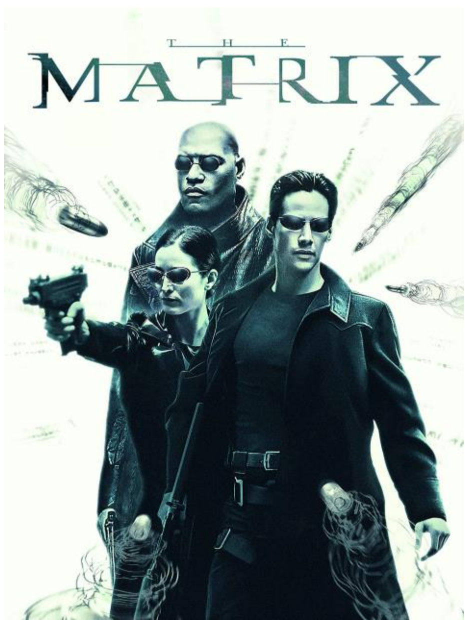
CentraleSupélec

Generative AI for Risk and Reliability

Lect 2: How GPT works and how to improve its performance



Zhiguo Zeng, Professor,

Chaire on Risk and Resilience of Complex Systems,

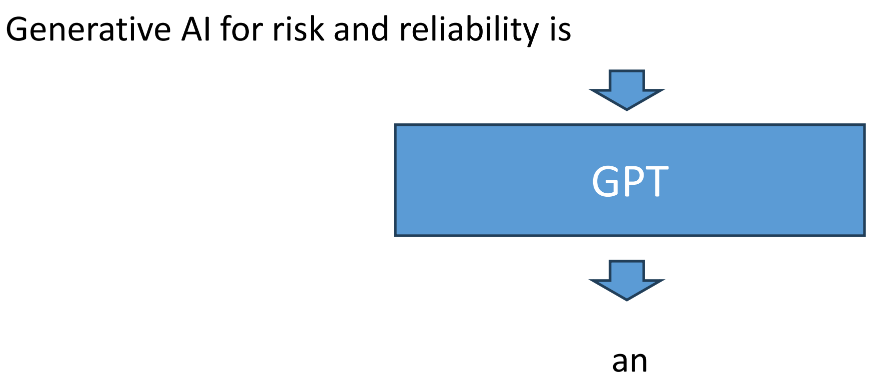
CentraleSupélec, Université Paris-Saclay

zhiguo.zeng@centralesupelec.fr

06/December/2024

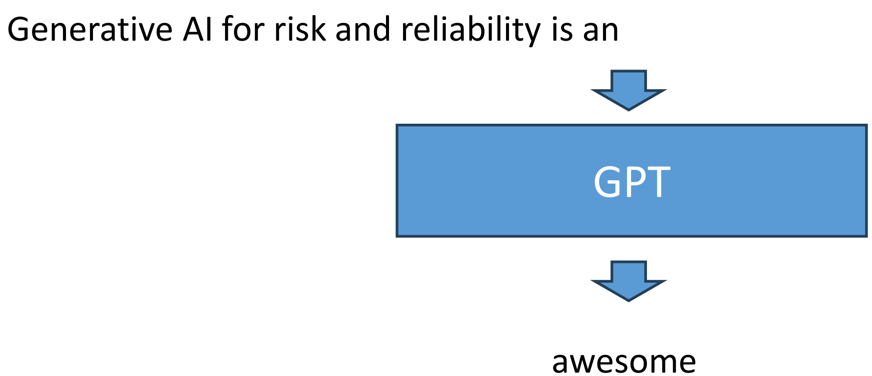
What GPT does

- GPT: Generative Pre-trained Transformer
- The basic task: Next token prediction
  - Recursively
  - Basis for designing AI assistant like ChatGPT



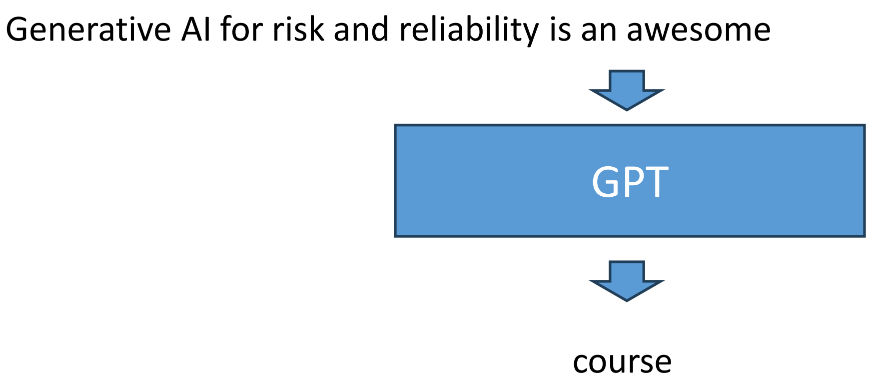
What GPT does

- GPT: Generative Pre-trained Transformer
- The basic task: Next token prediction
  - Recursively
  - Basis for designing AI assistant like ChatGPT



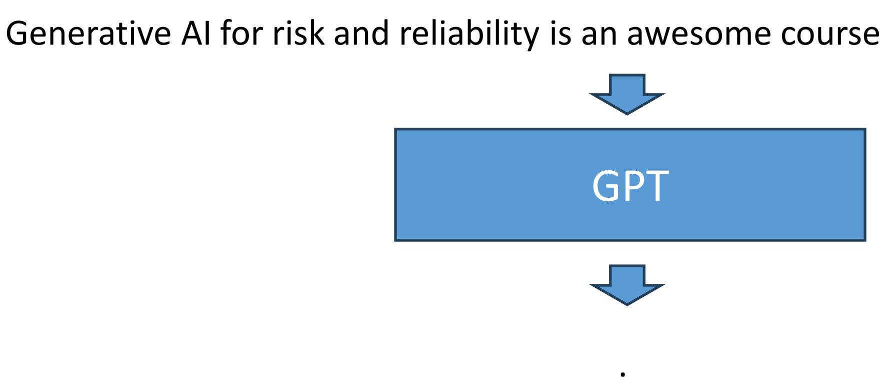
What GPT does

- GPT: Generative Pre-trained Transformer
- The basic task: Next token prediction
  - Recursively
  - Basis for designing AI assistant like ChatGPT



What GPT does

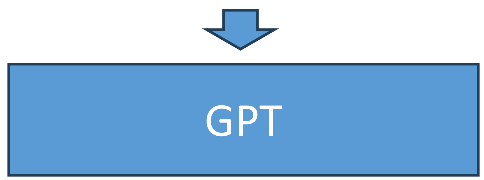
- GPT: Generative Pre-trained Transformer
- The basic task: Next token prediction
  - Recursively
  - Basis for designing AI assistant like ChatGPT



What GPT does

- GPT: Generative Pre-trained Transformer
- The basic task: Next token prediction
  - Recursively
  - Basis for designing AI assistant like ChatGPT

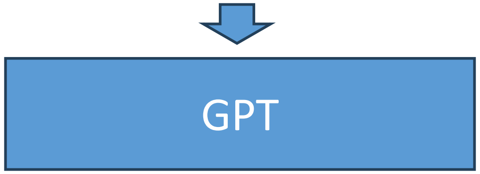
Generative AI for risk and reliability is an awesome course.



What GPT does

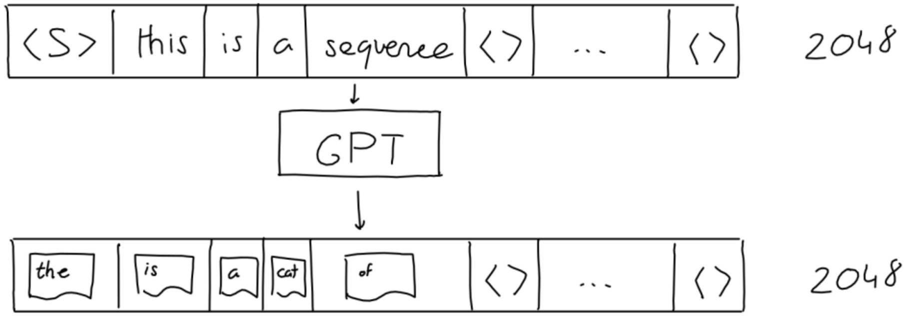
- GPT: Generative Pre-trained Transformer
- The basic task: Next token prediction
  - Recursively
  - Basis for designing AI assistant like ChatGPT
- But what's inside the block "GPT"?

Generative AI for risk and reliability is an awesome.

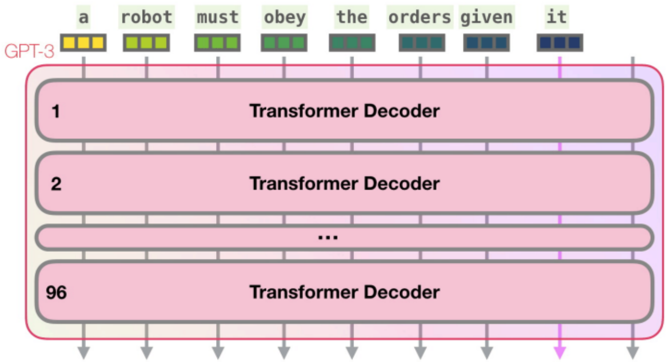


How GPT works: The global picture

- GPT is transformer architecture that uses only decoder.
  - Transformer: A deep learning architecture that uses attention mechanism to catch the relations among input tokens.
  - Input: A vector of tokens
    - Context size of GPT 3: 2048 tokens
  - Output: The same length of vector. But each element is a prediction of next token.
  - Only the last element on the output vector is taken as the prediction.
    - Each token will be predicted, and, in this way, the information of the entire sequence is compressed in the last token.



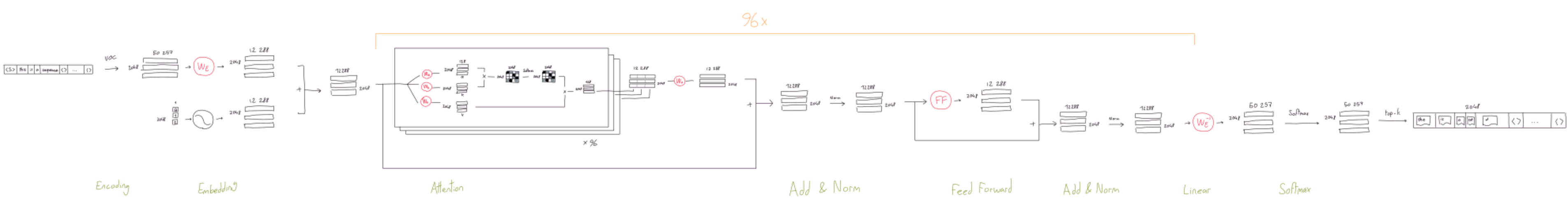
https://dugas.ch/artificial_curiosity/GPT_architecture.html



https://jalammar.github.io/how-gpt3-works-visualizations-animations/

How GPT works: The global picture

- GPT is a complex model but built on simple modules:
  - Embedding: Tokens (50257 for GPT 3) -> high-dimensional space (12,288 for GPT3)
  - Attention (is all you need): Learns the contexts
  - Feed-forward (MLP): Simple neural network to create nonlinear mappings
  - Unembedding: Transfer back to tokens
  - SoftMax: Generate a probability distribution from the neuron activations.



https://dugas.ch/artificial_curiosity/GPT_architecture.html

Embeddings and unembedding

- Map between the tokens (words) to a high-dimensional space of real numbers.
- In GPT 3, each token is represented by a vector of 12,288. Each element is a real number between (0, 1).
- The total amount of token is 50,257.
- Embedding matrix: 12,288*50,257.
- The parameters in this matrix are learnt from data. (We call all the parameters to be learnt as weights in general).

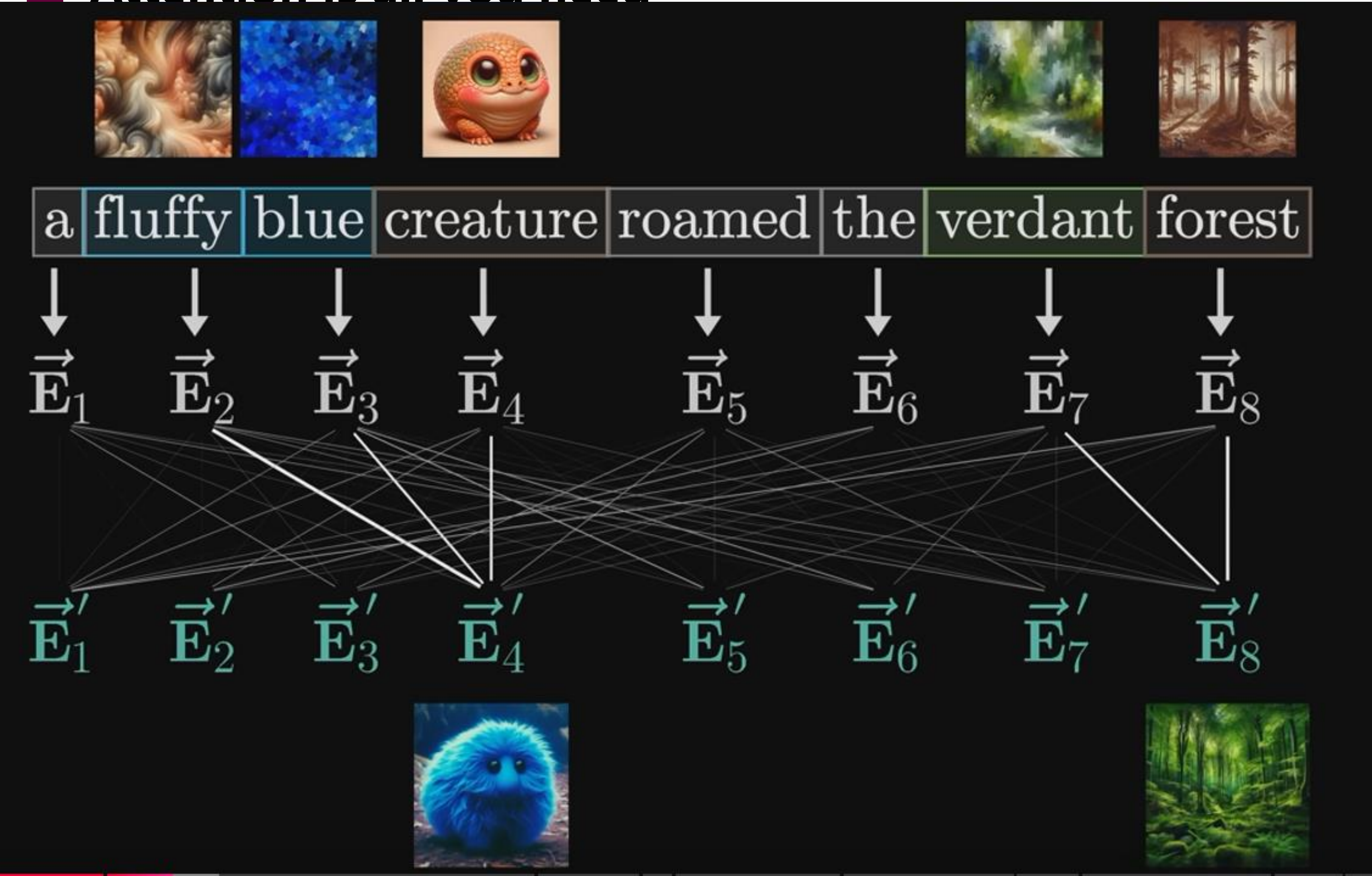
The secret of embedding

- After training, usually, the embeddings of tokens can represent semantic meanings.
- Close in the embedding space ⬄ close semantic meanings
  - http://projector.tensorflow.org/
- Difference in the vector space ⬄ Difference in semantic meanings
  - E('king')-E('queen') ~ E('man')-E('woman')
  - Try to verify that: https://colab.research.google.com/drive/1QSCr5myTwgVLLuIlhEXVA0TV2_9YOE2i?usp=sharing

Attention is all you need

- Motivation:
  - The meaning of a token depends on the context:
  - "The royal family, including Prince Harry, gathered to celebrate the annual Christmas festivities at Sandringham.“
  - "The young wizard, Harry Potter, embarked on a daring quest to uncover the secrets of the Chamber of Secrets hidden within Hogwarts.“
  - The embedding of “Harry” is the same, but the semantic meaning is different.
- Attention mechanism:
  - Modify the embeddings, so that it can be closer to the correct semantic meanings.
  - $E(X_i) = E(X_i) + \Delta E(X_i)$.
  - How to decide $\Delta E(X_i)$?
  - Let context helps!

Attention is all you need

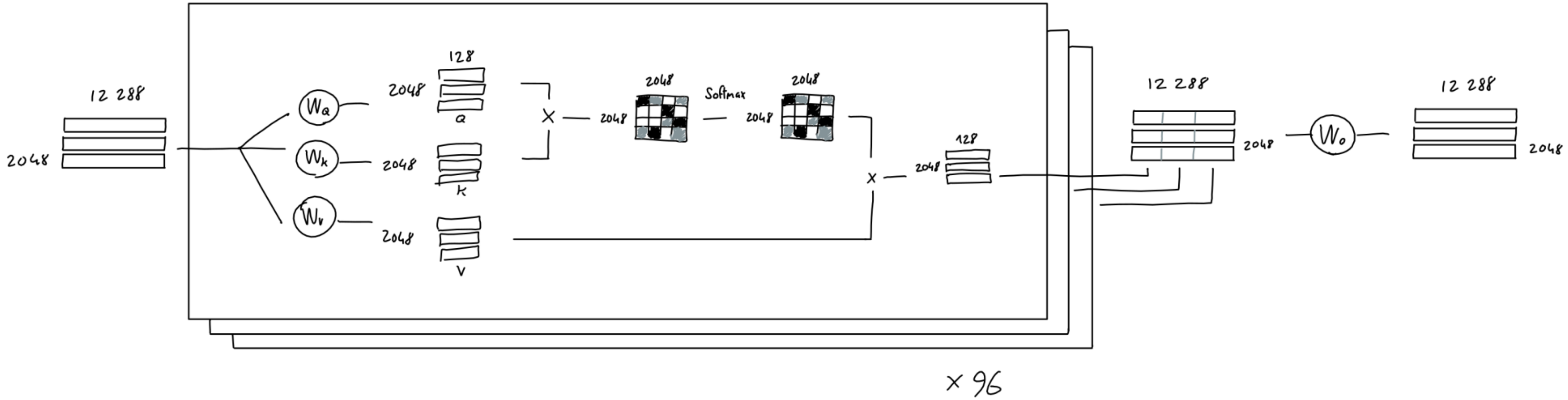


Attention is all you need

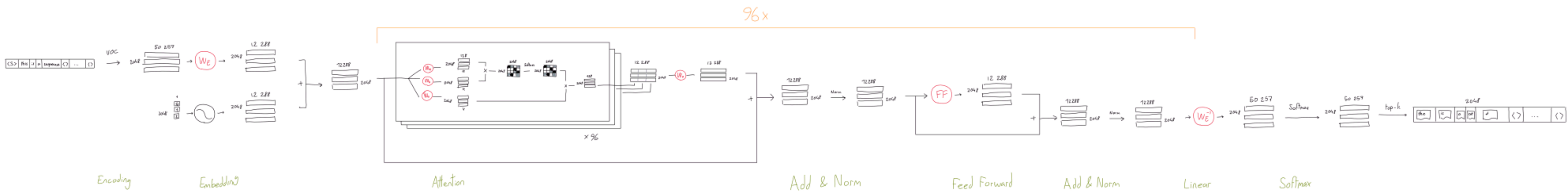
- $\Delta E(X_i) = \sum_{j=1}^{i} Attention_i$
- $Attention_i = softmax\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V$
  - Q – Query
  - K – Key
  - V – Value
  - Query and Key determines the weights (relevance) of a given token
  - Value – represent the amount of change we need for the adjustment.
  - V has the same dimension as $E(X_i)$. (GPT3: 12,228)
  - While Q and K usually has lower dimension (GPT3: 128).
- How to calculate Q, K and V?
  - Matrix operation:
  - $Q = W_Q \cdot E(X_i), K = W_K E(X_i), V = W_V E(X_i)$
  - Parameters in W_Q, W_K and W_V needs to be estimated from data

Multi-head attention

- The same attention block is repeated 96 times in GPT3 and the results are concatenated and weighted.
- Each head has its own weight matrices and can learn different dependencies.
- This is a way for the model to “see different patterns”

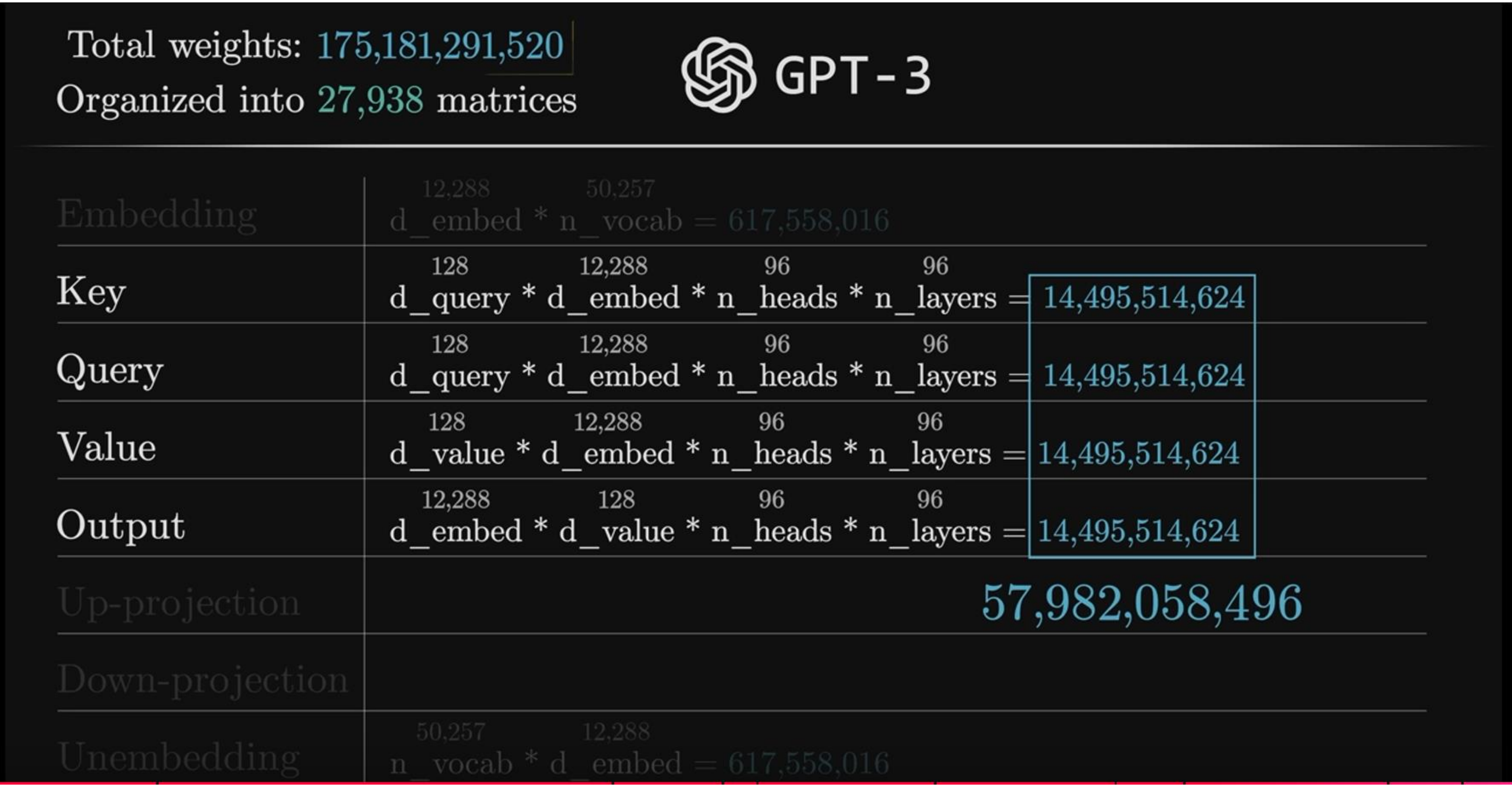


Predicting the output token



- Softmax with temperature:
  - Softmax: $p_i = \frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}$.
    - Normalize the output to get a pdf.
  - Softmax with temperature: $p_i = \frac{e^{x_i/T}}{\sum_{i=1}^{n} e^{x_i/T}}$
    - T: temperature
    - T large -> The pdf is more uniform -> The output is more random.
    - T small -> The pdf is more concentrated -> The output is more deterministic.
- To think: When do we need high temperature, and when do we need low temperature?

Not just the architecture, the scale matters!

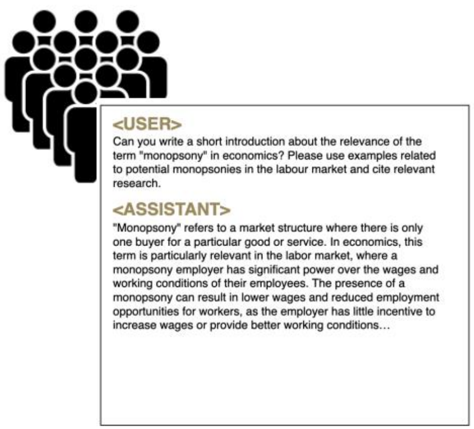


How to improve a pre-trained model on downstream tasks?

What we have discussed last lecture

every ~year

Stage 1: Pretraining
1. Download ~10TB of text.
2. Get a cluster of ~6,000 GPUs.
3. Compress the text into a neural network, pay ~$2M, wait ~12 days.
4. Obtain **base model**.



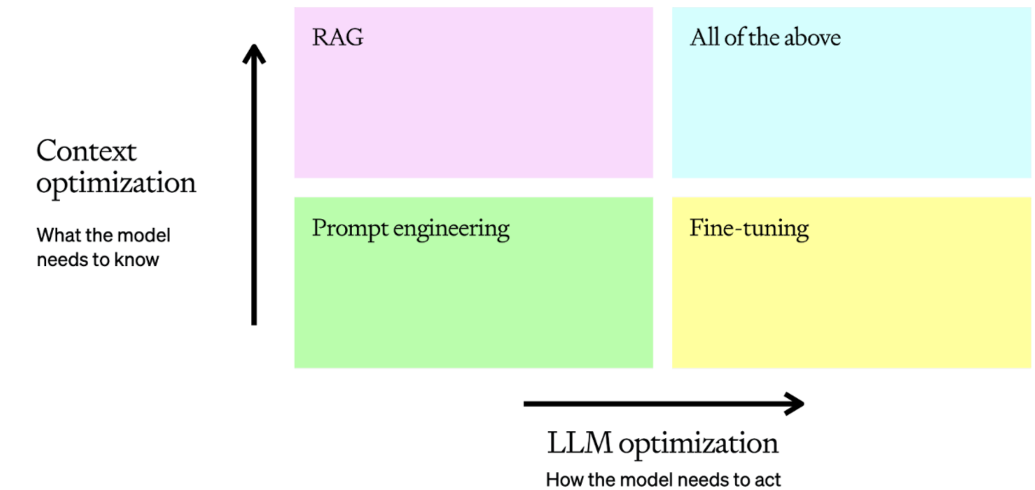
every ~week

Stage 2: Finetuning
1. Write labeling instructions
2. Hire people (or use scale.ai!), collect 100K high quality ideal Q&A responses, and/or comparisons.
3. Finetune base model on this data, wait ~1 day.
4. Obtain **assistant model.**
5. Run a lot of evaluations.
6. Deploy.
7. Monitor, collect misbehaviors, go to step 1.

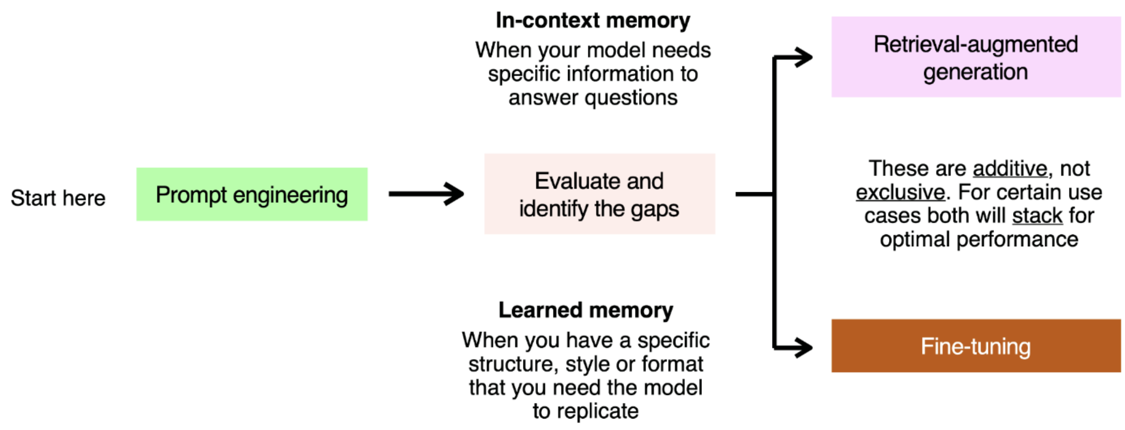
Source: Andrej Karpathy, Introduction to LLM, https://www.youtube.com/watch?v=zjkBMFhNj_g

How to improve a pre-trained model on downstream tasks?

If you want to improve further?



https://platform.openai.com/docs/guides/optimizing-llm-accuracy#understanding-the-tools



Thank you! Questions?