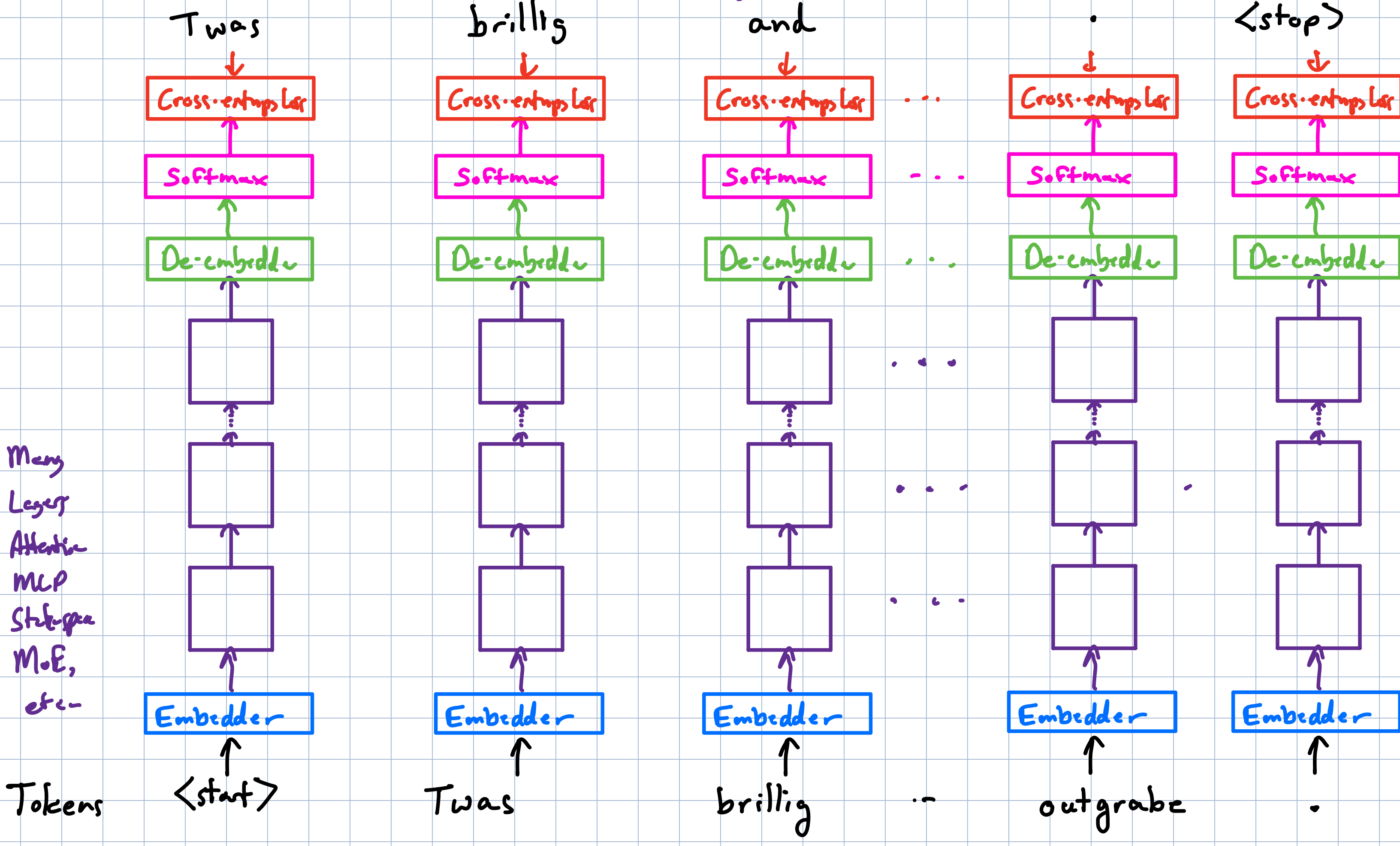
Today: ICL/Prompting
Fine-tuning
Parameter-efficient Fine-tuning

Announce: Start on Projects
Compute Offered
(NAIRR & Tinker)

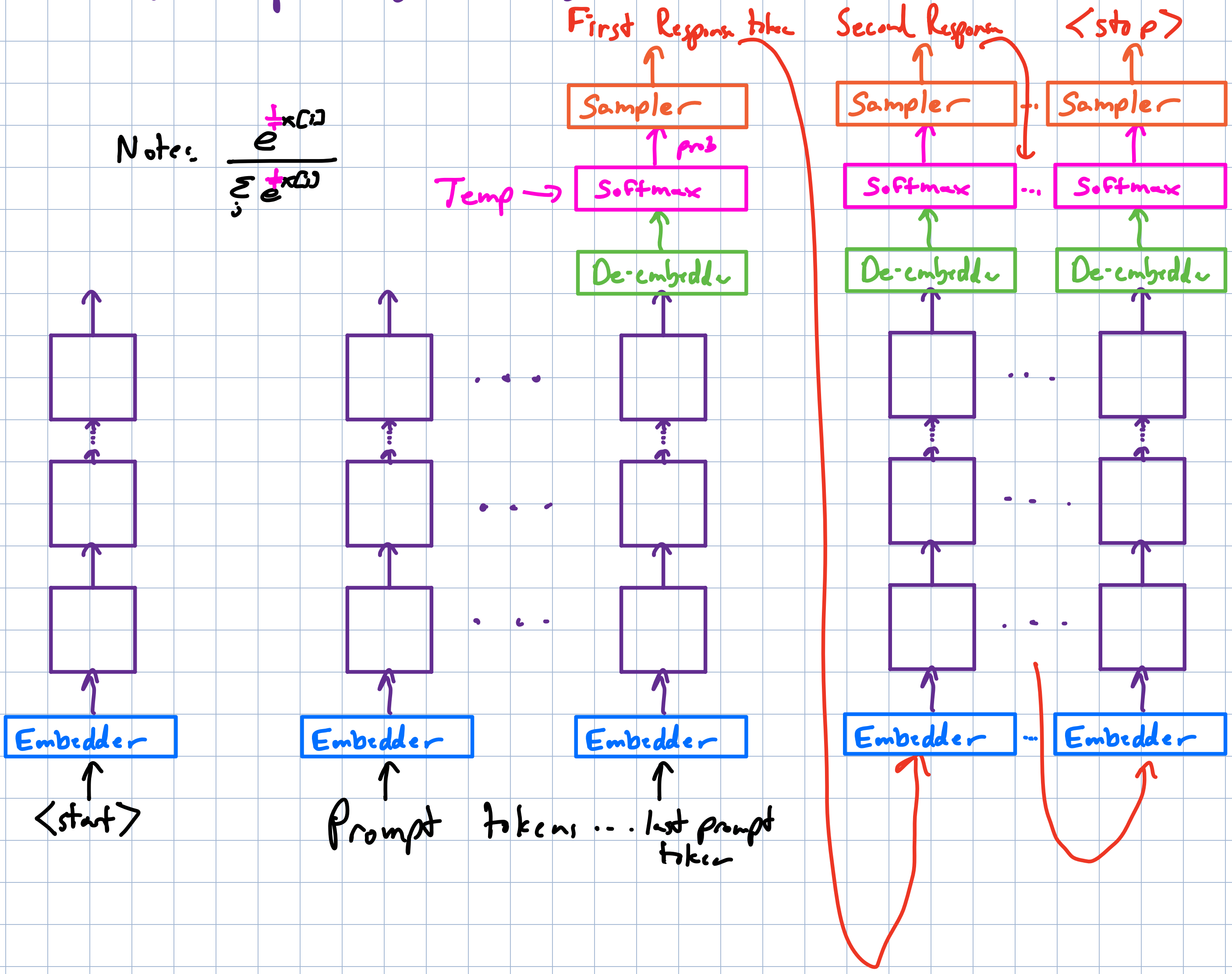
GPT-style models... Recall Pretraining: (Next token Prediction)



Data mix matters.

vs Inference... (Autocomplete Style Autoregressive generation)

Note: $\dfrac{e^{\frac{1}{T}x[i]}}{\sum_j e^{\frac{1}{T}x[j]}}$



Some history:

GPT-1 (2018) : Next-token prediction gives good embeddings

GPT-2 (2019) : With bigger models, AR completions result in natural prose/poetry and the model knows facts about the world that can be elicited by the right prompt. ("zero-shot learning")

Example: The capital of France is Paris.

GPT-3 (2020) : With even bigger models, also get in-context learning. ("Few shot" learning)

Example: Red : Ruby ; Blue : Saphire

vs

Red : Raspberry ; Blue : Blueberry.

Revolutionary... Can now solve a learning problem with a few examples without any fine-tuning or gradient descent. Pure Prompting.

Challenge: all training data must fit in context.

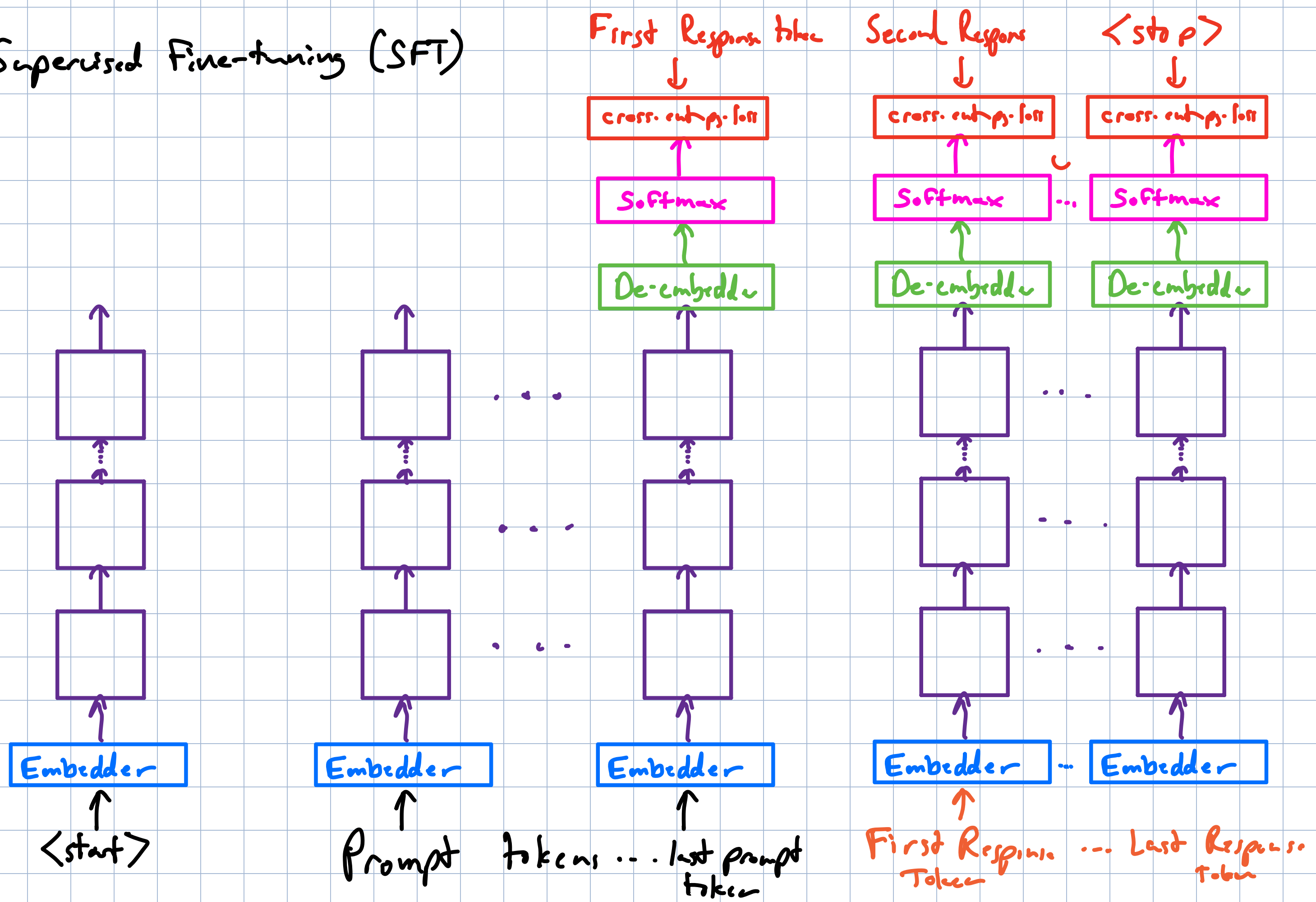
Other natural questions: (A) How can we find better prompts?

(B) How can we make more usefully promptable models?

Start with (B): Critical Paper: Instruct GPT (2021)

Fine-tuning For Instruction Following or Question Answering...

Supervised Fine-tuning (SFT)



In 2023, Llama model released. → Alpaca (showed instruction-following examples are key)

Dolly → showed it was enough examples that mattered.

Step Back: Fine-tuning or using a pretrained model for a new task:

0) Pure prompting — No gradient-descent steps

1) Treat as embedding — a feature extractor. Do classical ML to train a separate model
   Also called "linear probing"

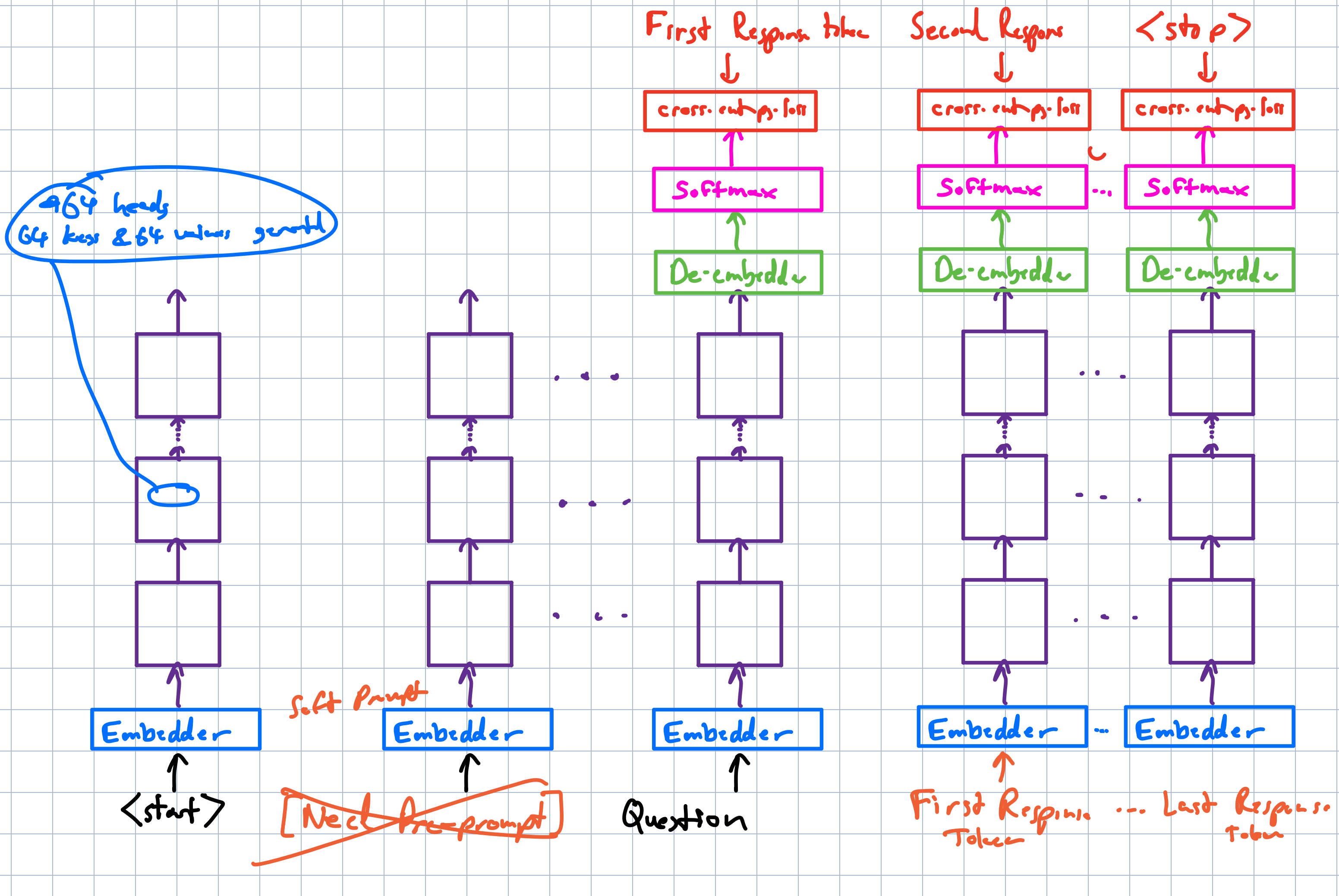
2) Full fine-tune — adjust the weights of the pretrained model

Return to (A): How to get better prompts?

Aside: If you want to do this, and only have API access to the model, tools like DSPy are what you want to use.
New algorithms inside like GEPA (this summer) that use LLMs to reflect on the execution of prompts (related to TextGrad in spirit) "textual gradients"

Genetic Algorithm style...

Assume we have model...



To be precise: If Pre-prompt is length 100

Call it $\langle special_1 \rangle \cdots - - - \langle special_{100} \rangle \leftarrow$

when the Embedder has 100 new columns added for

These 100 new columns are learnable parameters

Soft Prompts & Prefixes work pretty well.

And far far fewer parameter than the entire model.

example for a soft-prompt: Embedding 4096 (Llama 3, Olmo 2, etc.)

Prompt-length 100

Total Params: 40K vs 7 or 8 Billion

Soft Prefix (Just the K-V cache) Assume 32 layers

$$100 \times (\underset{\text{keys}}{4096} + \underset{\text{values}}{4096}) \times \underset{\text{layers}}{32} = 262K \times 100 = 26M$$