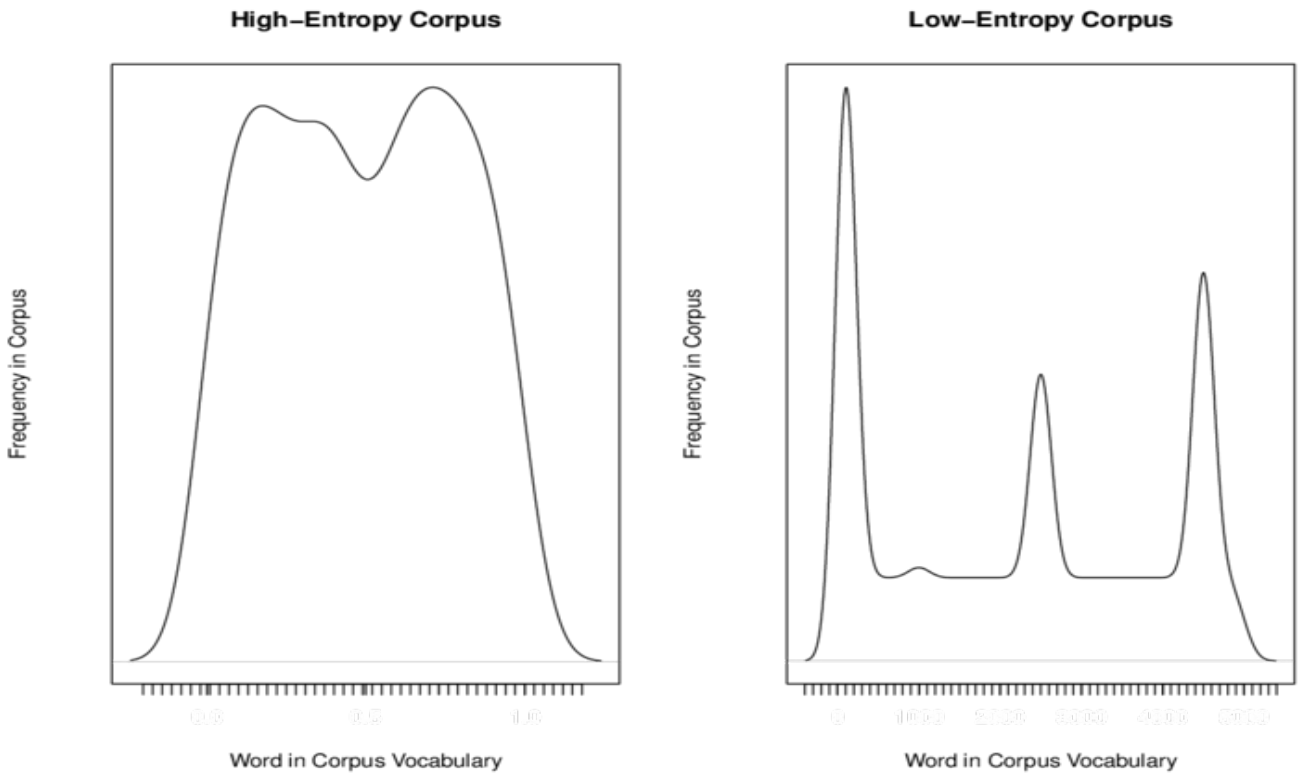
Fewer bits to encode it in a compressed form



created with Copilot and DALL-E 3

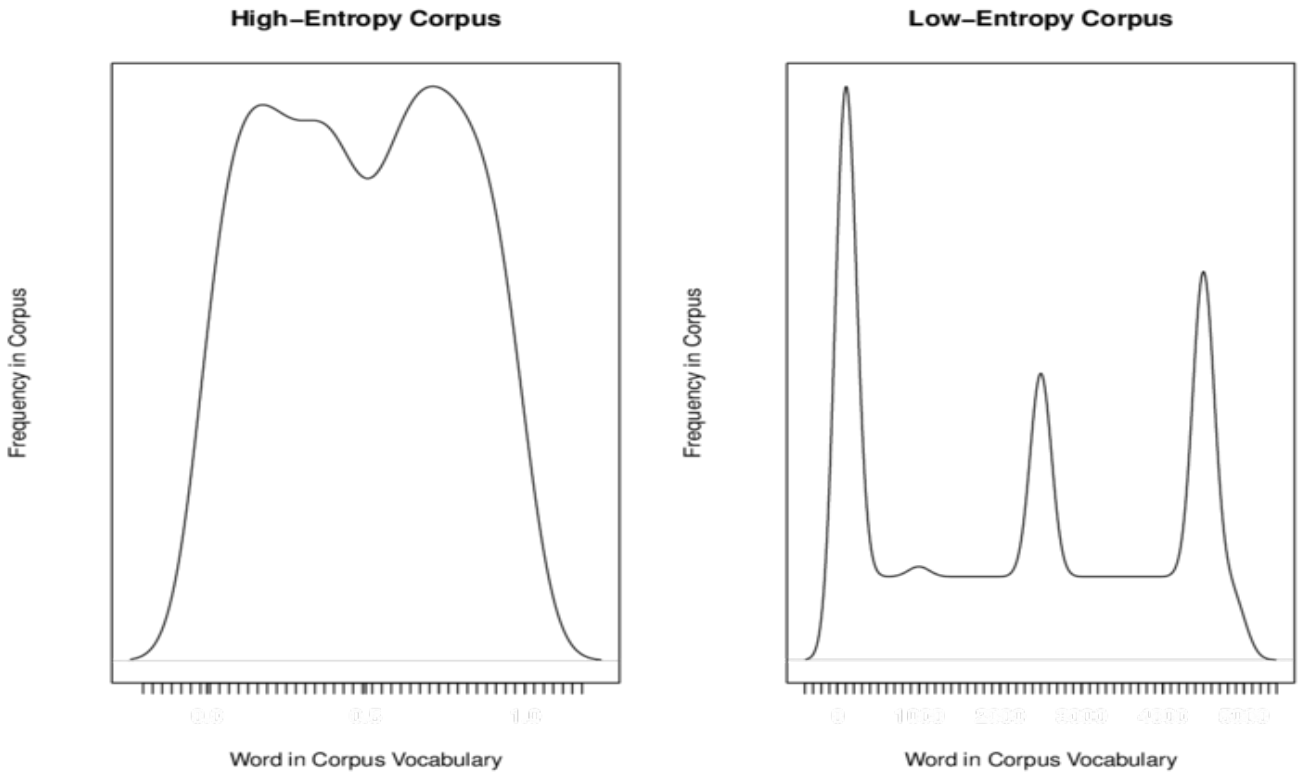
High vs low entropy for language models

- A uniform distribution has high entropy
  - Harder to predict a random draw from it.
- A peaked distribution has low entropy
  - Easier to predict a random draw from it.



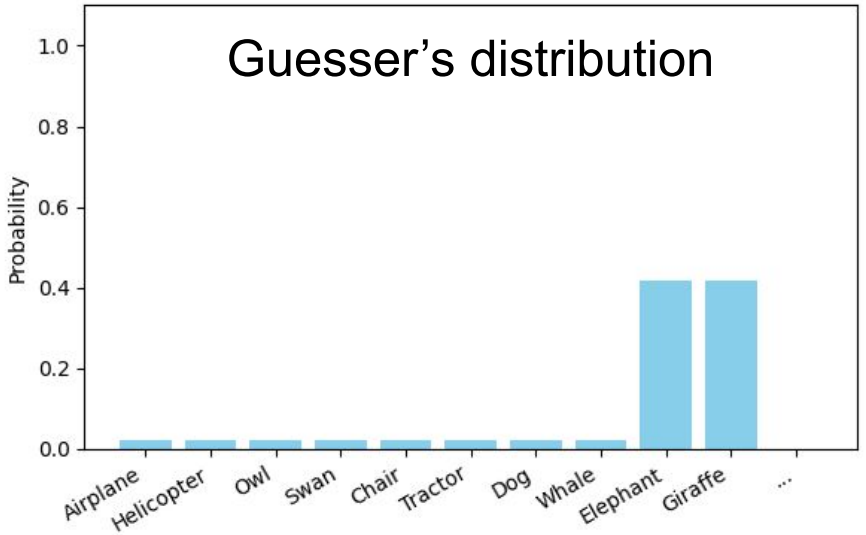
High vs low entropy for language models

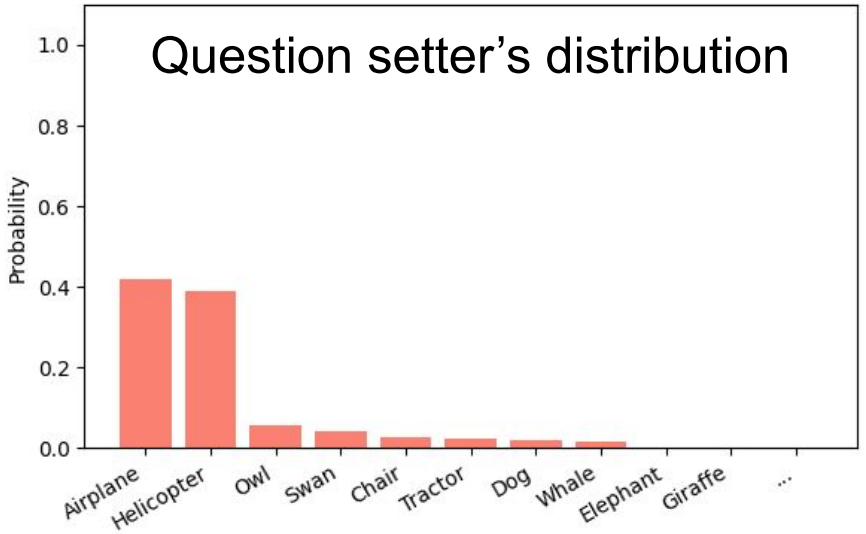
- A uniform distribution has high entropy
  - Harder to predict a random draw from it.
- A peaked distribution has low entropy
  - Easier to predict a random draw from it.



We know from Zipf's Law that natural language corpora will usually look like this

Playing twenty questions with a mismatched model

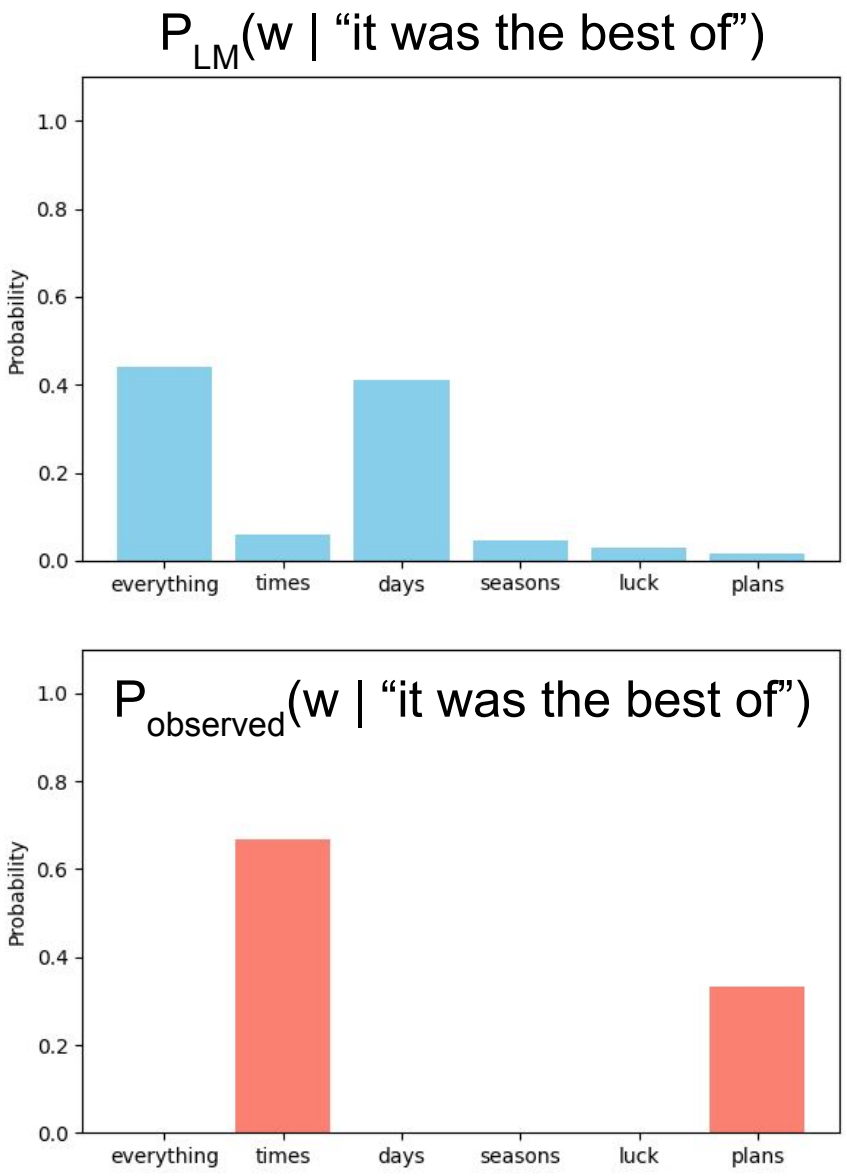




- You think I'll likely pick Elephant or Giraffe
  - **Model's distribution**
- I've changed my strategy and picking Airplane/Helicopter (and others more often)
  - **Observed distribution**
- How many Yes/No questions to get the answer from the **observed distribution** given this assumption of the **model's distribution**?

This is cross-entropy

Cross-entropy (for language models)



- Measure between two probability distributions
  1. The probability distribution from our language model
  2. The observed distribution of words (in our evaluation dataset)
- It is measured in bits
- Related to perplexity
  1. Reciprocal of our model’s probability of the observed data

$$\text{Perplexity} = 2^{\text{cross-entropy}}$$

Comparing language models with perplexity and cross-entropy

	Language Model A	Language Model B
Perplexity	32.7	16.4
Cross-entropy	5.03	4.04

Example perplexities on a held-out text set

- Perplexity often used to find a “better” language model
  - Specific to each dataset
  - Also, summarizing a system’s ability to model language by a single number is “limiting”
- Here, Model B seems to perform better for this dataset
  - Model A requires ~1 more bit to encode the observed data than Model B

Generating text with a language model

Generating Text with a Language Model

Recall that we can find the probability of a word in a sequence using a language model:

$$P(w_{k+1}|w_1, w_2, \ldots, w_k)$$

Generating Text with a Language Model

Recall that we can find the probability of a word in a sequence using a language model:

$$P(w_{k+1}|w_1, w_2, ..., w_k)$$

This property allows us to envision what a likely continuation of the text would look like! E.g., you can take the maximum:

$$\max_{x_1 \in V} P(x_1|w_1, w_2, ..., w_k)$$

“The dog ___” -> barks

Generating Text with a Language Model

Recall that we can find the probability of a word in a sequence using a language model:

$$P(w_{k+1}|w_1, w_2, ..., w_k)$$

This property allows us to envision what a likely continuation of the text would look like! E.g., you can take the maximum:

$$\max_{x_1 \in V} P(x_1|w_1, w_2, ..., w_k)$$ “The dog ___” -> barks

$$\max_{x_2 \in V} P(x_2|w_1, w_2, ..., w_k, x_1)$$ “The dog barks ___” -> at

Generating Text with a Language Model

Recall that we can find the probability of a word in a sequence using a language model:

$$P(w_{k+1}|w_1, w_2, ..., w_k)$$

This property allows us to envision what a likely continuation of the text would look like! E.g., you can take the maximum:

$$\max_{x_1 \in V} P(x_1|w_1, w_2, ..., w_k)$$ "The dog ___" -> barks

$$\max_{x_2 \in V} P(x_2|w_1, w_2, ..., w_k, x_1)$$ "The dog barks ___" -> at

$$\max_{x_3 \in V} P(x_3|w_1, w_2, ..., w_k, x_1, x_2)$$ "The dog barks at ___" -> …

This is a *greedy* generation strategy; the actual most likely sequence may be different. But it's too expensive to explore the whole space.

Picking the most likely next token doesn't work well

Previous Tokens	P(that)	P(the)	P(dog)	…
the, dog	0.3	0.01	0.02	
dog, that	0.03	0.4	0.02	
that, the	0.02	0.04	0.2	
…				

Example tri-gram language models

Always picking the most probable next token often creates boring outputs

Boring output:
the dog
the dog that
the dog that the
the dog that the dog
the dog that the dog that
the dog that the dog that the

Sampling from the next token probability distribution

Previous Tokens	P(that)	P(the)	P(dog)	…
the, dog	0.3	0.01	0.02	
dog, that	0.03	0.4	0.02	
that, the	0.02	0.04	0.2	
…				

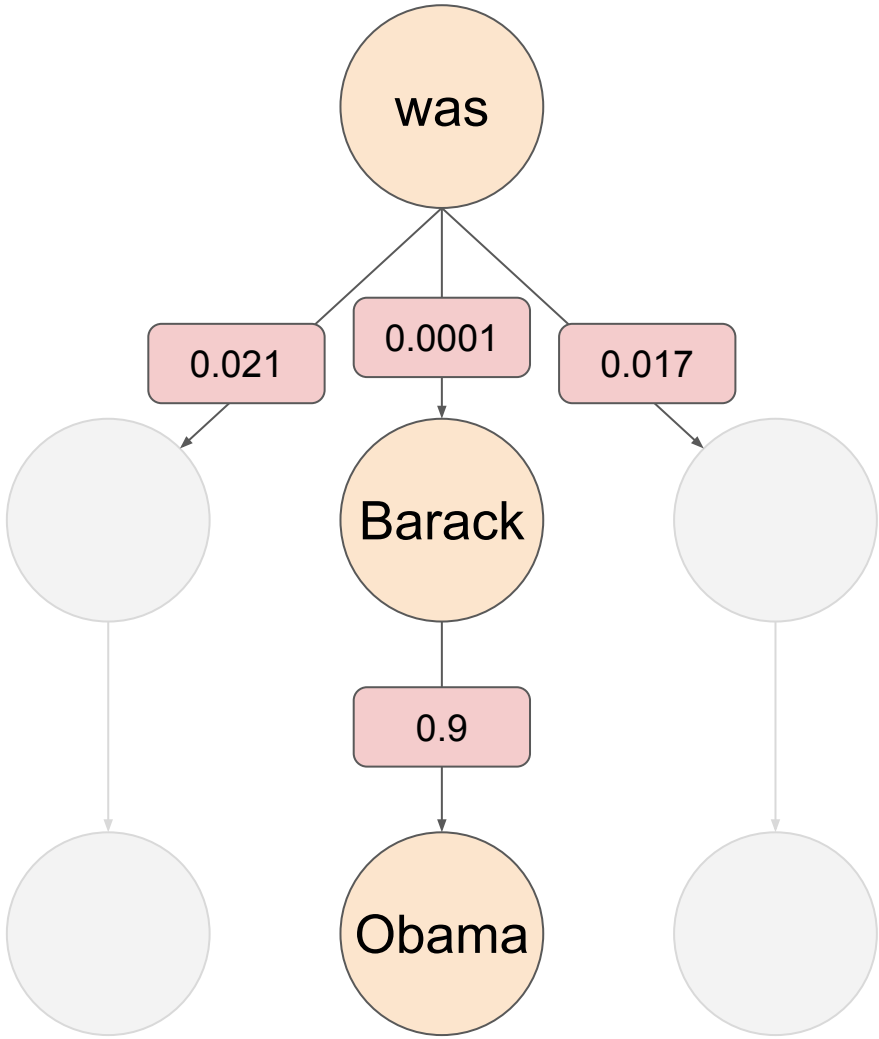
Example tri-gram language models

Instead, treat the probabilities as a distribution and sample from them.

The token with highest probability is still the likeliest output, but now other words have a chance.

One token at a time may not be ideal

- Some tokens are rare and unlikely to be generated
  - e.g. Barack
- However, as part of a longer phrase, they can be quite common and reasonable to be generated
  - e.g. Barack Obama
- This can be very common with names and phrases that start with uncommon words
  - e.g. esoteric knowledge, serendipitous encounter, etc
- So we need generation to look a bit further ahead

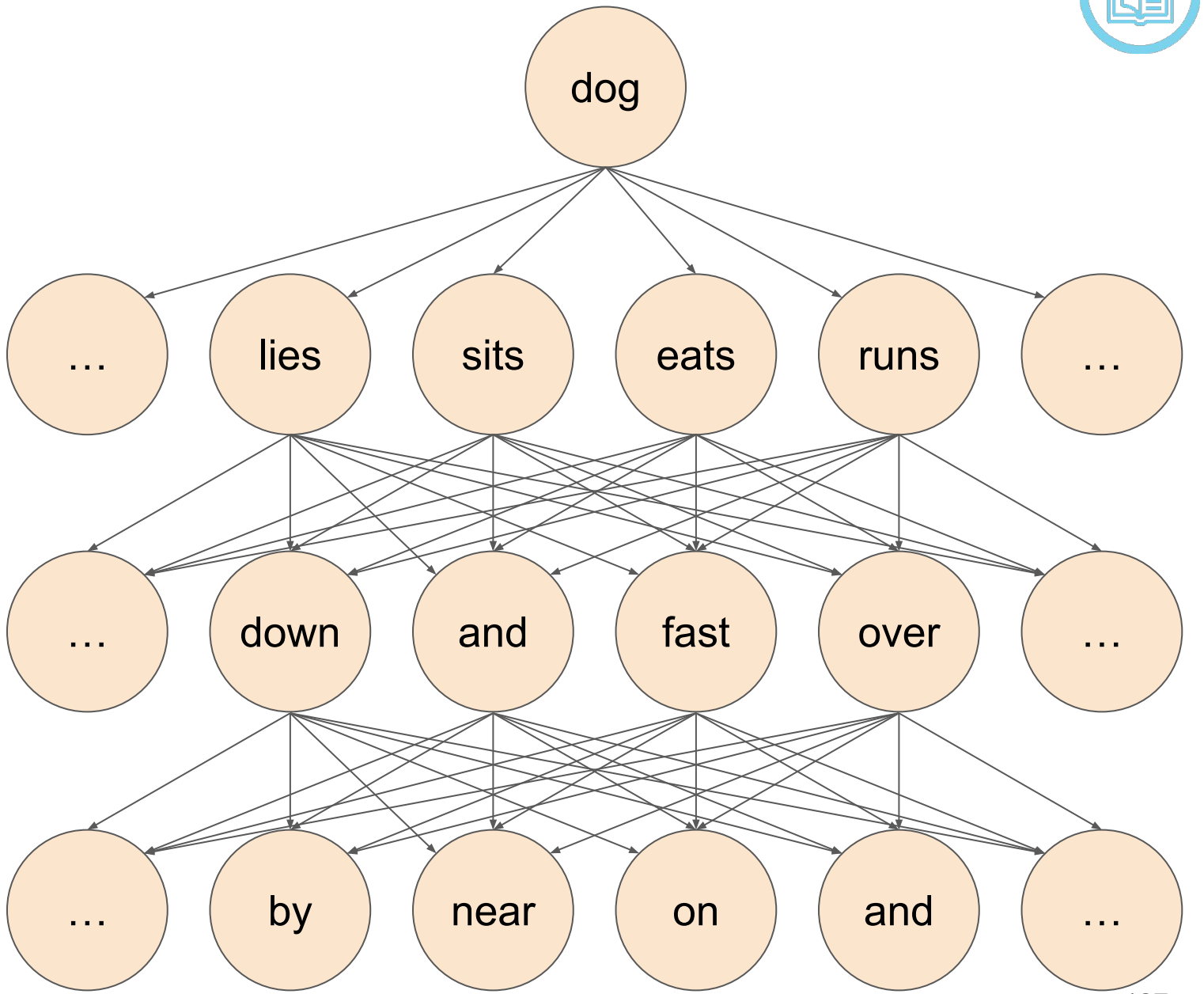


Beam search

Cannot search all possible generations (even for a small number of tokens) to find likeliest

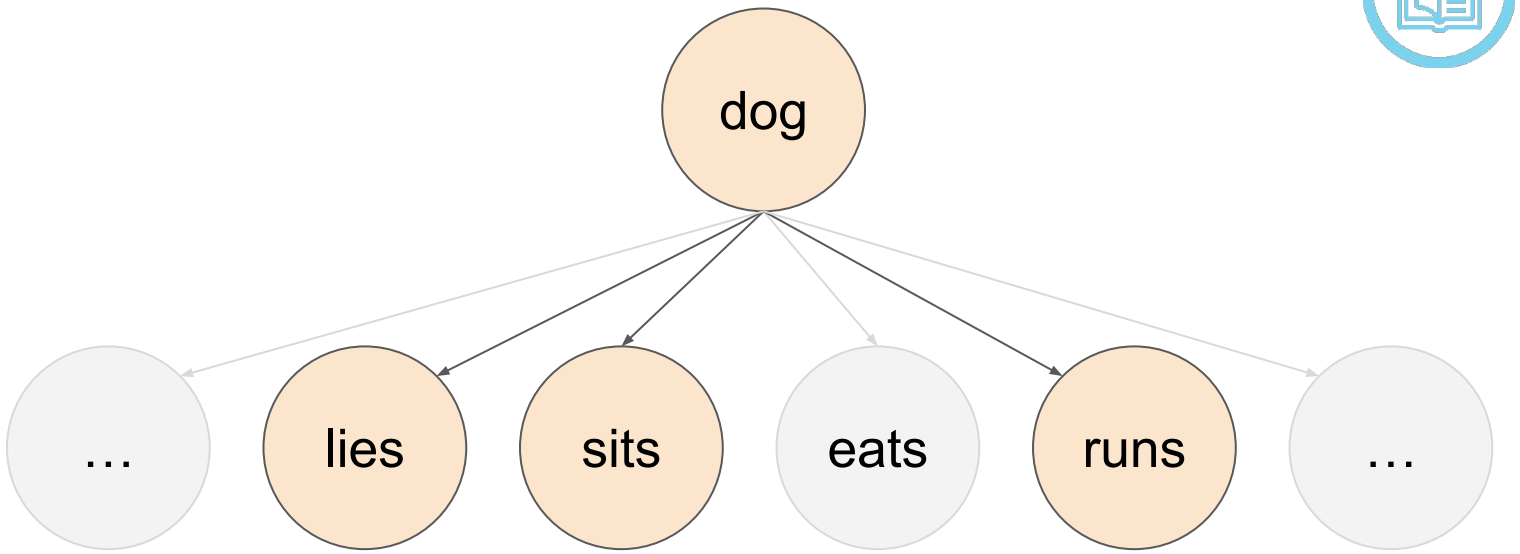
Idea: Grow several candidate short sequences (token-by-token) and discard ones that become too unlikely

The **beam width** parameter decides how many to keep



Beam search example

For beam width = 3, let's keep the top 3 tokens for the first step and grow those paths

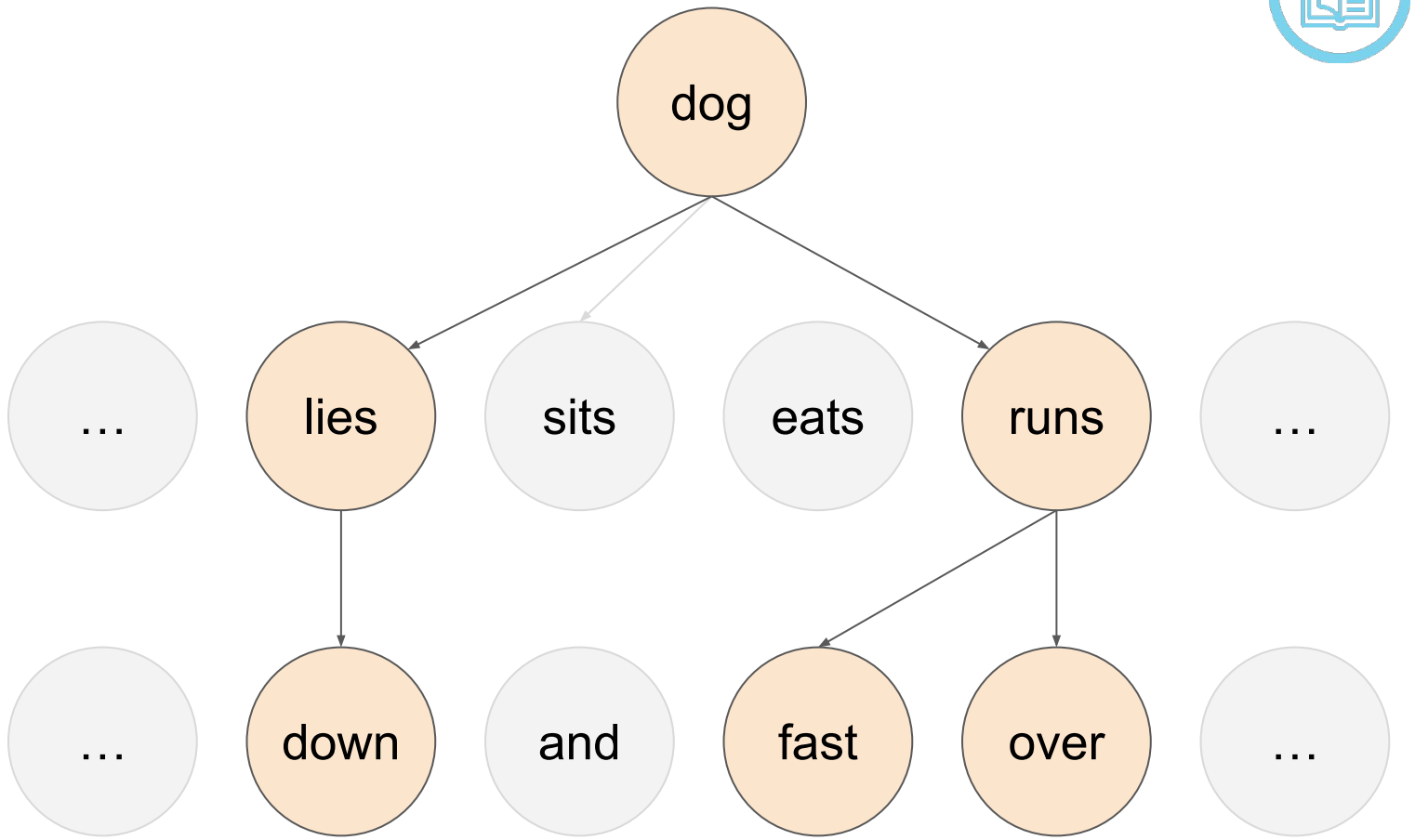


Beam search example

Next we add one extra token to the previous beams and keep the paths with the highest probability

Example probability:
P(“lies down”) = P(lies | down) * P(lies)

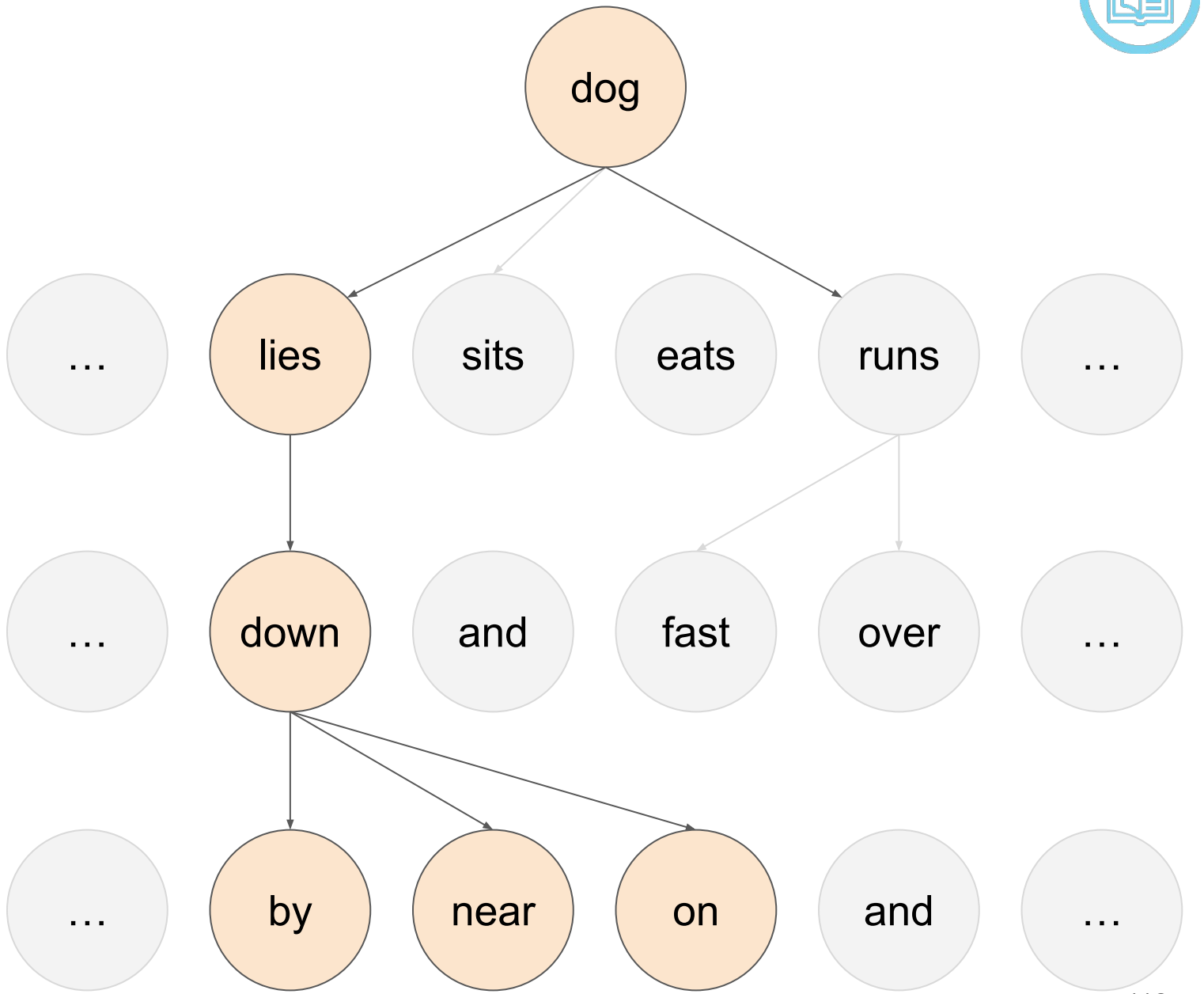
In this example, we actually discard the beams starting with “sits” as they had lower probability than “runs fast”, “runs over” and “lies down”



Beam search example

We try extending all of the previous paths and keep only the three most likely (for beam width=3).

And now we discard some of the existing ones and find that the highest probabilities are with sequences starting “lies down”.



More ideas for generating text

rabble obnixely yarborough lamprophony gobbledygook nebulize macaroni saccadic jentacular jaculiferous erf rabelaisian nudiustertian quadragenarian baboonery ragamuffin halfpace hent hullabaloo paleobotany gardyloo octothorpe valetudinarian xanthopsia sabbulonarium ratoon bibble tentigo dragoman cabotage gablock falsiloquence karozzin lackadaisical caffoy imago zoanthropy bacchanal dactylioglyph yellowplush taffeta bumfuzzle eellogofusciouhipoppokunurious agastopia flummox obrotund jargoon cacodemonomania whiffler decadarchy xertz kakorrhaphiophobia cattywampus uguisu erinaceous nainsook vaniloquence poppycock ulotrichous lollygag macrosmatic oxter ucalegon abatjour salopettes largiloquent yabba abaft xiphoid quomodocunquizing frankenfood walleteer vacherin naze kerfuffle quire impignorate meldrop finifugal whippersnapper quackle umbel jabberwock snickersnee hallux pauciloquent kennebecker xylocarp vainglory argle gadzooks paean elchee wakerife discombobulate mabble ickle taradiddle

- Avoid extremely rare words
  - top k: Filter to only top k words before sampling
  - top p: Filter to words with top X% of probability - also known as nucleus sampling
- Avoid repetition
  - Block repeated n-grams
  - Advanced **contrastive search** algorithm that blocks repeated text that looks similar to previous text

Summary

- Language modelling is the task of building predictive models.
  - Predict the next word in a sequence
  - Predict the probability of observing a sequence in a language.
- n-gram models
  - Markov independence assumption
  - Unigrams takes no context, Bigrams uses the prior token
- Smoothing
  - Zero probabilities cause real problems for many language model tasks
  - Smoothing offers way to give non-zero probabilities (often by stealing probability from elsewhere)
- Evaluating with perplexity and cross-entropy
  - Links to information theory (and guessing in twenty questions)
- Text generation
  - One word at a time, with or without sampling
  - Beam search