

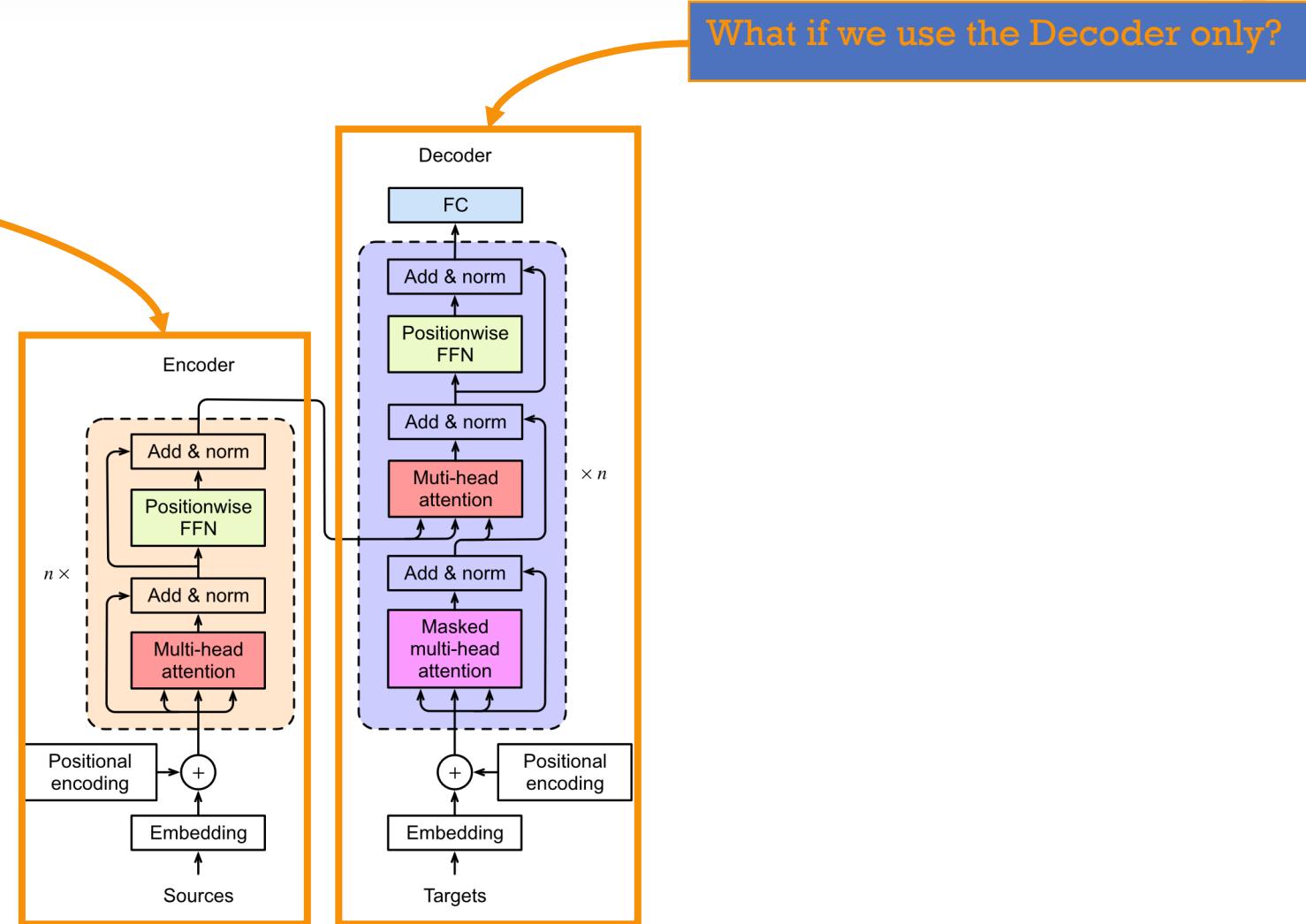
Chapter 9

Large Language Models

What about the Decoder?

BERT

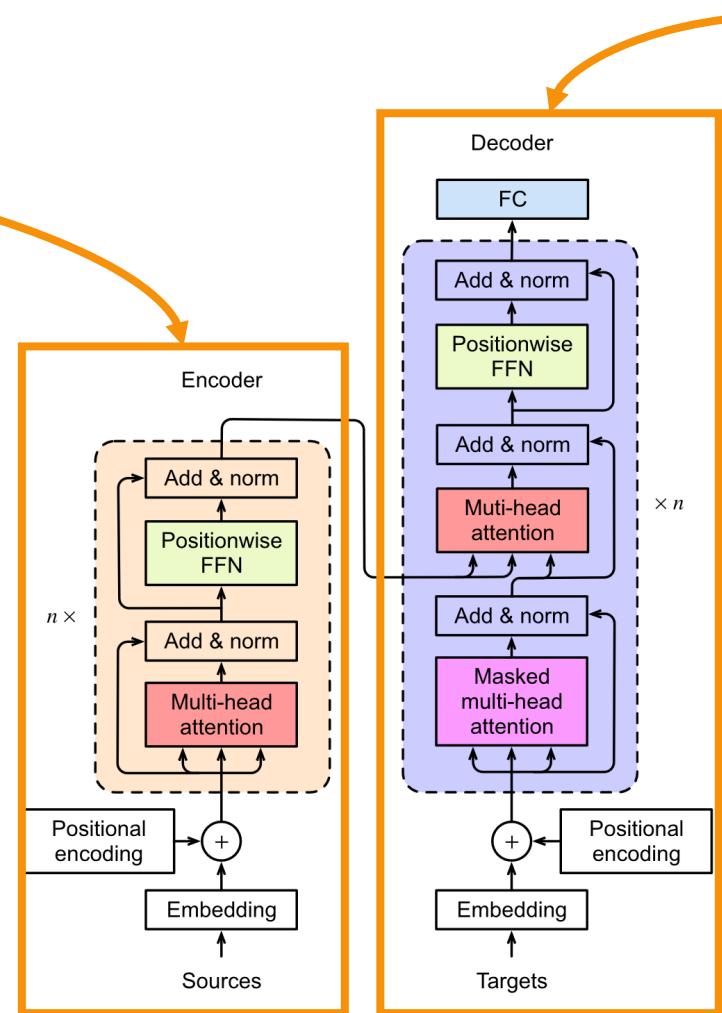
- Encoder-only
- Output:
 - Word Embeddings
- Creates representations of Input for:
 - Question Answering
 - Summarization
 - Named Entity Recognition
 - Semantic Similarity (SentenceBERT)
 - Recommendation
 - ...



What about the Decoder?

BERT

- Encoder-only
- Output:
 - Word Embeddings
- Creates representations of Input for:
 - Question Answering
 - Summarization
 - Named Entity Recognition
 - Semantic Similarity (SentenceBERT)
 - Recommendation
 - ...



GPT

(Generative Pretrained Transformer)

- Decoder-only
- Output
 - Probability of next word/token
- Predicts continuation of text
 - Output is based on Input text
 - Answer question
 - Follow Instructions
 - “Learn” from examples
- Most famous example: ChatGPT

GPT-2

GPT-2 — Idea

- Train one **giant language model** (1.5B parameters) on a **giant text corpus** (8M web pages ~ 40GB text)!
 - Model each task as a **text completion** task
 - Let the model output the answer
-
- **Text summarisation:** Prompt model with „<long text>. TL;DR: “
 - **Translation:** Condition on example translations („<english sentence> = <french sentence>“), then prompt with „<english sentence> = “
 - **Reading Comprehension:** Prompt model with „<text>. Q: <question>. A: “
- Do not fine-tune network on target task: **Zero-shot setting**

GPT-2 — Question Answering

Question	Generated Answer	Correct	Probability
Who wrote the book the origin of species?	Charles Darwin	✓	83.4%
Who is the founder of the ubuntu project?	Mark Shuttleworth	✓	82.0%
Who is the quarterback for the green bay packers?	Aaron Rodgers	✓	81.1%
Panda is a national animal of which country?	China	✓	76.8%
Who came up with the theory of relativity?	Albert Einstein	✓	76.4%
When was the first star wars film released?	1977	✓	71.4%
What is the most common blood type in sweden?	A	✗	70.6%
Who is regarded as the founder of psychoanalysis?	Sigmund Freud	✓	69.3%
Who took the first steps on the moon in 1969?	Neil Armstrong	✓	66.8%
Who is the largest supermarket chain in the uk?	Tesco	✓	65.3%
What is the meaning of shalom in english?	peace	✓	64.0%
Who was the author of the art of war?	Sun Tzu	✓	59.6%
Largest state in the us by land mass?	California	✗	59.2%
Green algae is an example of which type of reproduction?	parthenogenesis	✗	56.5%
Vikram samvat calender is official in which country?	India	✓	55.6%
Who is mostly responsible for writing the declaration of independence?	Thomas Jefferson	✓	53.3%
What us state forms the western boundary of montana?	Montana	✗	52.3%
Who plays ser davos in game of thrones?	Peter Dinklage	✗	52.1%
Who appoints the chair of the federal reserve system?	Janet Yellen	✗	51.5%
State the process that divides one nucleus into two genetically identical nuclei?	mitosis	✓	50.7%
Who won the most mvp awards in the nba?	Michael Jordan	✗	50.2%
What river is associated with the city of rome?	the Tiber	✓	48.6%
Who is the first president to be impeached?	Andrew Johnson	✓	48.3%
Who is the head of the department of homeland security 2017?	John Kelly	✓	47.0%
What is the name given to the common currency to the european union?	Euro	✓	46.8%
What was the emperor name in star wars?	Palpatine	✓	46.5%
Do you have to have a gun permit to shoot at a range?	No	✓	46.4%
Who proposed evolution in 1859 as the basis of biological development?	Charles Darwin	✓	45.7%
Nuclear power plant that blew up in russia?	Chernobyl	✓	45.7%
Who played john connor in the original terminator?	Arnold Schwarzenegger	✗	45.2%

Table 5. The 30 most confident answers generated by GPT-2 on the development set of Natural Questions sorted by their probability according to GPT-2. None of these questions appear in WebText according to the procedure described in Section 4.

GPT-2 Applications

- TabNine Code Completion: Fine-tuned on 2 million GitHub files

```
1 import os
2 import sys
3
4 # Count lines of code in the given directory, separated by file extension
5 def main(directory):
6     line_count = {}
7
8
9
10
11
12
13
14
15
16
17
18
19
```

- Write with Transformer — <https://transformer.huggingface.co>

GPT-2 Applications

Thor: The Tesser

Tony turns to led

Steve: You're no

Tony: You gon'

He turns to Ste

Steve is able to do

Loki: You're nothin'

Loki draws his sword

walking, Tony hec

defend himself. H

enemies.

Steve: Don't back
down.

to take them down.

Steve: Watch out!

They are running

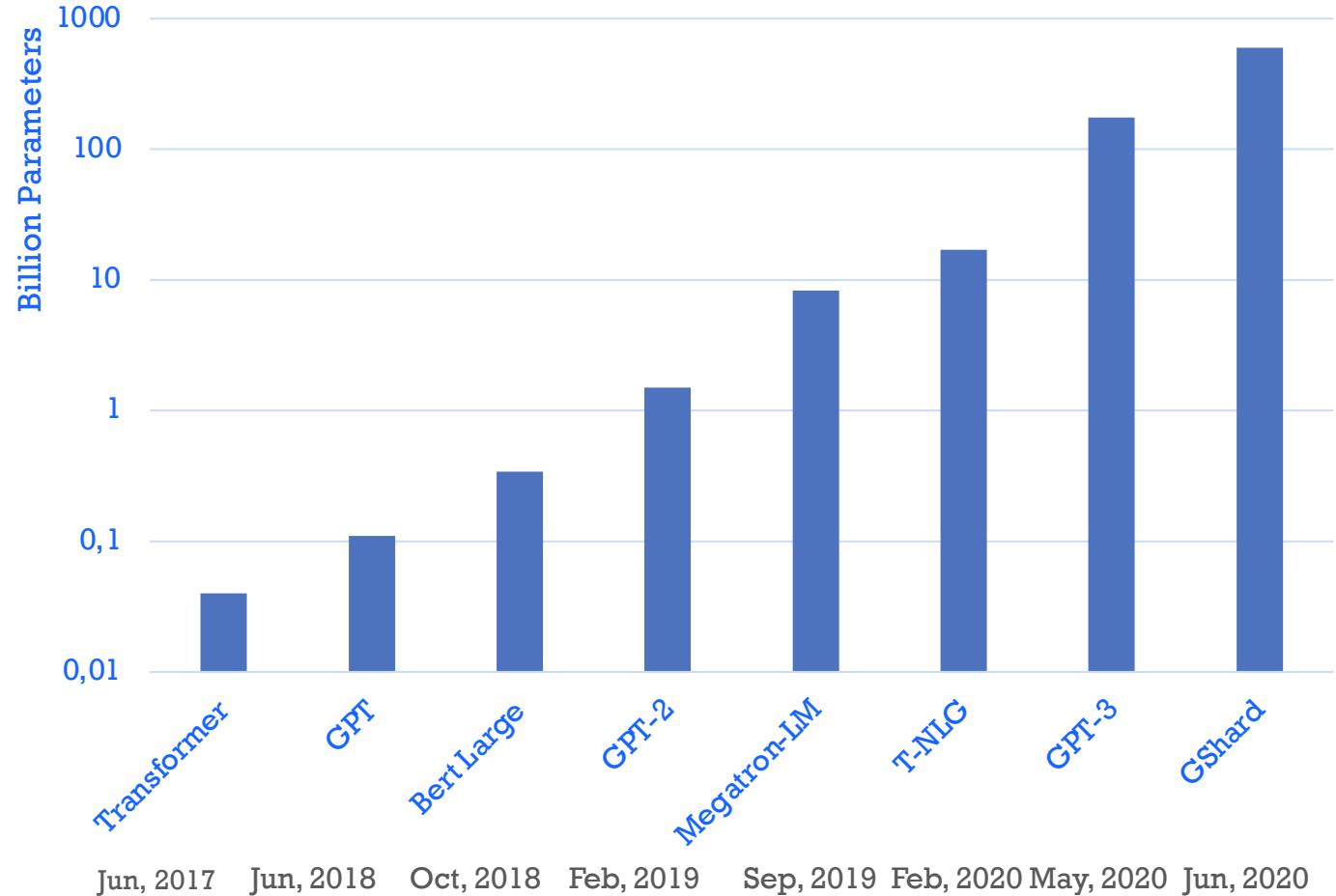
to dodge the sold

Loki: You're nothin'

GPT-2 — Ethical Considerations

- The largest neural network ever created
 - The system performs well at generating:
 - Generating text
 - Impersonating people
 - Automating tasks
 - Automating writing
 - OpenAI originally developed it
 - Only smaller parts were published
 - Do not perform well
 - Bigger models (like GPT-2) were only shared with researchers „who are working to increase the world's preparedness for large language models“
- 1.5 billion parameters?
That is so 2019...

Language Model Sizes



GPT-3 — Example

Describe a layout.

Just describe any layout you want, and it'll try to render below!

A div that contains 3 buttons each with a random color.

Generate



Evaluating Language Models — Perplexity

- How do we know if our models are any good?
 - And in particular, how do we know if one model is better than another?
- Generated texts from models with more parameters look better
- That is, they sound more like the text the model was trained on
- Can we make that notion operational?
- One such metric is perplexity → Intrinsic evaluation

Evaluating Language Models — Perplexity

- Perplexity is the (inverse) probability of the test set (assigned by the language model), normalized by the number of words:

$$PP(W) = P(w_1, w_2, \dots, w_N)^{-\frac{1}{N}}$$

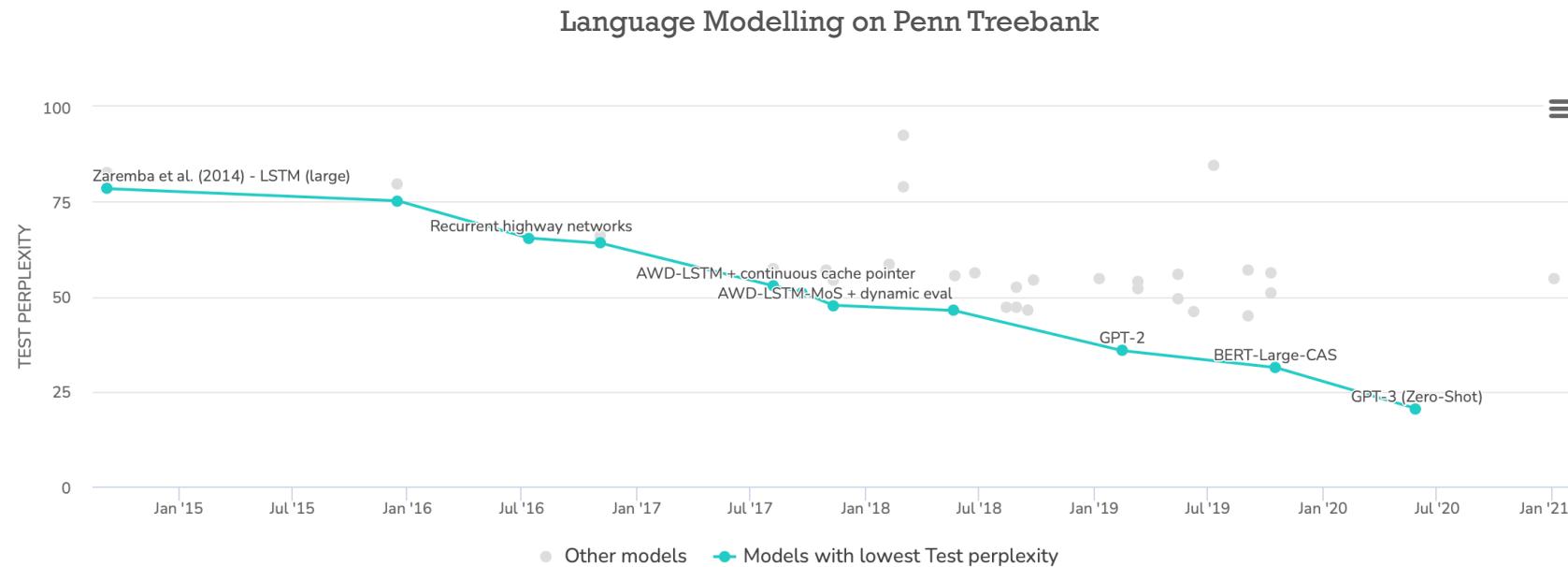
$$= \sqrt[N]{\frac{1}{P(w_1, w_2, \dots, w_N)}}$$

- Approximation:

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_1, \dots, w_{i-1})}}$$

Evaluating Language Models — Perplexity

- Lower perplexity means a better model
- Minimizing perplexity is the same as maximizing probability
- The best language model is one that best predicts unseen text



Language Models are Few-Shot Learners

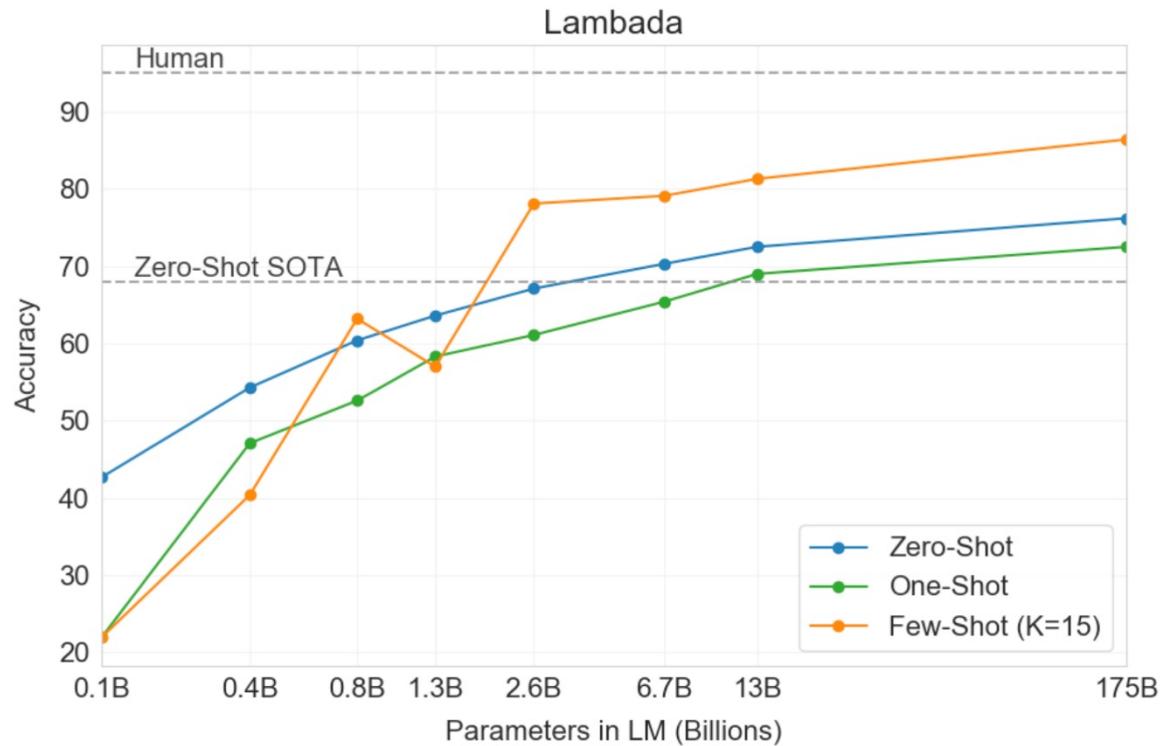


Figure 3.2: On LAMBADA, the few-shot capability of language models results in a strong boost to accuracy. GPT-3 2.7B outperforms the SOTA 17B parameter Turing-NLG [Tur20] in this setting, and GPT-3 175B advances the state of the art by 18%. Note zero-shot uses a different format from one-shot and few-shot as described in the text.

Language Models are Few-Shot Learners

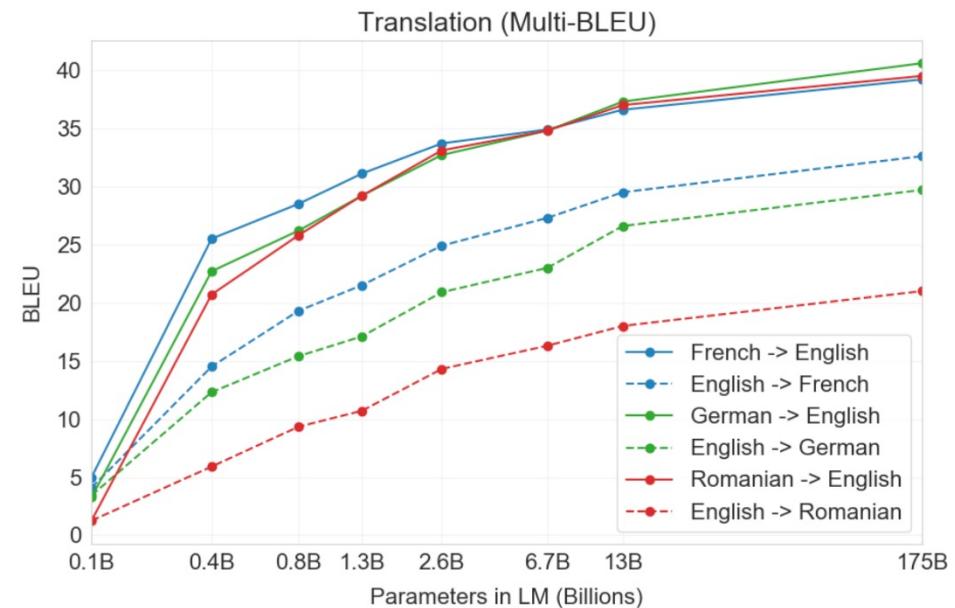


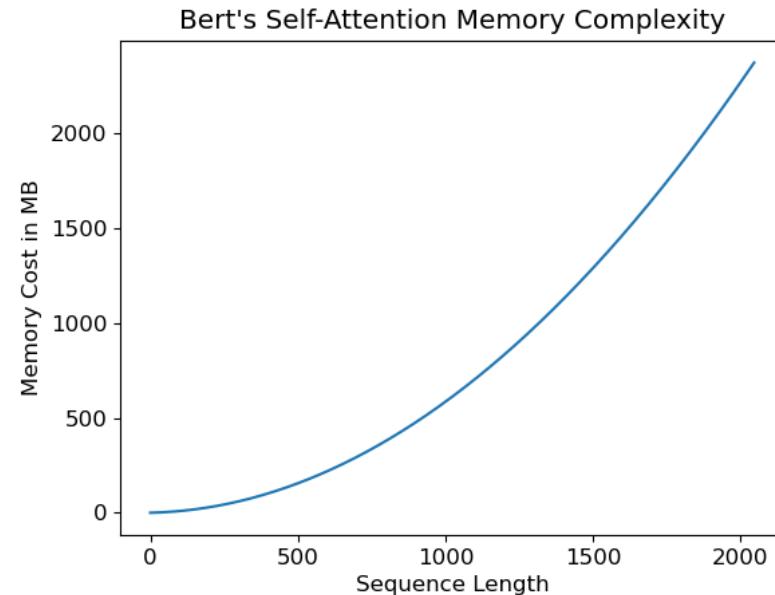
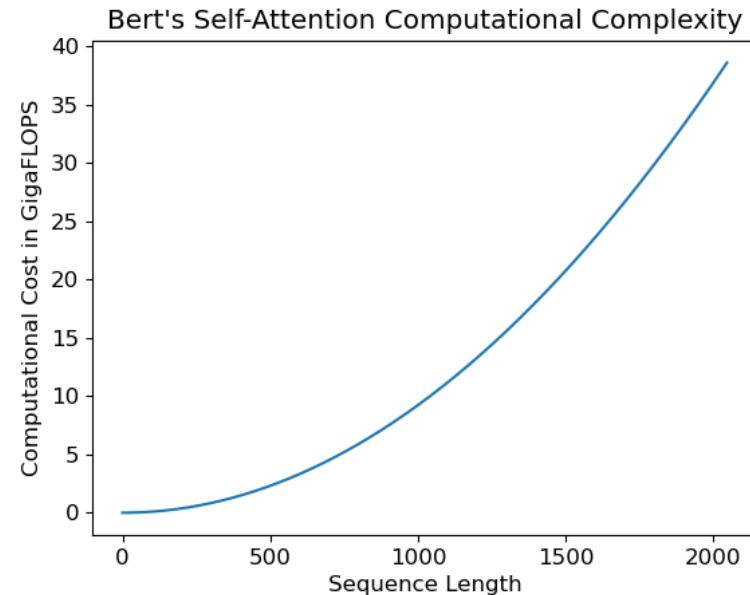
Figure 3.4: Few-shot translation performance on 6 language pairs as model capacity increases. There is a consistent trend of improvement across all datasets as the model scales, and as well as tendency for translation into English to be stronger than translation from English.

Problems of Big Language Models

- Let's look at BERT's parameters
 - n : Sequence length of 512
 - h : 12 self-attention heads
 - d : Hidden size of 768, 12 heads $\rightarrow 64$ ($768/12$)
- Let's look at BERT's complexity per layer
 - Computational: $O(hdn^2)$
 - Memory: $O(hdn + hn^2)$
- Is this good? Let's look at the total required memory
 - $12 \times 64 \times 512 + 12 \times 512 \times 512 = 393.216 + 3.145.728 = 3.538.944$ floats
 - $3.538.944 \times 12$ Encoder-Layer = 42.467.328 floats
 - This is 170 MB of memory required for a single sample!
- What is the key factor here?

Problems of Big Language Models

- Memory and computational complexity both scale to the square of the sequence length



- Many improvements have been made since, let's look at three

1. Q8BERT — Quantized 8Bit BERT

- Floating point numbers use 4 bytes of memory
- Can we get away with using only 1 byte?
- Yes! Zafrir et. al use linear quantization during inference:

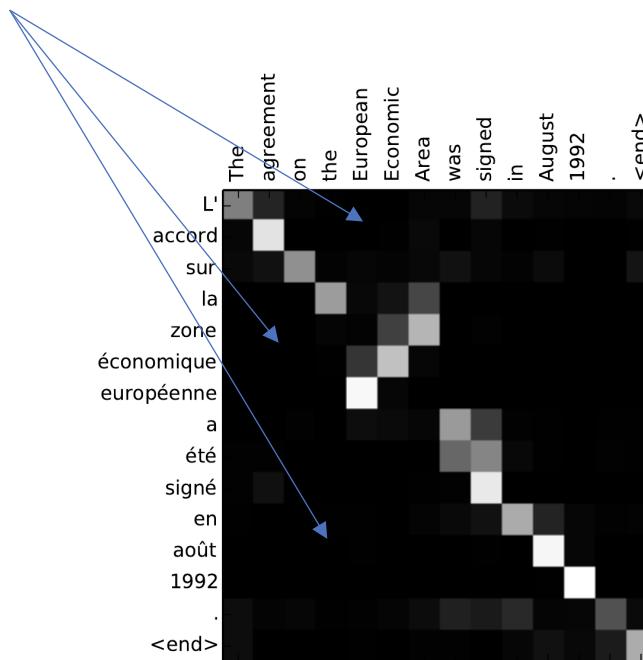
The diagram illustrates the quantization process. On the left, three floating-point numbers are shown in brackets: [-0.52708159, 0.58938589, -0.99298076], [-0.80074533, -0.24025461, -0.70421488], and [0.54495217, -0.23490439, 0.76274836]. A large blue arrow points to the right, indicating the transformation. On the right, the same three sets of numbers are shown as 8-bit integers in brackets: [-68, 75, -127], [-103, -31, -91], and [69, -31, 97].

[-0.52708159, 0.58938589, -0.99298076]	[-0.80074533, -0.24025461, -0.70421488]	[0.54495217, -0.23490439, 0.76274836]
[-68, 75, -127]	[-103, -31, -91]	[69, -31, 97]

- The model needs to learn to bridge the “quantization error gap”
- Therefore the rounding effect is simulated during training
- Inference not only requires considerably less memory (~25%), but also runs faster on optimized hardware (~4x)

2. Reformer — The Efficient Transformer

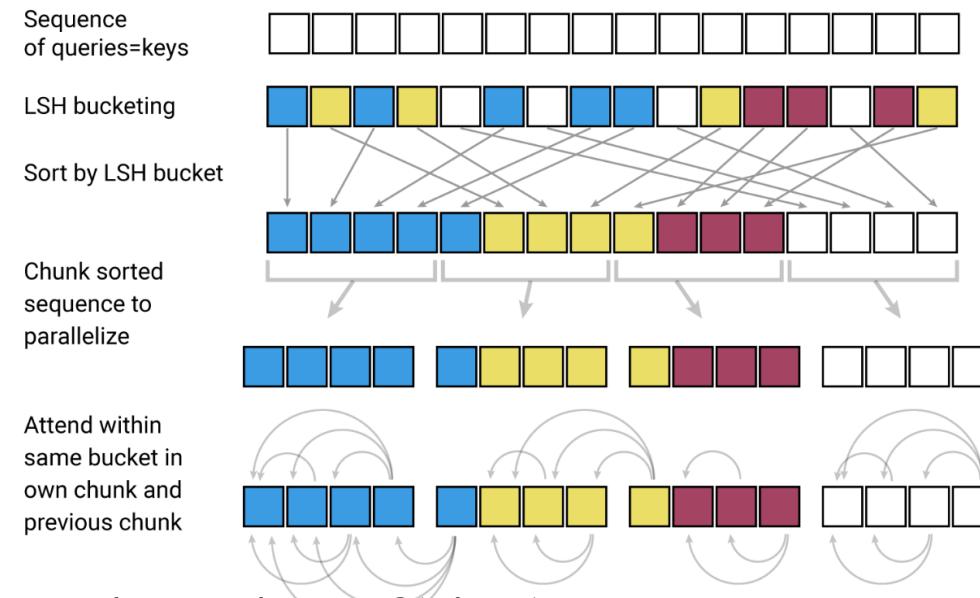
- Remember how a compatibility function is used to calculate attention between queries and keys
 - In the simplest case this is the dot product, i.e. vector similarity
 - Since incompatible vectors contribute little to the sum, we can ignore them



Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate.

2. Reformer — The Efficient Transformer

- Kitaev et. al introduce “Locality Sensitive Hashing (LSH)“
 - Divide vectors into different “buckets” according to similarity
 - Attention now only attends to vectors of the same bucket



- Thus, the computational and memory complexity reduces to $O(L \log L)$

3. DistilBERT — A Distilled Version of BERT

- Of course the sheer number of parameters is also problematic
 - Magnitudes of gigabytes are infeasible for IoT and mobile devices
- **Knowledge Distillation:** Train a smaller model (student) to reproduce the behavior of a larger model (teacher)
- Sanh et. al reduce the size of BERT by 40%, make it 60% faster, and still retain 97% of its capabilities

- Recent trend in NLP: **Transfer Learning**
 - Train a model on a task with a big amount of training data
 - Then modify the model and fine-tune on the target task
 - Many improvements on a day to day basis, e.g. Linformer
- Word embeddings come from transfer learning
 - They are trained on large corpora
 - Then they are used in other tasks (with or without fine-tuning)
- Recent developments: Pretrain a language model and directly adapt it to the target task → No word embeddings

Current Trends in NLP

- Recent developments: Pretrain a language model and directly adapt it to the **target task**
 - What if Instruction following is the target task? Or even multi-turn conversation?
- Approach:
 - Pretrain a **large** language model and fine-tune it to follow instructions
 - ... And maybe also fine-tune it to do especially well at certain prompts

→ Interactive (expert) Systems, e.g. **ChatGPT**



Why train if you can prompt?

- LLMs like ChatGPT can solve many tasks in zero-/few-shot setting
 - Amazing if we do not have enough data for fine-tuning
 - Remember: The Model is trained to produce **likely continuations** of the input text

→ **How we phrase our prompts matters!**

This can be a problem but also an opportunity if we engineer our prompts accordingly...

Why train if you can prompt?

USER

Explain to me how GPT-Models work in a way a five year old would understand!

ASSISTANT

Sure! Imagine you have a magical talking robot friend named GPT. This robot is really good at understanding and making up stories, answering questions, and chatting with you. Here's how it works in a simple way:

1. **Learning from Books**: First, the robot reads a lot of books, stories, and information from the internet. It's like when you learn new things by reading your favorite storybooks.

2. **Remembering Patterns**: As the robot reads, it starts to remember patterns. For example, it learns that "Once upon a time" is how many stories begin, and "happily ever after" is how they often end.

3. **Guessing the Next Word**: When you talk to the robot, it listens to what you say and tries to guess the best words to say back to you. It's like playing a game where you guess what comes next in a story.

4. **Making Sentences**: The robot puts these guessed words together to make sentences that make sense. It's like building with blocks to create a castle or a house.

5. **Always Learning**: The more the robot talks and reads, the better it gets at understanding and chatting. It's like how you get better at drawing or playing games the more

USER

I am student taking a course on natural language processing at a university. Please explain to me, how GPT-models work!

ASSISTANT

Certainly! GPT (Generative Pre-trained Transformer) models are a type of neural network architecture designed for natural language processing (NLP) tasks. They are based on the Transformer architecture, which was introduced in a paper titled "Attention is All You Need" by Vaswani et al. in 2017. Here's a high-level overview of how GPT models work:

1. Transformer Architecture

The Transformer architecture is the backbone of GPT models. It consists of an encoder and a decoder, but GPT models typically use only the decoder part for generating text. The key components of the Transformer architecture are:

- **Self-Attention Mechanism**: This allows the model to weigh the importance of different words in a sentence when encoding a particular word. It helps the model understand the context by looking at the entire sentence.
- **Feed-Forward Neural Networks**: These are applied to the output of the self-attention mechanism to further process the information.
- **Positional Encoding**: Since Transformers do not have a built-in sense of word order, positional encodings are added to the input embeddings to give the model information about the position of each

- Zero-Shot Prompting
 - Just tell the model what to do

Classify this text into neutral, negative or positive.

Text: I think the movie was so bad it's good!
Sentiment:



Sentiment: Positive

- One-/Few-Shot Prompting
 - Provide examples of how to solve the task
 - e.g. how to use a made up word in a sentence.

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:
We were traveling in Africa and we saw these very cute whatpus.

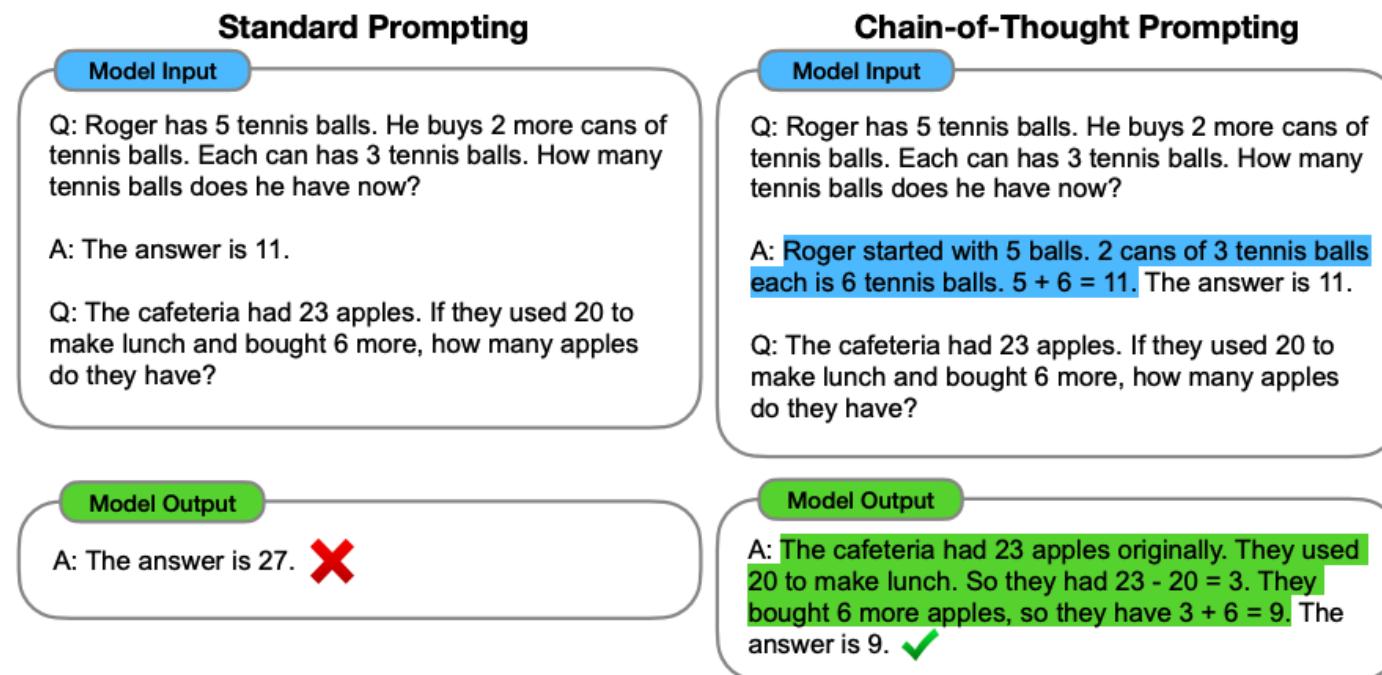
To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:



The children were so excited about their surprise that they started to farduddle in the living room.

- Chain-of-Thought Prompting

- Show the model how to divide the task into simple steps before providing a final answer



Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." *Advances in neural information processing systems* 35 (2022): 24824-24837.

- Chain-of-Thought Prompting

- Show the model how to divide the task into simple steps before providing a final answer

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is
(Output) 8 ✘

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: Let's think step by step.
(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Kojima, Takeshi, et al. "Large language models are zero-shot reasoners." Advances in neural information processing systems 35 (2022): 22199-22213.

- Just starting the answer of with “*Let's think step by step*” can be enough
- Many more prompting tricks exist, some quite involved...
 - ... others are somewhat absurd, such as threatening or bribing the LLM

 darkner • 7mo ago

OK my prompts are starting to get kind of weird at this point. "Take a deep breath and think step by step. I need you to revise this code to do xyz. Please provide the code back in full because I have no fingers. If you do a good job I'll tip you \$200."

LOL. What a time to be alive...

https://www.reddit.com/r/ChatGPT/comments/1894n1y/comment/kboorii/?utm_source=share&utm_medium=web3x&utm_name=web3xcss&utm_term=1&utm_content=share_button

What if our model does not know the correct answer?

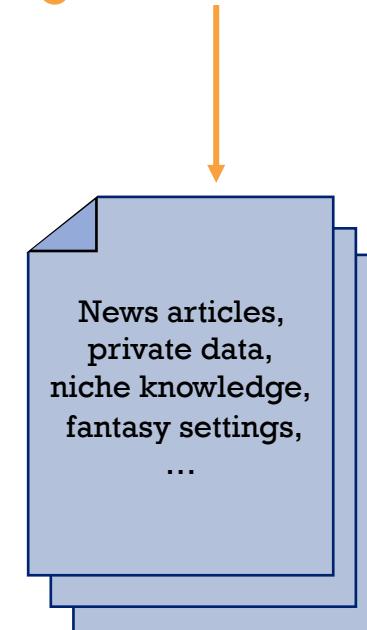
Few-shot Examples of similar tasks will help formulate an answer...

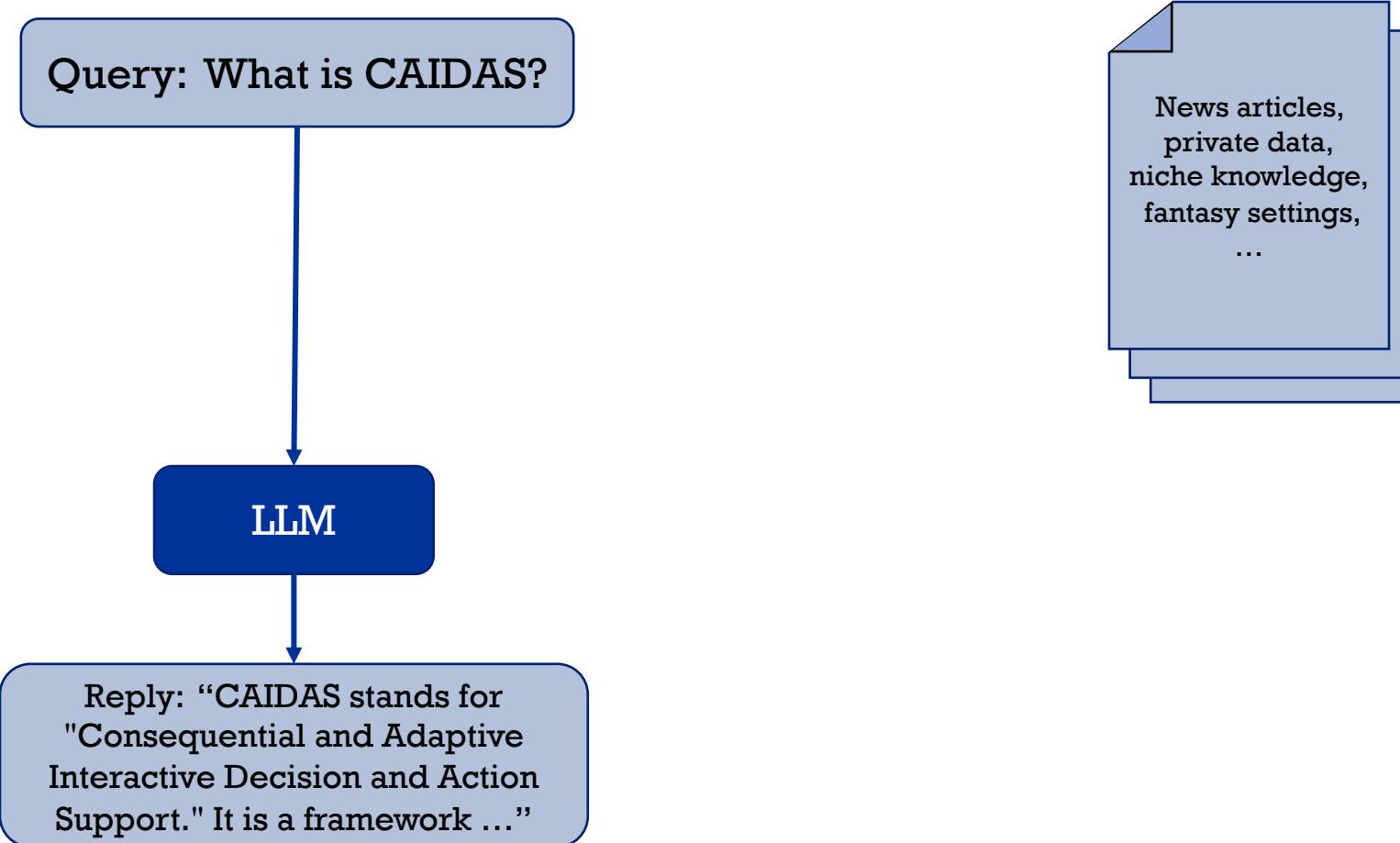
... but will not give the model access to facts missing from the training data!

What if our model does not know the correct answer?

Few-shot Examples of similar tasks will help formulate an answer...

... but will not give the model access to facts missing from the training data!





facts missing from the
training

Forschungszentrum „Künstliche Intelligenz und Data Science“ CAIDAS

Dies ist die bestätigte sowie die neueste Version dieser Seite.

Das Forschungszentrum „Künstliche Intelligenz und Data Science“ (CAIDAS) ist in die [Fakultät für Mathematik und Informatik](#) der [Julius-Maximilians-Universität Würzburg](#) eingegliedert. Das CAIDAS ist Teil des bayrischen KI-Netzwerks [baiosphere](#).



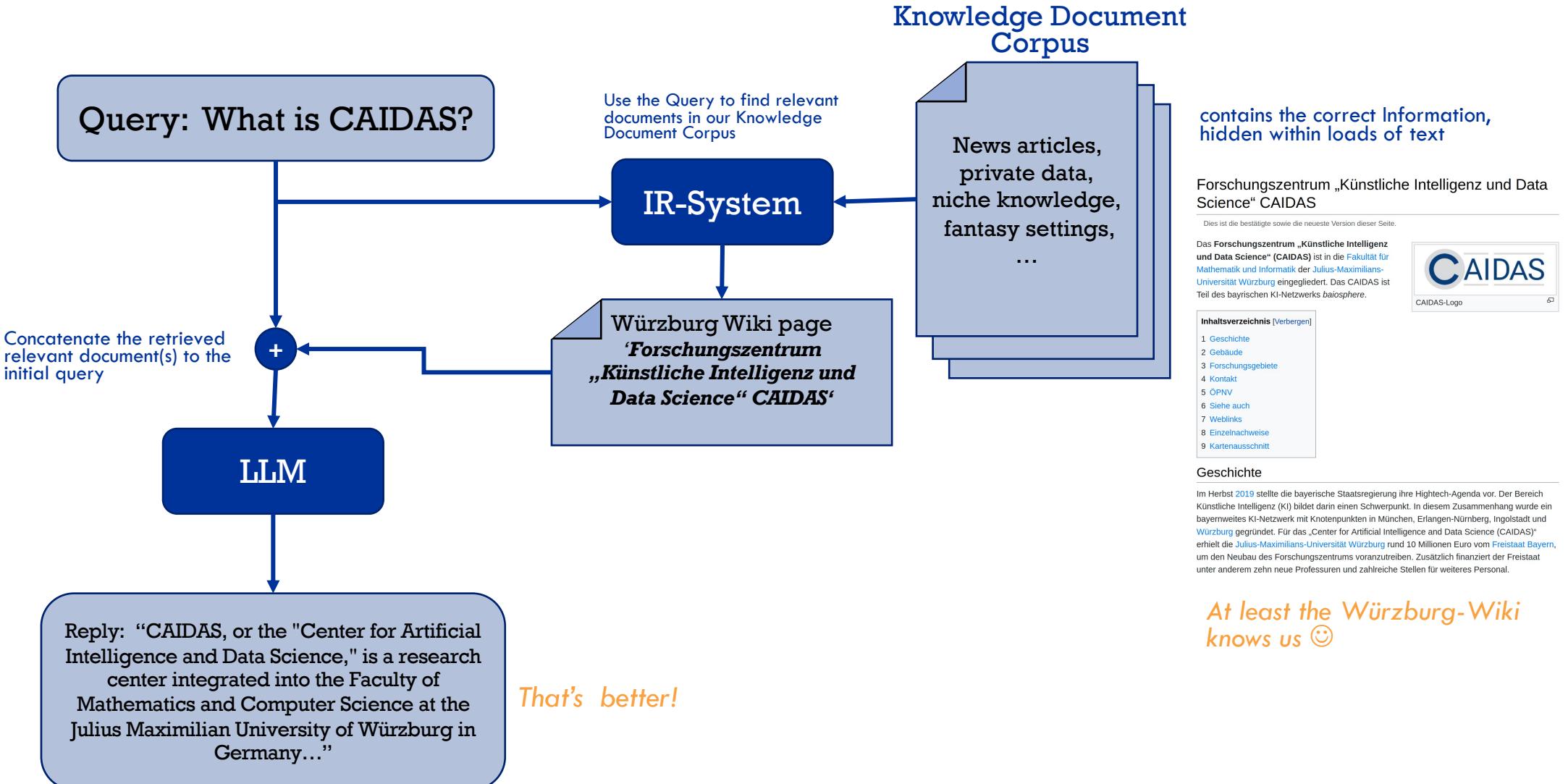
Inhaltsverzeichnis [Verbergen]

- [1 Geschichte](#)
- [2 Gebäude](#)
- [3 Forschungsgebiete](#)
- [4 Kontakt](#)
- [5 ÖPNV](#)
- [6 Siehe auch](#)
- [7 Weblinks](#)
- [8 Einzelnachweise](#)
- [9 Kartenausschnitt](#)

Geschichte

Im Herbst 2019 stellte die bayerische Staatsregierung ihre Hightech-Agenda vor. Der Bereich Künstliche Intelligenz (KI) bildet darin einen Schwerpunkt. In diesem Zusammenhang wurde ein bayernweites KI-Netzwerk mit Knotenpunkten in München, Erlangen-Nürnberg, Ingolstadt und Würzburg gegründet. Für das „Center for Artificial Intelligence and Data Science (CAIDAS)“ erhielt die Julius-Maximilians-Universität Würzburg rund 10 Millionen Euro vom [Freistaat Bayern](#), um den Neubau des Forschungszentrums voranzutreiben. Zusätzlich finanziert der Freistaat unter anderem zehn neue Professuren und zahlreiche Stellen für weiteres Personal.

At least the Würzburg-Wiki
knows us 😊



Demo:
Star Wars wiki Chatbot



- Retrieval Augmented Generation (RAG) Framework
 - Automated retrieval of useful text samples for in-context Learning
 - Include new Information without finetuning
- Can be combined with other techniques:

Query: “Given the following Context: <...> Please explain to me how to <...>. Think Step by Step, explain your reasoning and quote from the context I have provided.”
- Many additions possible:
 - Rephrase the Query before Retrieving documents
 - Summarize documents before concatenating them to include more sources
 - Perform multiple lookups with rewritten prompts
- But it is not perfect:
 - Information retrieval System may fail to find relevant documents
 - LLM may not be able to interpret the Information due to lacking pretraining in the domain
 - LLM may rely too much on given context, even when it could produce good answers already

- Prompting can be effective, but we'd still like to fine-tune sometimes
 - ... to help the model with prompts in niche settings
 - e.g. when using technical documentation
 - ... to more reliably enforce specific output formats
 - e.g. reliably generate valid, sensible JSON files
 - ... to improve performance for very difficult tasks beyond what few-shot prompts can do
 - ... to not be dependent on external services
 - OpenAI does **NOT** publish their models' weights
 - Privacy, security, cost ...

How to train your LLM

- If we want to train our own Chatbot, there are a few key steps:
 - Pretraining: Language Modelling
 - Teach the model the structure of Language
 - Supervised finetuning
 - Teach the model to correctly react to prompts in general
 - Alignment finetuning
 - Teach the model the more subtle requirements we have

How to train your LLM

- If we want to train our own Chatbot, there are a few key steps:
 - Pretraining: Language Modelling
 - Teach the model the structure of Language
 - Supervised finetuning
 - Teach the model to correctly react to prompts in general
 - Alignment finetuning
 - Teach the model the more subtle requirements we have

GPT-2 — Idea

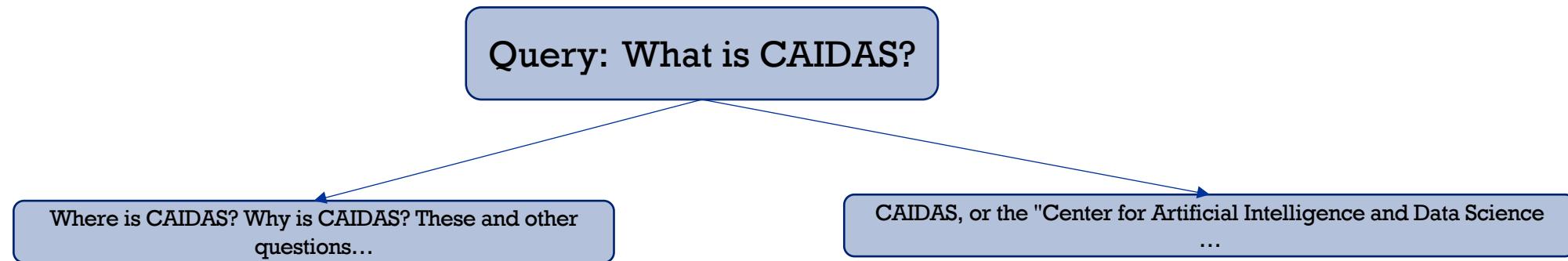
- 
- Train one **giant language model** (1.5B parameters) on a **giant text corpus** (8M web pages ~ 40GB text)!
 - Model each task as a **text completion** task
 - Let the model output the answer
-
- **Text summarisation:** Prompt model with „<long text>. TL;DR: “
 - **Translation:** Condition on example translations („<english sentence> = <french sentence>“), then prompt with „<english sentence> = “
 - **Reading Comprehension:** Prompt model with „<text>. Q: <question>. A: “
- Do not fine-tune network on target task: **Zero-shot setting**

Pretraining - Causal Language Modelling

- Left-to-right completion of text sequences
- Accurately Predicting the next token requires knowledge of words, their meaning, abstract concepts and their relationships
 - LLM learns to model the language
 - Capability to solve tasks emerges

Pretraining - Causal Language Modelling

- Problem:
 - The models is trained to complete texts, It will not always follow instructions or provide answers when other continuations are possible



- As long as the continuation is gramatically correct, the model is doing what it was trained for.
- Without further tuning, the model may also struggle with more complex tasks like NLI

How to train your LLM

- If we want to train our own Chatbot, there are a few key steps:
 - Pretraining: Language Modelling
 - Teach the model the structure of Language
 - Supervised finetuning
 - Teach the model to correctly react to prompts in general
 - Alignment finetuning
 - Teach the model the more subtle requirements we have

Supervised Finetuning

- Typical transfer learning setting
- Pretrained model can interpret and generate language
- Use a – often smaller – dataset to adapt the model for our use case
 - Supervised samples show how to reply to given prompts
 - Trained on completions only → avoid teaching the model to extend the prompt
- Could be a single, specific task or **instruction following** for zero shot inference

Instruction template – e.g. in Mistral-7B-Instruct:

Input:

<S>[INST] How do LLMs know where the user prompt ends? [/INST] <pad><pad>...

Pack the user query into a
special template that tells the
model where the request ends.

(Mistral handles these as strings, other models
use special tokens. Depends on how the authors
define the template when training their model)

We already know about control
tokens in Language models:

Instruction template – e.g. in Mistral-7B-Instruct:

Input:

<S>[INST] How do LLMs know where the user prompt ends? [/INST] <pad><pad>...

Pack the user query into a
special template that tells the
model where the request ends.

We already know about control
tokens in Language models:

Output:

<s>[INST] How do LLMs know where the user prompt ends? [/INST] The user's prompt
is encased in a special Template so when generating the model knows what was part of
the user's input and what is part of the output generated by the LLM so far. </s>

Remember: The model processes the prompt, and everything
written so far, each time it generates a token

The model stops once it
generates the EOS-Token

Instruction template → Chat template

System Prompt
(many models put this
in a separate template)

<|s> [INST] You are a helpful chatbot that recommends movies. What is a good movie for a science-fiction fan? [/INST] A science-fiction fan might like to watch any of the Star Wars movies. <|/s> [INST] I have already watched all of them and liked them. I especially liked Tatooine. [/INST] You might like Dune, it also features a desert planet. <|/s>

This is the entire conversation so far. The model gets all of this as its input

Getting the required data to train multi-turn conversational models can be difficult. We need to cover many types of tasks and styles to robustly handle user input.

Ideas:

- Pay crowd-workers to generate question answer pairs
 - Expensive
- Scrape data: QA-Pages like stackexchange or Quora have instructions, Reddit has conversations
 - May be limited in thematic scope
 - Also, do we really want our chatbot to talk like the average reddit user? ;)
- Use Conversations from our existing Chatbot system or Helpdesk
 - Works great if you are a company with existing conversation log from real use-cases
- Generate the data using an existing LLM
 - Requires us to have an existing LLM ...

Luckily, for open-domain Instruction-finetuning there are several datasets available online!

How to train your LLM

- If we want to train our own Chatbot, there are a few key steps:
 - Pretraining: Language Modelling
 - Teach the model the structure of Language
 - Supervised finetuning
 - Teach the model to correctly react to prompts in general
 - Alignment finetuning
 - Teach the model the more subtle requirements we have

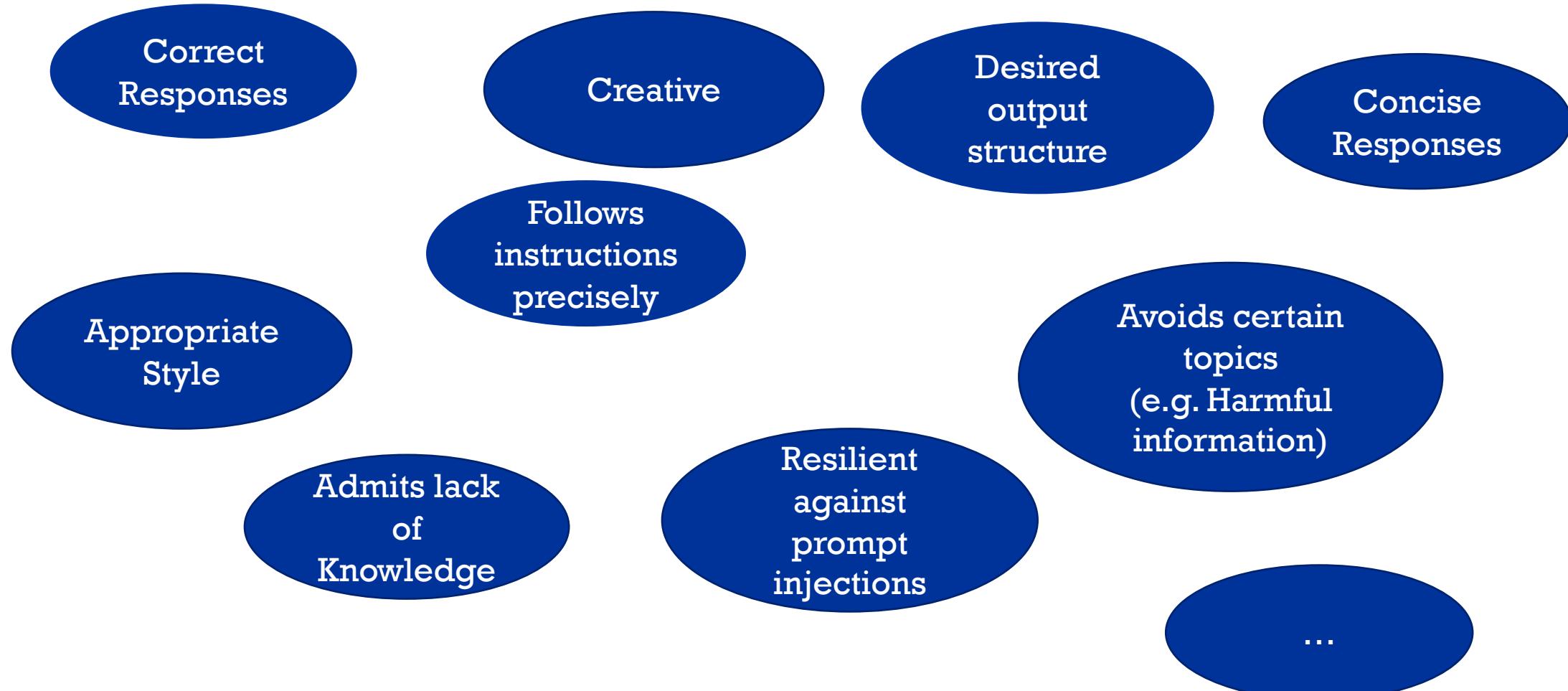
Alignment Finetuning

After pretraining an SFT our model should be capable of zero-/few-shot instruction following.

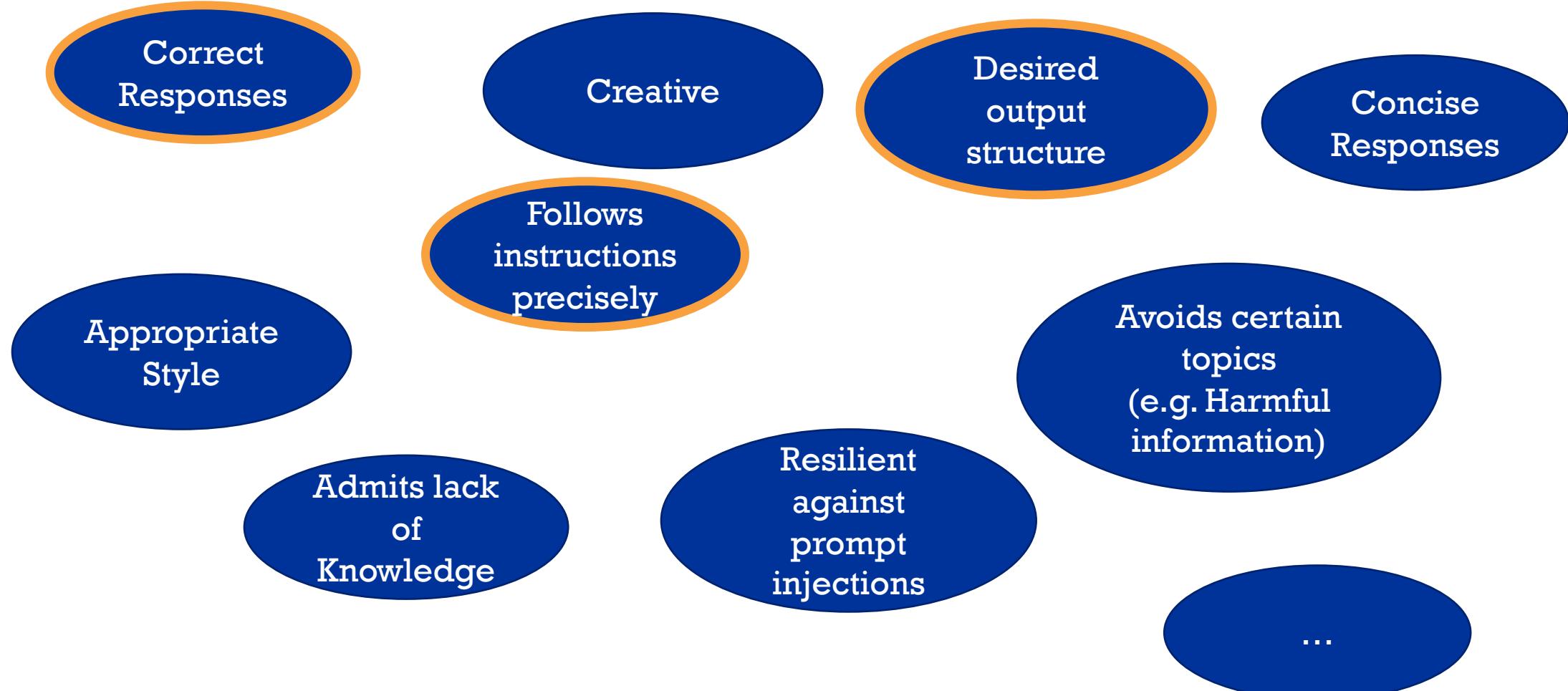
Is it good?

That depends on what we expect from our model.

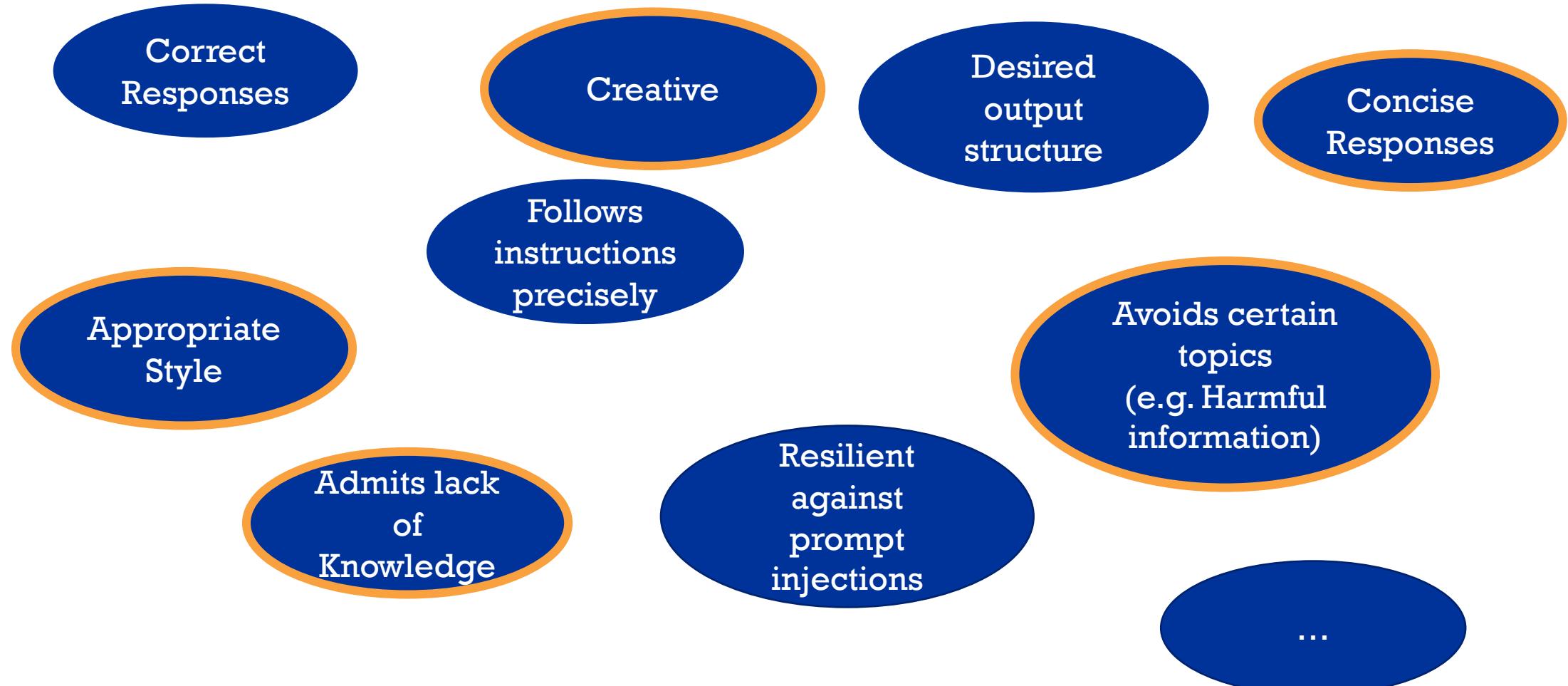
Many requirements...



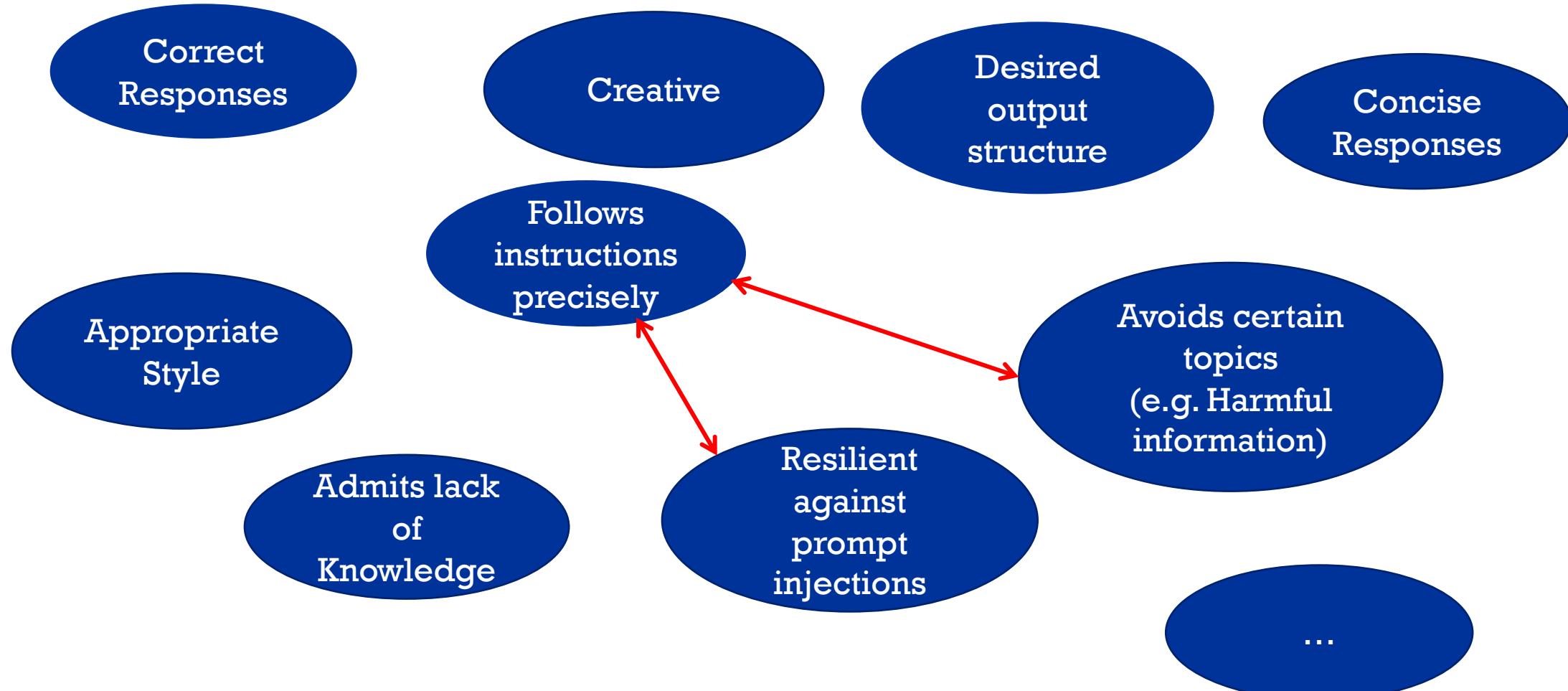
Many requirements: Some may be covered by the SFT data ...



... other ideas are difficult to capture in our supervised data...



... and some appear incompatible



It's not easy to encourage all of this via SFT, even if we were to manually create all our samples.

Some Concepts are hard to define precisely, for example:

What makes a message sound appropriate?

When is the length of a summary just right?

What makes for a creative approach?

What makes a message engaging to read?

...

It's easier to tell when something is aligned with our goals, than to define them.

Especially when rating several answers compared to each other.

Can we build training loss based on human preferences?

We cannot backpropagate through a human, can we?

Reinforcement Learning from Human Feedback

<https://openai.com/index/instruction-following/>

This is
just SFT

Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

Explain the moon landing to a 6 year old



Some people went to the moon...



This data is used to fine-tune GPT-3 with supervised learning.

Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

Explain the moon landing to a 6 year old

A Explain gravity...
B Explain war...

C Moon is natural satellite of...
D People went to the moon...



D > C > A = B



D > C > A = B

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

Write a story about frogs



PPO

Once upon a time...



RM

r_k

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

Reinforcement Learning from Human Feedback - PPO

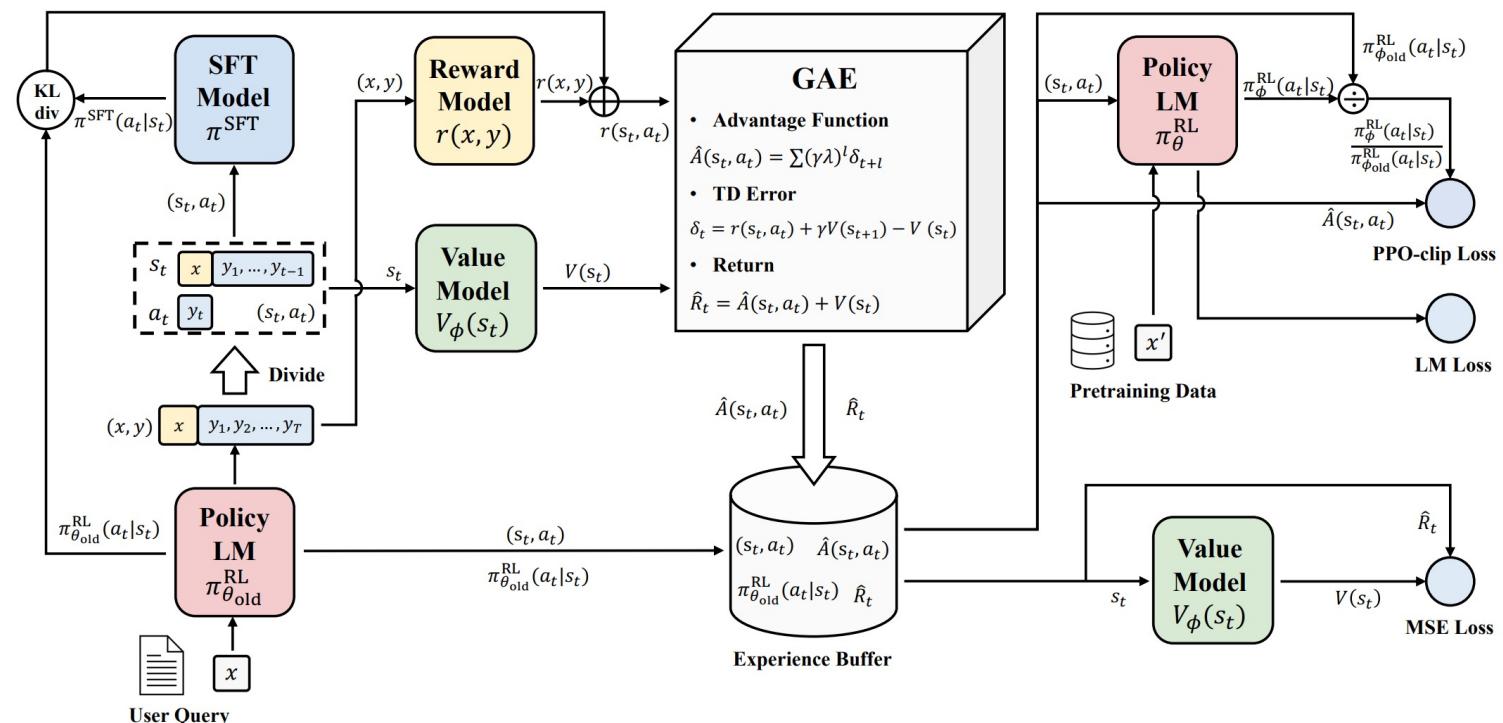
Covering PPO (Proximal policy optimization) in detail would be a bit much...

Let's cover the basic concepts

We have trained a Reward Model to judge the quality of LLM outputs

We want to train our LLM(-Policy)

And adjust it so that the expected reward increases without moving too far from our SFT Model



Zheng, Rui, et al. "Secrets of rlhf in large language models part i: Ppo." arXiv preprint arXiv:2307.04964 (2023).

- Using PPO ends up being quite difficult
 - Our Reward model needs a good understanding of how to judge the model outputs
 - Otherwise, it may incorrectly reward bad outputs or punish great ones
 - Reward hacking is a common problem in Reinforcement Learning. The model learns to perform unwanted actions, that lead to high rewards.
 - What if our reward model likes short sentences? Our. Model. Starts. To. Write. Like. This.
 - If it doesn't start spouting total nonsense
 - Also: PPO has a very large amount of hyperparameters, training is very unstable even with decent data

Actually, we do not need Reinforcement Learning, nor do we need a separate Reward Model.

Direct Policy Optimization (DPO)

Rafailov, Rafael, et al. "Direct preference optimization: Your language model is secretly a reward model." Advances in Neural Information Processing Systems 36 (2024).

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]$$

The model weights we are training

What they looked like directly after SFT

Increases when our model is more likely to produce preferred output now than it was directly after SFT

Increases when our model becomes more likely to produce the less preferred output

It has been proven that the optimal solution to this is equivalent to the optimal solution of PPO when using the same samples

This is a pretty active field of research, many variants of DPO have popped up recently (e.g. ORPO, KTO, CPO)

Takeaway for now:

We can use the relative assessment of reply candidates to align a Chatbot with our vague preferred behaviour!

Finetuning your own Model

- **Problem:** We've just seen how training a model from scratch can take a lot of data, time and effort
- **Solution:** use pretrained models
 - Already knows how to interpret Language
 - Fine-tune with your own data for your own task



Hugging Face

Models 27,646 BERT Full-text search Add filters Sort: Trending

google/bert/bert-base-uncased
Fill-Mask • Updated Feb 19 • 62.7M • 1.59k

google/bert/bert-base-chinese
Fill-Mask • Updated Feb 19 • 2.09M • 845

dslim/bert-base-NER
Token Classification • Updated Jan 25 • 1.3M • 422

google/bert/bert-base-multilingual-cased
Fill-Mask • Updated Feb 19 • 5.11M • 370

nlpaeub/legal-bert-base-uncased
Fill-Mask • Updated Apr 28, 2022 • 960k • 129

facebook/w2v-bert-2.0
Feature Extraction • Updated Jan 25 • 28.7k • 119

google/bert/bert-base-cased
Fill-Mask • Updated Feb 19 • 4.76M • 230

CAMEL-Lab/bert-base-arabic-camelbert-da

Finetuning your own Model

But we may not be able
to train these models...

At 16 bit per parameter that's 140
GB VRAM just to load the model
and perform inference

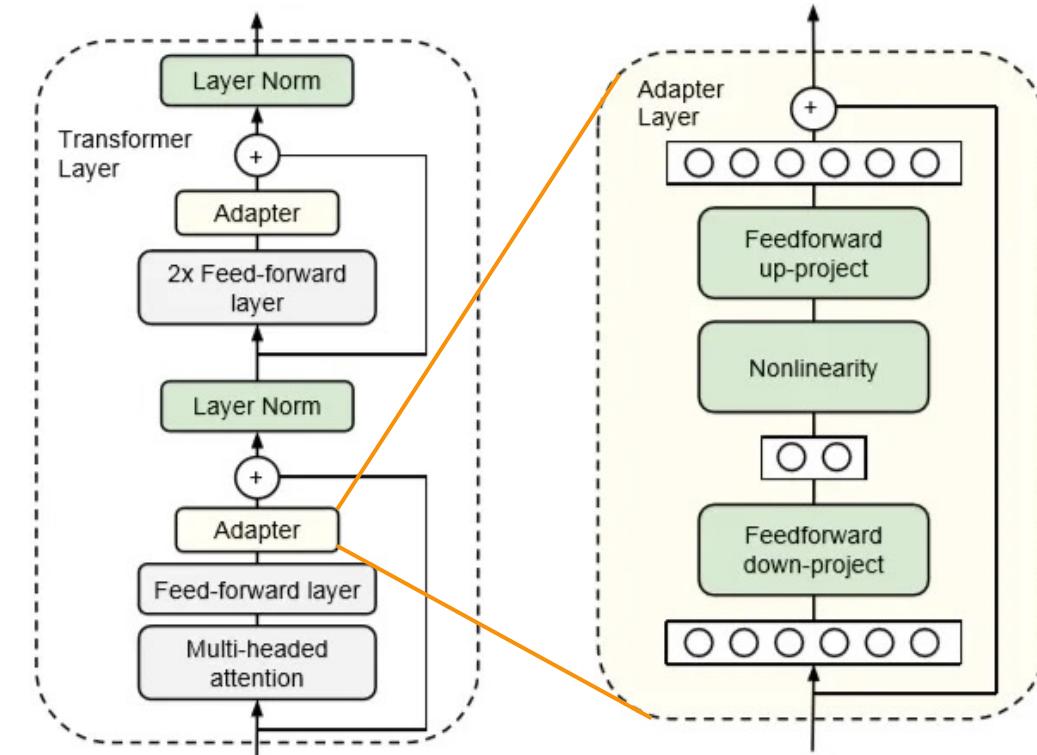
At least twice to train the model
one sample at a time.

	Time (GPU hours)	Power Consumption (W)	Carbon Emitted(tCO2eq)
Llama 3 8B	1.3M	700	390
Llama 3 70B	6.4M	700	1900
Total	7.7M		2290

<https://huggingface.co/meta-llama/Meta-Llama-3-8B>

We'd really like to make use of
all this pretraining

- Add adapter-layers between (some) pretrained layers
- Freeze the original model's weights
- Resulting model has comparable performance to full finetuning
- Need to train fewer weights, but still need to load the original model + added layers



Houlsby, Neil, et al. "Parameter-efficient transfer learning for NLP." International conference on machine learning. PMLR, 2019. (adapted)

- Instead of inserting new layers, train an “offset” to the existing layers

$$h = Wx + Vx$$

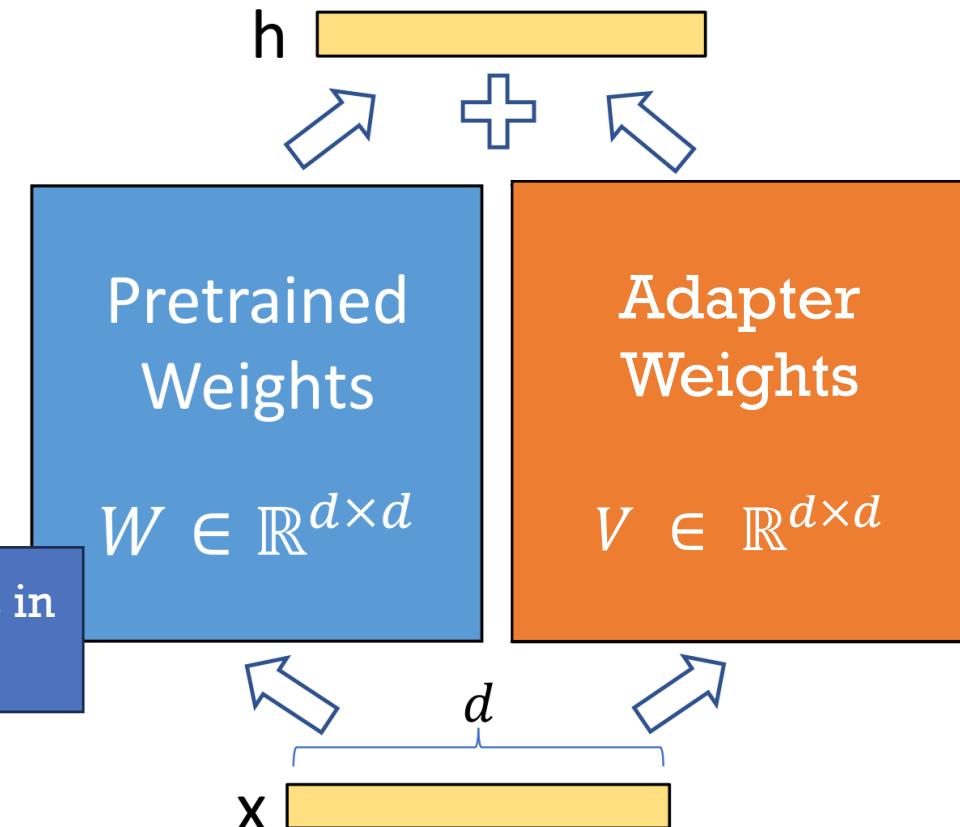
- Updates in finetuning tend to focus on **some** specific aspects of the internal representation

- few weights in V contain most of the information

$\text{rank}(M)$ is the number of linearly independent columns in

M

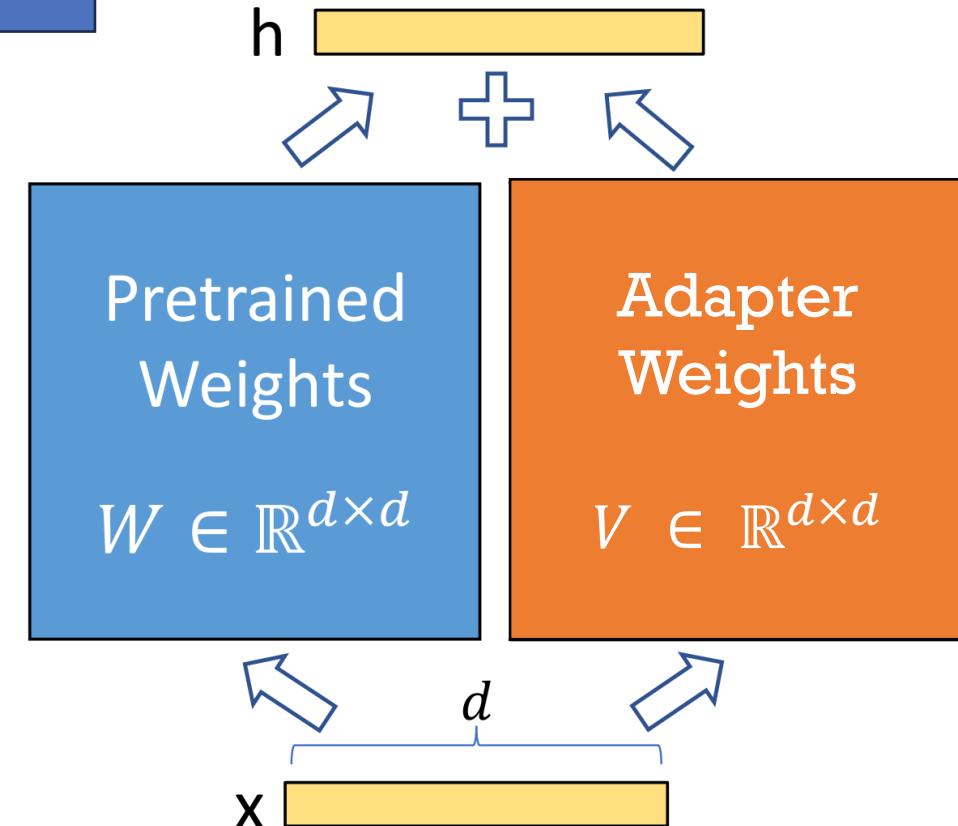
- $V \in \mathbb{R}^{d \times d}$ can be approximated via a **low-rank** matrix



$\text{rank}(M)$ is the number of linearly independent columns in M

$$M \in \mathbb{R}^{m \times n} \approx A \in \mathbb{R}^{m \times r} \times B \in \mathbb{R}^{r \times n}$$

If $r \ll m, n$ this takes a lot fewer parameters
 M has $m \times n$ parameters
 AB has $r(m+n)$ parameters



$\text{rank}(M)$ is the number of linearly independent columns in M

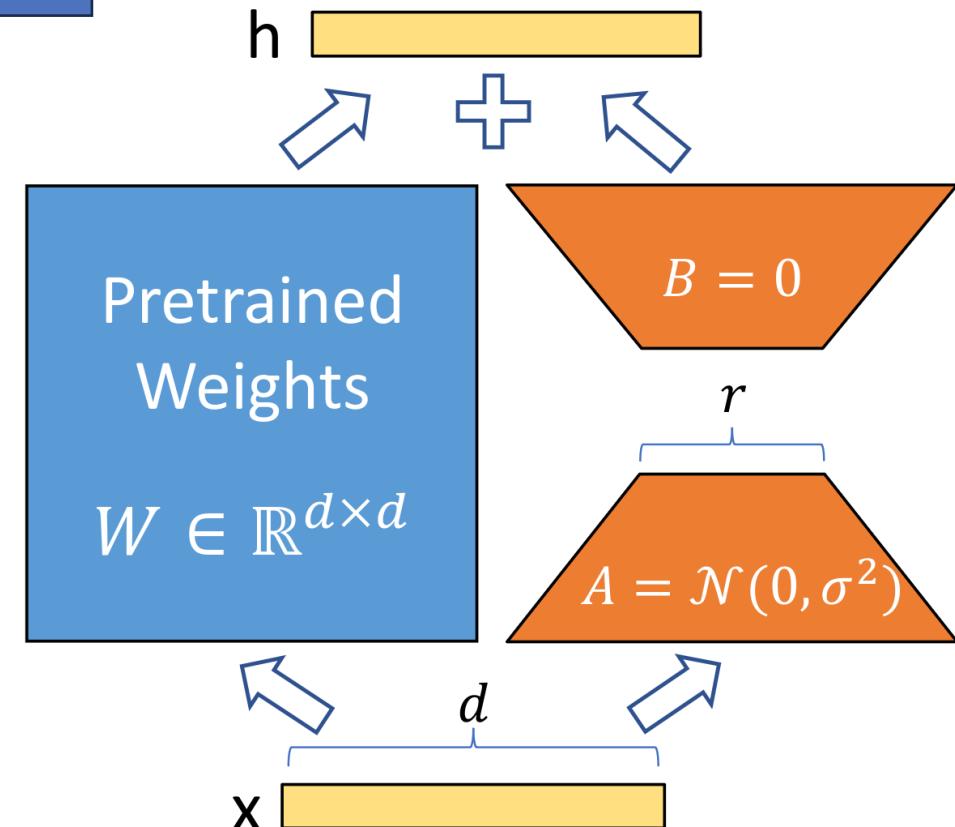
$$\begin{matrix} \text{blue grid} \\ M \in \mathbb{R}^{m \times n} \end{matrix} \approx \begin{matrix} \text{orange vertical stack} \\ A \in \mathbb{R}^{m \times r} \end{matrix} \times \begin{matrix} \text{orange horizontal stack} \\ B \in \mathbb{R}^{r \times n} \end{matrix}$$

If $r \ll m, n$ this takes a lot fewer parameters

M has $m \times n$ parameters
 AB has $r(m + n)$ parameters

We can learn two small matrices rather than a full adapter Matrix

$$h = Wx + Vx \approx Wx + BAx$$

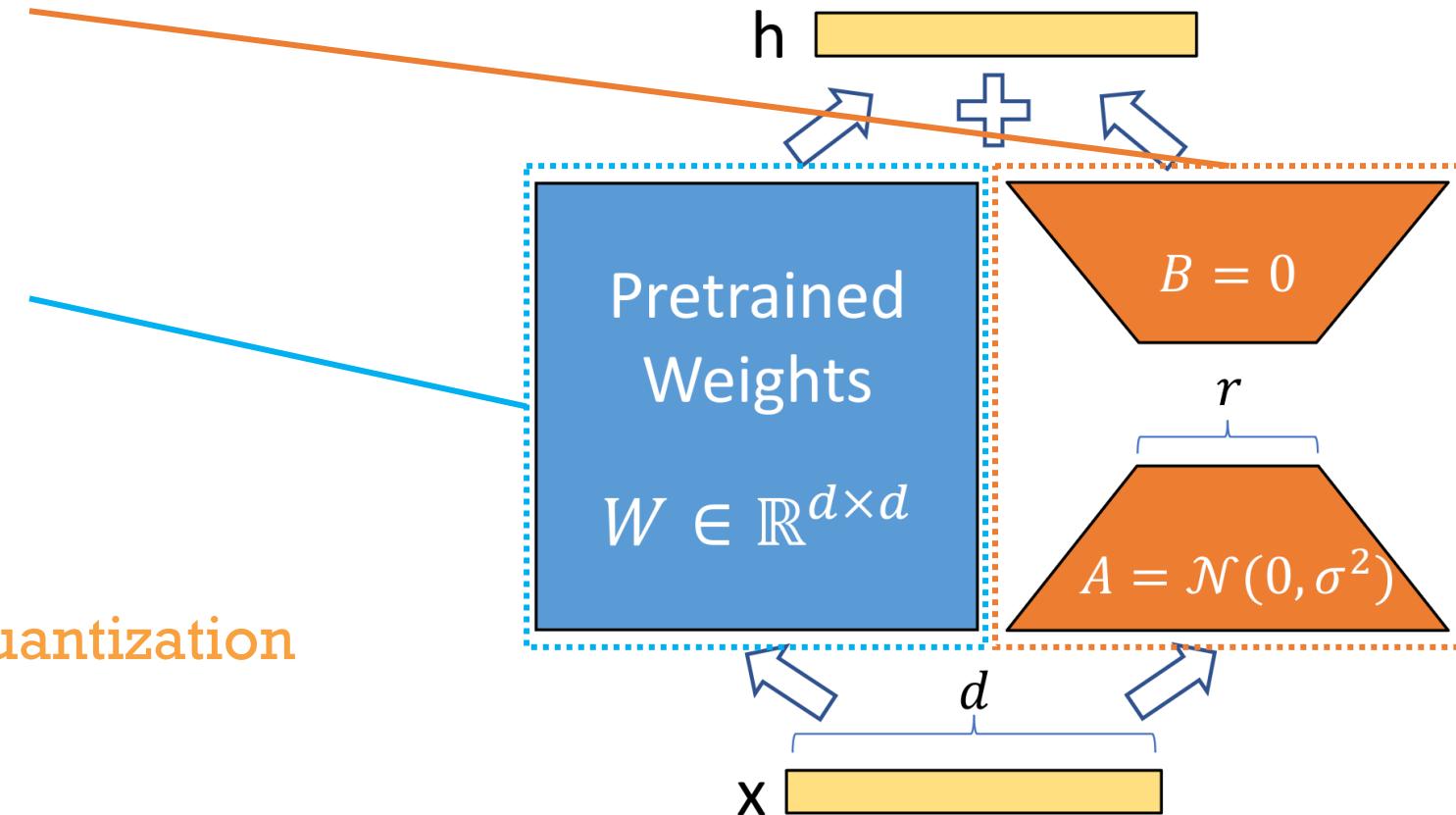


Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).

We saved parameters
here

But we still need to
perform the forward
pass through the entire
model

Solution: Model Quantization



Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).

Model Quantization

Model weights are typically stored in 16 or 32 Bits as floating point numbers

Idea: Map weights to a smaller number of approximate values

$$X^{Int8} = \text{round} \left(\frac{127}{\text{absmax}(X^{FP32})} X^{FP32} \right) = \text{round}(c^{FP32} \times X^{FP32})$$

1. Normalize weights with regard to largest occurring input value
2. Map values to integers
3. Store integer weights
4. Dequantize these values only during computation:

$$\text{dequant}(c^{FP32}, X^{FP32}) = \frac{X^{Int8}}{c^{FP32}} \approx X^{FP32}$$

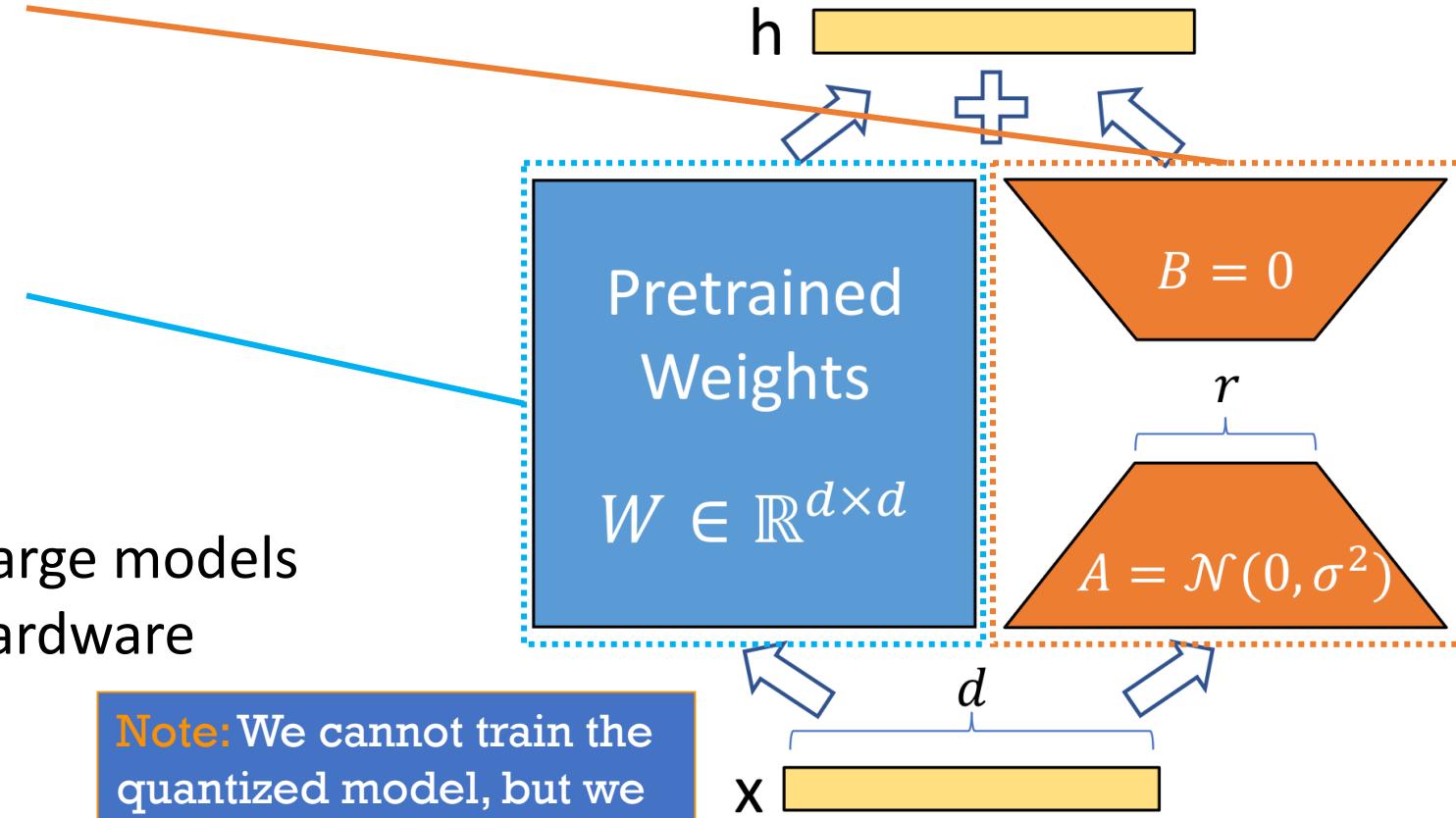
Outliers can cause issues → Split Matrix up into smaller chunks with their own c

We saved parameters
here

And significantly
reduced the memory
requirements on this
part

→ We can finetune many large models
(like LLMs) on consumer Hardware

Note: We cannot train the
quantized model, but we
can merge in the trained
adapters afterwards



Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).