Language Models

Task: generating text

Ex: Input: "Once upon a time, in a land far away,"
Output: "there lived a wise old dragon who loved to read books."

Ex: Input: "Why did the chicken cross the road?"
Output: "To get to the other side."

Ex: How ChatGPT works

Input:

System prompt

"Your are an AI assistant trained to provide accurate, concise, and helpful responses to user inquiries."

+

"Why did the chicken cross the road?"

Output: "To get to the other side."

We will only study auto-regressive models

<u>Ex</u>: ChatGPT, Gemini, Claude

<u>Auto-regressive</u>: generates output sequentially by using its previous outputs and inputs

<u>Ex</u>: Input: "Why did the chicken cross the road?"

Output: "To"

Input: "Why did the chicken cross the road?
To"

Output: "get"

Input: "Why did the chicken cross the road?
To get"

Output: "to"

⋮

Input: "Why did the chicken cross the road?
To get to the other side."

Output: "<EOS>" End Of Sequence token

We want a model $f : \mathcal{X} \Rightarrow \mathcal{Y}$

$x \in \mathcal{X}$ is a sequence of ~~words~~ tokens

$f(x) \in \mathcal{Y}$ is the next ~~word~~ token

$\mathcal{Y} = \{$ all words + punctuation + <EOS> + <PAD> $\}$

$= \{1, \ldots, K\}$ Vocabulary $|\mathcal{Y}| = K$

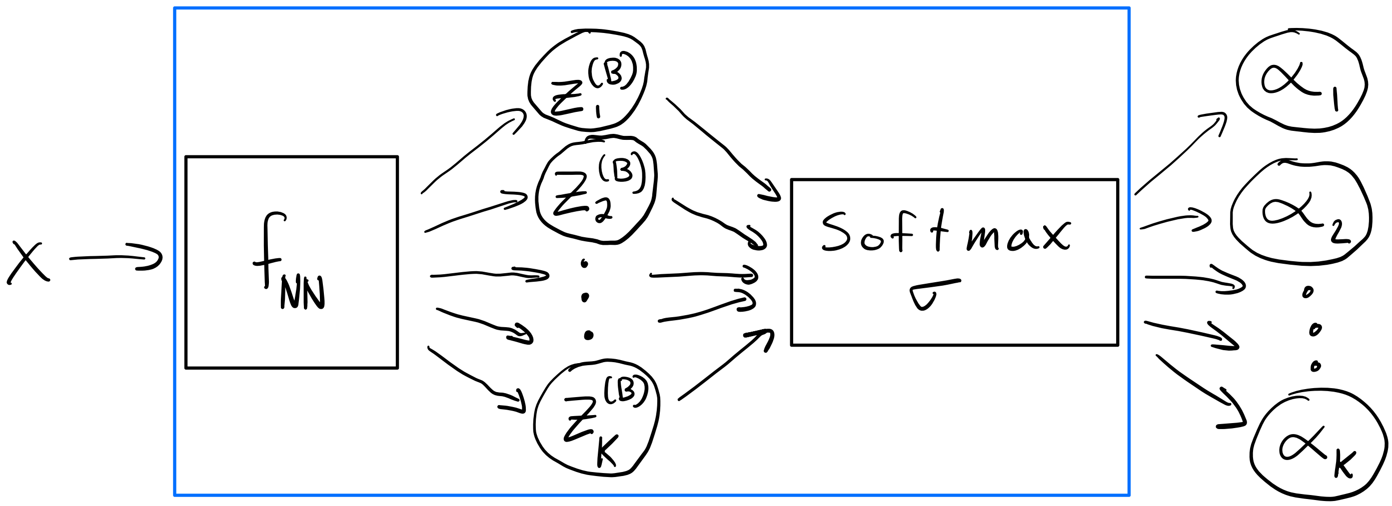
Discrete labels make optimization hard

We want a model $f_{prob} : \mathcal{X} \Rightarrow \mathcal{Y}_{prob}$ $= [0,1]^K$

$x \in \mathcal{X}$ is a sequence of tokens

$f_{prob}(x) \in \mathcal{Y}_{prob}$ is a vector of K probabilities with $d^{(B)} = K$

We can use a NN $f_{NN}(\vec{x})$ with $h^{(B)}(z) = z$ and then apply softmax to get a probability



$f_{prob}(\vec{x}) = \sigma(f_{NN}(x))$

$$\sum_{q=1}^{K} \alpha_q = 1 \quad , \quad \alpha_q \in [0,1]$$

How do we represent a sequence of tokens s as a vector $\vec{x} \in \mathbb{R}^{d+1}$?

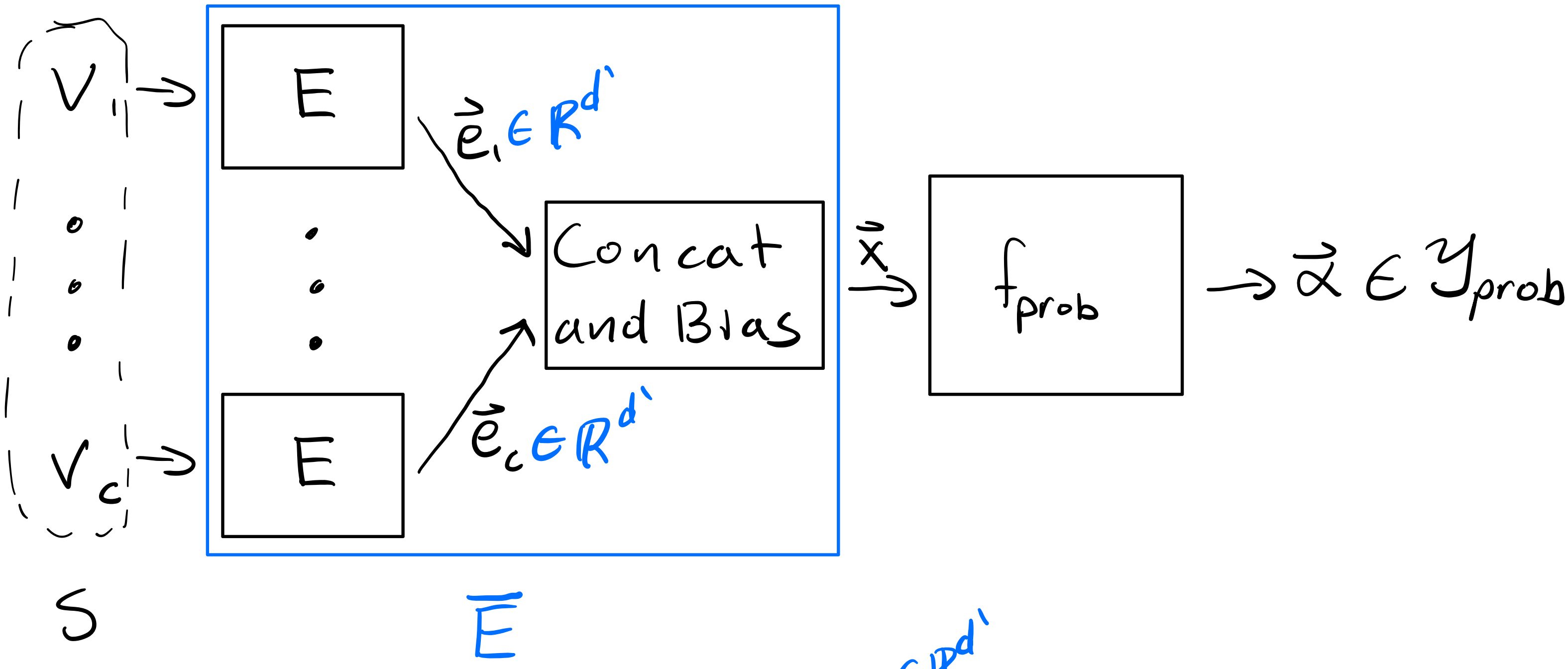
$$s = (v_1, \ldots, v_c) \in \mathcal{Y}^c$$

Ex: " Why did the chicken"

$$s = (\text{"Why"}, \text{"did"}, \text{"the"}, \text{"chicken"}) \in \mathcal{Y}^4$$

$E : \mathcal{Y} \to \mathbb{R}^{d'}$ embedding function

assume it is given



$$\bar{E}(s) = \bar{E}(v_1, \ldots, v_c) = (1, \overset{\in \mathbb{R}^{d'}}{E(v_1)}, E(v_2), \ldots, E(v_c))^T = \vec{x} \in \mathbb{R}^{d+1}$$

$$d = cd'$$

context length

If a sequence is longer than c tokens take the last c tokens

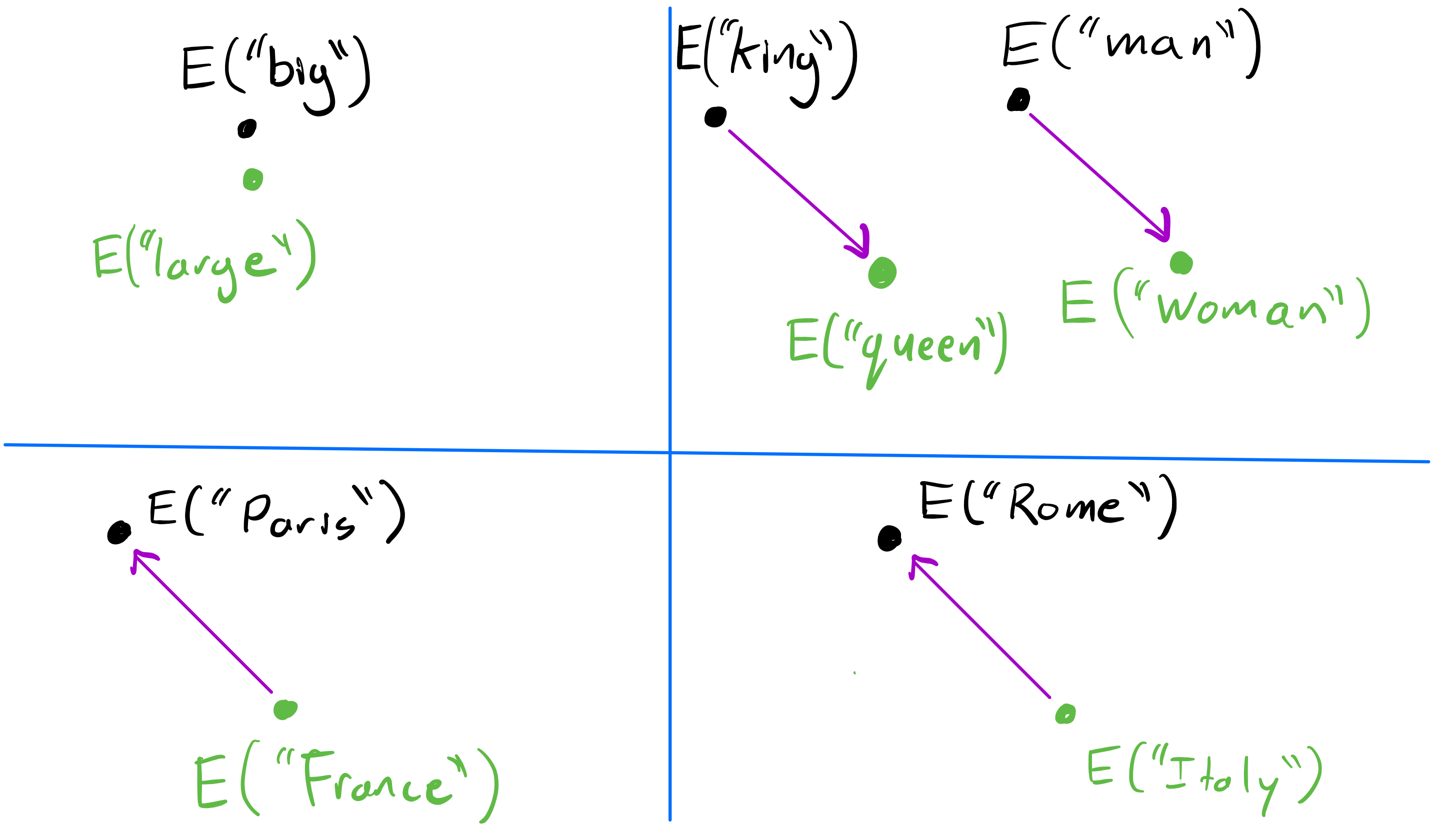
To handle sequences shorter than c tokens a padding token "<PAD>" is added

<u>Ex:</u> $c = 3$

S = ("Why", "did") ⇒ ("<PAD>", "Why", "did")

Embeddings

Ex: $d' = 2$ $\qquad E : \mathcal{Y} \to \mathbb{R}^{d'}$



Creating a Dataset

$$S = (V_1, V_2, V_3, \ldots, V_c, V_{c+1}, \ldots, V_{n+1})$$

$$S_1 = (\overbrace{\langle PAD\rangle, \ldots, \langle PAD\rangle}^{c-1}, V_1), \quad y_1 = V_2$$

$$S_2 = (\overbrace{\langle PAD\rangle, \ldots, \langle PAD\rangle}^{c-2}, V_1, V_2), \quad y_2 = V_3$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$S_c = (V_1, \ldots, V_c), \qquad y_c = V_{c+1}$$

$$\vdots$$

$$S_n = (V_{n-c+1}, \ldots, V_n), \qquad y_n = V_{n+1}$$

<u>Ex</u>: $C=2$, $S = (\text{"Why"}, \text{"did"}, \text{"the"})$

$$S_1 = (\text{"<PAD>"}, \text{"Why"}), \quad y_1 = \text{"did"}$$

$$S_2 = (\text{"Why"}, \text{"did"}), \quad y_2 = \text{"the"}$$

$$\vec{x}_1 = \bar{E}(s_1),\ \vec{x}_2 = \bar{E}(s_2),\ \ldots,\ \vec{x}_n = \bar{E}(s_n) \in \mathbb{R}^{d+1}$$

$$\vec{y}_1 = \mathrm{onehot}(y_1) \in \{0,1\}^K \subseteq [0,1]^K$$

$$\vdots$$

$$\vec{y}_n = \mathrm{onehot}(y_n) \in \{0,1\}^K$$

$$D = ((\vec{x}_1, \vec{y}_1), \ldots, (\vec{x}_n, \vec{y}_n))$$

ERM Learner:

$$\mathcal{A}(D) = \hat{f}_{prob} = \underset{f \in \mathcal{F}}{\arg\min}\ \hat{L}(f)$$

$\mathcal{F} = \{ f \mid f : \mathcal{X} \to \mathcal{Y}_{prob}$ where $f = \sigma(f_{NN})$ and f_{NN} is a NN with a fixed architecture $\}$

ℓ is multiclass cross-entropy loss

For a new sequence $s = (v_1, \dots, v_c) \in \mathcal{Y}^c$

The Language model is:

$$\hat{f}_{LM}(s) = \underset{q \in \{1, \dots, k\}}{\arg\max}\ \hat{y}_q \qquad \hat{\vec{y}} = \hat{f}_{prob}(\vec{E}(s))$$

Notes

- Transformers are the NN architecture currently used
- Embedding E can be learned
- Vocabulary can use characters or sub-words instead of words "tokenization"
- Most probable word is not always chosen, instead can sample based on probabalities
- "Large" language models (LLM) means a NN with a lot of weights

Ex: GPT-3 has 175 Billion weights

GPT-5 Publically estimated to have ≈ 2 Trillion weights

The brain has ≈ 100 trillion connections between neurons