From last time:

LMs measure:

↳ $P(w_1, w_2, \ldots, w_n)$ ⇒ prob of a text

↳ $P(w_n \mid w_1, w_2, \ldots w_{n-1})$ ⇒ cond. prob of next word given prefix

n-gram models:

↳ cheap way to get these probabilities by counting/dividing over a training dataset

Perplexity

↳ a metric used to eval LMs

↳ PPL is exponentiated avg. neg log likelihood

↳ ideal case:

$$\left.\begin{array}{l} \text{PPL (train) is low} \\ \text{PPL (test) is also low} \end{array}\right\} \text{generalization}$$

↳ bad:

PPL (train) is low
PPL (test) is high } overfitting

↳ to compute PPL, you need an existing dataset of text

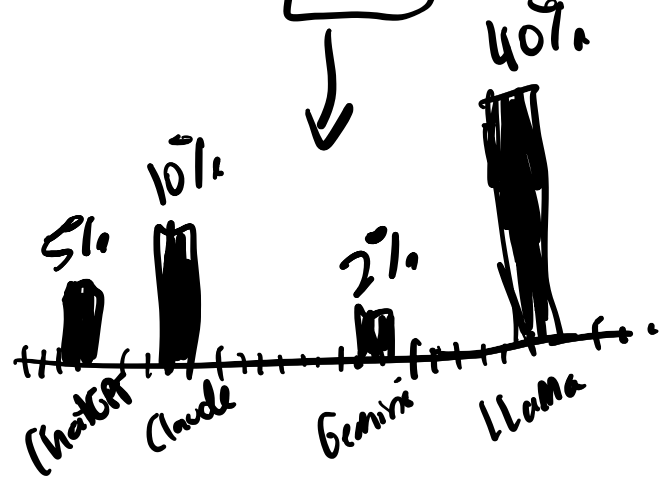
how do we use a LM to generate new text?

↳ this process is called decoding from the LM

↳ part of the higher-level stage known as inference

My favorite LLM is → $\boxed{\text{LM}}$

$P(w_i \mid \text{prefix})$
distribution over vocabulary



deterministic

↳ greedy decoding: choose $\arg\max_{w_i} P(w_i \mid \text{prefix})$

↳ sampling word w_i from this cond. distribution

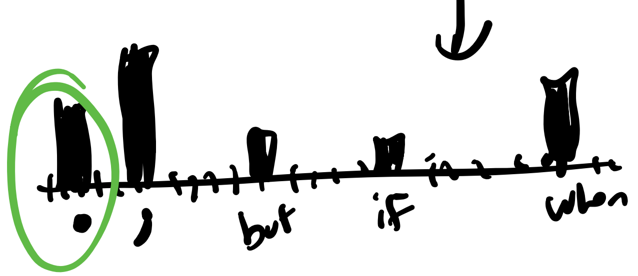
↳ 40% of the time choose LLaMa

↳ 5% we choose ChatGPT

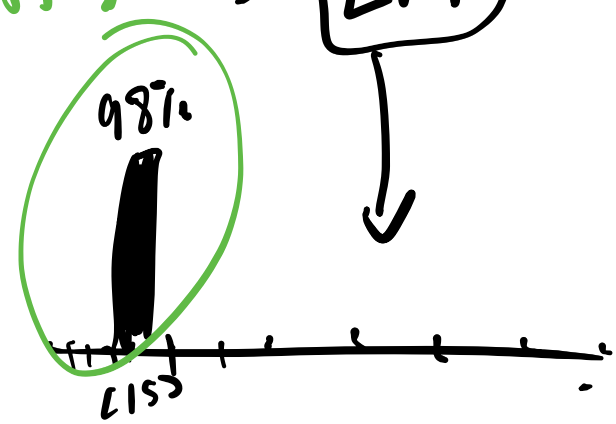
⋮

after generating a word/token, we add it to our prefix and then generate the next word

My favorite LLM is ChatGPT → LM



My fav. LLM is ChatGPT . → LM



My fav. LLM is ChatGPT . </s>
<eos>

↳ autoregressive decoding

neural language models:

↳ move away from "count + divide"

↳ core concepts:

  ↳ forward propagation

    ↳ how we go from input seq to output prob. dist

  ↳ backpropagation

    ↳ how we "train" the model to make better predictions

    ↳ gradient descent

↳ embedding
↳ composition

} today

embeddings:

↳ in an n-gram model, "movie" and "film" are treated as completely different

↳ "one-hot" representation of a word

$$\text{movie} = \langle \overbrace{0, 0, 0, \ldots, \underset{\substack{\uparrow \\ \text{movie}}}{1}, 0, 0, \ldots}^{\text{V-dimensional}} \rangle$$

$$\text{film} = \langle 0, 0, \underset{\substack{\uparrow \\ \text{film}}}{1}, \ldots \underset{\substack{\uparrow \\ \text{movie}}}{0}, 0, 0, \ldots \rangle$$

dot product

$$\vec{\text{movie}} \cdot \vec{\text{film}} = 0 \ \} \text{ orthogonal}$$

ideally, we want a vector space where words/phrases/docs w/ similar meanings have similar representations

↳ what if we switch from sparse V-dim vector to a dense, low-dim vector

↳ $d = \{50, 100, 768, 1024\}$ common values

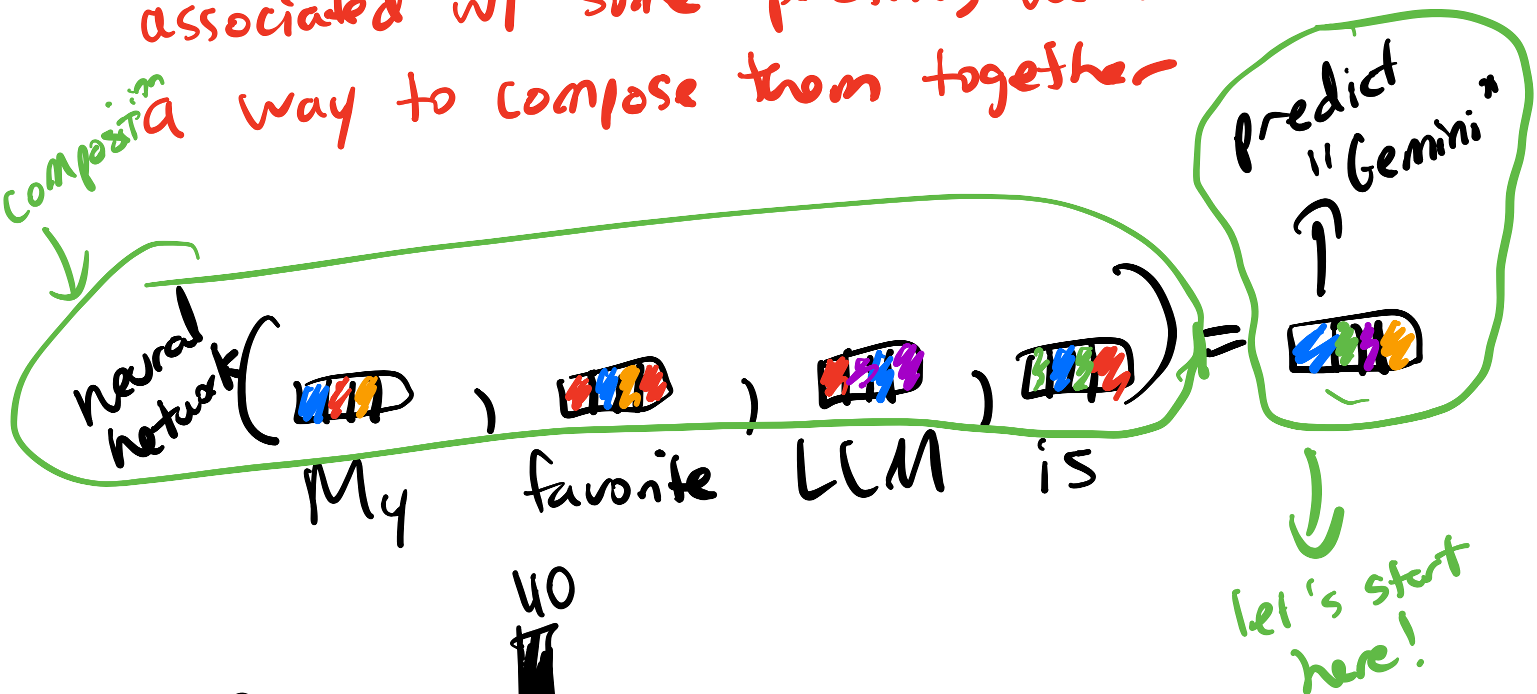
$$\text{movie} = \langle 0.3, -1.4, 5.8 \rangle$$

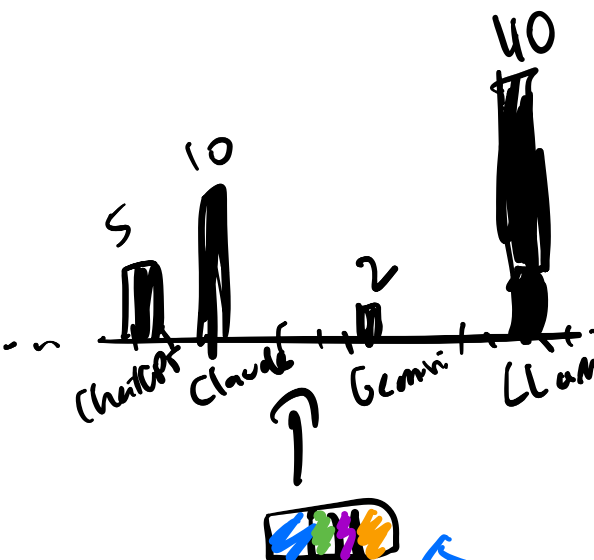
$$\text{film} = \langle 0.2, -1.8, 6.3 \rangle$$

$$\text{zebra} = \langle 7.9, 3.2, -17.5 \rangle$$

↳ these are examples of ==word embeddings==

now, given a sequence of word embs associated w/ some prefix, we need a way to compose them together





← output: V-dim prob dist over next word
↳ sum to 1
↳ non-negative

← d-dim prefix vector

Softmax layer:

↳ convert a d-dim dense vector to a V-dim prob. dist example: output vocab = { ChatGPT, Claude, Gemini, Llama }

given prefix vector  $\leftarrow x$

$$x = \langle -2.3,\ 0.9,\ 5.4 \rangle$$

$$d = 3,\ V = 4$$

1. project x to 4-d using a weight matrix W

↳ weight matrices contain parameters that we update in backprop to change the next word dist.

$$W = \begin{Bmatrix} 1.2 & -0.3 & 0.9 \\ 0.2 & 0.4 & -2.2 \\ 8.9 & -1.9 & 6.5 \\ 4.5 & 2.2 & -0.1 \end{Bmatrix}$$

this is a 4×3 matrix and x is 3 dimensional

↳ matrix-vector product

Wx is a 4-d vector

$$W = \begin{bmatrix} 1.2 & -0.3 & 0.9 \\ 0.2 & 0.4 & -2.2 \\ 8.9 & -1.9 & 6.5 \\ 4.5 & 2.2 & -0.1 \end{bmatrix} \begin{array}{l} \rightarrow \text{ChatGPT} \\ \rightarrow \text{Claude} \\ \rightarrow \text{Gemini} \\ \rightarrow \text{LLaMa} \end{array}$$

feature weights for

$x = \langle -2.3, 0.9, 5.4 \rangle$

each dim of x intuitively corresponds to a "feature" of the prefix

$Wx = \langle 1.8, -11.9, 12.9, -8.9 \rangle$

enter the **softmax fn**

$$\text{softmax}(x) = \frac{e^{x}}{\sum_i e^{x_i}}$$

x is a vector

x_j is dimension j of x

$$\text{Softmax}(Wx) =$$

$$\langle 1.5 \cdot 10^{-5},\ 1.6 \cdot 10^{-11},\ 0.999984,\ 3.4 \cdot 10^{-10} \rangle$$

P(ChatGPT | ...)

P(Claude | prefix)

P(Gemini | prefix)

P(LLama | prefix)