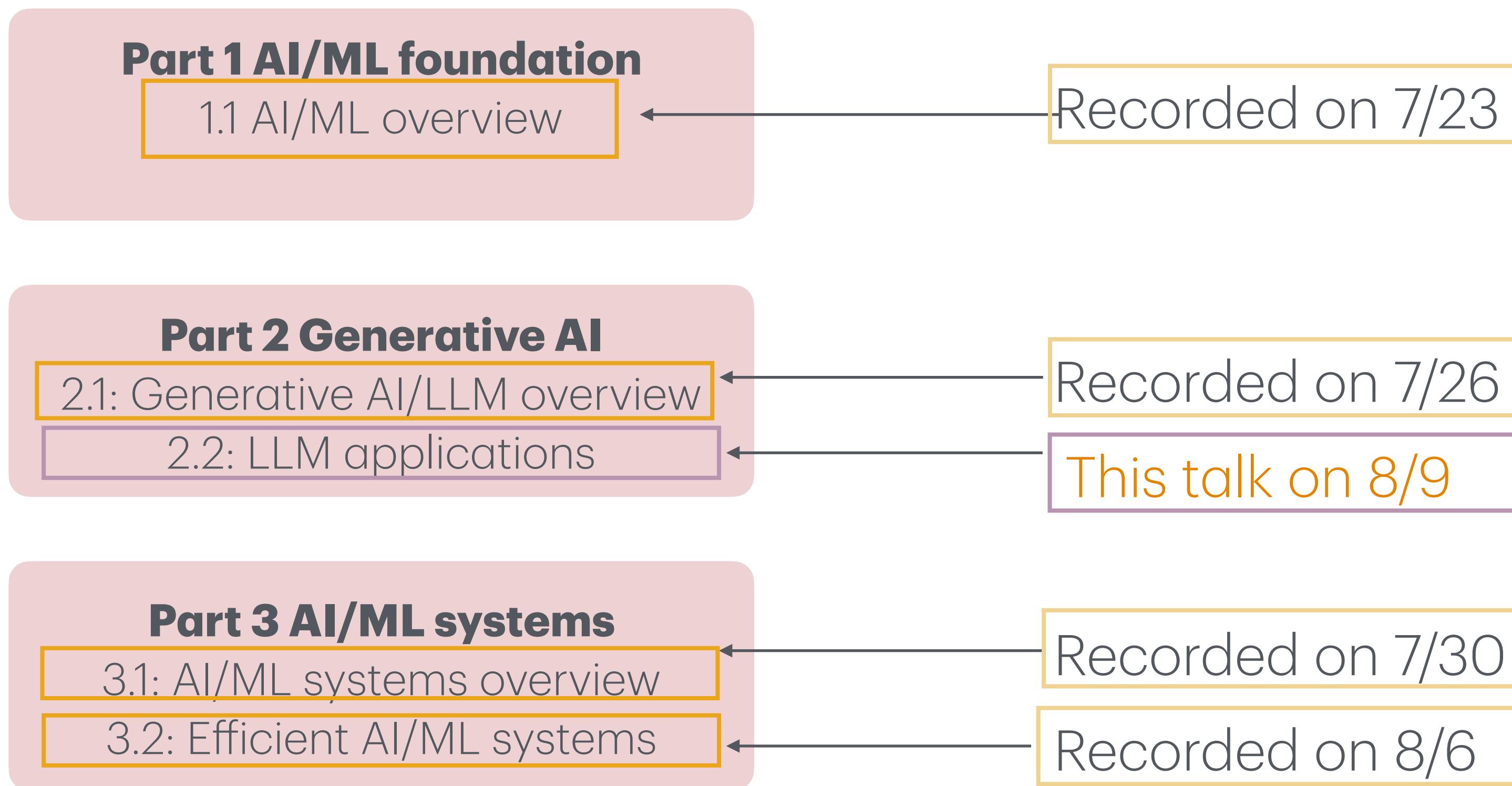


Part 2. Generative AI

2.2: LLM applications

AI learning modules



2. Generative AI

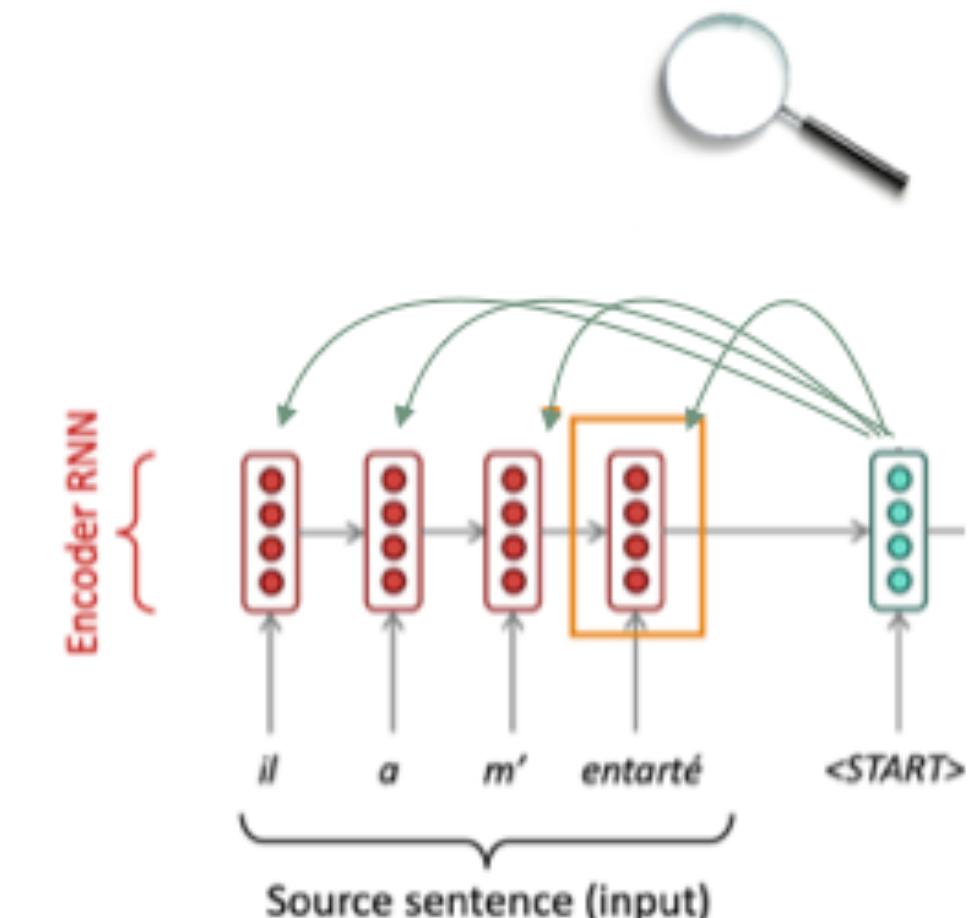
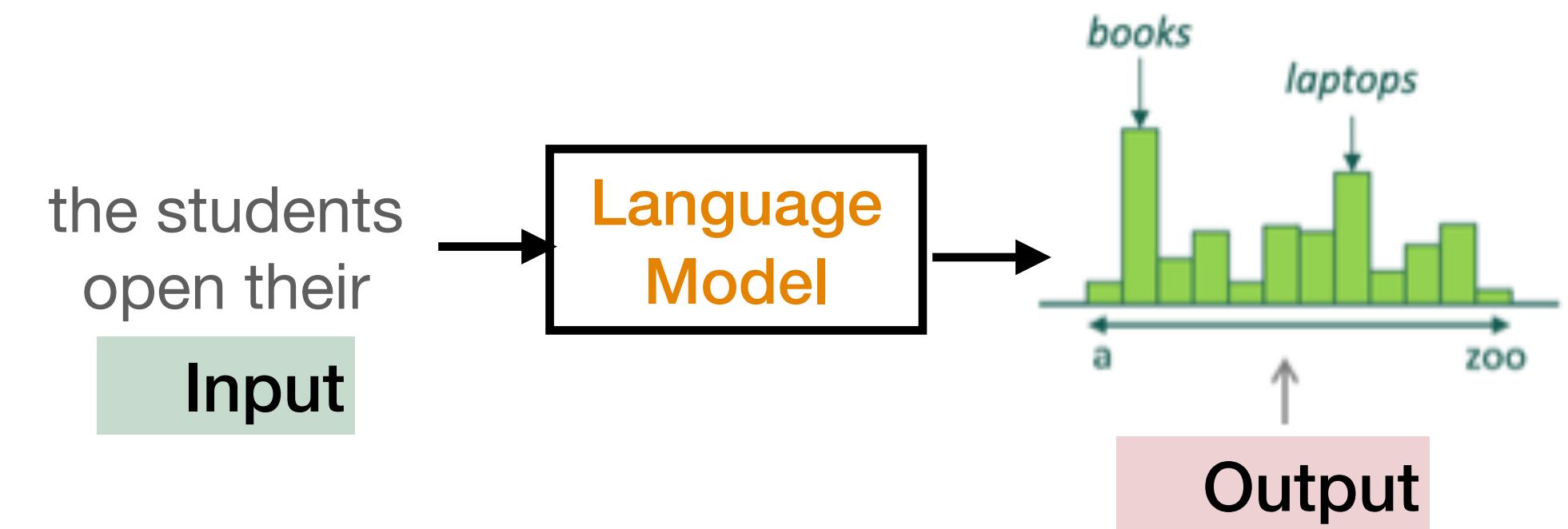
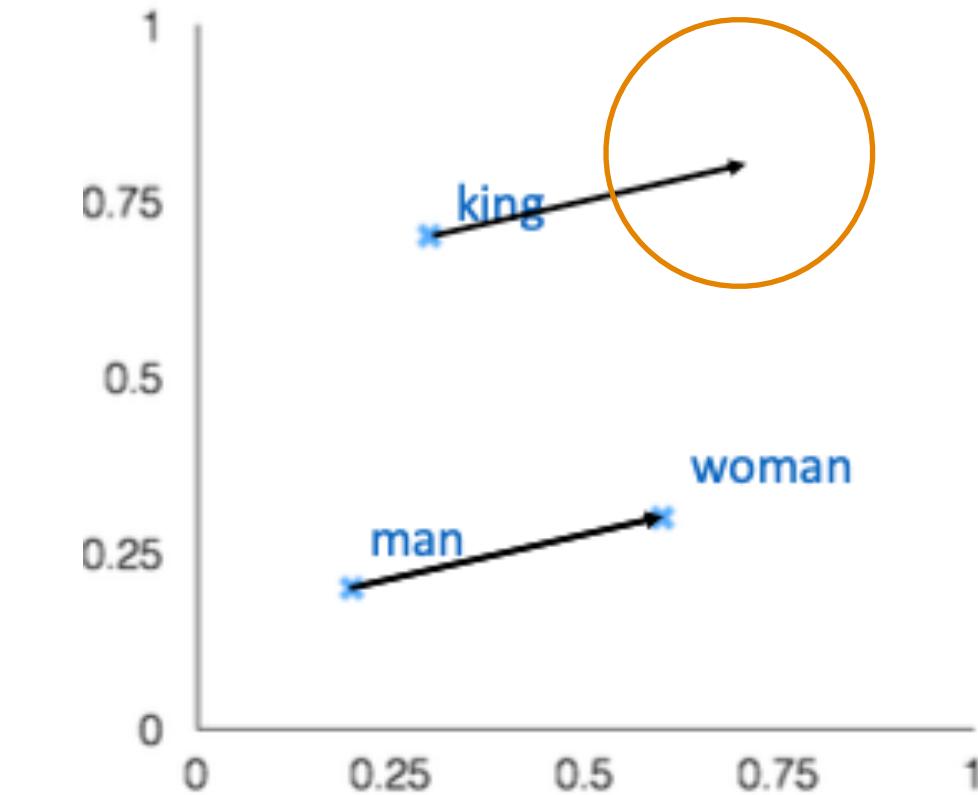
Contents

2.2: LLM applications

Review

From Part 2.1: Generative AI/LLM overview

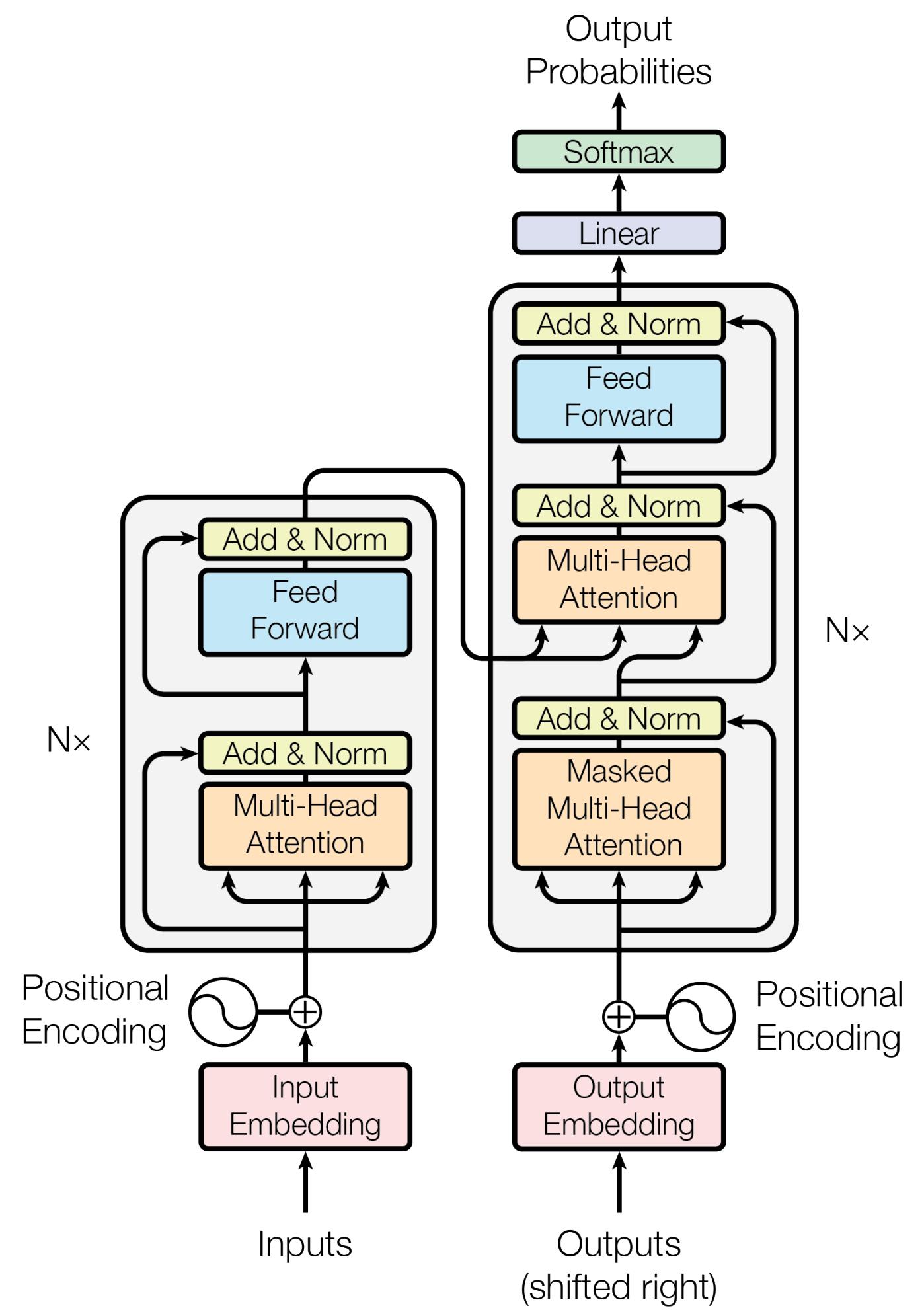
- **Word2vec**: a model that trained to converts words to vector (word embeddings)
- **LM (language model)**: predicts next word given input
- **RNN**: type of neural network that can process sequential data, which allows stateful computation since it uses hidden state from previous time. Any input sequence length can be processed
- **Sequence to sequence model**: consists of encoder block and decoder block and used for neural machine translation model
- **Attention**: a general method in neural network that solves information bottleneck, which directly computes the similarities with all input and focus on the specific part of the input



Review

From Part 2.1: Generative AI/LLM overview

- **Transformer architecture/model**: improves the issues with RNN, such as providing attention paths, enabling parallel computations.
- **Transformer block**: a core block for the Transformer architecture, which includes *multi-headed self attention* and *feedforward network*.
- **Self-supervision**: in language model training, it uses the input text to predict the next word without label.
- **Pre-training**: teaches the model how the world works in general by using self-supervision training
- **Fine-tuning**: customizes it to a specific task to easier to use.
- **Generation (LM inference)**: the model continues its input (prompt) sequence.

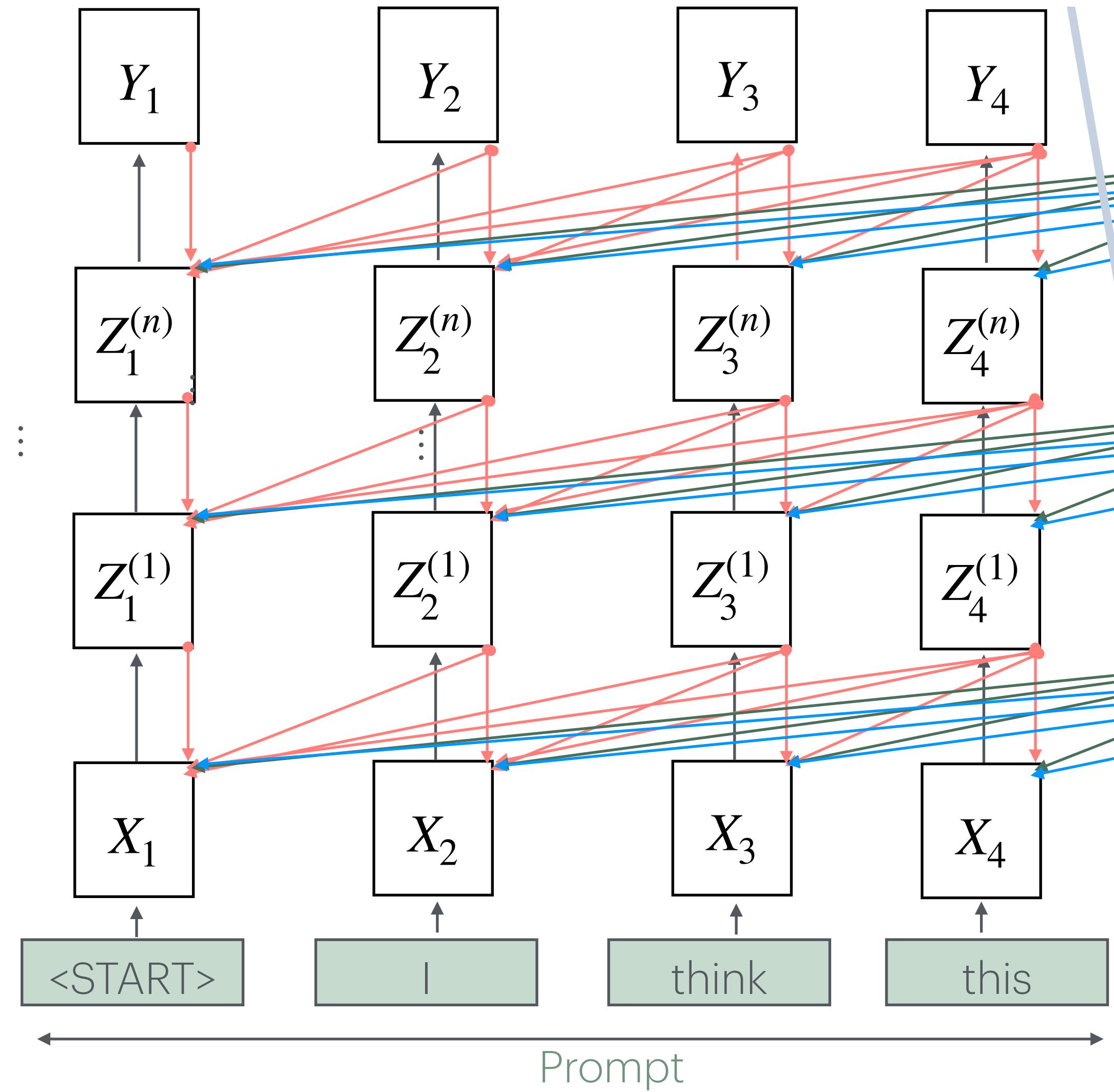


Review

From Part 3.2: Efficient AI/ML systems

Language model is built with transformer blocks

$$Z_t^{(i+1)} = \text{Transformer_block}(Z_t^{(i)})$$

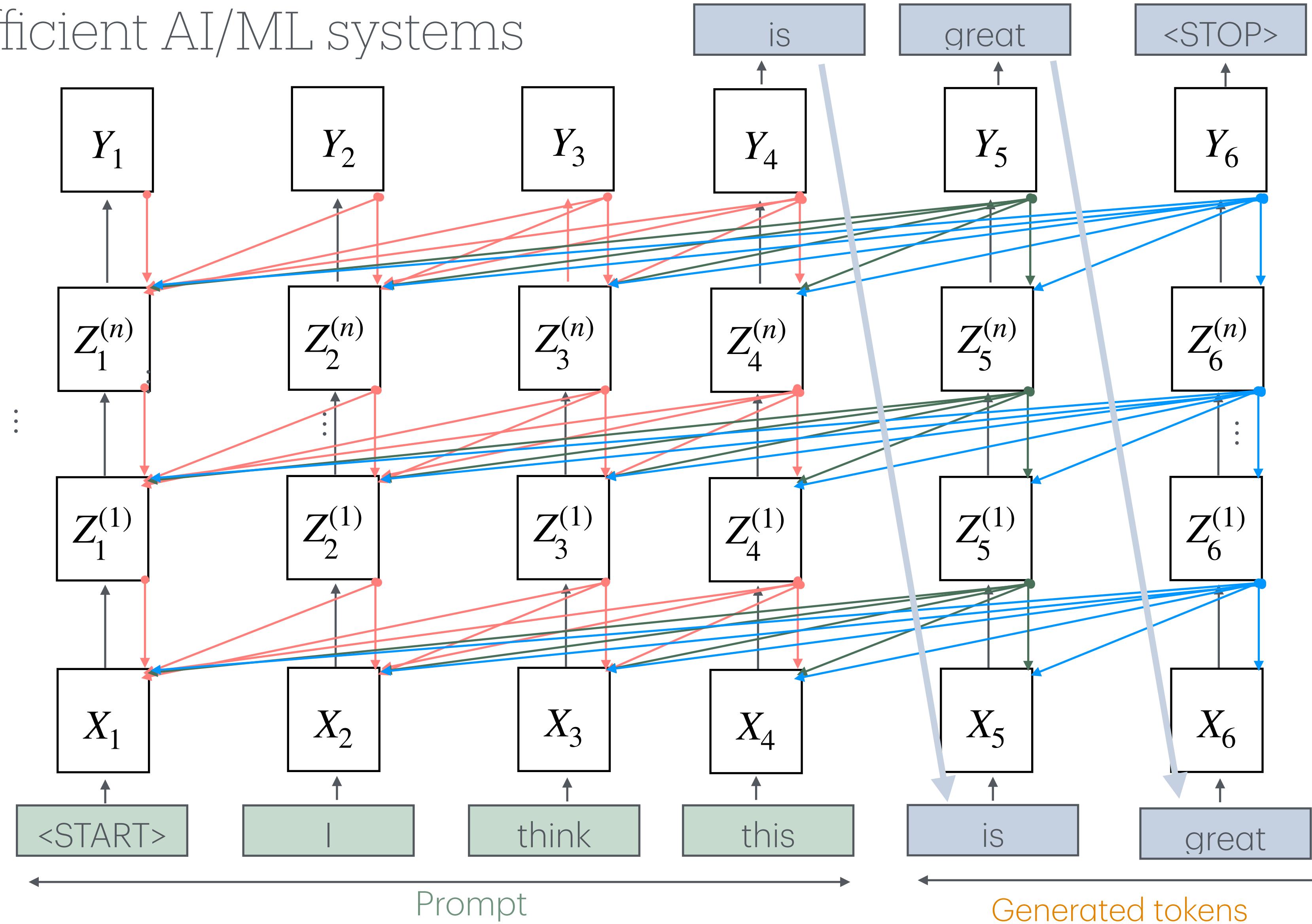


Review

From Part 3.2: Efficient AI/ML systems

Language model is built with transformer blocks

$$Z_t^{(i+1)} = \text{Transformer_block}(Z_t^{(i)})$$



2.2: LLM applications

Motivations

- LLMs are getting powerful to help your work
- LLMs are easy to access
- Many new and existing ways to use LLMs together with external tools

What you will learn from this session?

- How to use LLMs text generation for your use cases
- How to identify and prototype tasks that you can use LLMs
- Playbook for using LLMs

Plan for today

- Opportunities with LLMs
- Basic LLM usage
- LLMs with external tools
- Agentic LLMs
- Local LLMs
- Fine-tuning LLMs
- Playbook for LLM-based applications

LLMs and related applications: Current status

- LLMs are getting better and better, and new LLMs and products are introduced
- There are many already existing applications using LLMs, e.g. Github copilot, other copilot, ChatGPT, ...
 - Use available applications as they are to help your tasks at work or personal tasks
- There are various ways to use LLMs with APIs from remote providers and locally running LLMs
 - Allows us to use LLMs in opportunities that are not addressed by existing products

Opportunities

- There are many tasks around you that are not solved by existing AI products
- LLMs is still new, the industry is going through learning period, and new and better ways are introduced
- If you see the potential areas (even if small) that LLMs can help, that's an opportunity
- Use LLMs with API using remote and local LLMs
- Many examples using LLMs can be found
- Easy to prototype and check the feasibility
- Small tools using LLMs can help save time
- When new commercial tools available in the future, you can make educated/experienced decisions to review and evaluate new tools

Basic LLM usage

Text generation

How to interact with LLMs (such as gpt)?

- Prepare 'question or instruction' (also called as [prompt](#)) for your task and send to LLMs
- Prepare 'question/instruction together with a few examples' for your task and send to LLMs

How to send your 'prepared prompt' to LLMs?

- Use [playground](#) (chat style) UI, manually paste your prompt and test
- Use [API](#), use APIs to send your prompt to LLM

Quality of the generated output varies per models used, such as output from gpt-4o likely better than that of gpt-3.5-turbo

Text generation is a simple interface, sending text and receiving text with a small number of [configurations](#).

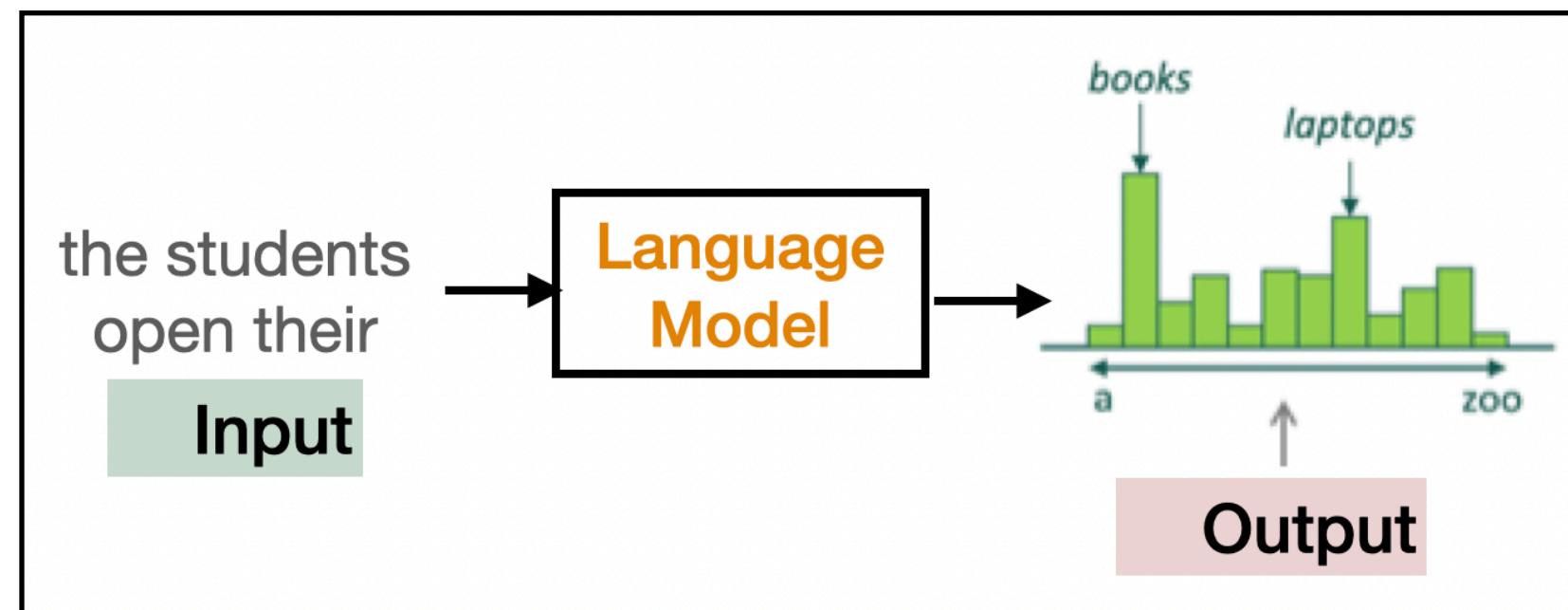
Message types and configuration for text generation

Message types:

- **System** prompt/message: instruction to LLMs for handling user messages
- **User** prompt/message: main request for a task

The screenshot shows the Hugging Face Playground interface. On the left, there is a sidebar with the title "Playground". Inside the sidebar, under the "SYSTEM" section, there is a text box containing the instruction: "You will be provided with statements, and your task is to convert them to standard English." To the right of the sidebar, the main workspace displays a conversation between "USER" and "ASSISTANT". The "USER" message is "She no went to the market." and the "ASSISTANT" message is "She did not go to the market.". Below the messages, there is a button labeled "+ Add message". On the far right of the interface, there are several configuration options: "Mode" set to "Chat", "Model" set to "gpt-3.5-turbo", "Temperature" (0), "Maximum length" (256), "Stop sequences" (empty input field), "Top P" (1), "Frequency penalty" (0), "Presence penalty" (0). At the bottom right, there is a note: "API and Playground requests will not be used to train our models. [Learn more](#)".

Message types and configuration for text generation



Configurations for text generation

- Temperature: controls randomness. 0 is more deterministic and repetitive
- Maximum length: prompt+output
- Stop sequence: tokens to stop generation
- Top P: top p probability mass, if 1 then 100% probability mass are considered for being sampled. Use either this or Temperature for tuning
- Live visualization demo*: [link](#)

The screenshot shows the Hugging Face Playground interface. At the top, there are buttons for "Grammar correction", "Save", "View code", "Share", and three dots. Below this is a section for "Mode" set to "Chat" and "Model" set to "gpt-3.5-turbo". Sliders for "Temperature" (0), "Maximum length" (256), and "Stop sequences" (empty field) are visible. At the bottom, a note states: "API and Playground requests will not be used to train our models. [Learn more](#)".

Playground

SYSTEM
You will be provided with statements, and your task is to convert them to standard English.

USER She no went to the market.

ASSISTANT She did not go to the market.

+ Add message

Playground: live demo*

SYSTEM

When I ask for help to write something, you will reply with a document that contains at least one joke or playful comment in every paragraph.

USER

Write a thank you note to my steel bolt vendor for getting the delivery in on time and in short notice. This made it possible for us to deliver an important order.

Suggestions for getting better results

1. Write clear instructions:

LLMs cannot read your mind, so be more specific. The less the model has to guess at what you want, the more likely you'll get it.

Tactic: Include details in your query to get more relevant answers

In order to get a highly relevant response, make sure that requests provide any important details or context. Otherwise you are leaving it up to the model to guess what you mean.

| Worse | Better |
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| How do I add numbers in Excel? | How do I add up a row of dollar amounts in Excel? I want to do this automatically for a whole sheet of rows with all the totals ending up on the right in a column called "Total". |
| Who's president? | Who was the president of Mexico in 2021, and how frequently are elections held? |
| Write code to calculate the Fibonacci sequence. | Write a TypeScript function to efficiently calculate the Fibonacci sequence. Comment the code liberally to explain what each piece does and why it's written that way. |
| Summarize the meeting notes. | Summarize the meeting notes in a single paragraph. Then write a markdown list of the speakers and each of their key points. Finally, list the next steps or action items suggested by the speakers, if any. |

Suggestions for getting better results

Playground: live demo*

1. Write clear instructions:

LLMs cannot read your mind, so be more specific. The less the model has to guess at what you want, the more likely you'll get it.

Tactic: Ask the model to adopt a persona

The system message can be used to specify the persona used by the model in its replies.

SYSTEM When I ask for help to write something, you will reply with a document that contains at least one joke or playful comment in every paragraph.

USER Write a thank you note to my steel bolt vendor for getting the delivery in on time and in short notice.
This made it possible for us to deliver an important order.

Playground: live demo*

Tactic: Ask the model to adopt a persona

SYSTEM

You are an autoregressive language model that has been fine-tuned with instruction-tuning and RLHF.

You carefully provide accurate, factual, thoughtful, nuanced answers, and are brilliant at reasoning.

If you think there might not be a correct answer, you say so.

Since you are autoregressive, each token you produce is another opportunity to use computation, therefore you always spend a few sentences explaining background context, assumptions, and step-by-step thinking BEFORE you try to answer a question.

Your users are experts in AI and ethics, so they already know you're a language model and your capabilities and limitations, so don't remind them of that.

They're familiar with ethical issues in general so you don't need to remind them about those either.

Don't be verbose in your answers, but do provide details and examples where it might help the explanation.

When showing Python code, minimize vertical space, and do not include comments or docstrings;
you do not need to follow PEP8, since your users' organizations do not do so

USER

Explain me what is softmax function

Suggestions for getting better results

1. Write clear instructions:

LLMs cannot read your mind, so be more specific. The less the model has to guess at what you want, the more likely you'll get it.

Tactic: Use delimiters to clearly indicate distinct parts of the input

Delimiters like triple quotation marks, XML tags, section titles, etc. can help demarcate sections of text to be treated differently.

USER

Summarize the text delimited by triple quotes with a haiku.

"""
Insert text here
"""

Suggestions for getting better results

1. Write clear instructions:

LLMs cannot read your mind, so be more specific. The less the model has to guess at what you want, the more likely you'll get it.

Tactic: Use delimiters to clearly indicate distinct parts of the input

Delimiters like triple quotation marks, XML tags, section titles, etc. can help demarcate sections of text to be treated differently.

USER Summarize the text delimited by triple quotes with a haiku.

"""insert text here"""

SYSTEM You will be provided with a pair of articles (delimited with XML tags) about the same topic. First summarize the arguments of each article. Then indicate which of them makes a better argument and explain why.

USER <article> insert first article here </article>

<article> insert second article here </article>

Suggestions for getting better results

1. Write clear instructions:

LLMs cannot read your mind, so be more specific. The less the model has to guess at what you want, the more likely you'll get it.

Tactic: Use delimiters to clearly indicate distinct parts of the input

Delimiters like triple quotation marks, XML tags, section titles, etc. can help demarcate sections of text to be treated differently.

USER Summarize the text delimited by triple quotes with a haiku.

"""insert text here"""

SYSTEM You will be provided with a pair of articles (delimited with XML tags) about the same topic. First summarize the arguments of each article. Then indicate which of them makes a better argument and explain why.

USER <article> insert first article here </article>

<article> insert second article here </article>

SYSTEM You will be provided with a thesis abstract and a suggested title for it. The thesis title should give the reader a good idea of the topic of the thesis but should also be eye-catching. If the title does not meet these criteria, suggest 5 alternatives.

USER Abstract: insert abstract here

Title: insert title here

Suggestions for getting better results

1. Write clear instructions:

LLMs cannot read your mind, so be more specific. The less the model has to guess at what you want, the more likely you'll get it.

Tactic: Specify the steps required to complete a task

Some tasks are best specified as a sequence of steps. Writing the steps out explicitly can make it easier for the model to follow them.

SYSTEM Use the following step-by-step instructions to respond to user inputs.

Step 1 - The user will provide you with text in triple quotes. Summarize this text in one sentence with a prefix that says "Summary: ".

Step 2 - Translate the summary from Step 1 into Spanish, with a prefix that says "Translation: ".

USER """insert text here"""

Suggestions for getting better results

Playground: live demo*

1. Write clear instructions:

LLMs cannot read your mind, so be more specific. The less the model has to guess at what you want, the more likely you'll get it.

Tactic: Provide examples

Providing general instructions that apply to all examples is generally more efficient than demonstrating all permutations of a task by example, but in some cases providing examples may be easier. For example, if you intend for the model to copy a particular style of responding to user queries which is difficult to describe explicitly. This is known as "few-shot" prompting.

SYSTEM

Answer in a consistent style.

USER

Teach me about patience.

ASSISTANT

The river that carves the deepest valley flows from a modest spring; the grandest symphony originates from a single note; the most intricate tapestry begins with a solitary thread.

USER

Teach me about the ocean.

Playground: live examples

Tactic: Provide examples

SYSTEM

Answer in a consistent style.

USER

Q: Teach me about patience.

A: The river that carves the deepest valley flows from a modest spring; the grandest symphony originates from a single note; the most intricate tapestry begins with a solitary thread.

Q: Teach me about the ocean.

Tactic: Provide examples

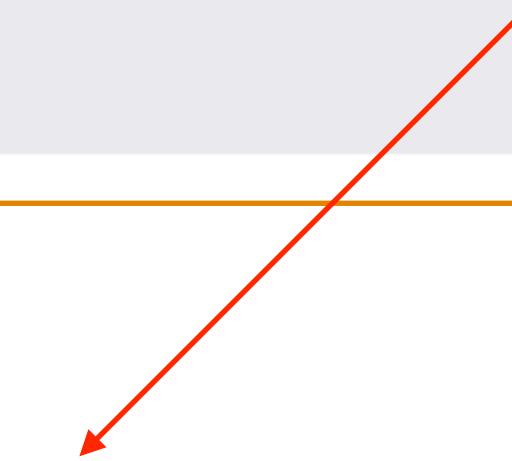
Providing general instructions that apply to all examples is generally more efficient than demonstrating all permutations of a task by example, but in some cases providing examples may be easier. For example, if you intend for the model to copy a particular style of responding to user queries which is difficult to describe explicitly. This is known as "few-shot" prompting.

SYSTEM Answer in a consistent style.

USER Teach me about patience.

ASSISTANT The river that carves the deepest valley flows from a modest spring; the grandest symphony originates from a single note; the most intricate tapestry begins with a solitary thread.

USER Teach me about the ocean.



Suggestions for getting better results

1. Write clear instructions:

LLMs cannot read your mind, so be more specific. The less the model has to guess at what you want, the more likely you'll get it.

Tactic: Specify the desired length of the output

You can ask the model to produce outputs that are of a given target length. The targeted output length can be specified in terms of the count of words, sentences, paragraphs, bullet points, etc. Note however that instructing the model to generate a specific number of words does not work with high precision. The model can more reliably generate outputs with a specific number of paragraphs or bullet points.

USER Summarize the text delimited by triple quotes in about 50 words.

"""insert text here"""

USER Summarize the text delimited by triple quotes in 2 paragraphs.

"""insert text here"""

USER Summarize the text delimited by triple quotes in 3 bullet points.

"""insert text here"""

You can also ask output to be **structured format**, such as JSON.

Suggestions for getting better results

2. Provide reference text:

LLMs can generate fake answers for those they don't know. Provide reference text (context), like cheat sheet for a student exam, LLMs will use reference instead of making things up

Tactic: Instruct the model to answer using a reference text

If we can provide a model with trusted information that is relevant to the current query, then we can instruct the model to use the provided information to compose its answer.

SYSTEM Use the provided articles delimited by triple quotes to answer questions. If the answer cannot be found in the articles, write "I could not find an answer."

USER <insert articles, each delimited by triple quotes>

Question: <insert question here>

We will look more details in this tactic

Suggestions for getting better results

3. Split complex tasks into simpler subtasks:

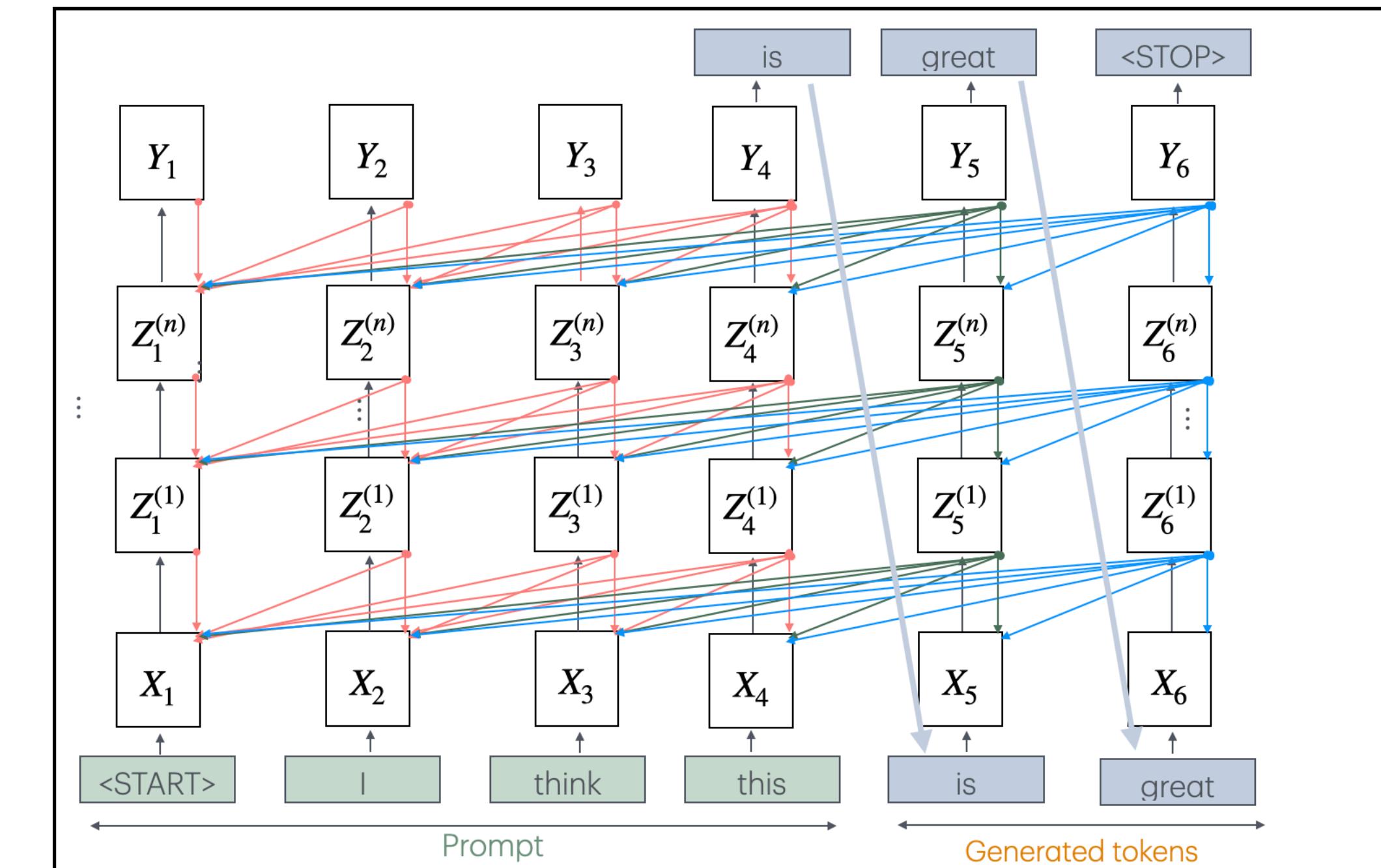
Complex tasks result in higher errors. Decomposing a complex task into a set of simpler tasks is a good idea

Tactics:

- Use intent classification to identify the most relevant instructions for a user query
- For dialogue applications that require very long conversations, summarize or filter previous dialogue
- Summarize long documents piecewise and construct a full summary recursively

Suggestions for getting better results

Playground: live demo*



4. Give the model time to “think”:

LLMs make more reasoning errors when asked answer right away instead of taking time to work it out. Ask for “chain of thought (CoT)” method before answering can help the model to reason toward correct answers

Playground: live demo*

Strategy: Give models time to "think"

SYSTEM

Determine if the student's solution is correct or not.

USER

Problem Statement: I'm building a solar power installation and I need help working out the financials.

- Land costs \$100 / square foot
- I can buy solar panels for \$250 / square foot
- I negotiated a contract for maintenance that will cost me a flat \$100k per year, and an additional \$10 / square foot

What is the total cost for the first year of operations as a function of the number of square feet.

Student's Solution: Let x be the size of the installation in square feet.

1. Land cost: $100x$
2. Solar panel cost: $250x$
3. Maintenance cost: $100,000 + 100x$

Total cost: $100x + 250x + 100,000 + 100x = 450x + 100,000$

This answer is incorrect

Playground: live demo*

Strategy: Give models time to "think"

SYSTEM

First work out your own solution to the problem. Then compare your solution to the student's solution and evaluate if the student's solution is correct or not. Don't decide if the student's solution is correct until you have done the problem yourself.

USER

Problem Statement: I'm building a solar power installation and I need help working out the financials.

- Land costs \$100 / square foot
- I can buy solar panels for \$250 / square foot
- I negotiated a contract for maintenance that will cost me a flat \$100k per year, and an additional \$10 / square foot

What is the total cost for the first year of operations as a function of the number of square feet.

Student's Solution: Let x be the size of the installation in square feet.

1. Land cost: $100x$
2. Solar panel cost: $250x$
3. Maintenance cost: $100,000 + 100x$

Total cost: $100x + 250x + 100,000 + 100x = 450x + 100,000$

This answer is incorrect

Suggestions for getting better results

SYSTEM

You can write and execute Python code by enclosing it in triple backticks, e.g. ```code goes here```. Use this to perform calculations.

USER

Find all real-valued roots of the following polynomial: $3x^{10} - 5x^9 - 3x^8 - 7x^7 - 10$.

5. **Use external tools:** 'LLMs for all' may have limitations. Using outputs from the external tools can help LLMs to generate better answers. Such as retrieved text for generations, or external tool to validate generated output

Suggestions for getting better results

SYSTEM

You can write and execute Python code by enclosing it in triple backticks, e.g. `` `code goes here` `` . Use this to perform calculations.

USER

Find all real-valued roots of the following polynomial: $3x^{**5} - 5x^{**4} - 3x^{**3} - 7x - 10$.

SYSTEM

You can **write and execute Python code** by enclosing it in triple backticks. Also note that you have access to the following module to help users send messages to their friends:

```
```python
import message
message.write(to="John", message="Hey, want to meetup after work?")````
```

**WARNING: Executing code produced by a model is not inherently safe and precautions should be taken in any application that seeks to do this. In particular, a sandboxed code execution environment is needed to limit the harm that untrusted code could cause.**

5. **Use external tools:** 'LLMs for all' may have limitations. Using outputs from the external tools can help LLMs to generate better answers. Such as retrieved text for generations, or external tool to validate generated output

# Suggestions for getting better results

Have evaluation plan early

Prepare evaluation data

LLM as a judge together with human evaluation  
will be a good starting point

## 6. Test changes systematically:

Improving performance is easier if we can measure it. Prepare ways to evaluate systematically

# Suggestions for getting better results

1. Write clear instructions:  
LLMs cannot read your mind, so be more specific. The less the model has to guess at what you want, the more likely you'll get it.
2. Provide reference text:  
LLMs can generate fake answers for those they don't know. Provide reference text, like cheat sheet for a student exam, LLMs will use reference instead of making things up
3. Split complex tasks into simpler subtasks:  
Complex tasks result in higher errors. Decomposing a complex task into a set of simpler tasks is a good idea
4. Give the model time to "think":  
LLMs make more reasoning errors when asked answer right away instead of taking time to work it out. Ask for "chain of thought" method before answering can help the model to reason toward correct answers
5. Use external tools: 'LLMs for all' may have limitations. Using outputs from the external tools can help LLMs to generate better answers.  
Such as retrieved text for generations, or external tool to validate generated output
6. Test changes systematically:  
Improving performance is easier if we can measure it. Prepare ways to evaluate systematically

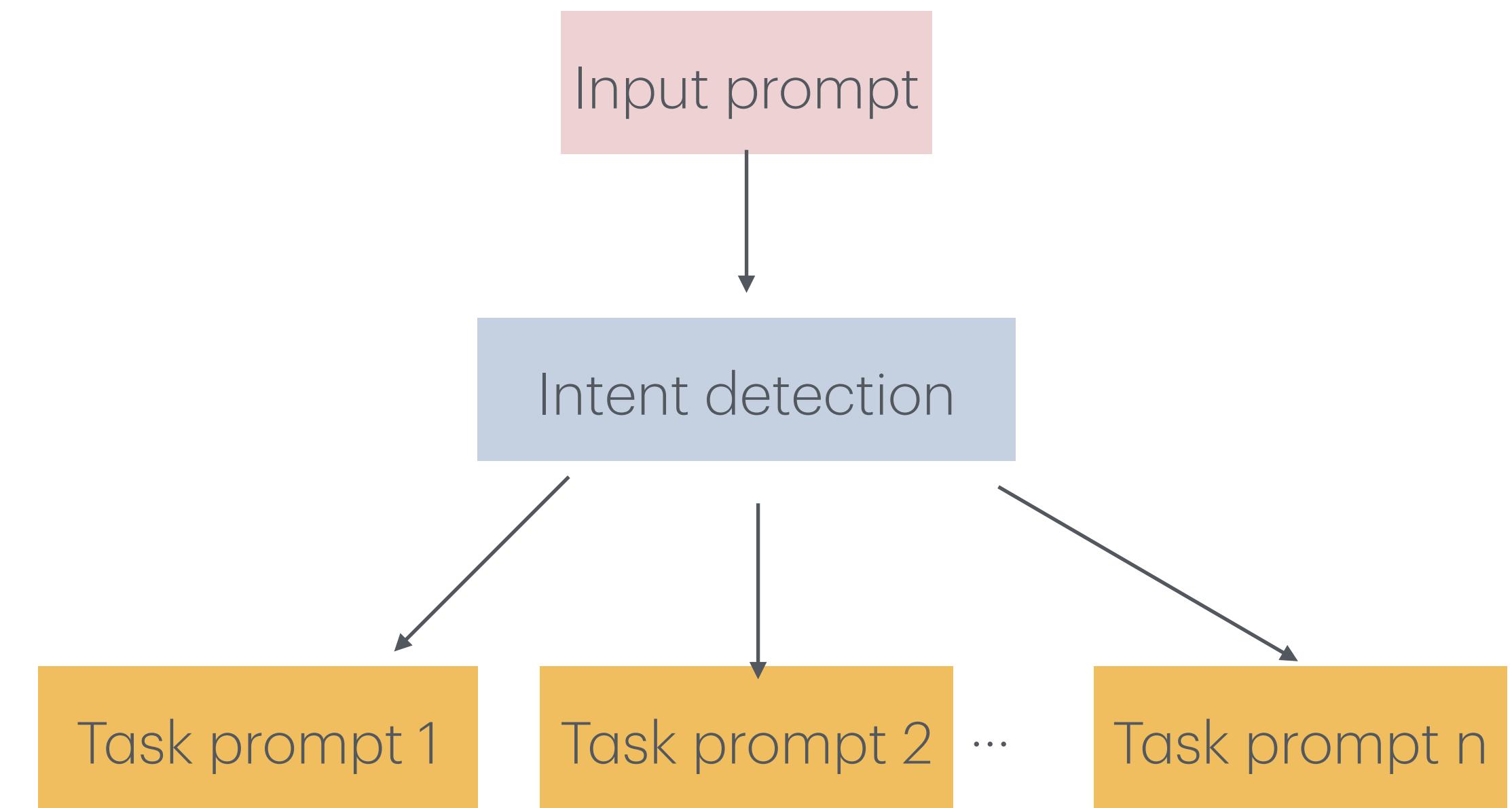
# Intent detection

When we have many tasks to handle that requires different prompts, it would help to classify an intent from the original query.

Intent detection will determine which prompts are needed, so we will use the selected prompts to follow up the task

Consider this as a decomposing original task and process iteratively

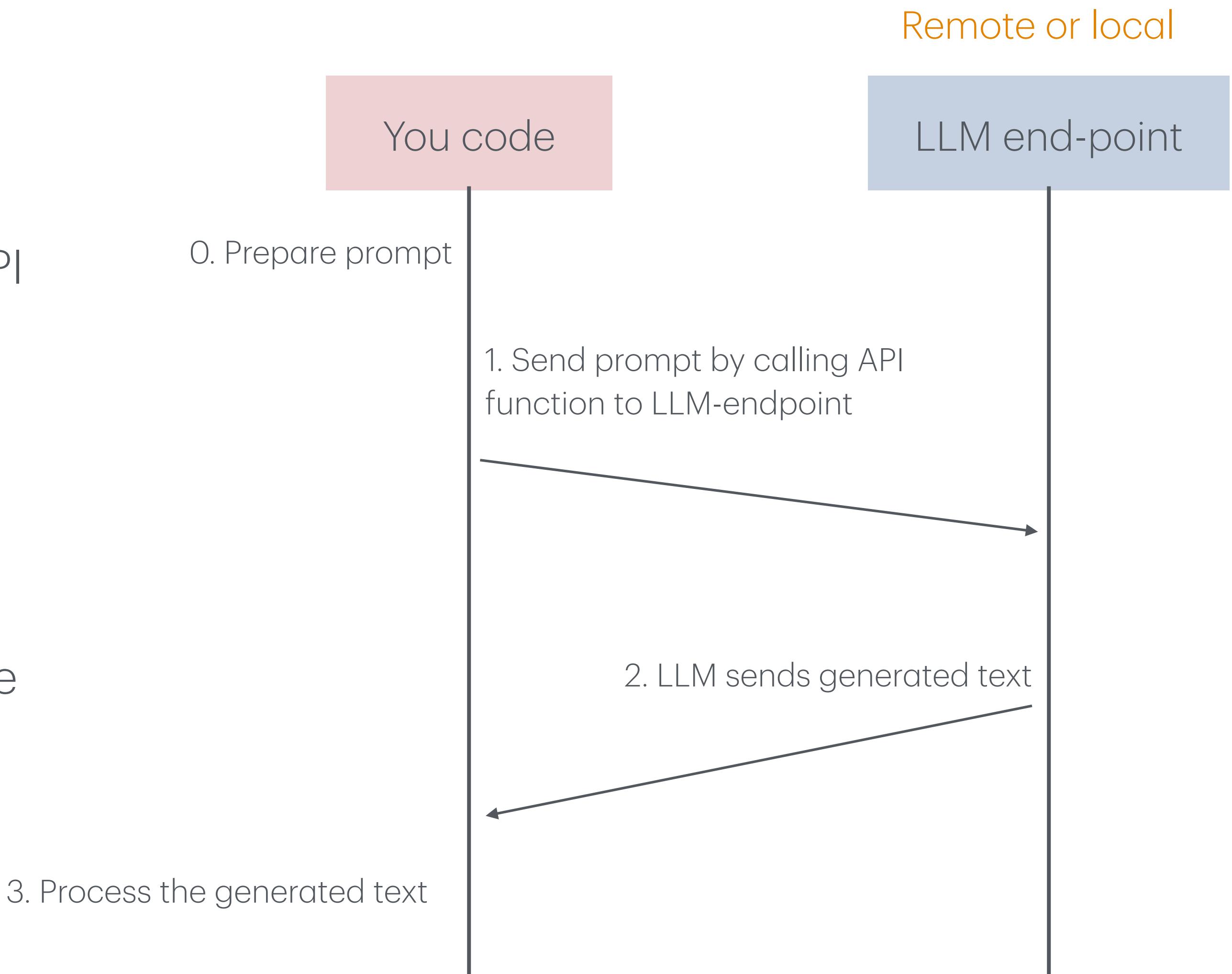
This method also help to determine different LLMs to use depending on the detected intent



Prompt to identify intent:  
Based on the intent use different prompt to follow up  
It can be iterative calls to LLMs  
  
It can be different LLMs based on the detected intent

# LLMs access using API

- Use API to do ‘text generation’
- Use service providers’ end point and call API
- Use code (python) to prepare your prompt and send
- Post process received text
- If necessary and based on the result, iterate or call other tools, or more LLM calls



# LLM access using API: live demo\*

# An example of code documentation

LLM access using API: live demo\*

Task: create code comment in specific style for C function

Strategy:

- describe what needs to be done specifically and clearly
- Provide example(s), ‘few-shot examples’

Kernel-doc code comment style will be used as an example

```
/**
 * INIT_LIST_HEAD - Initialize a list_head structure
 * @list: list_head structure to be initialized.
 *
 * Initializes the list_head to point to itself. If it is a list header,
 * the result is an empty list.
 */
static inline void INIT_LIST_HEAD(struct list_head *list)
{
 WRITE_ONCE(list->next, list);
 WRITE_ONCE(list->prev, list);
}
```

# An example of code documentation

LLM access using API: live demo\*

Task: create code comment in specific style for C function

Prompt with example

Strategy:

- describe what needs to be done specifically and clearly
- Provide example(s), ‘few-shot examples’

You are a skilled developer.

You have been asked to review a function delimitated by triple backticks.

Please review the function and provide maximum two lines of clear, helpful, and short documentation comment for the function follow the format of the example:

{example}

```{code}...``

An example of code documentation

LLM access using API: live demo*

Putting all together a prompt to generate comment for the C code in specific style as follows:

- Task description (system message)
- Few-shot example (example C code)
- Task (C-code)

You are a skilled developer.

You have been asked to review a function delimited by triple backticks.

Please review the function and provide maximum two lines of clear, helpful, and short documentation comment for the function follow the format of the example:

```
/**  
 * INIT_LIST_HEAD - Initialize a list_head structure  
 * @list: list_head structure to be initialized.  
  
 * Initializes the list_head to point to itself. If it is a list header,  
 * the result is an empty list.  
 */  
static inline void INIT_LIST_HEAD(struct list_head *list)  
{  
    WRITE_ONCE(list->next, list);  
    WRITE_ONCE(list->prev, list);  
}
```

```
...  
static inline void list_replace(struct list_head *old,  
                           struct list_head *new)  
{  
    new->next = old->next;  
    new->next->prev = new;  
    new->prev = old->prev;  
    new->prev->next = new;  
}  
...
```

LLMs with external tools

Weaknesses of LLMs

- Making things up (hallucination) rather confidently
 - LLMs compressed vast amount of knowledge, but it could make things up
- Training data cut off
- Missing citations
 - Do not provide the source of the generated output
- Proprietary data
 - LLMs will not know the data that is not trained on

LLMs with external tools

Compensate the weakness of LLMs by feeding outputs of other tools.

- Use embedding based search for information retrieval
- Use code executor to run the generated code and use the output
- Use API calls to external services

Embedding-based knowledge retrieval

Use cases:

- When want to find information from long text (e.g. book(s))
- We want LLMs to answer only based on proprietary document(s)

Embedding-based knowledge retrieval

Use cases:

- When want to find information from long text (e.g. book(s))
- We want LLMs to answer only based on proprietary document(s)

Approach 1. Feed the long text to text as part of the prompt

- Pros: no external is needed
- Cons: context length is limited, long context process is slow, lost-in-the-middle can be a problem

Embedding-based knowledge retrieval

Use cases:

- When want to find information from long text (e.g. book(s))
- We want LLMs to answer only based on proprietary document(s)

Approach 2. Use external tool to retrieve related information from the long text. Then only feed retrieved information and create prompt

- Pros: we don't need long context
- Cons: we need preprocess long context and external tool

SYSTEM

Use the provided articles delimited by triple quotes to answer questions. If the answer cannot be found in the articles, write "I could not find an answer."

USER

<insert articles, each delimited by triple quotes>

Question: <insert question here>

Review

Word embedding

Word2vec: a model that trained to converts words to vector (**word embeddings**)

It *learns* the word embedding by training the model to predict probability of given word and its contexts.

Similar words are clustered each other. In the example high dimensional vectors are projected to 2 dim for visualization.

Relationship of words also captured by word embedding. In the example, an answer for "**man:woman=king:?**" can be with cosine distance in the embedding space.

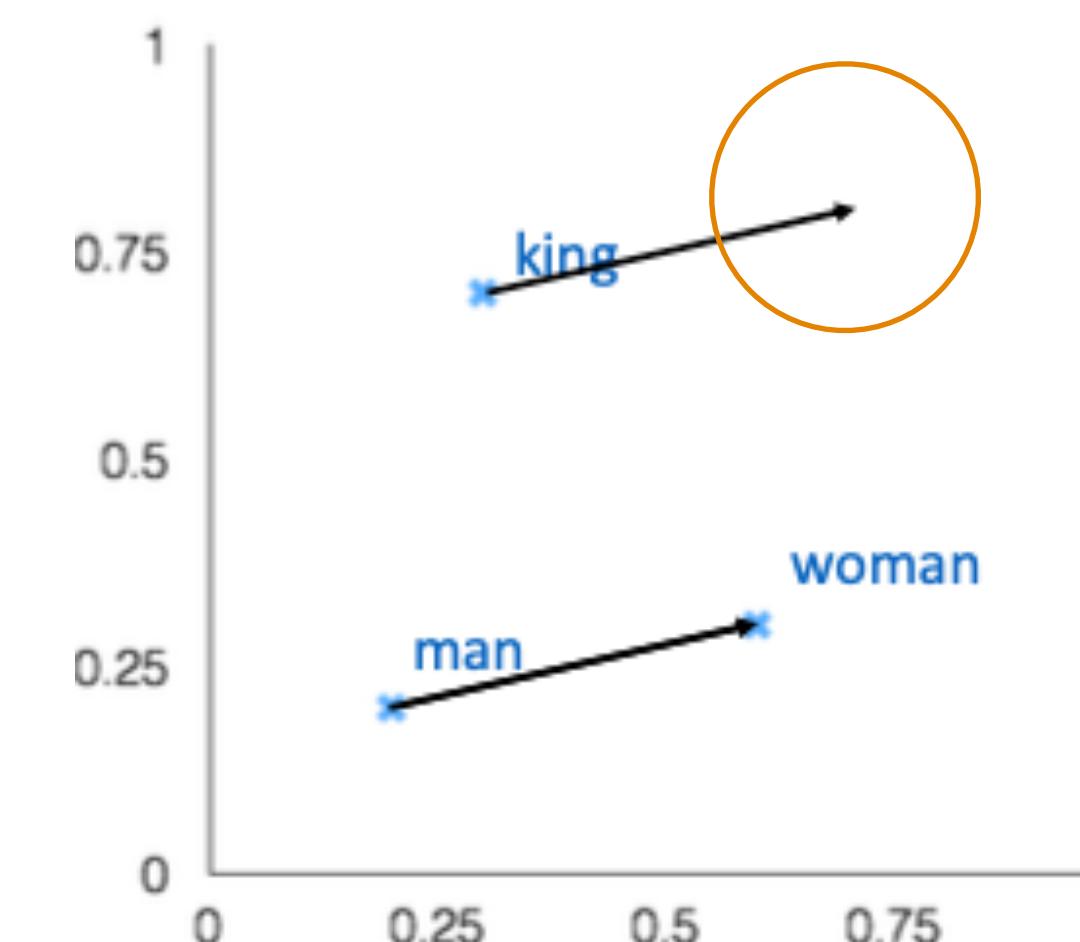
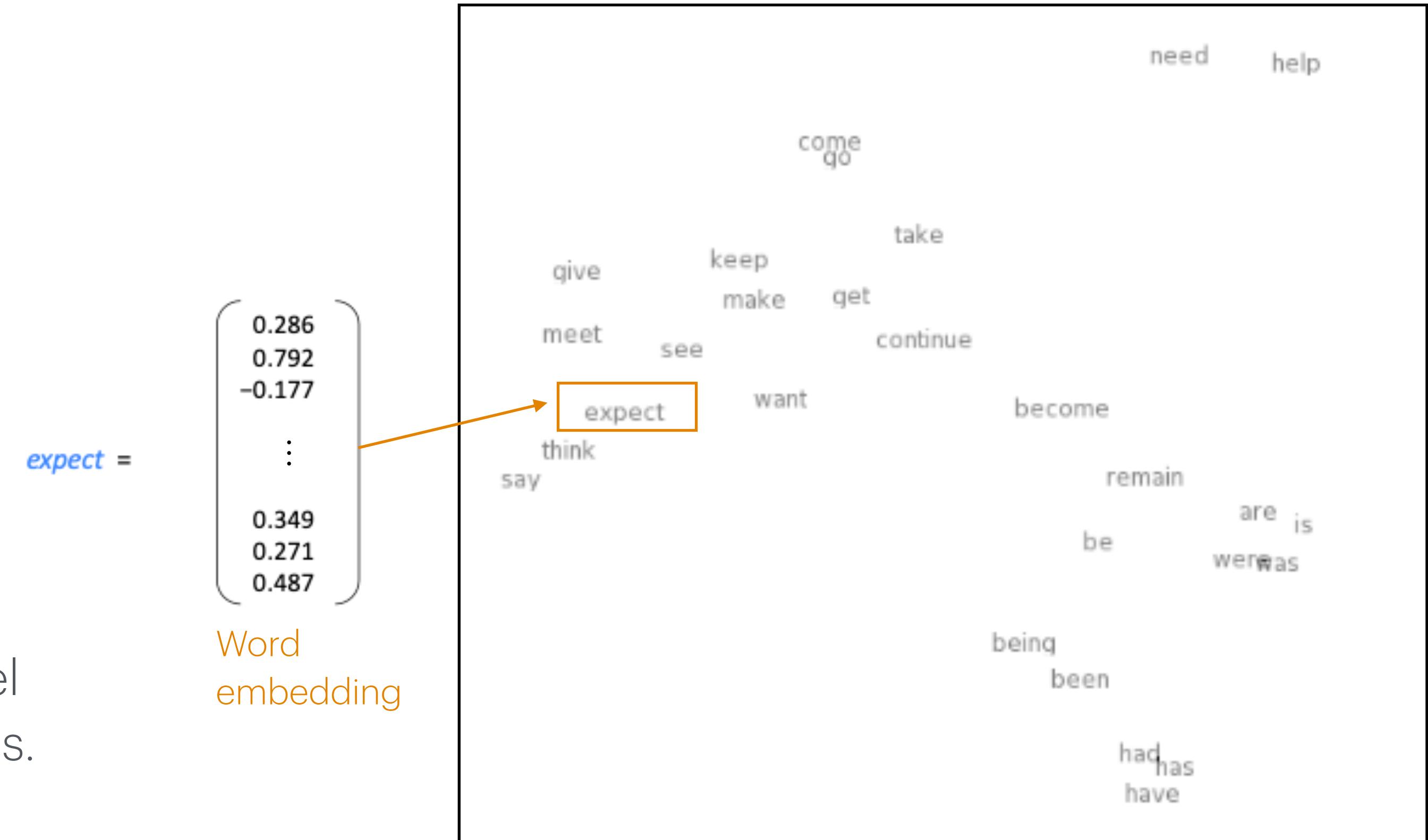


Image source: [3]

Text (sentence) Embedding

Text embedding: high-dim embedding (vector) that contains the semantic meaning of the text

Sentence embeddings

| Sentence | Numbers | | | |
|------------------------------|---------|-------|-----|-------|
| Hello, how are you? | 0.419 | 1.28 | ... | -0.06 |
| I'm going to school today | -0.74 | 1.02 | ... | 1.35 |
| ... | | | ... | |
| Once upon a time | -0.82 | -1.32 | ... | 0.23 |

 Embedding dimension

Image source: [link](#)

Text (sentence) Embedding

Text embedding: high-dim embedding (vector) that contains the semantic meaning of the text

Project high-dimension embeddings to 2d space for visualization

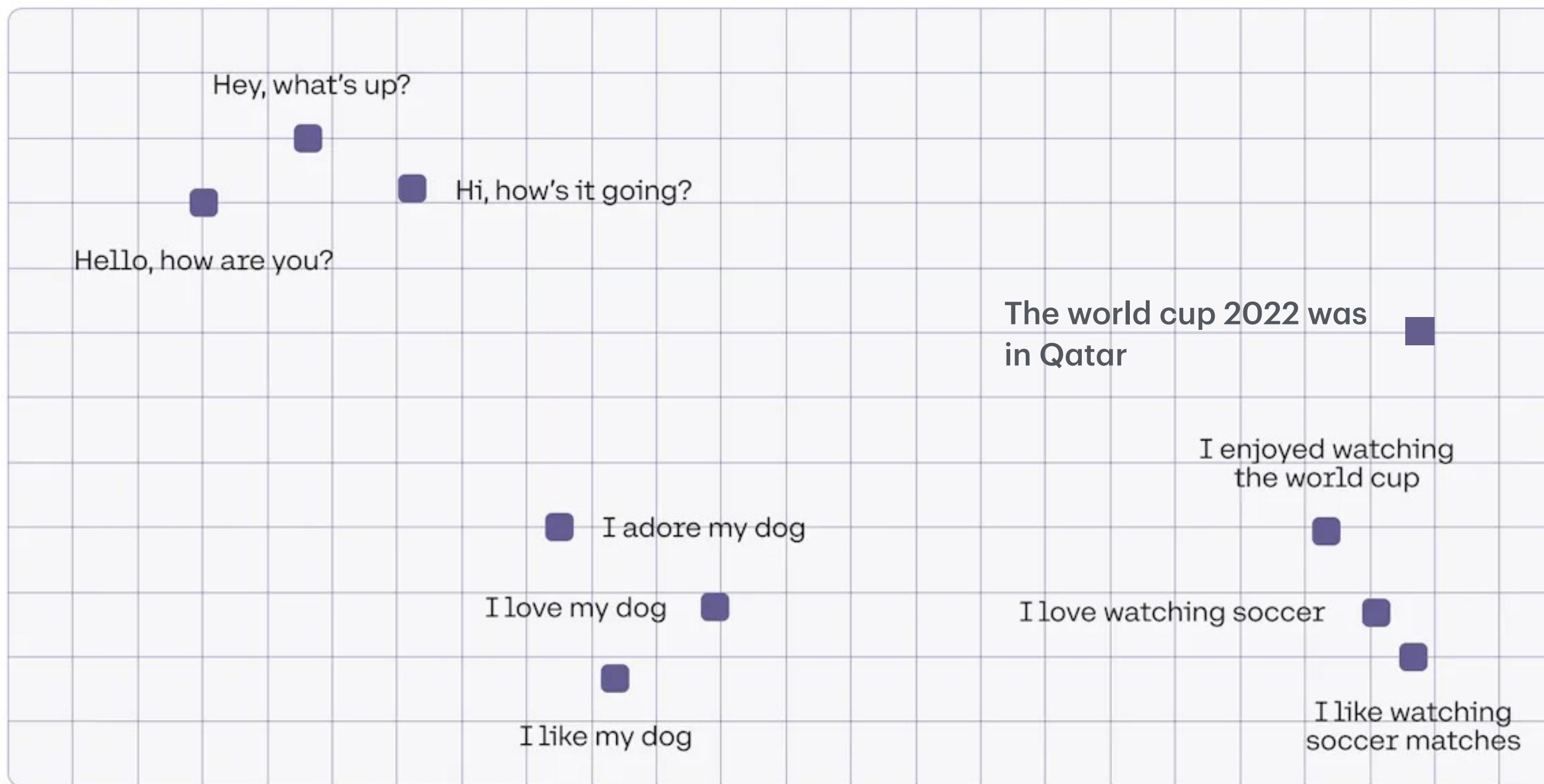
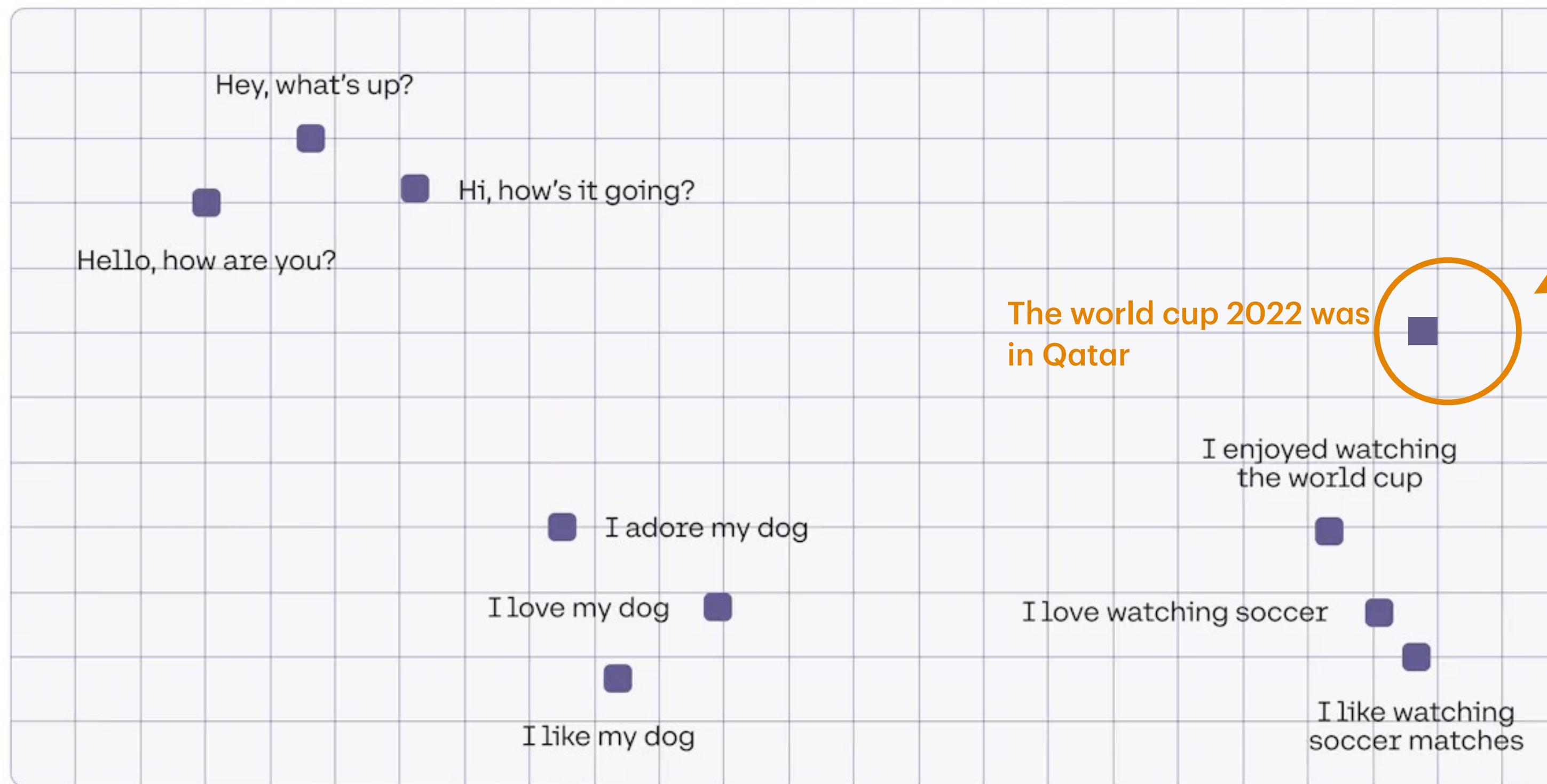


Image source: [link](#)

Text (sentence) Embedding

Text embedding: high-dim embedding (vector) that contains the semantic meaning of the text

Project high-dimension embeddings to 2d space for visualization



If I want to find an answer for a question of "where was world cup 2022?"

The text embedding of this question will be near at here.

So I can extract the text of this embedding

Motivational example

- When ask a question about specific domain?
- For example, "mamba" is a new modeling proposed for handling sequential information (such as text).
- When asking this to chatgpt, then it will only answer based on data what it is trained on

The screenshot shows a conversation between a user and ChatGPT. The user, represented by a blue circle with 'IN', asks: "what is mamba in LLM?". The AI, represented by a green circle with a swirl icon, responds: "In the context of large language models (LLM), 'mamba' does not have a specific or widely recognized meaning. If you are referring to a specific term or concept related to LLMs, please provide more context or specify the domain of interest. Otherwise, it's possible that 'mamba' might be a term not commonly associated with LLMs, and additional information would be needed to provide a more accurate answer." The AI's response is partially highlighted with a red rectangle around the phrase "does not have a".

You
what is mamba in LLM?

ChatGPT
In the context of large language models (LLM), "mamba" does not have a specific or widely recognized meaning. If you are referring to a specific term or concept related to LLMs, please provide more context or specify the domain of interest. Otherwise, it's possible that "mamba" might be a term not commonly associated with LLMs, and additional information would be needed to provide a more accurate answer.

Motivational example

- When we provide related information as 'context' LLM is able to answer question.
- LLM can answer as 'in-context learning'



You

Answer the following question using the below context in less than three sentences.

Question

What is mamba in LLM?

Context

Foundation models, now powering most of the exciting applications in deep learning, are almost universally based on the Transformer architecture and its core attention module. Many subquadratic-time architectures such as linear attention, gated convolution and recurrent models, and structured state space models (SSMs) have been developed to address Transformers' computational inefficiency on long sequences, but they have not performed as well as attention on important modalities such as language. We identify that a key weakness of such models is their inability to perform content-based reasoning, and make several



ChatGPT

Mamba in LLM refers to a sequence model backbone introduced as an improvement to Transformer architectures. It integrates selective structured state space models (SSMs) into a simplified neural network without attention or MLP blocks, resulting in fast inference and linear scaling in sequence length. Mamba achieves state-of-the-art performance across various modalities, including language modeling, outperforming Transformers of similar or larger size.

Embedding-based knowledge retrieval

Steps that we use retrieval augmented generation:

- Convert text into embedding (may require to chunk long text into smaller size text)
- Keep embeddings and texts (may need to use vector database to keep them)
- When a query is requested
 - convert the query into an embedding and search the most close embeddings from the vector database)
- Create a prompt using a query and retrieved text from the search and send to LLM

Agentic LLMs

- Current LLM usage: zero-shot, few-shot
- Agent workflow: ask LLM to *iterate* over a document or task, *use external tools or code*
 - Research on topics using web data and summarize the search and present output
 - Prepare plan for software fix and iteratively propose code to fix the code
- Ask LLMs with different roles (such as generator and critics) iteratively
- Generate plans for using external tools
- Agentic LLM usage can achieve more complex tasks than non-iterative and LLM only pattern
- Design and SW on top of LLMs can help many applications

Agentic LLM

Example

- Examples:
 - [Devin \(startup partnered with MSFT\)](#)
 - And similar tools



Agentic LLM

- Examples:
 - Github copilot workspace (recent GitHub preview feature)
 - “I think AI agent workflows will drive massive AI progress this year — perhaps even more than the next generation of foundation models.” by Andrew Ng (ref)



Agentic LLM design patterns

- Here are some of the agents design patterns
 - **Reflection**: examines its own work and improve
 - **Tool usage**: use web search, code executions
 - Planning: multistep planning to achieve a goal
 - Multi-agent collaboration: splitting work and discuss

Reflexion

- A pattern that's quick to implement and lead to good performance
- Example of **fixing** the existing code base
 - Ask LLM to provide feedback
 - “Here is the <code>. Check the code and provide constructive feedback”*
 - Then ask LLM using the response:
 - “Here is the code and feedback. Use the feedback and rewrite the code”
 - Repeating the criticism and regenerate process could improve the generation.
 - This self-reflection helps LLM to find issues and improves its generation

*: prompt will be more elaborated, it is shorten only for the simplicity for presentation.

Ref: <https://www.deeplearning.ai/the-batch/how-agents-can-improve-lm-performance/>

Tool usage

- Tool usage is a key design pattern:
 - Executing external tools, such as **search** information, taking actions, modifying data
- Example: “What is the best cafe based on user review?”
 - LLM will generate '{tool: web-search, query: "cafe reviews"}'
 - External tool searches, and provide result to LLM
- Example: “If I invest \$100 at compound 7% interest for 12 years, what do I have at the end?”
 - LLM generates python code with this: '{tool: python-interpreter, code: "100 * (1+0.07)**12"}'
 - Generate code run and produce the output

Local LLMs

Local LLMs

Small LM can be run on local machine (i.e. your laptop) using the following tools.

- MLC: less easier to install and use, but maybe more optimized in using memory and compute. Recent llama3 favors MLC.
- Ollama: easier to use, but maybe less optimize in using memory and compute

Models can be used standalone or launched as a REST API endpoint from the local machine.

Local LLMs usage : live demo*

Fine-tuning LLMs

Fine-tuning LLMs

Further train closed or open source LLMs since you need a custom model

- Continuous pre-training: train the model the prepared text using next token prediction
- Supervised fine-tuning (SFT): train the model with Q&A pair

Both closed and open LLMs can be used for this approach

Why consider custom LLMs

Data residency and
privacy

Intellectual property

Customization

Cost

Performance

Latency

Control over updates

Ownership

Transparency

Playbook for LLM-based apps

Playbook for LLM-based applications

Identify use case(s), define clean inputs and outputs of the case

Create and test prompt using playground. Use descent models, such as gpt-35-turbo, gpt-4o

Think about how to measure the generated output. Create some test cases

Once you determine LLM can help your use case using simple testing, then move to the next step and use API calls to automate development.

Write python scripts that can automate the experiment, test and evaluation

Use a descent model API endpoint, or use local LLMs API endpoint depends on the task complexity

Evaluation the results and iterate

Depends on the task complexity and task nature, consider using external tools, which includes embedding based retrieval or calling external APIs to get more information.

Intent detection and iterative generation can improve output performance, and methods used in agentic LLMs can be helpful for certain types of task.

For work related text (code, documentation) be sure to use LLMs that are approved by the company. The reason is that we do not want to send work related text to externally hosted LLMs, which might be used for training and be read.

References

1. From deeplearning.ai, ChatGPT Prompt Engineering for Developers
2. Prompt engineering, OpenAI
3. Natural Language Processing with Deep Learning

Free Books and additional materials

1. Short course from deeplearning.ai , [AI Python for Beginners](#)
2. AWS w/ deeplearning.ai, [Generative AI with Large Language Models](#)
3. Databricks, EdX, [Large Language Models: Foundation Models from the Ground Up](#)
4. Databricks, EdX, [Large Language Models: Application through Production](#)
5. Databricks, [Data+AI Summit 2023](#)
6. AWS, [Generative AI Foundations on AWS Technical Deep Dive Series](#)

Additional slides

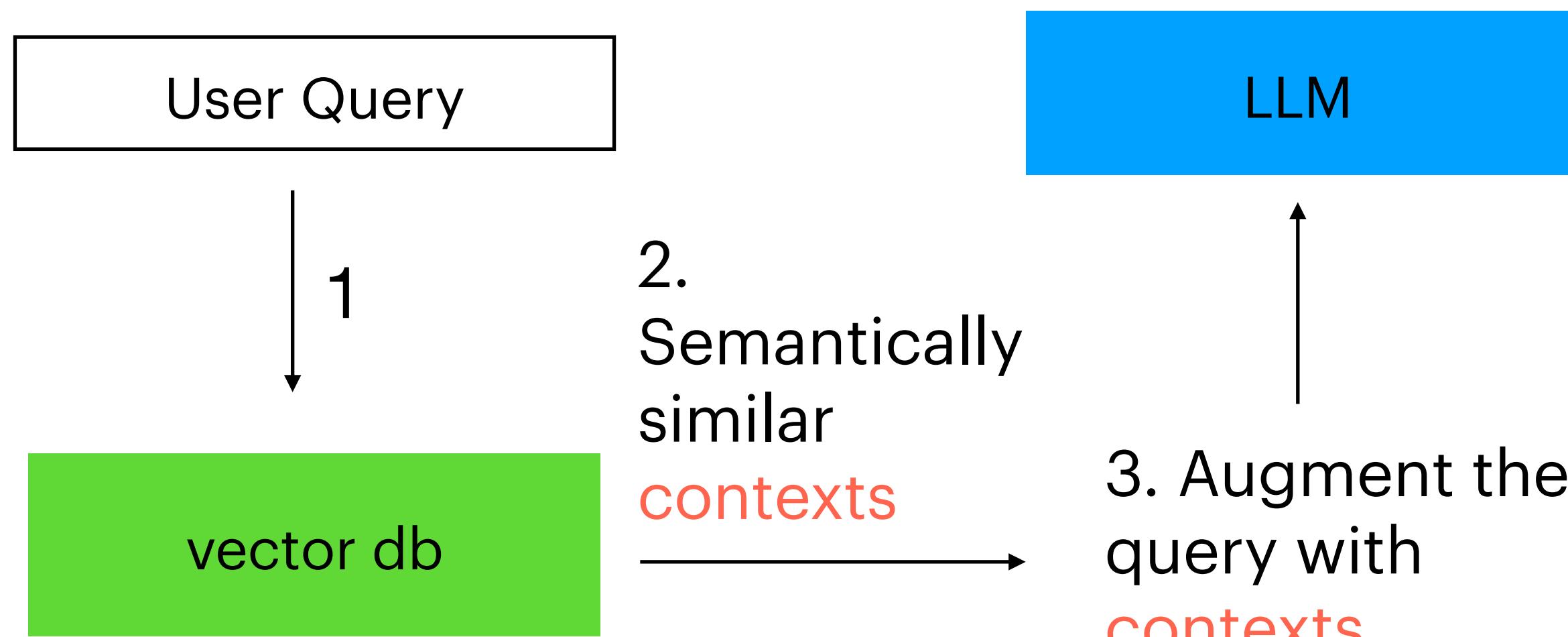
Write clear instructions

- Include details in your query to get more relevant answers
- Ask the model to adopt a persona
- Use delimiters to clearly indicate distinct parts of the input
- Specify the steps required to complete a task
- Provide examples
- Specify the desired length of the output

Retrieval augmented generation (RAG)

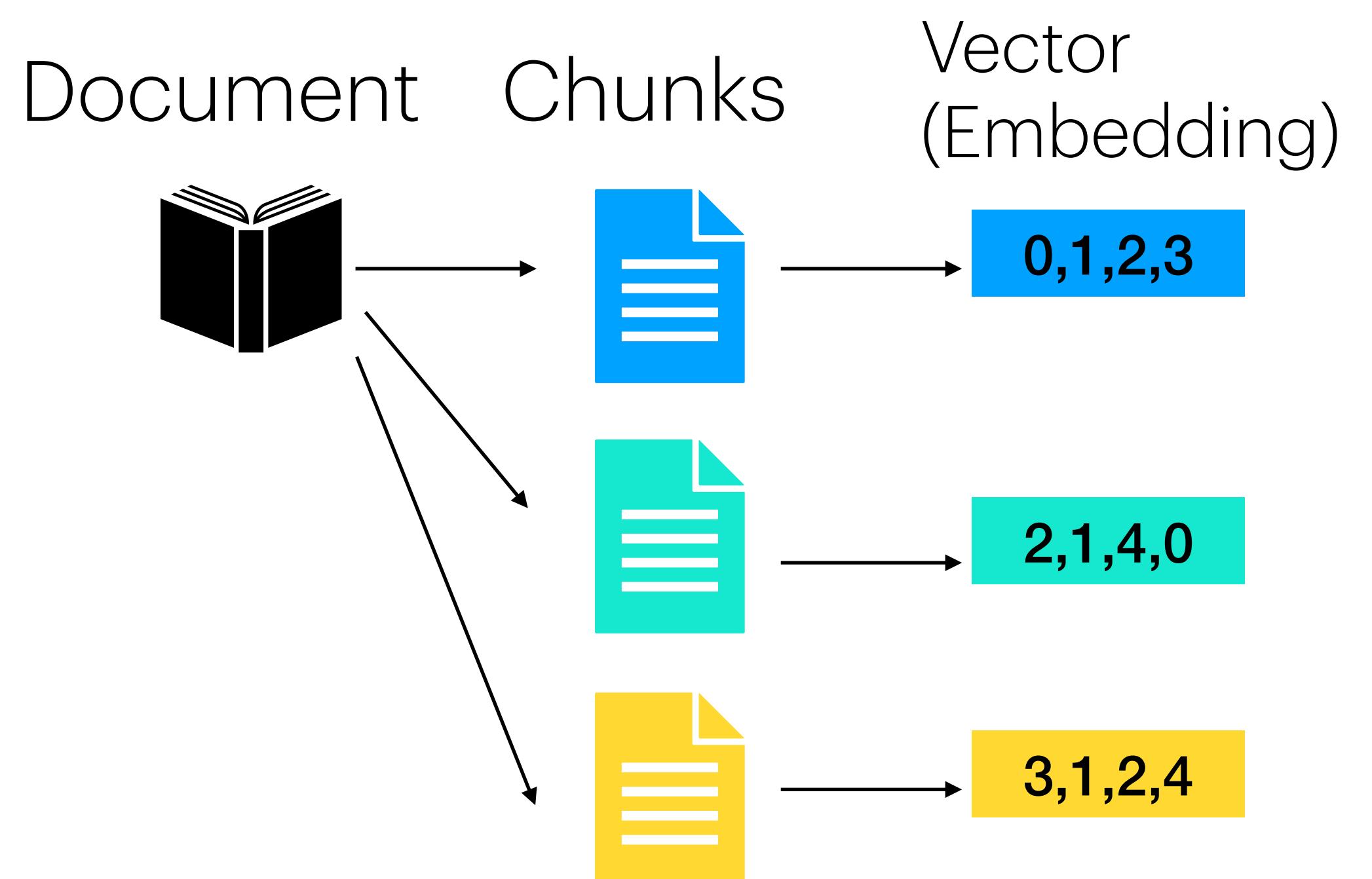
Overview

- Retrieval augmented system augments a query with semantically related text from vector db
- Two components
 - Vector database
 - LLM



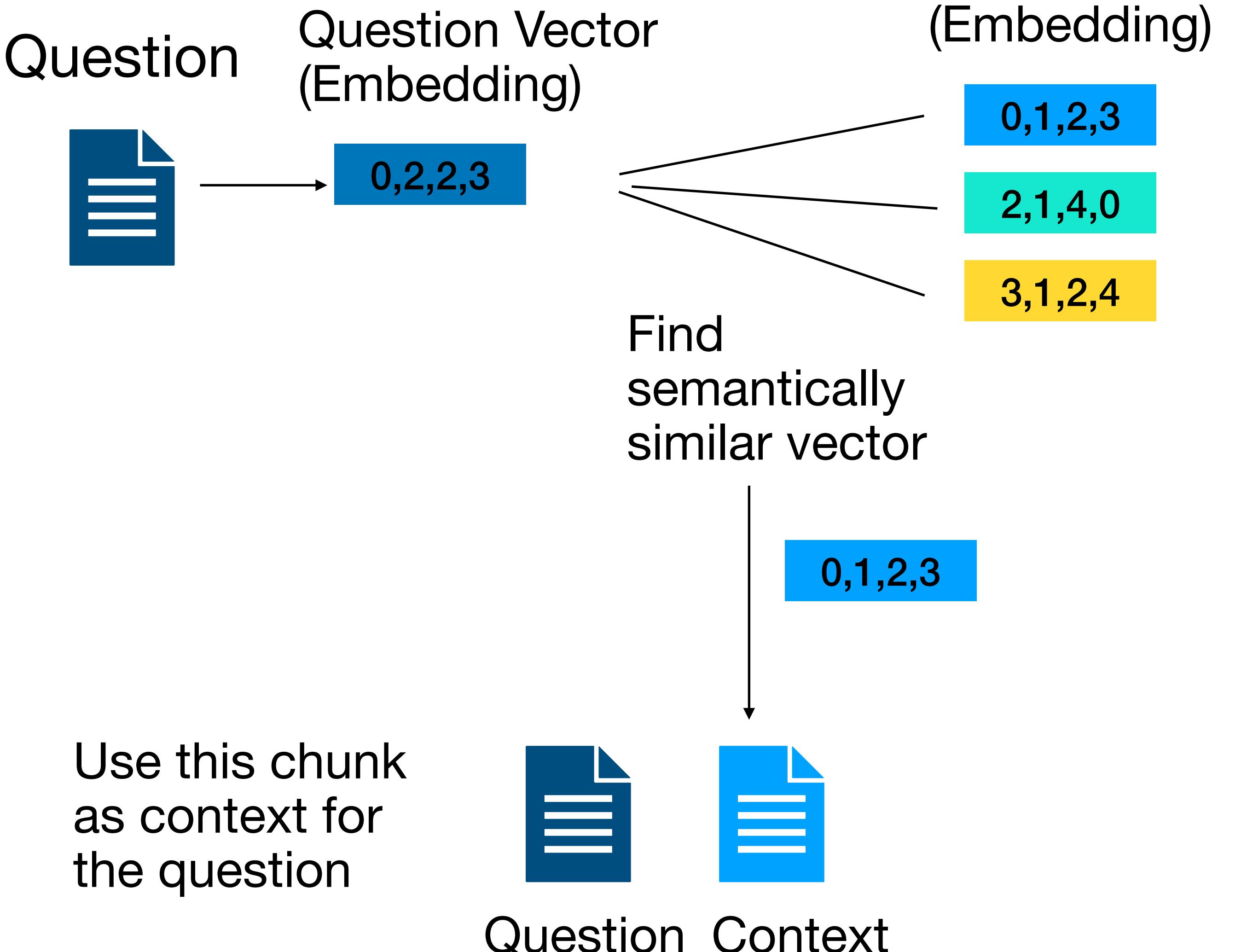
Indexing (turn documents meaningful vector)

- Chunk document to small chunk
- Convert text chunks to vectors
 - A dedicated model does this
- Store text chunks and vectors
(vector dbs are used to store and provide search)
- We do this off-line



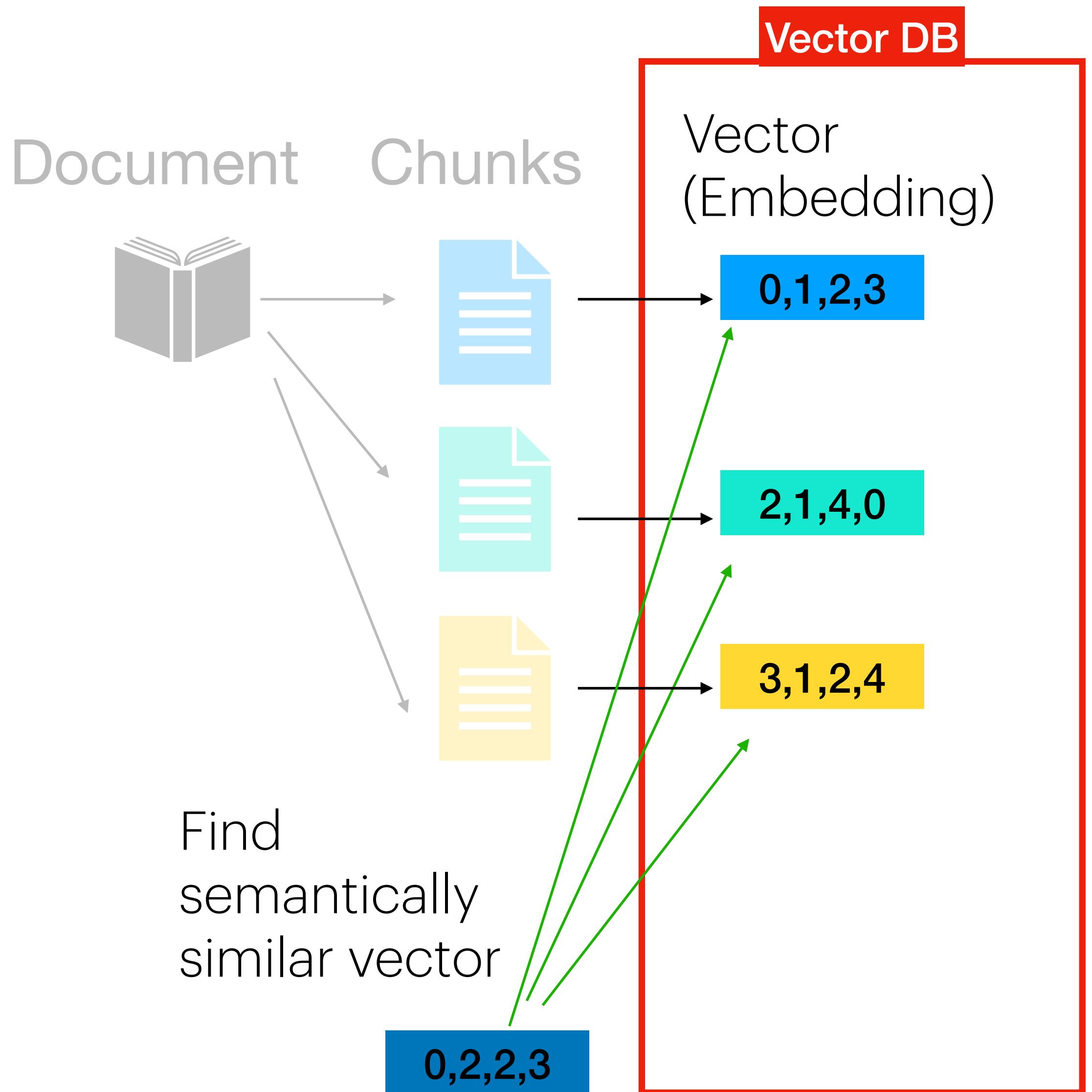
Similarity search

- Convert the question to vector
- Compare the question vector and the chunk vectors
- Find similar chunk using cosine similarity (measure angles of the vectors)
 - Select **top-k** similar chunks
 - In the example (right), we use top-1 chunk ($k=1$)
- Use the similar chunk as context with the question



Vector DB

- Manages large dimension vectors
- Find similar vectors based measures (cosine similarity or dot product)



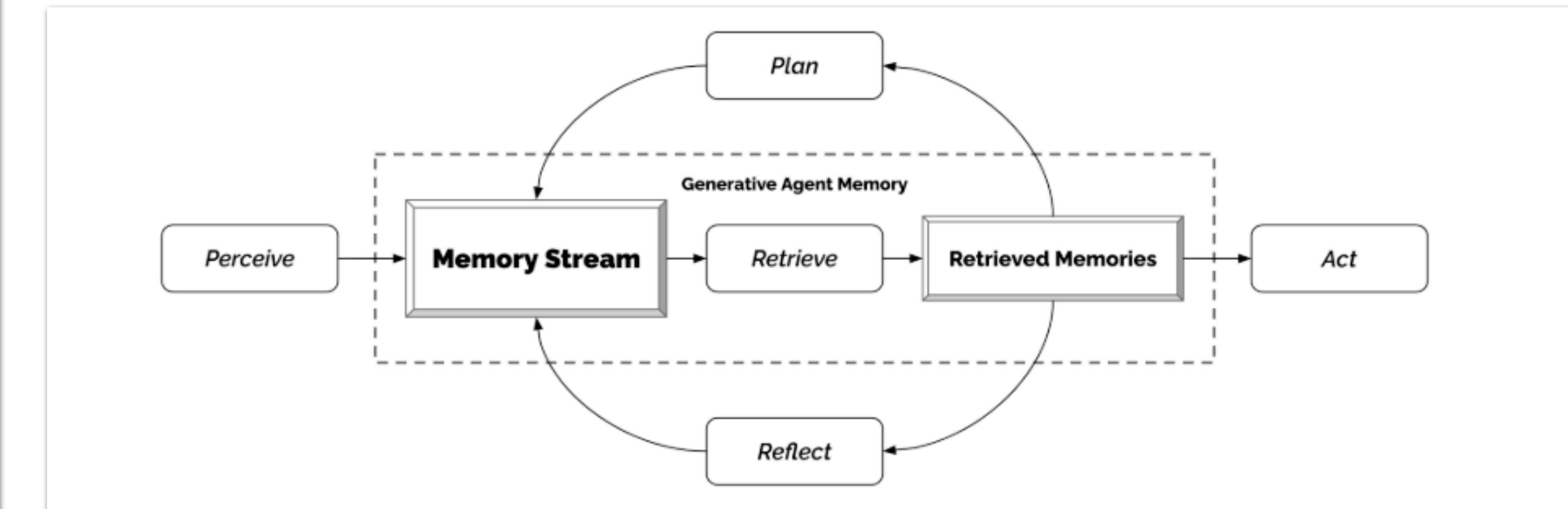
Design considerations for retrieval based system

- Hybrid search: combine semantic and syntactic (traditional) search
- Rerank: find coarse grain match, then apply fine grain re-ranking
- Use meta data to help filter out the semantic search results
- Various ways to pre and post process prompts
- Evaluation is important. Evaluate retrieval and final generation using prepared dataset.

Agentic LLM

Example

- Examples:
 - Generative agent (simulated village)

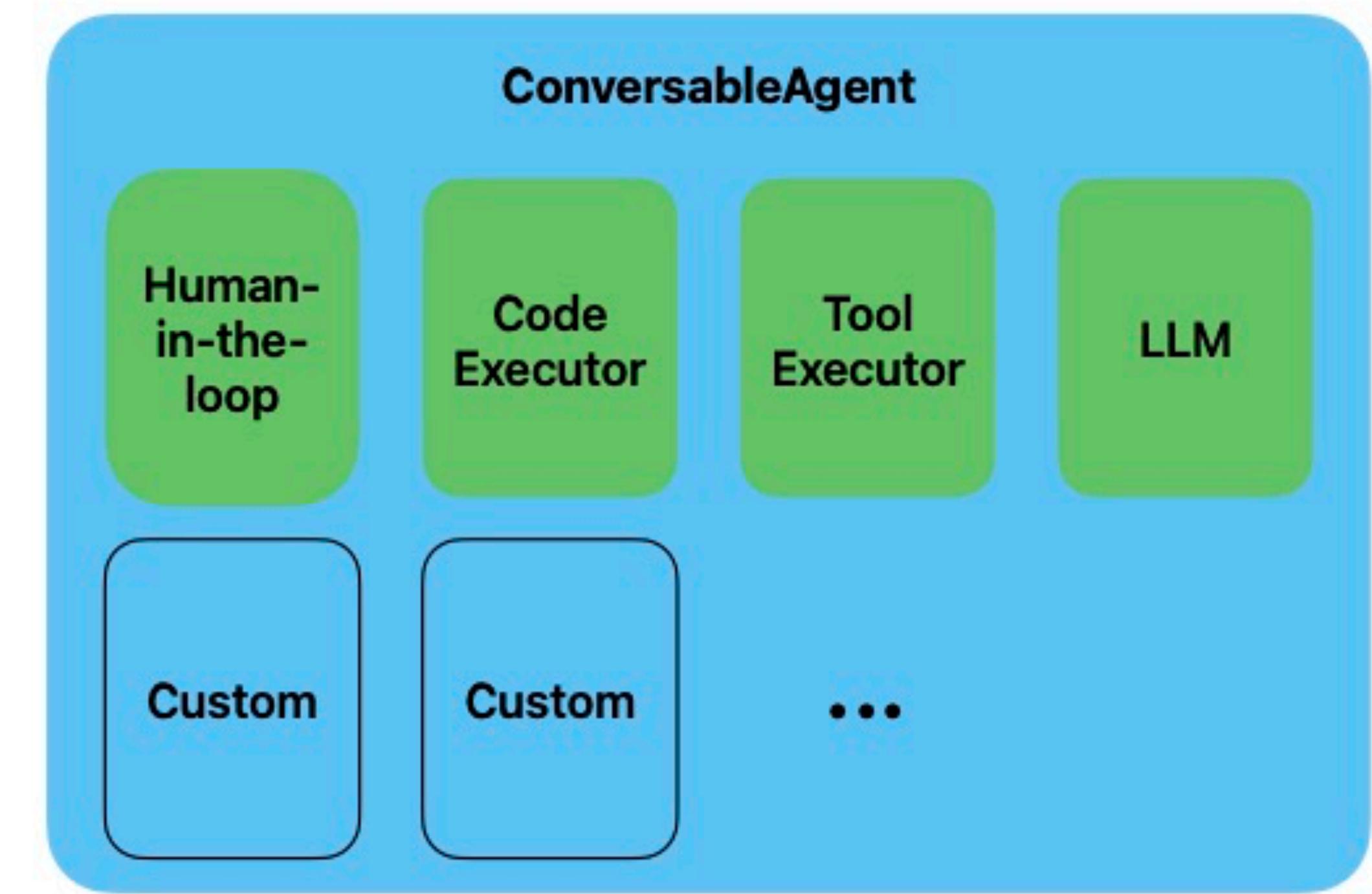


Each agent is backed by LLM with memory

Ref: <https://www.deeplearning.ai/the-batch/how-agents-can-improve-lm-performance/>

AutoGen SW framework

- Many OSS SW framework: [AutoGen](#), [MetaGPT](#), [ChatDev](#), [SWE-agent](#) (open source version of Devin)

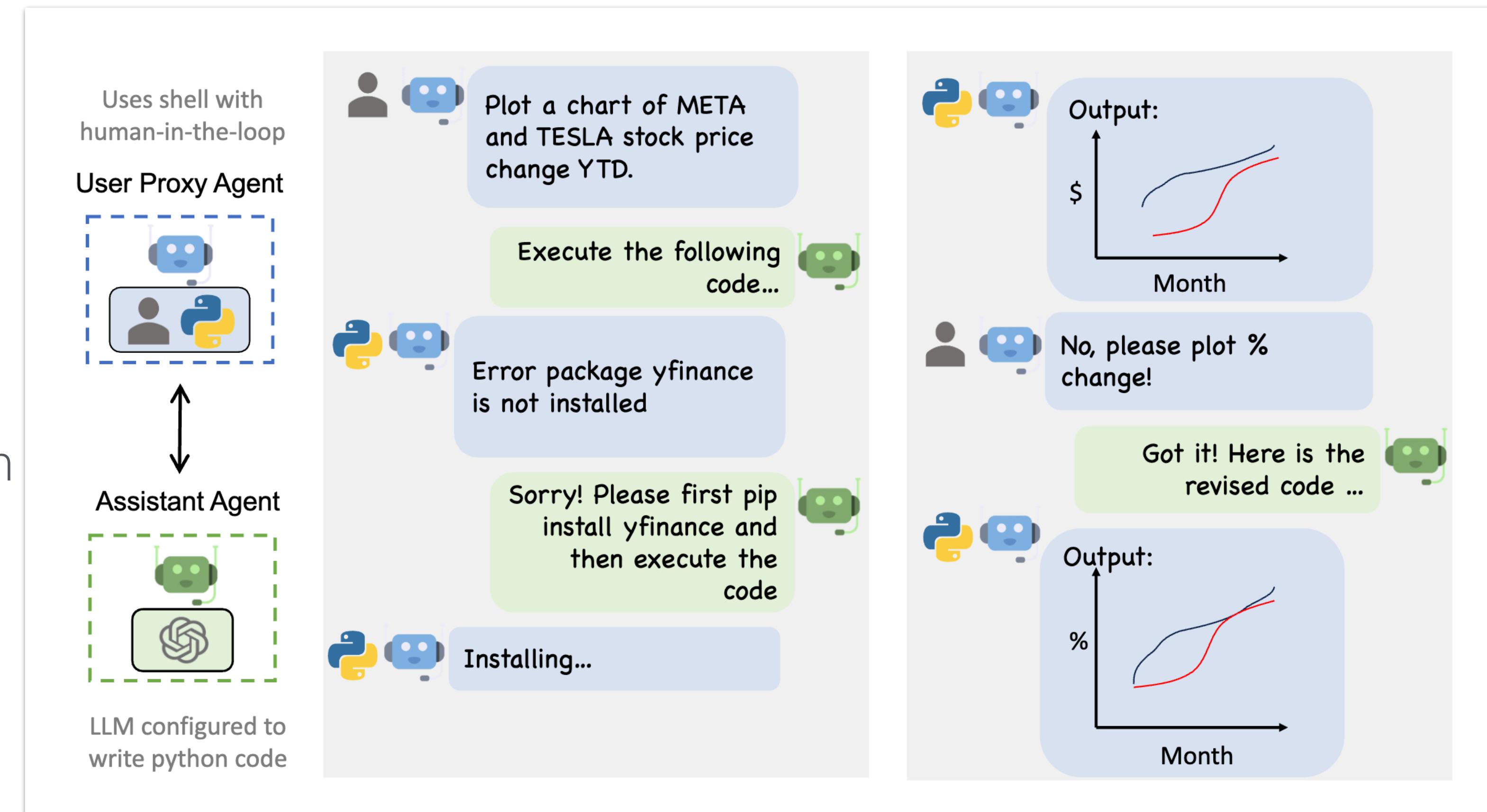


From <https://microsoft.github.io/autogen/docs/Getting-Started>

From <https://microsoft.github.io/autogen/docs/Getting-Started>

AutoGen SW framework

- We will use AutoGen as an example for this presentation
- An agent is backed by LLM, code executers, can send & receive messages to and from other agents



From <https://microsoft.github.io/autogen/docs/Getting-Started>

From <https://microsoft.github.io/autogen/docs/Getting-Started>

Fine-tuning

Performance of Fine-tuned LLM

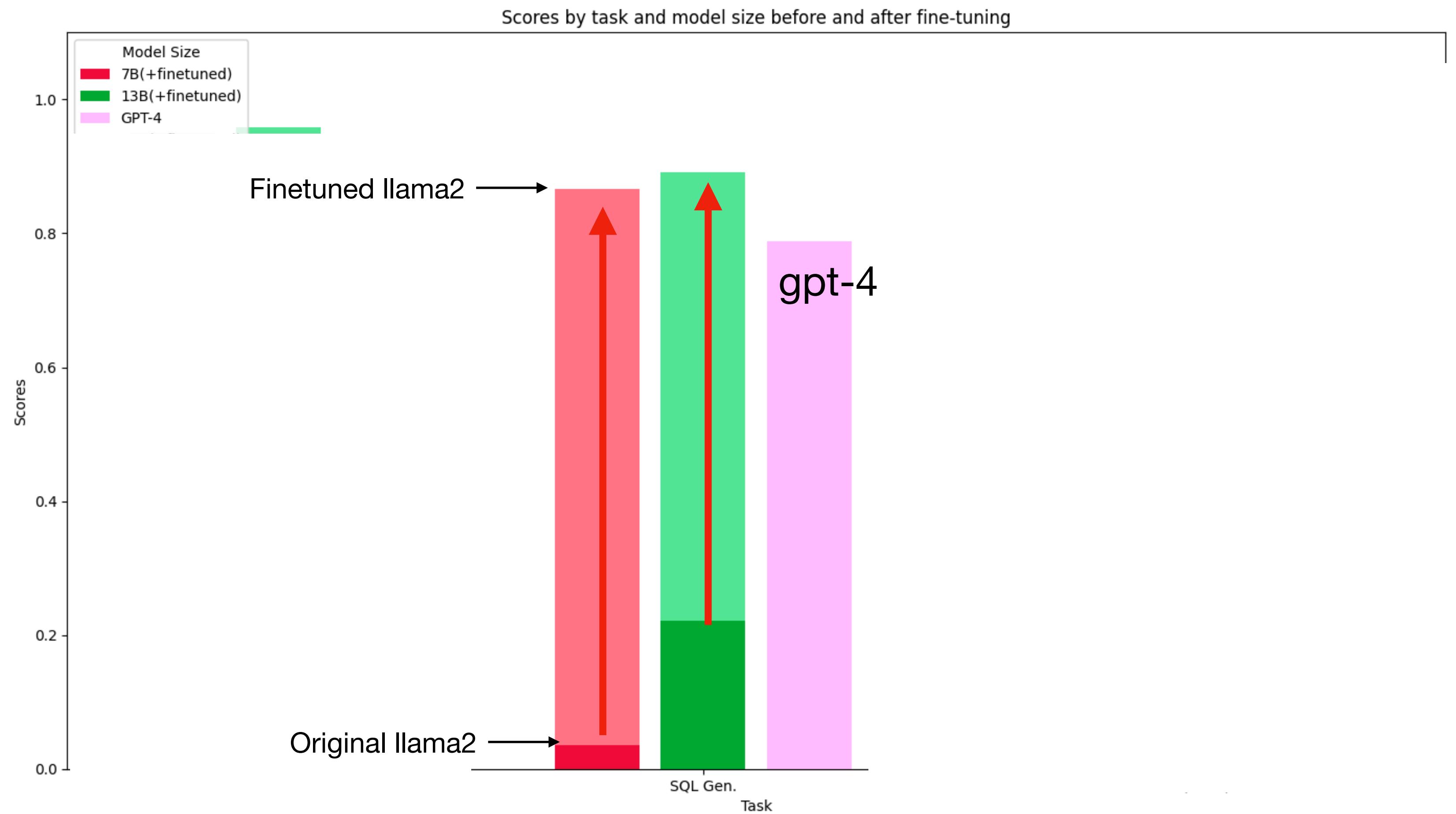
How is the performance of fine-tuned LLM?

- It depends
- For a specific task, fine-tuning open source LLM can improve the performance. It could surpass closed models' capability
- Depends on the complexity of task, degree of improvement can vary

OSS LLM Fine-tuning



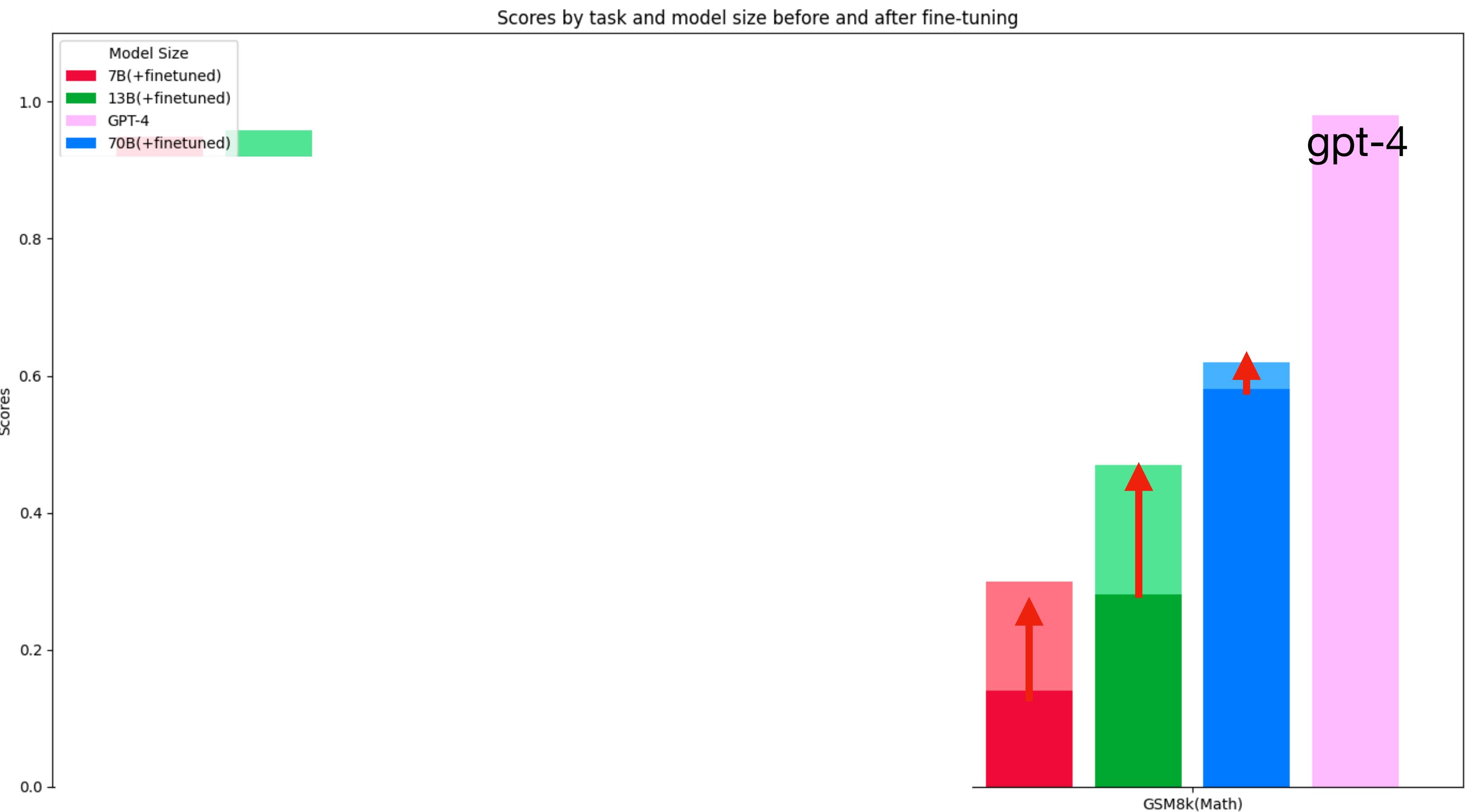
- Finetuned llama2 outperforms gpt-4 on SQL generation task
- Easier specific task can be benefited by fine-tuning



OSS LLM Fine-tuning



- Finetuned llama2 still behind gpt-4 on GSM8k (grade school math)
- Complex task may be be benefited by small scale fine-tuning

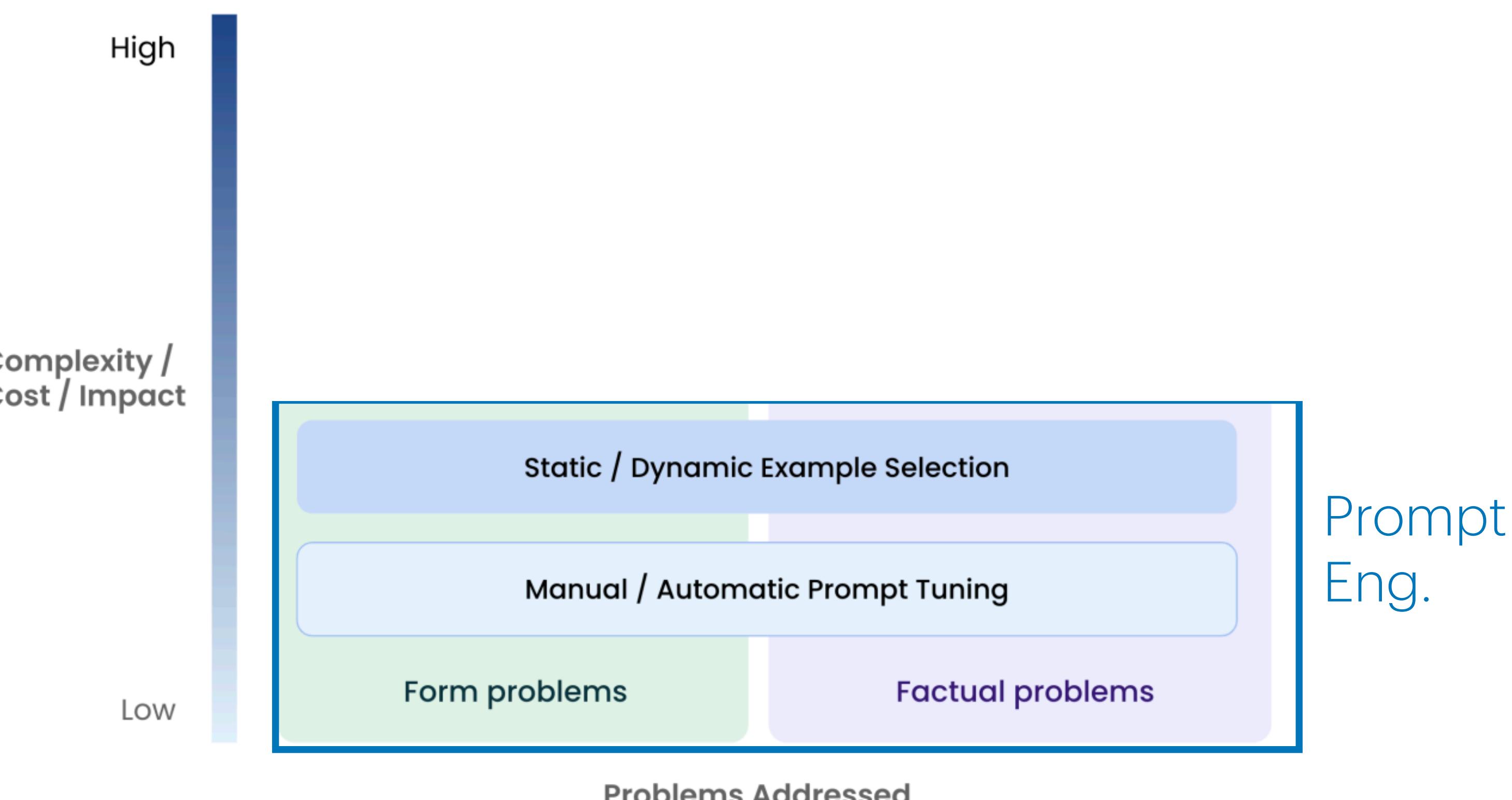


What type of tasks are good for fine-tuning?

Start with simple method first and gradually move the more involved methods

- In context learning
- RAG:
 - When dealing with factual data
- Fine tuning
 - Need to improve few shot behavior, domain adaptation
 - Reduce the known prompt patterns

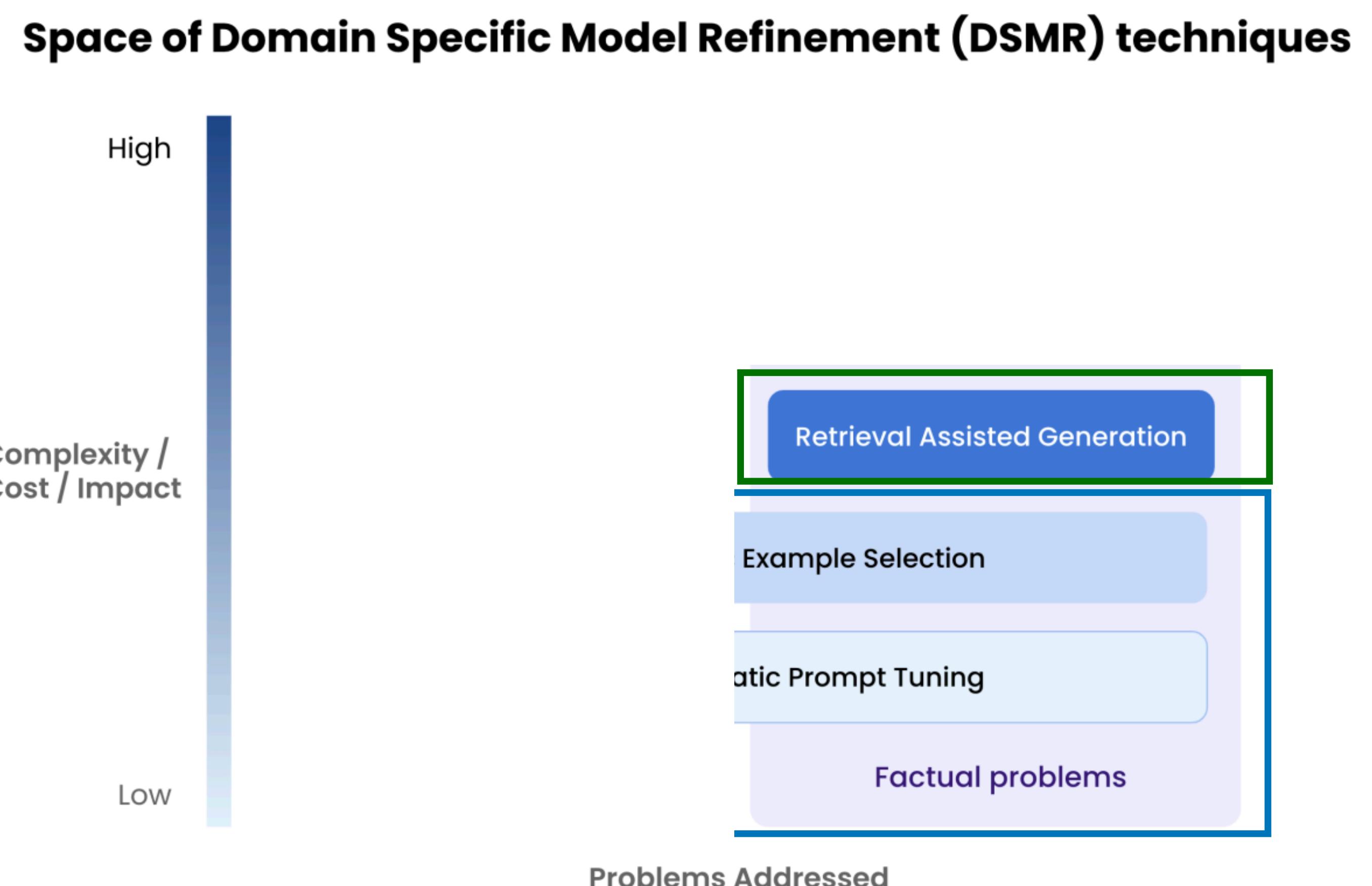
Space of Domain Specific Model Refinement (DSMR) techniques



What type of tasks are good for fine-tuning?

Start with simple method first and gradually move the more involved methods

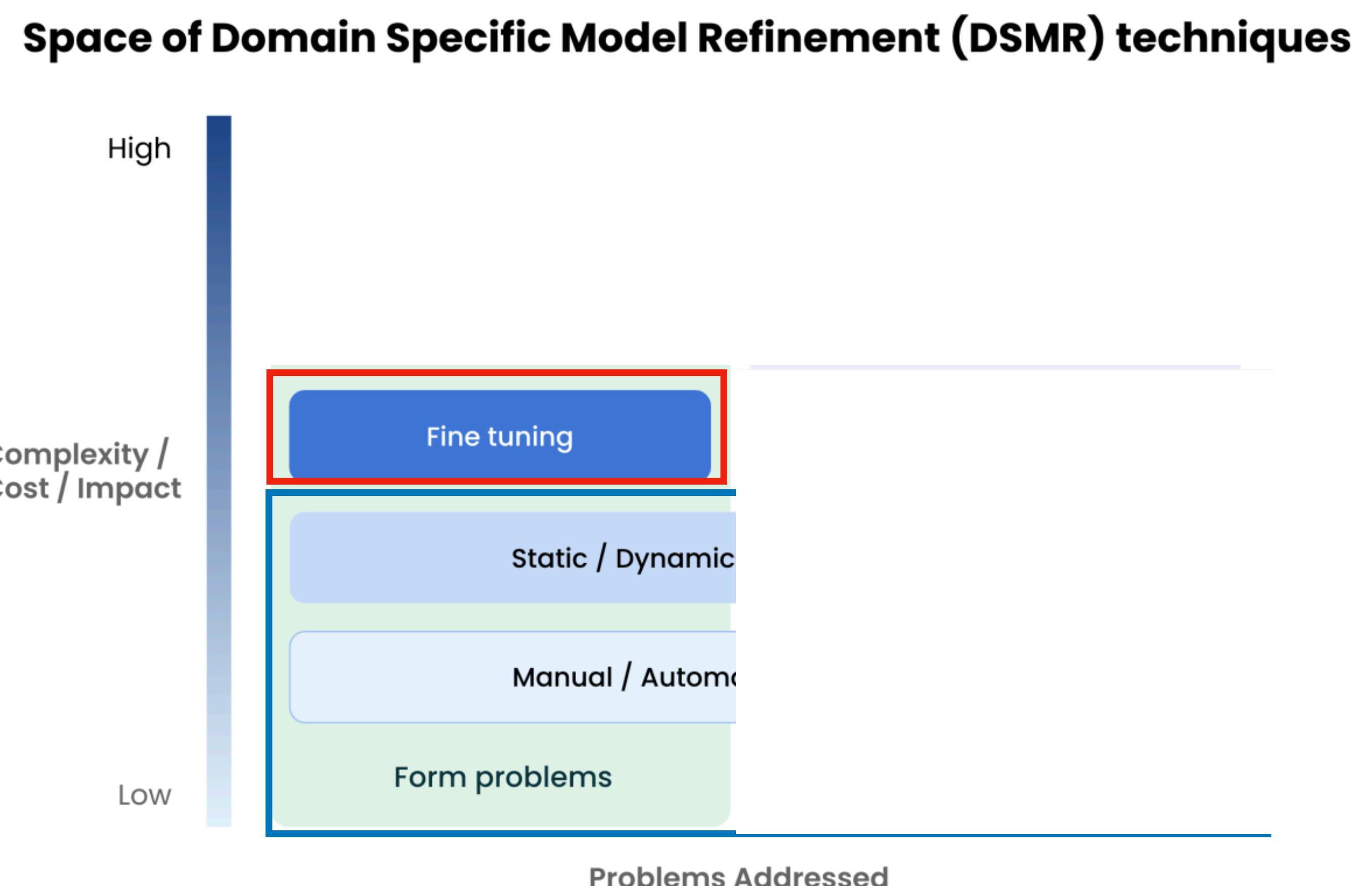
- In context learning
- RAG:
 - When dealing with factual data
- Fine tuning
 - Need to improve few shot behavior, domain adaptation
 - Reduce the known prompt patterns



What type of tasks are good for fine-tuning?

Start with simple method first and gradually move the more involved methods

- In context learning
- RAG:
 - When dealing with factual data
- Fine tuning
 - Need to improve few shot behavior, domain adaptation
 - Reduce the known prompt patterns

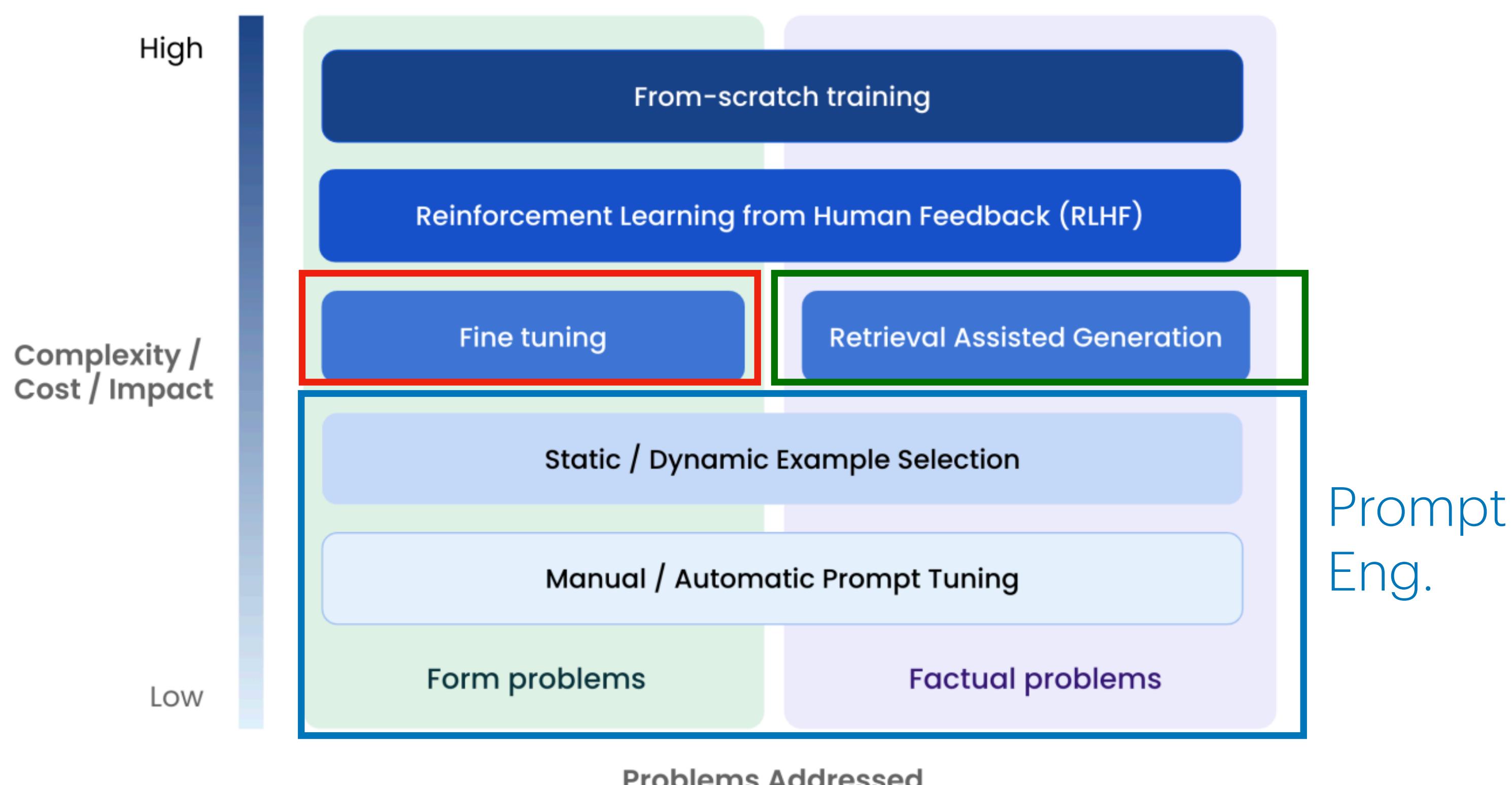


What type of tasks are good for fine-tuning?

Start with simple method first and gradually move the more involved methods

- In context learning
- RAG:
 - When dealing with factual data
- Fine tuning
 - Need to improve few shot behavior, domain adaptation
 - Reduce the known prompt patterns

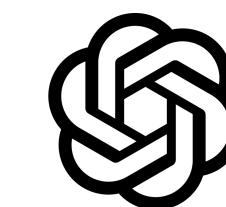
Space of Domain Specific Model Refinement (DSMR) techniques



Customization options

Fine-tuning commercial LLMs

Fine-tuning OSS LLMs



OpenAI fine-tuning service*:

- Prepare and upload training data
- Train a new fine-tuned model
- Evaluate results and go back to step 1 if needed
- Use your fine-tuned model

OpenAI fine-tune dataset example

```
{"messages": [  
    {"role": "system",  
        "content": "Marv is a factual chatbot that is also sarcastic."},  
    {"role": "user",  
        "content": "What's the capital of France?"},  
    {"role": "assistant",  
        "content": "Paris, as if everyone doesn't know that already."}  
]
```

<https://platform.openai.com/docs/guides/fine-tuning>
<https://docs.cohere.com/docs/training-custom-models>

<https://www.bloomberg.com/company/press/bloomberggpt-50-billion-parameter-llm-tuned-finance/>

<https://blog.replit.com/llm-training>

<https://openai.com/pricing>

<https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset>

<https://openai.com/blog/openai-partners-with-scale-to-provide-support-for-enterprises-fine-tuning-models>

*: GPT-3.5 Turbo

* GPT-4-turbo fine-tuning service is just started

Outlines

LLM Overview

Why consider your own LLMs

 Which LLMs to choose

Train your LLMs

Preparing Dataset for training

Evaluating LLMs

How to serve your own LLMs

Which Open source LLMs

Target tasks

Requirements

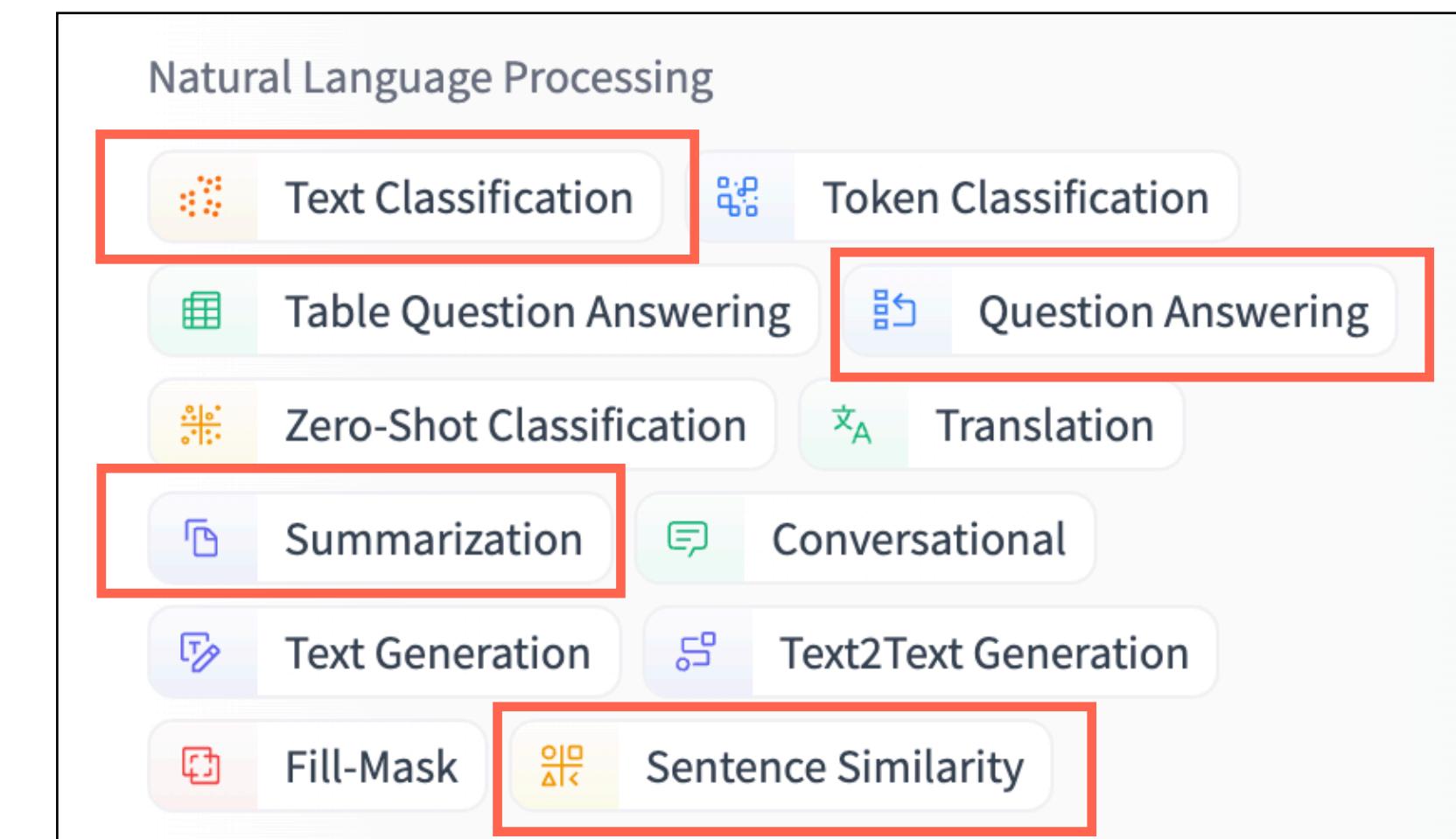
Model: size, structure, license

Dataset used

Training type: base, instruction-tuned

Evaluation: leaderboard

Iterative development



Which LLMs

Start with smaller model first

Target tasks

Requirements

Model: size, structure, license

Dataset used

Training type: base, instruction-tuned

Evaluation: leaderboard

Iterative development

Example model card

The screenshot shows a model card for the Llama-2-7b model on the Hugging Face platform. At the top, there's a logo for Hugging Face and a search bar. Below the search bar, the model name "meta-llama/Llama-2-7b" is displayed along with a "like" button and a count of 2.12k likes. There are also buttons for "Text Generation" and "PyTorch". A "Model card" tab is selected, showing the current page, and a "Files and versions" tab is also present. A "Gated model" notice indicates that access has been granted. The main content area is titled "Llama 2" and describes it as a collection of pretrained and fine-tuned generative text models ranging from 7 billion to 70 billion parameters. It links to the 7B pretrained model and other models. A "Model Details" section at the bottom includes a note about the Meta license and instructions to visit the website and accept the license before requesting access.

Hugging Face

meta-llama/Llama-2-7b 2.12k

Text Generation PyTorch English facebook meta llama llama-2 arxiv:2307.09288

Model card Files and versions

Gated model You have been granted access to this model

Llama 2

Llama 2 is a collection of pretrained and fine-tuned generative text models ranging in scale from 7 billion to 70 billion parameters. This is the repository for the 7B pretrained model. Links to other models can be found in the index at the bottom.

Model Details

Note: Use of this model is governed by the Meta license. In order to download the model weights and tokenizer, please visit the [website](#) and accept our License before requesting access here.

Which LLMs

Target tasks

Requirements

Model: size, structure, license

Dataset used

Training type: base, instruction-tuned

Evaluation:

Iterative development

- See if the model is used in similar tasks
- Check the evaluations score if the evaluation is similar to your task
- Start with small model and small size of dataset
- Run fraction of epoch to measure the performance
- Increase model size and/or increase dataset
- Iterate

Outlines

LLM Overview

Why consider your own LLMs

Which LLMs to choose

 Train your LLMs

Preparing Dataset for training

Evaluating LLMs

How to serve your own LLMs

Train your LLMs



Huggingface:

- One stop shop for LLMs
- Model repo
- Dataset repo
- LLM libraries
- Model serving
- Model training

A screenshot of the Huggingface website. At the top, there is a search bar with the placeholder "Search models, datasets, users...". Below the search bar is a navigation menu with links for "Models", "Datasets", "Spaces", "Docs", "Solutions", "Pricing", and "Log In". The main content area features a large yellow emoji-like character with hands clasped together. To its right, the text "The AI community building the future." is displayed in large, bold, white font. Below this text, a smaller paragraph reads "The platform where the machine learning community collaborates on models, datasets, and applications." On the right side of the page, there is a sidebar titled "Models 469,541" with a "Filter by name" button. The sidebar lists various model categories and specific models with their details: "meta-llama/Llama-2-70b", "stabilityai/stable-diffusion-xl-base-0.9", "openchat/openchat", "illyasviel/ControlNet-v1-1", "cerspense/zeroscope_v2_XL", "meta-llama/Llama-2-13b", "tiiuae/falcon-40b-instruct", "WizardLM/WizardCoder-15B-V1.0", "CompVis/stable-diffusion-v1-4", "stabilityai/stable-diffusion-2-1", and "Salesforce/xgen-7b-8k-inst". Each model entry includes a small icon, the model name, a brief description, and some statistics like "Text Generation • Updated 4 days ago • 25.2k • 64".

Train your LLMs

Dataset preparation

Train your LLMs

Model training

- Start with examples*
- Simple training code
- Systematic experiments

* see examples in reference section; due to rapid changes in libraries, you may need to change code/api depends on the library version you use

```
# load model
model = AutoModelForCausalLM.from_pretrained(
    model_path, device_map="auto",
)

# load tokenizer
tokenizer = AutoTokenizer.from_pretrained(
    model_path)

# load/set training arguments
training_arguments = TrainingArguments(...)

# load dataset
dataset = load_dataset(dataset_name)
train_data, val_data = dataset["train"], dataset["validation"]

# configure train
trainer = Trainer(
    model=model, args=training_args, train_dataset=train_data,
    eval_dataset=val_data
)

# train!!!
trainer.train()
```

Compute resource

GPU memory:

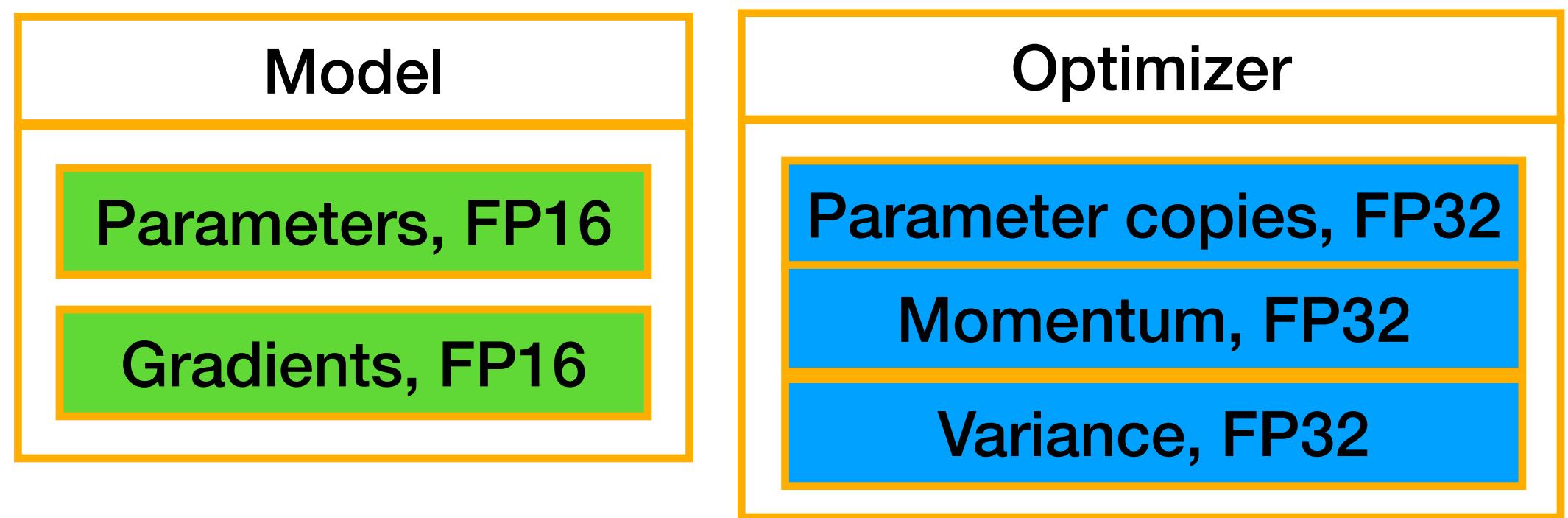
- RTX GPUs: ~24 GB
- A10G GPU: 24GB
- A100 GPU: 40/80 GB

| Numbers ¹⁰⁰ every LLM Developer  should know * | | | |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|-------------------------|---------------------------------------------------------------------------------------------------------------------|
|  Prompts | | |  Training and Fine Tuning |
| 40-90% | Amount saved by appending "Be Concise" to your prompt | ~\$1 million | Cost to train a 13 billion parameter model on 1.4 trillion tokens |
| 1.3 | Average tokens per word | <0.001 | Cost ratio of fine tuning vs training from scratch |
|  Price | | |  GPU Memory |
| ~50 | Cost Ratio of GPT-4 to GPT-3.5 Turbo | 16GB 24GB 40/80GB | V100 GRAM capacity A10G GRAM capacity A100 GRAM capacity |
| 5 | Cost Ratio of generation of text using GPT-3.5-Turbo vs OpenAI embedding | 2x number of parameters | Typical GPU memory requirements of an LLM for serving |
| 10 | Cost Ratio of OpenAI embedding to Self-Hosted embedding | ~1GB | Typical GPU memory requirements of an embedding model |
| 6 | Cost Ratio of OpenAI base vs fine tuned model queries | >10x | Throughput improvement from batching LLM requests |
| 1 | Cost Ratio of Self-Hosted base vs fine-tuned model queries | 1 MB | GPU Memory required for 1 token of output with a 13B parameter model |
| * Check out bit.ly/llm-dev-numbers for how we calculated the numbers | | | |
| Presented by  RAY &  anyscale with ❤️ Join the community ray.io or Request a Trial anyscale.com/signup today | | | |

Param. & GPU memory

To train, need to store the followings on the GPU memory

- model parameters, 2 Bytes (floating point 16, FP16)
- model gradients, 2 Bytes
- optimizer states (Adam), 12 Bytes



n : number of parameters

Lower bound of static memory using Adam optimizer:

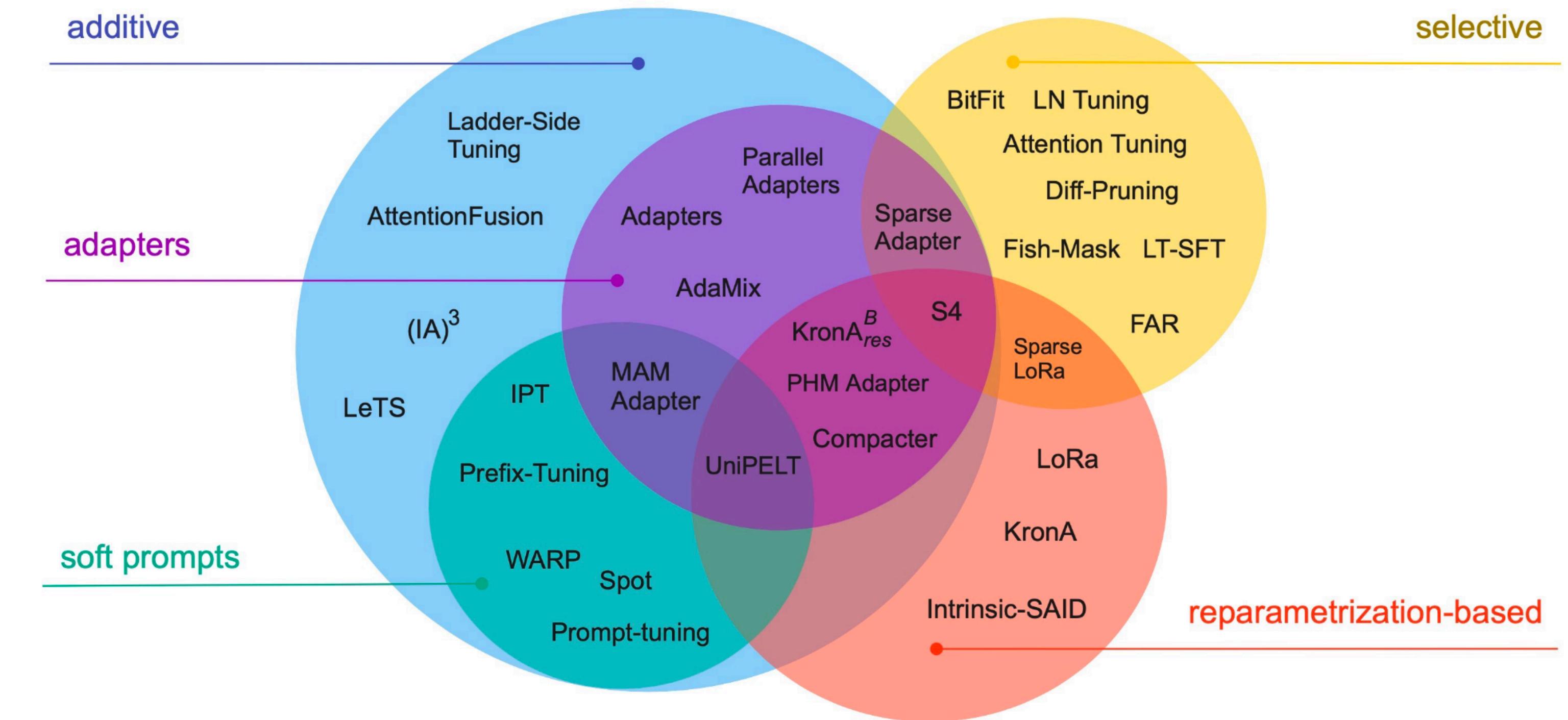
- 16 x n Bytes, (16=2+2+12)

ex> llama 2-7b, 7 b param. require ~ 112 GB GPU memory

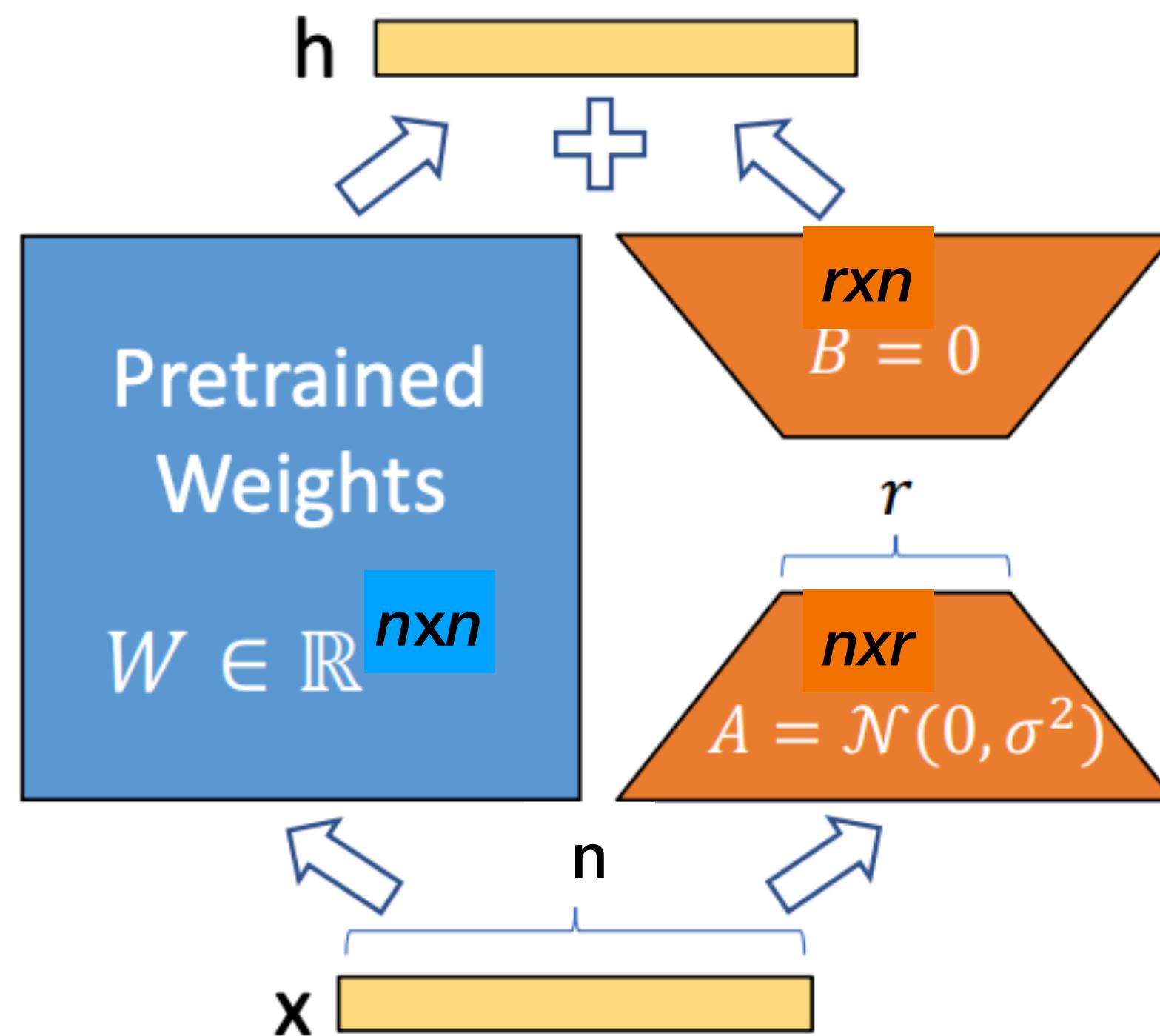
This will not fit to 80GB memory on A100 GPU!

Parameter efficient fine tuning (PEFT)

- train additional small subset, additive
- train small set of model, re-parameterization
- *Low rank adaptation (LoRa)*



Low rank adaptation (LoRa)



Low rank adaptation (LoRa)

- Freeze pretrained weights
- Only train lower rank matrices

```
def lora_linear(x):  
    """LoRA  
    W: nxn, frozen param  
    W_A: nxr, W_B: rxn, where r < n, learnable param  
    ....  
    h = x @ W # regular linear  
    h += x @ W_A @ W_B # low-rank update  
    return scale * h
```

Low rank adaptation (LoRa)

- Rank configuration
- target_module: which modules to use

Trainable parameter with lora rank 8 w.r.t original number of trainable parameters: 0.3 %

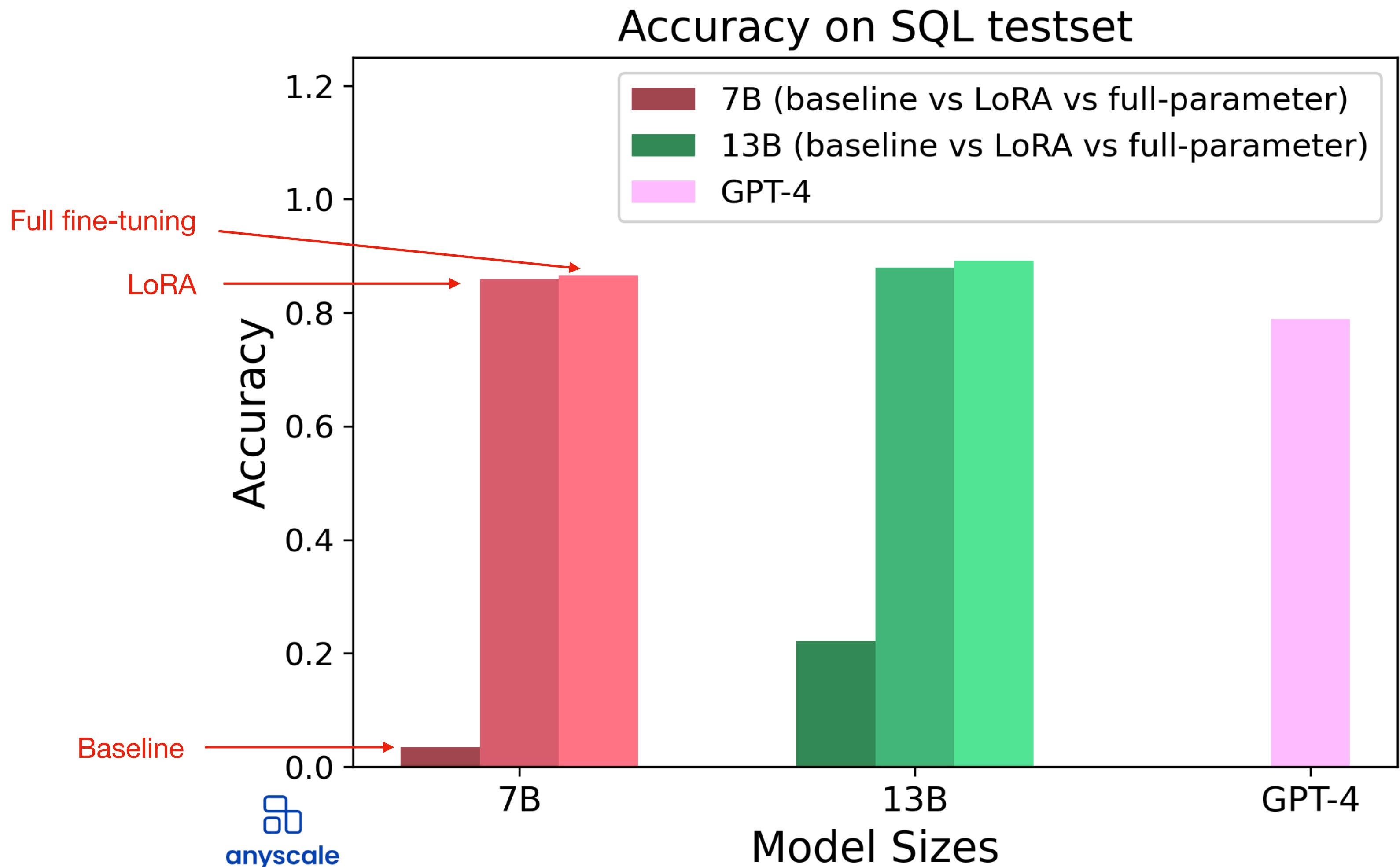
Memory required llama 2 7b model

- 112GB = 16x7
 - *14GB is for keep the frozen weight*
- => $(112 * 0.003 + 14)$ GB ~ 14 GB

```
peft_config = LoraConfig(  
    r=args.lora_r,  
    lora_alpha=args.lora_alpha,  
    lora_dropout=args.lora_dropout,  
    target_modules=modules,  
    bias="none",  
    task_type=TaskType.CAUSAL_LM,  
)  
  
model = get_peft_model(model, peft_config)
```

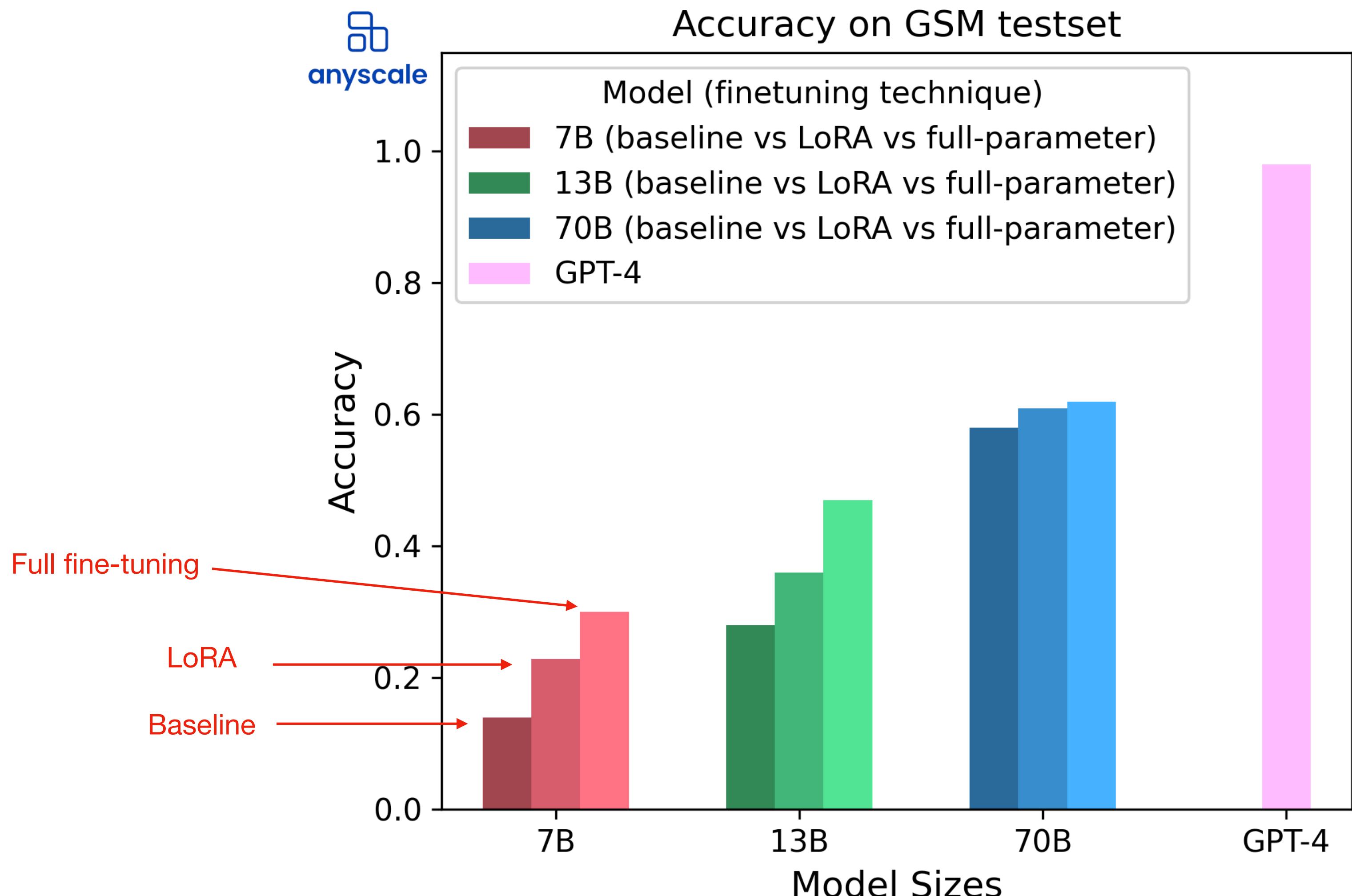
Full fine-tuning vs LoRA

- How is the performance of LoRA compared to full fine-tuning?
 - It *depends*, but small
 - Simple task (**SQL** generation)
 - Small performance difference



Full fine-tuning vs LoRA

- Complex task (**GSM8K**, grade school math)
 - Large performance difference in smaller model
 - small performance difference in larger model



Outlines

LLM Overview

Why consider your own LLMs

Which LLMs to choose

Train your LLMs

 Preparing Dataset for training

Evaluating LLMs

How to serve your own LLMs

Dataset preparation

Dataset for domain adaptation

Dataset for instruction tuning

- Prepare text dataset
- Verify dataset manually
- Apply filtering & dedup
- Continuous pre-training
- Next token prediction objective

Dataset preparation

Dataset for domain adaptation

Dataset for instruction tuning

- Prepare Q&A pair dataset
- Allows chatgpt-like behavior after this training
- High quality dataset is important
- Suggested number of dataset varies from 100 to 1,000 or more. So test incrementally based on the your own results
- Self-instruct method: use LLM to generate/filter/improve Q&A pairs for dataset. There are many dataset follows this idea.

<https://platform.openai.com/docs/guides/fine-tuning>

LIMA, less is more, <https://arxiv.org/pdf/2305.11206.pdf>

Self-instruct: <https://arxiv.org/abs/2212.10560>

dolly-dataset:<https://www.databricks.com/blog/2023/03/24/hello-dolly-democratizing-magic-chatgpt-open-models.html>

Codellama2: <https://ai.meta.com/research/publications/code-llama-open-foundation-models-for-code/>

*: frozen model only used to generate text

**: model under training

Outlines

LLM Overview

Why consider your own LLMs

Which LLMs to choose

Train your LLMs

Preparing Dataset for training

 Evaluating LLMs

How to serve your own LLMs

Evaluation

- Very important to have evaluations metric early
- Find out metric that can measure the performance of the task
- LLM based evaluation can be effective

Outlines

LLM Overview

Why consider your own LLMs

Which LLMs to choose

Train your LLMs

Preparing Dataset for training

Evaluating LLMs



How to serve your own LLMs

How to serve LLMs

Post train model conversion

End point serving

- GPTQ: post train quantization (8, 4, 3, 2-bit precision) using the GPTQ algorithm
- AutoGPTQ: library that applies GPTQ for LLMs
 - Huggingface integrates AutoGPTQ library, which makes it easier to use

How to serve LLMs

Post train model conversion

End point serving

- 😊 tgi (text generation inference)
 - LLM serving tool support
 - Support most OSS models
 - Feature rich tool
- vllm
 - Easy, Fast, and Cheap LLM Serving with PagedAttention
 - Performance number is very promising