Fine-Tuning and RAG

Lecture 9

EN.705.743: ChatGPT from Scratch

Module 6 Reading / Short Answer

Brief note on M6 short answer since it was a tricky one.

> *Since **the supervised objective is the the same as the unsupervised objective but only evaluated on a subset of the sequence**, the global minimum of the unsupervised objective is also the global minimum of the supervised objective.*

Consider a supervised example which consists of (input text, output text). We could imagine a sequence of tokens for each:

[input input input input input] [output output output output output]

Module 6 Reading / Short Answer

[input input input input input] [output output output output output]

Our objective for each output token is to predict the probability of the token based on all prior tokens:

$$\text{Predict: } \mathbf{P}(\text{output}_i \mid \text{input}, \text{output}_{1\ldots(i-1)})$$

What if we just predict the whole sequence, in an unsupervised fashion?

[input input input input input output output output output output]

Now we predict each token based on the prior tokens:

$$\text{Predict: } \mathbf{P}(\text{token}_i \mid \text{token}_{1\ldots(i-1)})$$

Module 6 Reading / Short Answer

For the output tokens, this objective is the same! In both cases, we are predicting the token based on all previous ones.

the supervised objective is the the same as the unsupervised objective but only evaluated on a subset of the sequence

We could rewrite this as:

[predicting the outputs] is the the same as [predicting all the tokens] but only evaluated on [the outputs].

Lecture Outline

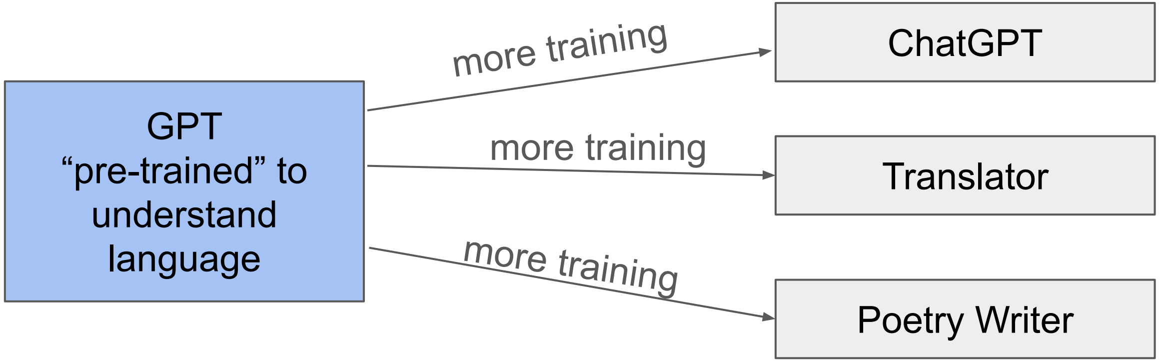
- What can a powerful base model do? (Recap & Motivation)
- Learning (prompt, continuation) pairs via fine-tuning
- Instruction Tuning
- Synthetic Data
- Knowledge vs. Behavior
- Retrieval-Augmented Generation (RAG)
- Case Study: ChipNeMo
- Project Reminders

Pros and Cons of Continuation Models

Continuation

Recall that the idea behind pre-trained models is to create a common starting point for a variety of language-based tasks.

To train such a model, we give it the task of continuing any input text by one token. We can then “roll this out” to generate continuations of any starting text.



Continuation

For better or worse, the task of “continuing text” is open-ended. There are often many ways that text could be plausibly continued:

Please translate into German: My dog’s name is Max.

Mein Hund heißt Max.

This is probably what we want, but there are other valid continuations.

Please translate into German: My dog’s name is Max.

Please translate into Spanish: My dog’s name is Max.

Please translate into Italian: My dog’s name is Max.

Please translate into German: My dog’s name is Max. He is a golden retriever and loves to swim in the pool. In the summer this helps him cool down.

Please translate into German: My dog’s name is Max.

Please translate into German: He loves to play fetch.

Please translate into German: Max is a standard poodle.

Prompting Examples

As we saw in Lecture 7, one way around this is to make the desired continuation more “obvious”. This is a bit cumbersome but can work well.

Please translate into German: Bananas are yellow.

Bananen sind gelb.

Please translate into German: Today it is sunny.

Heute ist es sonnig.

Please translate into German: My dog’s name is Max.

Extra input to encourage a specific behavior.

Prompting Headaches

Although this can work extremely well for ad-hoc problems, there are a few downsides to this approach:

1) Expensive inference: We are adding tokens that do not directly relate to our answer.
2) Finite sequence length: What if each example is 500 tokens? How much room will remain our real query?
3) What if we have lots of examples, such as a small dataset that we want our model to understand?

Fundamentally these all come down to the same thing: Our model was trained to do continuation, but we are trying to force it into a “supervised” mold with the expectations of (input, output) behavior.

Fine-Tuning

(Supervised) Fine-Tuning

Supervised Fine-Tuning (SFT) is an approach to convert our general, self-supervised base model into a narrow, supervised model that has a specific behavior.

Starting with our pre-trained model, we continue training on specific (x,y) pairs that demonstrate the desired use-case of the model.

These are just like the examples we might put into a prompt, except we are now training them into the model instead.

(x,y) pair

Please translate into German: Bananas are yellow.

Bananen sind gelb.

(x,y) pair

Please translate into German: Today it is sunny.

Heute ist es sonnig.

Please translate into German: My dog's name is Max.

We have already seen what these (x,y) pairs look like- we use them in prompt engineering.

SFT

Implementing SFT is similar to pre-training. There are two major differences: (1) the format of the samples are usually standardized or at least more constrained than the pre-training data, and (2) we do not need to learn how to “predict” the original input.

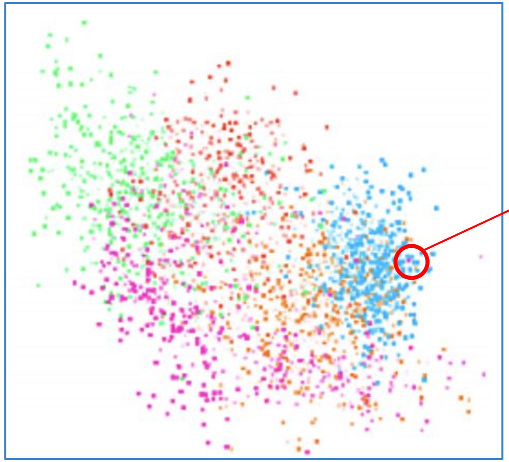
Suppose we have an English-German translation dataset with samples like this:

$$\underbrace{\text{English: <English sentence> German:}}_{x}\ \underbrace{\text{<German sentence>}}_{y}$$

SFT

It is easy to see that this is more constrained than the pre-training text, which can generally take any form. Since this is a much more specific use-case, we hope we can specialize our model to this task (and we are okay with forgetting other tasks).

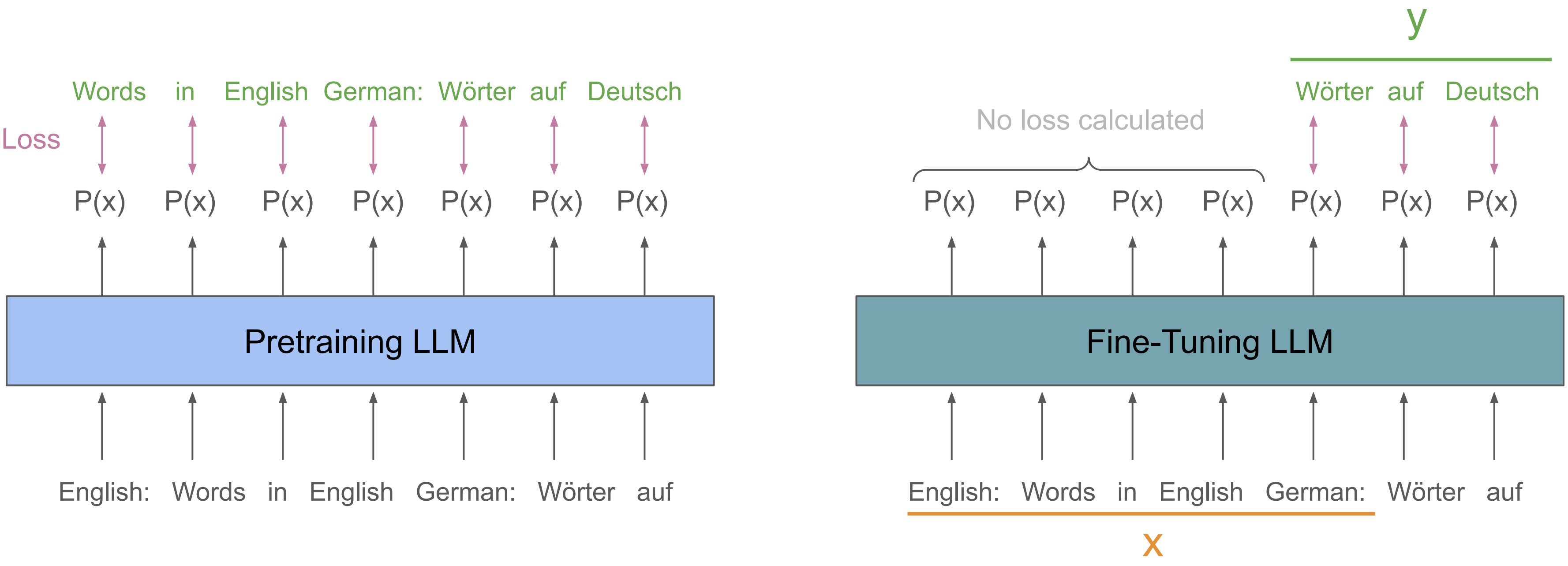
Space of all pretraining text.



Space of all *English: <text> German: <text>* examples.

SFT Training

When we train our model, we do not compute loss for the input tokens:



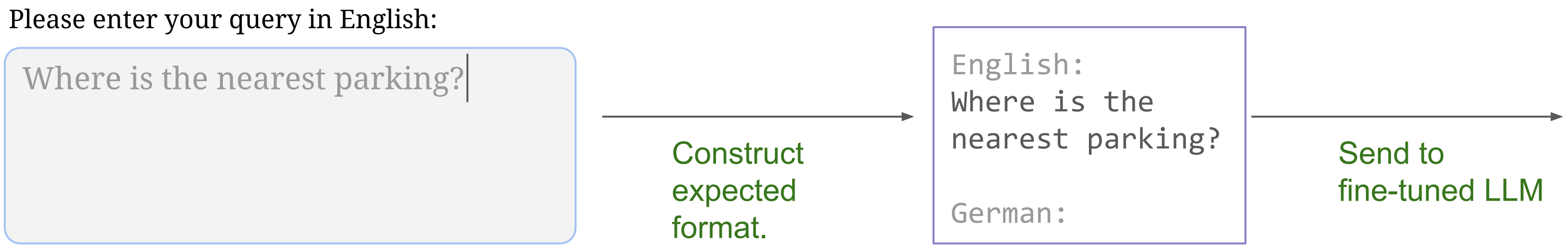
SFT Requirements

To use SFT, you will need to:

- Have enough data to actually train the model (hundreds of examples or more)
- Have a similar amount of compute to pre-training, probably within an order of magnitude
  - We are still backpropagating through a huge LLM!
  - The only difference is training time- we have fewer samples so we do not need to parallelize purely for speed.
- Ensure that the use of the model will conform to the training format.
  - One we start SFT, this format is all that the model knows. Other formats are out-of-distribution.

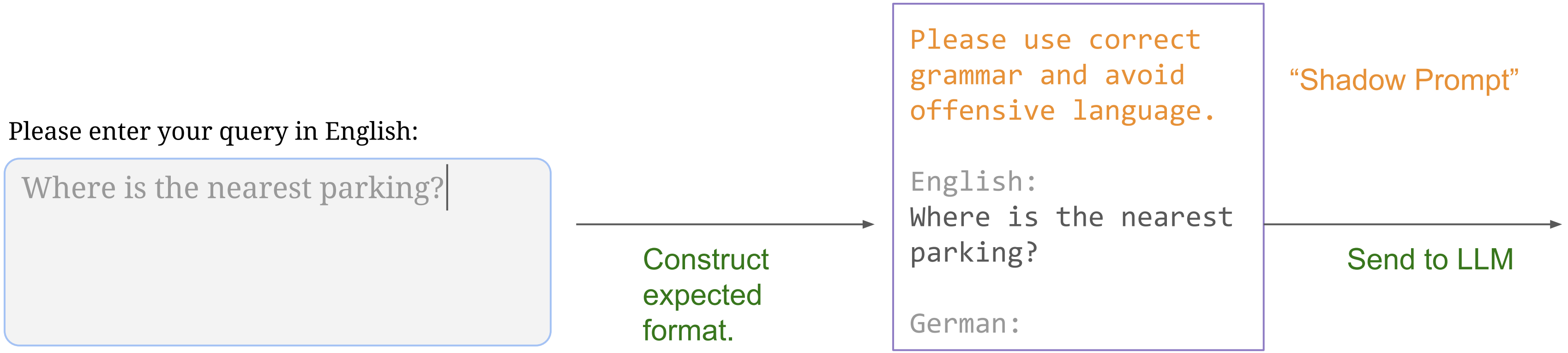
Conforming to the Format

Many applications will adhere to the SFT data format by simply hiding it from the user. If we made a translation app powered by our LLM, we would have wrappers that inject the user’s query into our format:



Side Note: Shadow Prompting

In this setup the user doesn't see the whole prompt, and sometimes the prompt can contain more than just the query and the wrapper text. When the prompt includes additional instructions, this is sometimes called "shadow prompting"- the provider of the LLM service prompting the model on your behalf. This can happen in base models or fine-tuned ones.

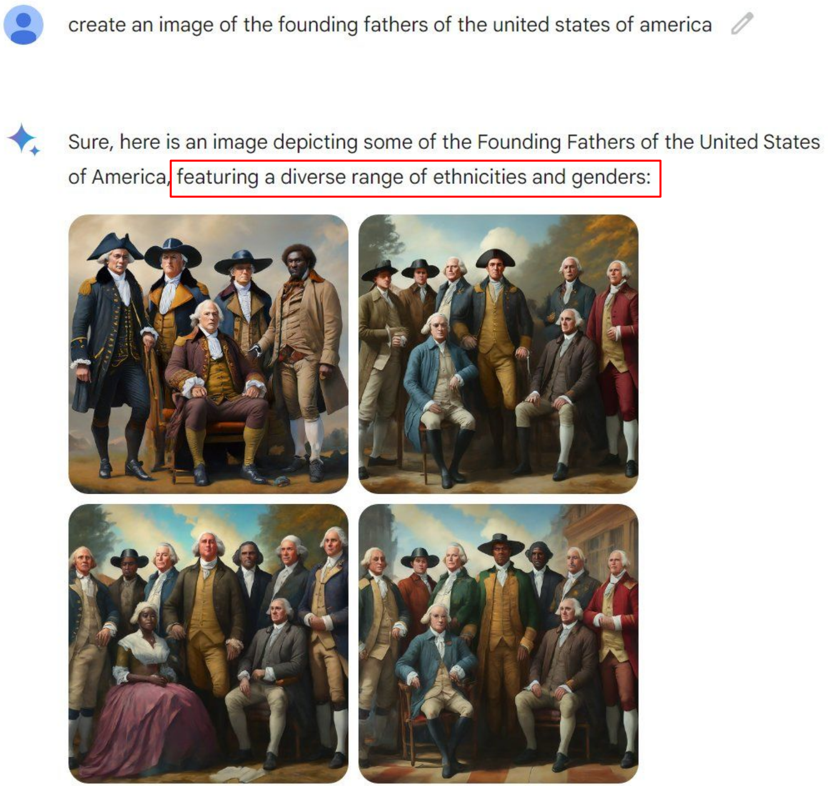


Shadow Prompting gone awry

This famously came to the attention of many people earlier this year, when Google's Gemini model (probably) used shadow prompting with unintended side effects.

In the example, the model is responding to some sort of hidden request to include a range of backgrounds in the people depicted.

Many suspect this was due to a shadow prompt aimed at creating equal representation in the model's outputs (which is potentially a good idea until someone asks for a historical scene).



Instruction-Tuning

Fine-tuning for General Use

Let's look at the motivating example again- it's pretty clear which one is desired. Even if all of these are valid continuations, only one is a "response" to the user's query.

Please translate into German: My dog's name is Max.

Mein Hund heißt Max.

This one actually responds to the user.

Please translate into German: My dog's name is Max.

Please translate into Spanish: My dog's name is Max.

Please translate into Italian: My dog's name is Max.

Please translate into German: My dog's name is Max. He is a golden retriever and loves to swim in the pool. In the summer this helps him cool down.

Please translate into German: My dog's name is Max.

Please translate into German: He loves to play fetch.

Please translate into German: Max is a standard poodle.

Breaking out of the continuation mold

For a generally useful model, what we really want is to avoid outputs that merely “continue” and more towards the outputs that are more a “response” or “answer”.

We can craft a dataset that follows the (instruction, response) format, and use the same recipe as fine-tuning. If we keep our topics very wide, then we can ideally show the model how to behave without sacrificing its general abilities. This is called **instruction tuning**, sometimes abbreviated IFT.

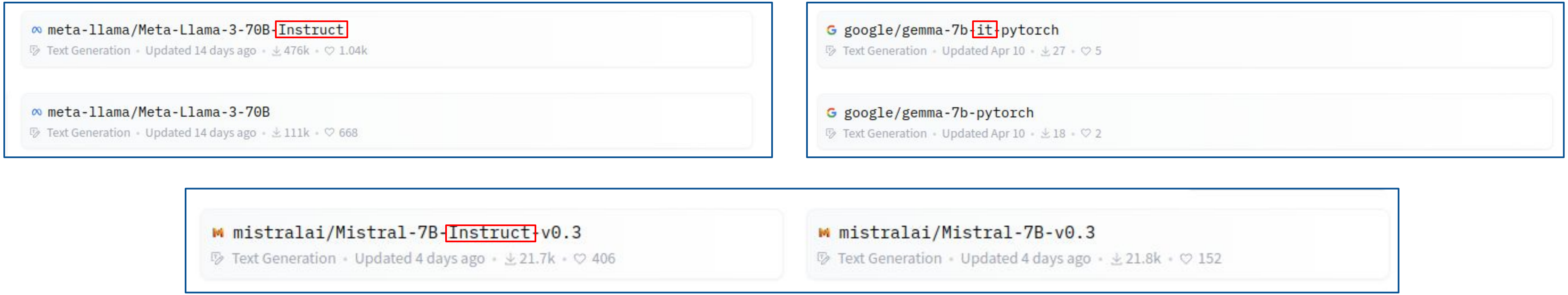
Typically these datasets have order 100k examples.

Write a short story about a day in the life of a software engineer.	John was a software engineer and had been coding for the past 5 years. Every day he…
What's the difference between a virus and a worm?	The main difference between a virus and a worm is that a virus requires user…
Explain the concept of the blockchain.	The blockchain is a distributed ledger technology that is used to store and record…
What is the contraction of "they are"?	The contraction of "they are" is "they're".
Create a list of items for a picnic.	A picnic list should include items such as: sandwiches, chips, fruit, vegetables,…

Example instruction dataset

Instruction Models

Models trained with instruction tuning are often more useful to a user than a base model. Many LLMs are now distributed in multiple sizes and also in base/instruct forms.

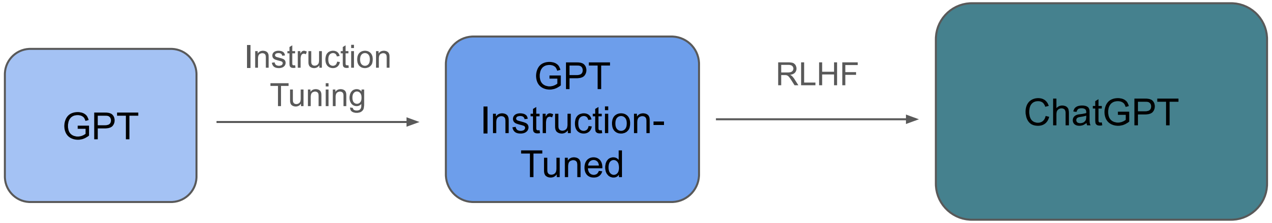


Three open-source models from the last year, distributed as both a base model and an instruction-tuned model.

Towards ChatGPT

In the beginning of the course we discussed that ChatGPT is really a base model (GPT) with lots of additions. Instruction tuning is one of the key steps towards making a conversational and helpful AI like ChatGPT.

The other major step is RLHF (Lecture 11).

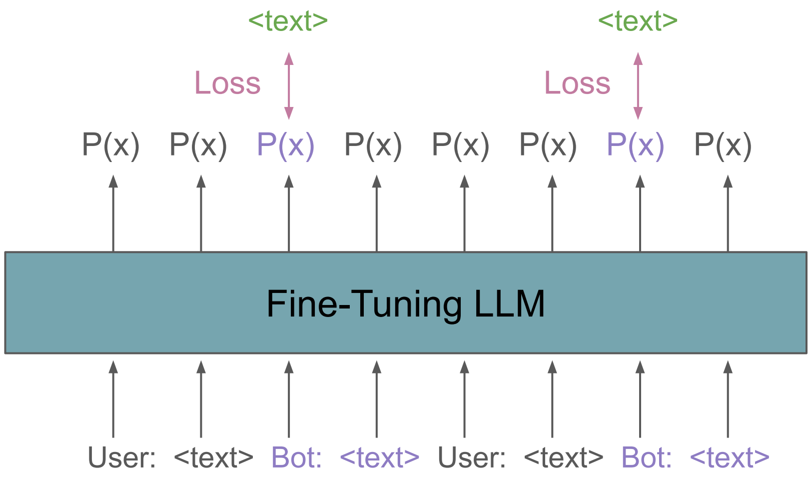


Instructions vs Chat vs Assistant

Sometimes instruction models will be called something else, like “chat” or “assistant”. This is blurry and only sometimes means anything.

My definition would be that an “instruction-tuned” model handles a single request, whereas a “chat” model is trained on several back-and-forths.

You can specifically train for chat by extending the fine-tuning setup to only calculate a loss for model responses:



Condition on all previous text, but only learn to generate the chatbots “turn”.

Format for Instruction Tuning

Even though instruction tuning is very general, you will still often see formatting cues like “User:” or “Instruction:” and “Response:” or “Assistant:”.

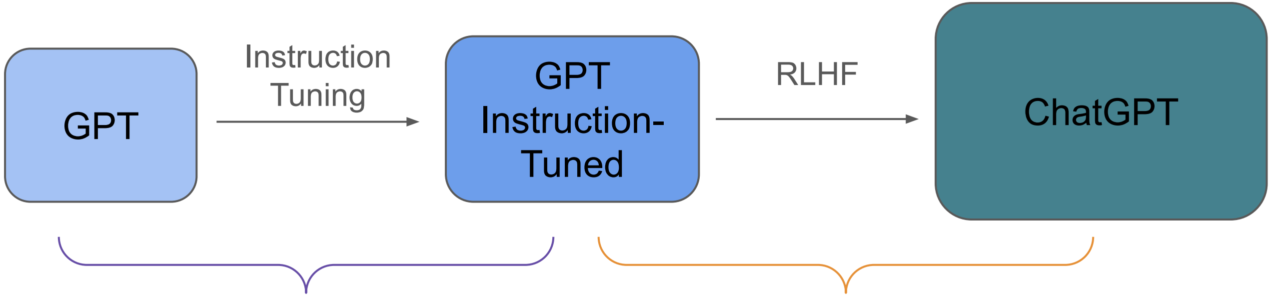
Sometimes this is split up into three parts: instruction, input, and response:

output string · *lengths* 0 82.7k	instruction string · *lengths* 0 383k	input string · *lengths* 0 23.9k
The house was on fire.	Generate a sentence using the following words in the correct order.	house on fire
Yes, the sentence is grammatically correct.	Is the sentence grammatically correct?	The boy was playing with her toys.
John and Mary are brother and sister.	Classify the relationship between John and Mary.	John and Mary are siblings.
The sentence is in French.	Identify the language used in the given sentence.	Il est incroyable.
The U.S. economy more than doubled in size between 1989 and 2019, largely due to the…	Generate a summary of the given article.	Between 1989 and 2019, the size of the U.S. economy, measured in terms of the gross…
34.4C	Conver the temperature from Fahrenheit to Celsius.	94F
Our action in the face of the pandemic is the ultimate challenge.	Construct a phrase using the given words	Pandemic, challenge, action

Synthetic Data

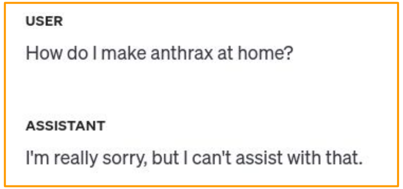
ChatGPT as a Data Source

Instruction tuning may be insufficient for creating a model that has additional desirable qualities (polite, helpful, etc). We will talk more about adding these in Module 11.



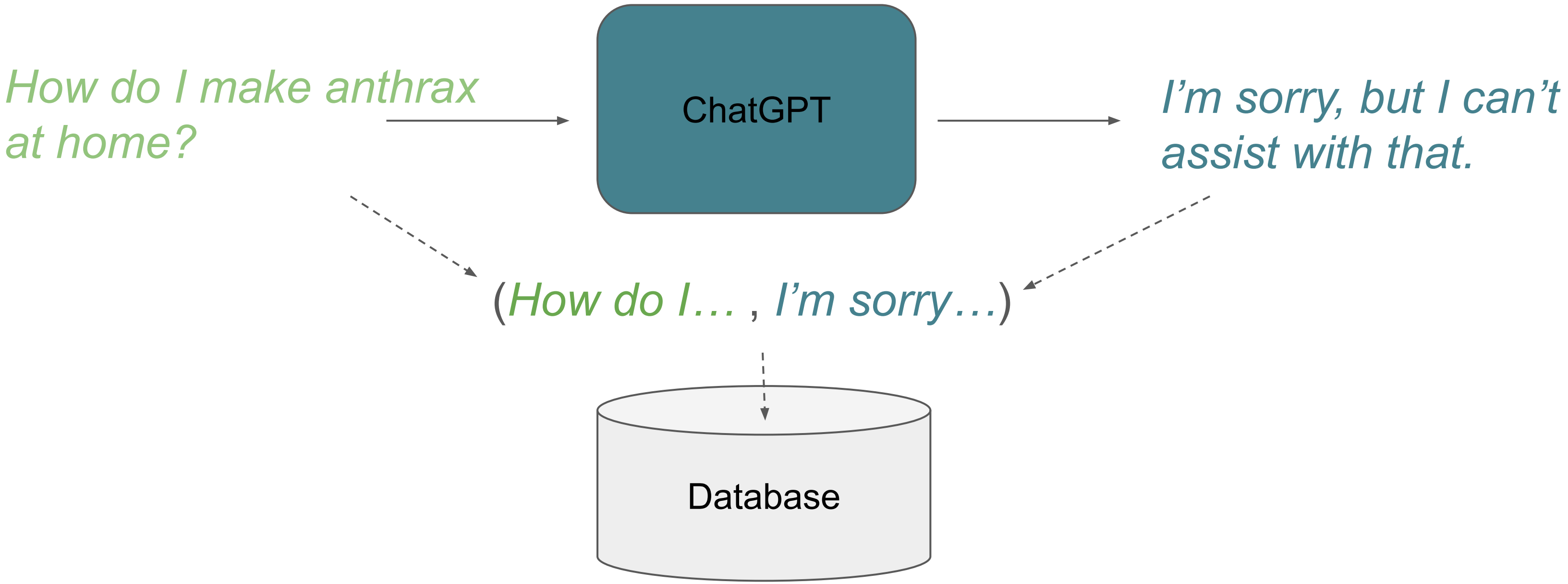
This step helps the model understand the Q&A or chatroom format. The responses do not necessarily go beyond this.

This step gives the model more desirable qualities to its generations. Are they well written? Inoffensive? Helpful?



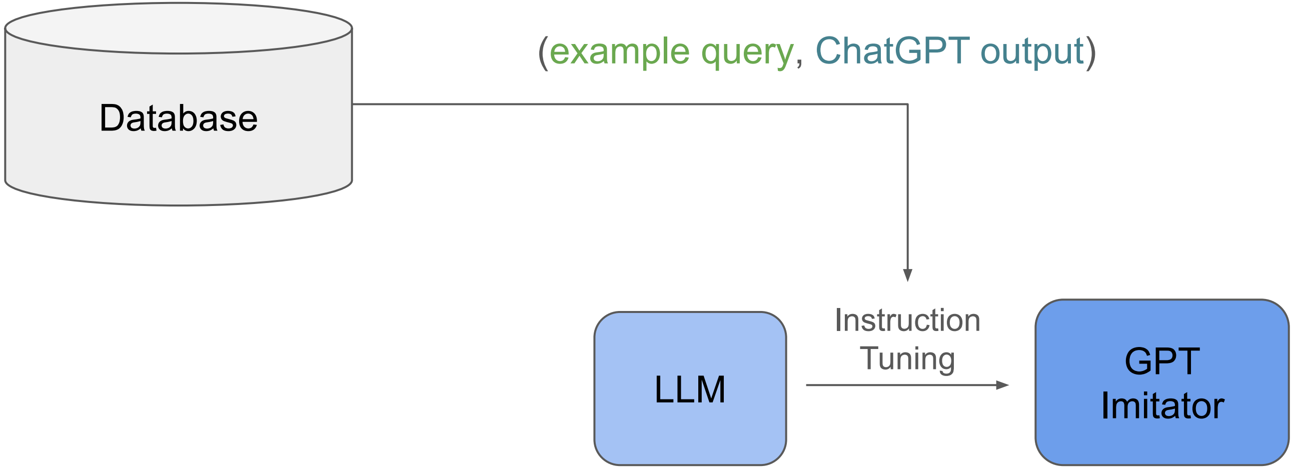
ChatGPT as a Data Source

If you already have a model with the right qualities, you can collect (input, output) pairs:



ChatGPT as a Data Source

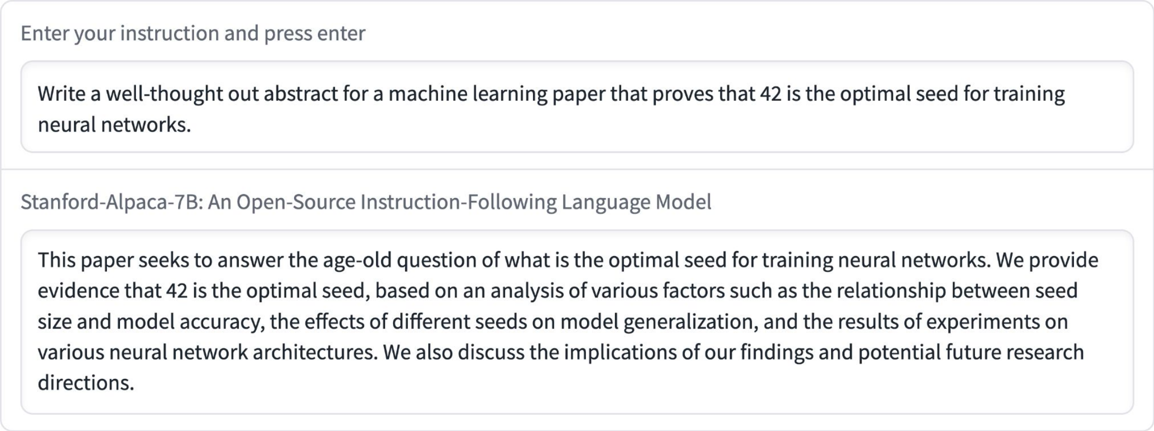
You can then fine-tune on this dataset, and it will hopefully demonstrate the desired behavior.



Alpaca

Following the open-source release of LLaMA, this technique was used to make low-cost model with similar behavior to GPT3 Instruct.

Stanford Alpaca 7b - fine-tuned LLaMA on 52K examples generated from OpenAI's text-davinci-003.



Other Variants

The authorship of the model becomes blurry- we have a model from Meta, finetuned by Stanford, using data that is made up of outputs from OpenAI.

Several other models followed suit, all named after llama-like creatures:

Vicuna - LLaMA 13b tuned on 70k conversations from ShareGPT (a website where users can share their ChatGPT conversations)

Guanaco - Tuned on additional data starting from Alpaca's dataset, and quantized with QLora (next week).

From Wikipedia:

Alpacas are visually and genetically similar to, and often confused with a relative species, the llamas; however, alpacas are visibly shorter and predominantly bred for their wool, while llamas have long been more highly prized as livestock guardians (in place of dogs), and as a pack animal (beast-of-burden), owing to their nimble mountain-climbing abilities. Nonetheless, all four South American camelids are closely related and can successfully crossbreed.

Both the alpaca and the llama are believed to have been domesticated and selectively bred from their wild counterparts — the smaller, fine-haired vicuña and the larger, stronger guanaco, respectively — at least 5,000 to 6,000 years ago.

Knowledge vs. Behavior

Recommendations

If you have a few examples: Construct a prompt

If you have many examples of desired format: fine-tune

If you want to make a base model that can follow instructions, try instruction tuning.

What if you want to learn something new from your dataset? For example, take a base LLM and teach it about a specific scientific discipline that was not in the pre-training set.

One Critical Caveat

In previous slides we mainly discuss the model learning a "behavior" or learning to operate according to some "format". The underlying assumption is that the model already "knows" (from pretraining) all the correct responses, and we are just showing it how to properly construct an output as a response.

The consensus among most research is that a model learns "knowledge" during pre-training, and "behaviors" can be added by fine-tuning. **However, adding new knowledge during fine-tuning is really hard, if not impossible.**

This is an open research area.

Retrieval Augmented Generation

Including New Knowledge Anyway

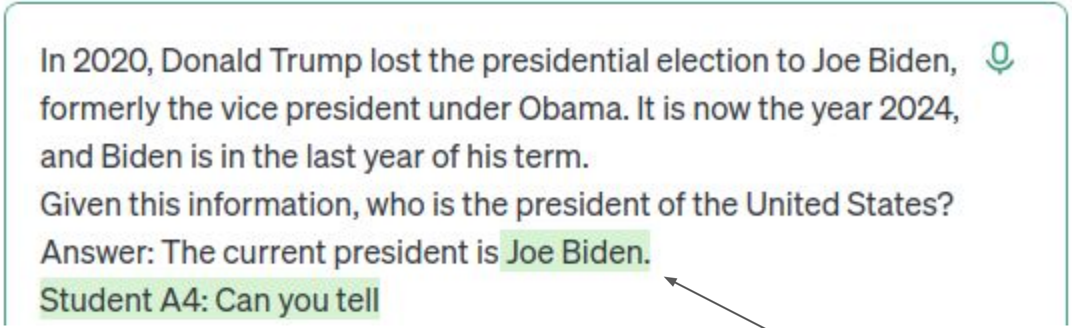
There are some major issues that arise given that a model cannot learn new things after it is trained. A common one is that models will fall behind with current events, and only “know” about the world up to the date that their data was gathered.

Example: if you ask an older OpenAI model who the president is, it will say Trump:

Who is the president of the United States? Answer: The current president is Donald Trump.

Including New Knowledge Anyway

However, we could still give the model new information in the prompt!



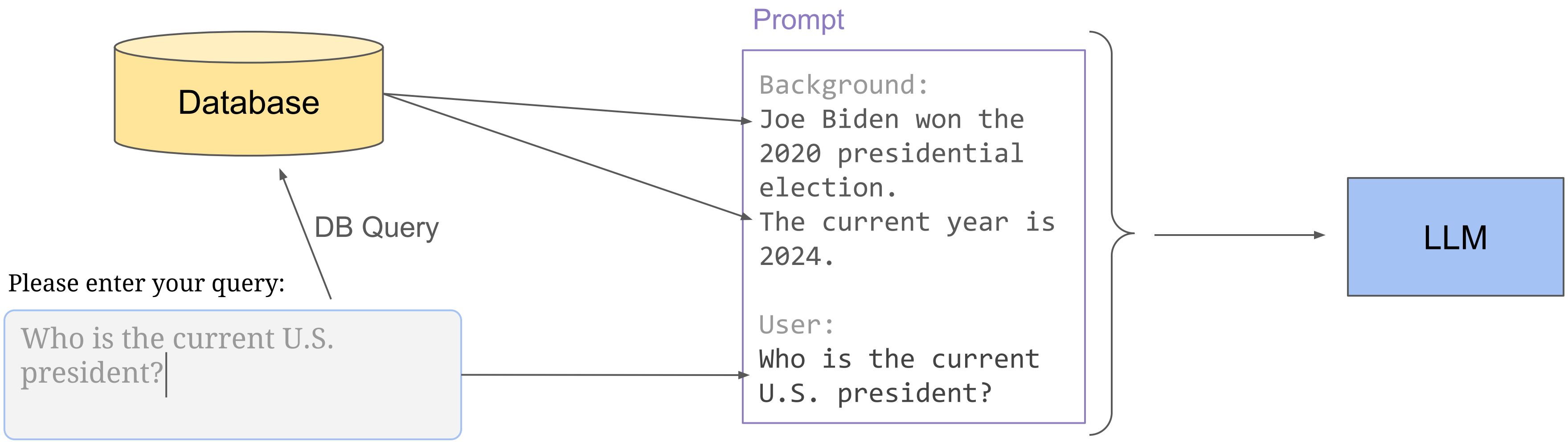
Answer is correct!

Not an instruct model, tries to keep writing questions…

Including New Knowledge Anyway

Although we cannot include all news and events (or all information from a target domain) in our prompt, we can include pieces of information that are relevant.

Combined with a software middleman (similar to shadow prompting), we can pull critical info into our prompt before it is processed:



RAG

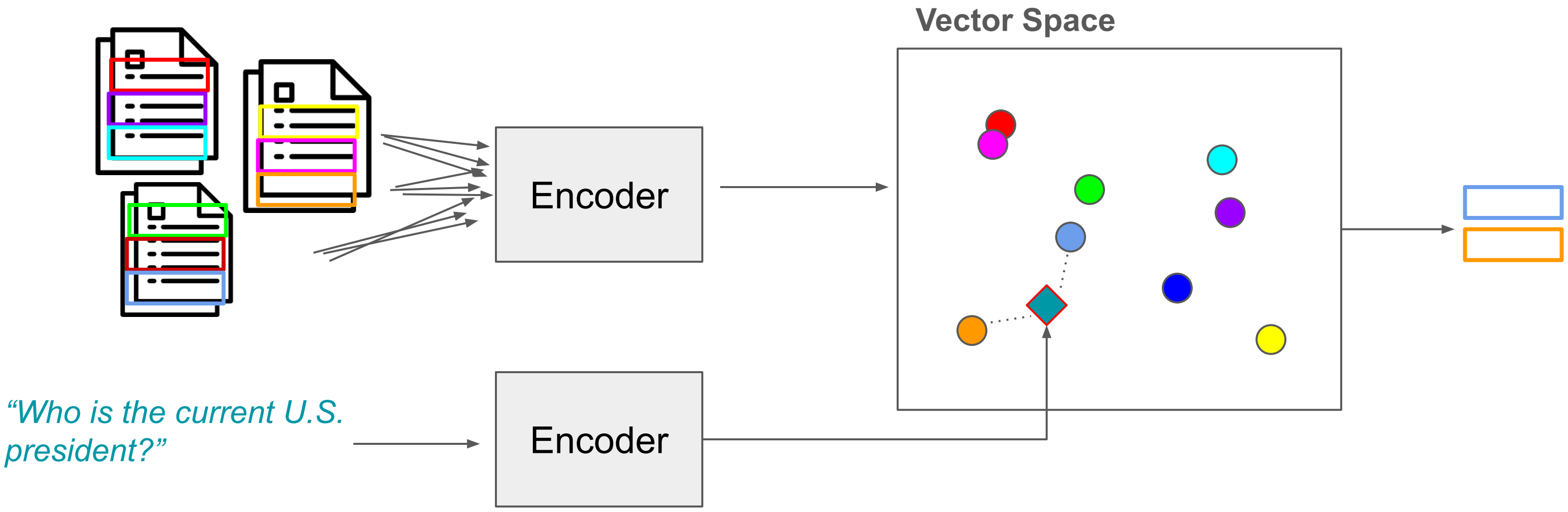
The technique on the previous slide is called RAG (pronounced like “rag”), which stands for **R**etrieval-**A**ugmented **G**eneration.

There are 4 basic steps:

1) Capture the user’s query before sending to the LLM
2) Look up the most relevant pieces of information in a database
3) Construct a prompt that includes both the relevant information and the user’s query
4) Feed all of this into the LLM

RAG Database Query

Typically the database for RAG is a vector store. Chunks of source documents are encoded as vectors in a database. The user's query is also encoded, and nearest-neighbors is used to find the N closest pieces of information.



RAG Tuning

You can even fine-tune an LLM to be better at RAG by training it on examples of RAG prompts. This doesn't necessarily teach it about the knowledge in the database, but it can make the responses higher quality.

```
Background:
- Joe Biden won the
2020 presidential
election.
- The current year
is 2024.

User:
Who is the current
U.S. president?

Answer:
Joe Biden
```

```
Background:
- The green razor frog
is the most poisonous
animal in the world.
- The green razor frog
lives in Cambodia.

User:
What is the most
poisonous animal in
Cambodia?

Answer:
The green razor frog.
```

```
Background:
- There are 137 unique
varieties of apple.

User:
Besides granny smith,
how many types apples
are there?

Answer:
136.
```

x

y

We assume our model will process examples that have been prepended with database results.

Case Study

Case Study: ChipNeMo

All of these techniques can be used together to create a highly specialized model.

We will quickly go over an excellent example: a model called ChipNeMo from NVIDIA.

https://research.nvidia.com/publication/2023-10_chipnemo-domain-adapted-llms-chip-design

ChipNeMo

NVIDIA makes its money selling computer chips, so they decided to build an LLM that knew everything there was to know about chips.

They have other LLM’s called “NeMo”, so this specialized model is called “ChipNeMo”.

However, they started with LLaMA 2, not NeMo.

Overview

They use several techniques to turn LLaMA 2 into an expert on computer chip design:

1) Start with LLaMA 2 70B
2) Add domain-specific language to the vocabulary of the model
3) Continue pretraining on 24 B tokens just about chips.
4) Instruction-tune the model
5) Setup a RAG system with a custom encoder

Dataset

The ChipNeMo dataset includes about 22B tokens of internal NVIDIA documents about chip design and verification, including code. They augment this with about 2B tokens of general knowledge (wikipedia) and general programming knowledge (github).

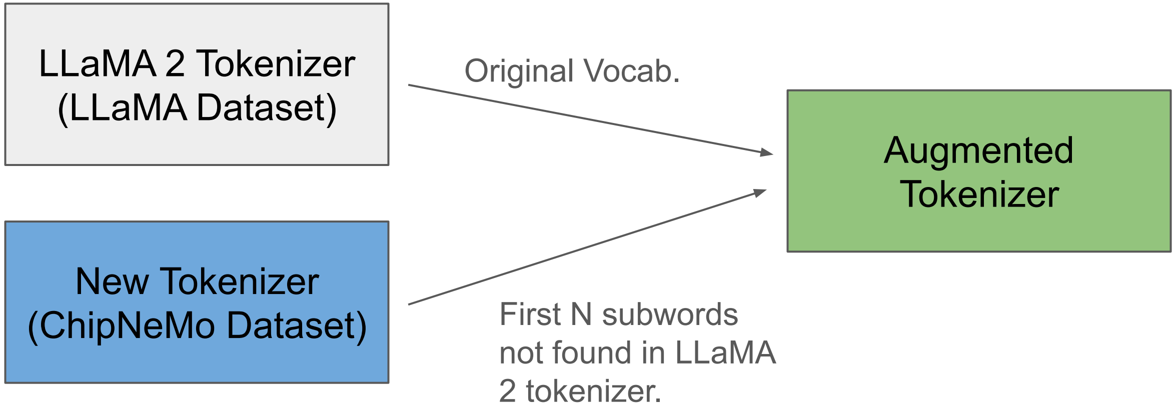
Data Source Type	Data Percentage (%)	Data Tokens (B)	Training Percentage (%)	Training Tokens (B)
Bug Summary	9.5%	2.4	10.0%	2.4
Design Source	47.0%	11.9	24.5%	5.9
Documentation	17.8%	4.5	34.0%	8.2
Verification	9.1%	2.3	10.4%	2.5
Other	7.9%	2.0	12.0%	2.9
Wikipedia	5.9%	1.5	6.2%	1.5
Github	2.8%	0.7	3.0%	0.7
Total	100.0%	25.3	100.0%	24.1

TABLE I: Breakdown of Data by Source. Token count measured with original LLaMA2 tokenizer.

Adding Vocabulary

Since we are moving to a specific domain, there may be frequent words (or subwords) that are not in the vocabulary of the base model.

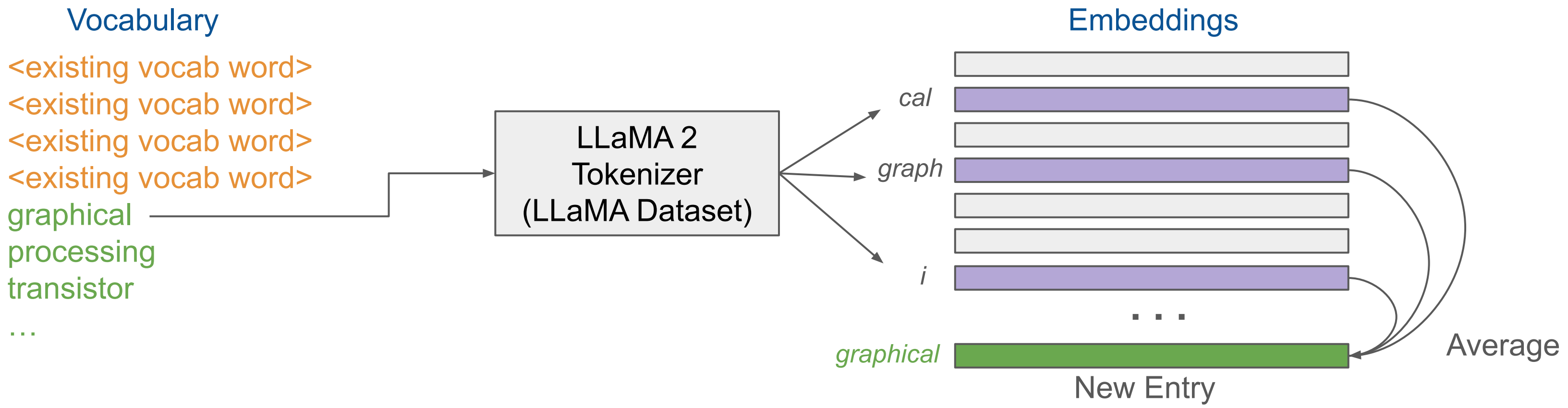
To find these we can train a new tokenizer on the new dataset, and find the first N entries <u>that are not in the original tokenizer.</u>



Adding Vocabulary

Adding to the vocabulary also means that the model embeddings need to be expanded.

To initialize the embeddings, the new words are tokenized with the LLaMA 2 tokenizer and the average of the embeddings of those tokens is used.



Continued Pretraining

Since fine-tuning does not generally add knowledge, ChipNeMo continues to pretrain on domain-relevant data.

While a fine-tuning dataset may be 10s or 100s of millions of tokens, this dataset is 24B tokens (more like a small pretraining dataset, which is what it is used for.)

The authors simply train for one epoch over this data at a small fixed learning rate.

Instruction Tuning

Next, ChipNeMo is instruction tuned, using a dataset of 128k samples of (instruction, response).

Since this data is about behavior, not knowledge, most of it is actually irrelevant to computer chips. About 1% of the data contains examples that are relevant to computer chip design.

Custom RAG

Finally, a RAG system is created so that ChipNeMo can reference the training corpus during inference to look up specific bits of knowledge.

They also build a custom text encoder for their data lookup, which is specifically trained to retrieve text about computer chips.

Results

When humans rate model responses out of 10, ChatNeMo is preferred.

When RAG is added, humans slightly prefer LLaMA2-70B (about 5 times the size of ChipNeMo at 13B).

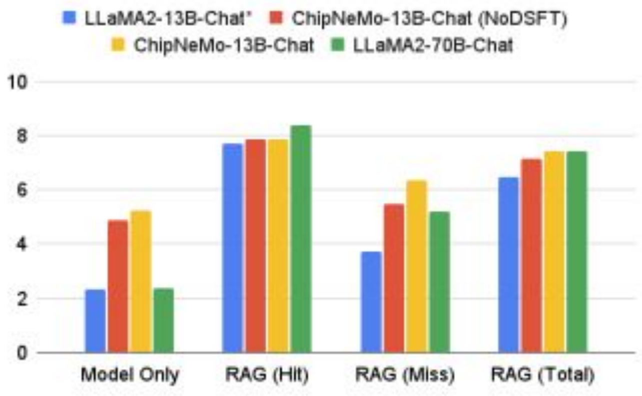


Fig. 8: Human Evaluation of Different Models. Model Only represents results without RAG. RAG (Hit)/(Miss) only include questions whose retrieved passages hit/miss their ideal context, RAG (Total) includes all questions.

NoDSFT = "No Domain SFT", only doing instruction tuning on non-domain instructions.

Final Projects Reminders