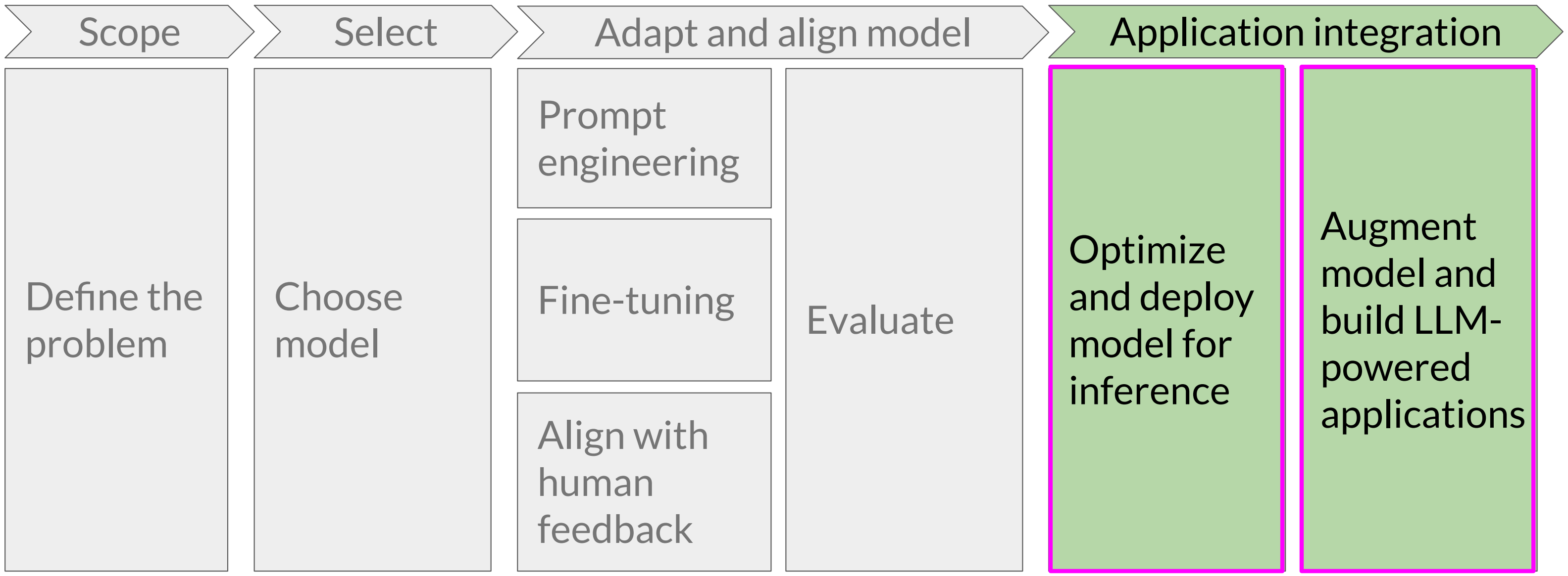
Optimize LLMs and build generative AI applications

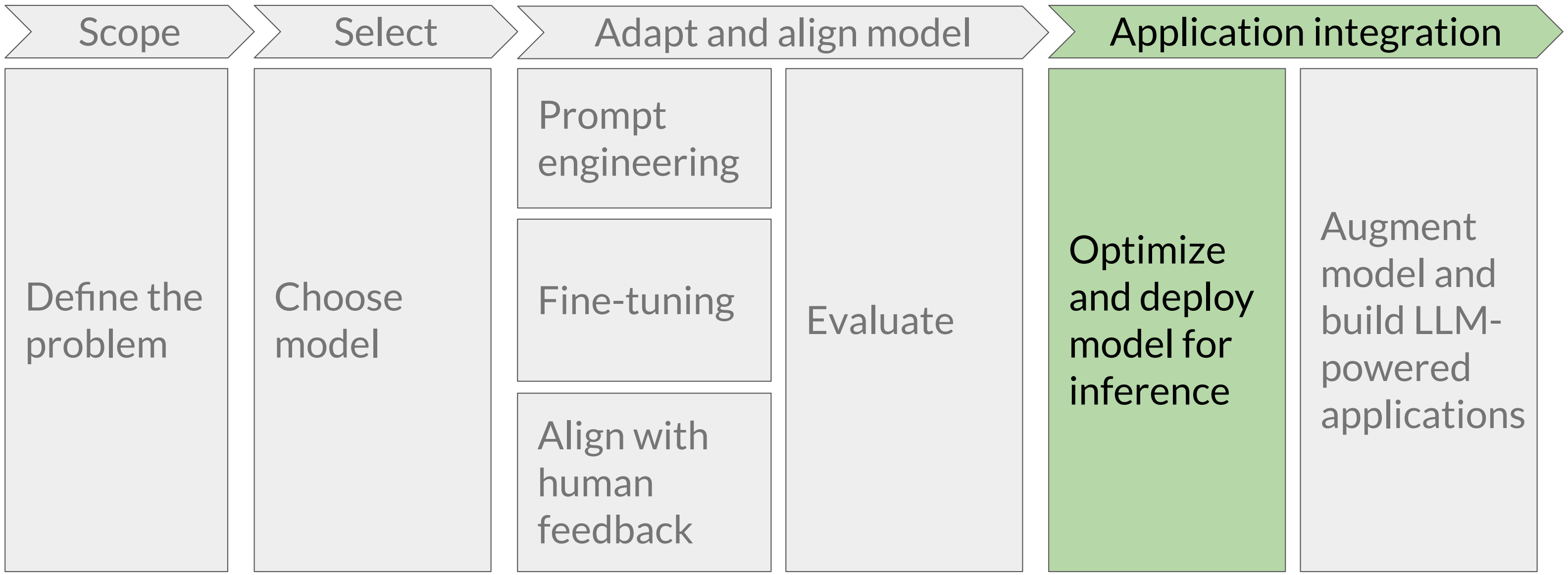
Generative AI project lifecycle

Scope	Select	Adapt and align model		Application integration	
Define the problem	Choose model	Prompt engineering Fine-tuning Align with human feedback	Evaluate	Optimize and deploy model for inference	Augment model and build LLM-powered applications

Generative AI project lifecycle

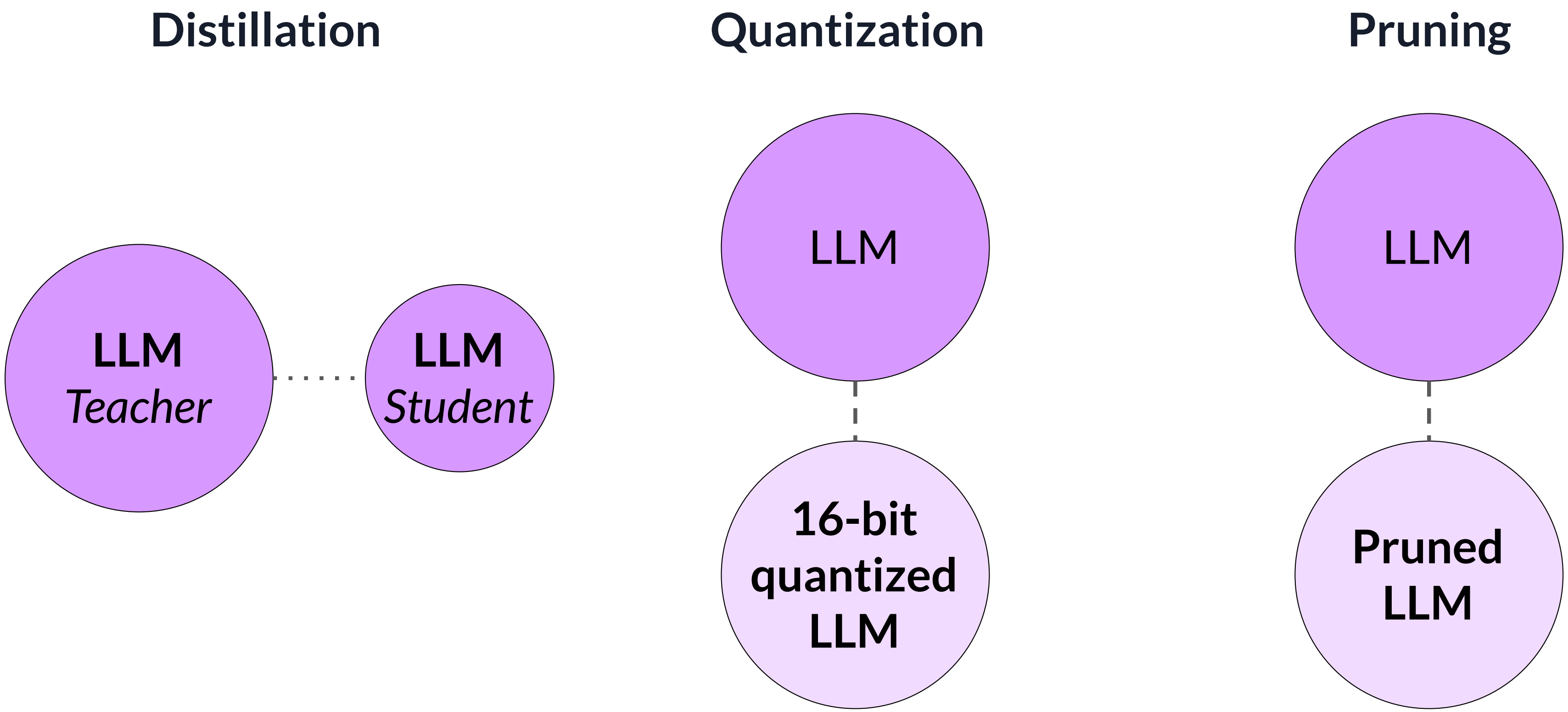


Generative AI project lifecycle

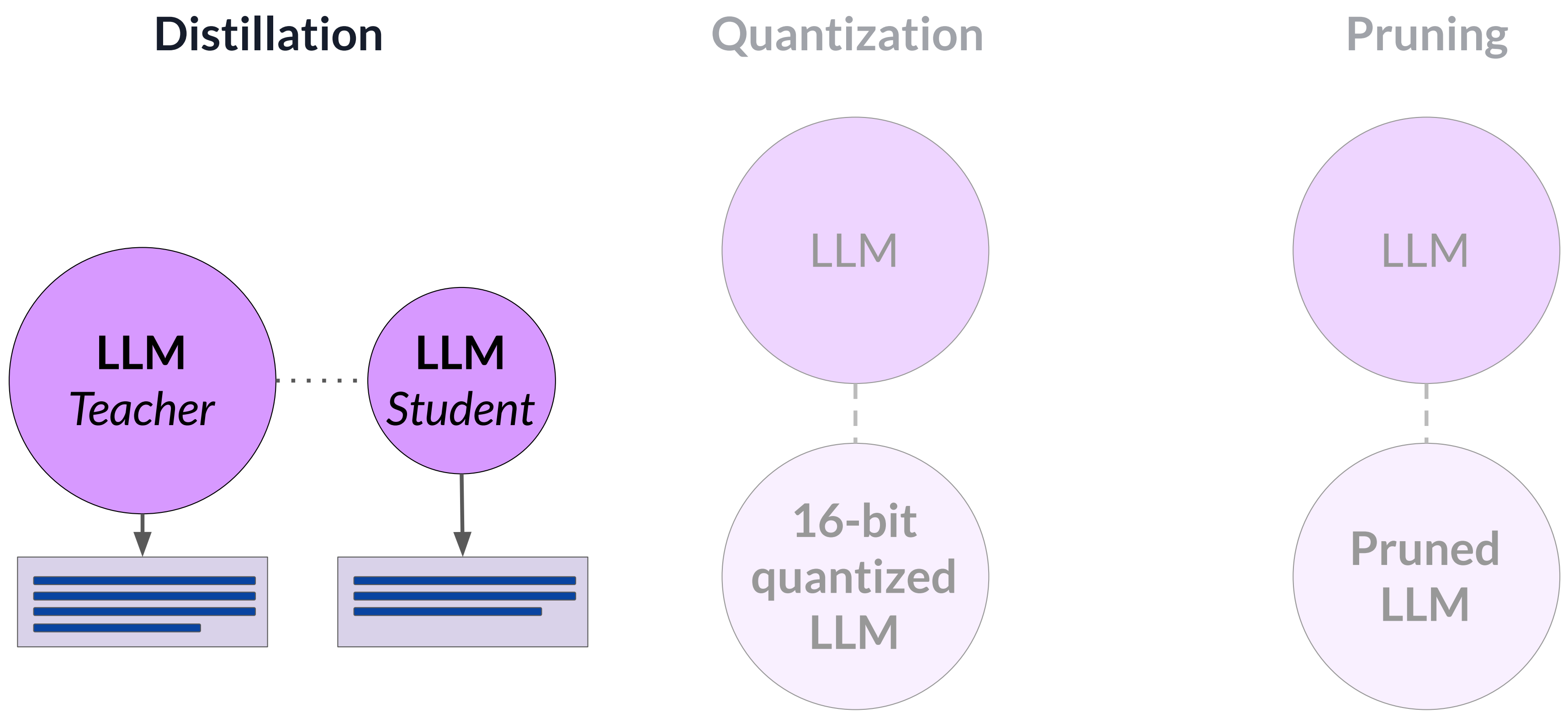


Model optimizations to improve application performance

LLM optimization techniques

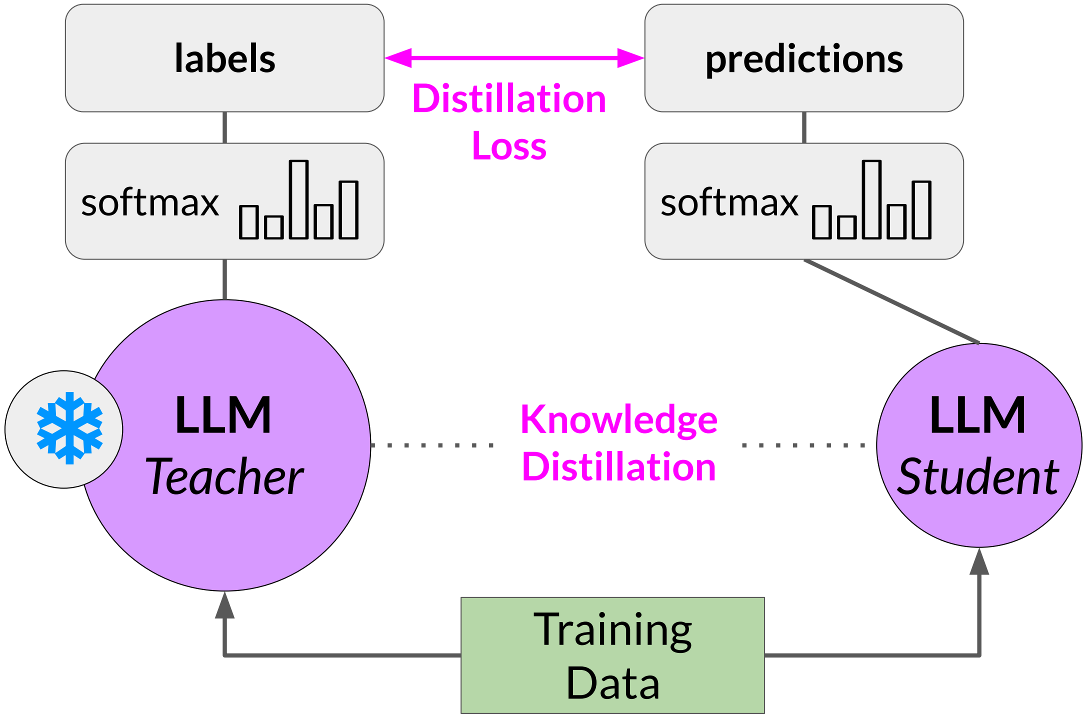


LLM optimization techniques



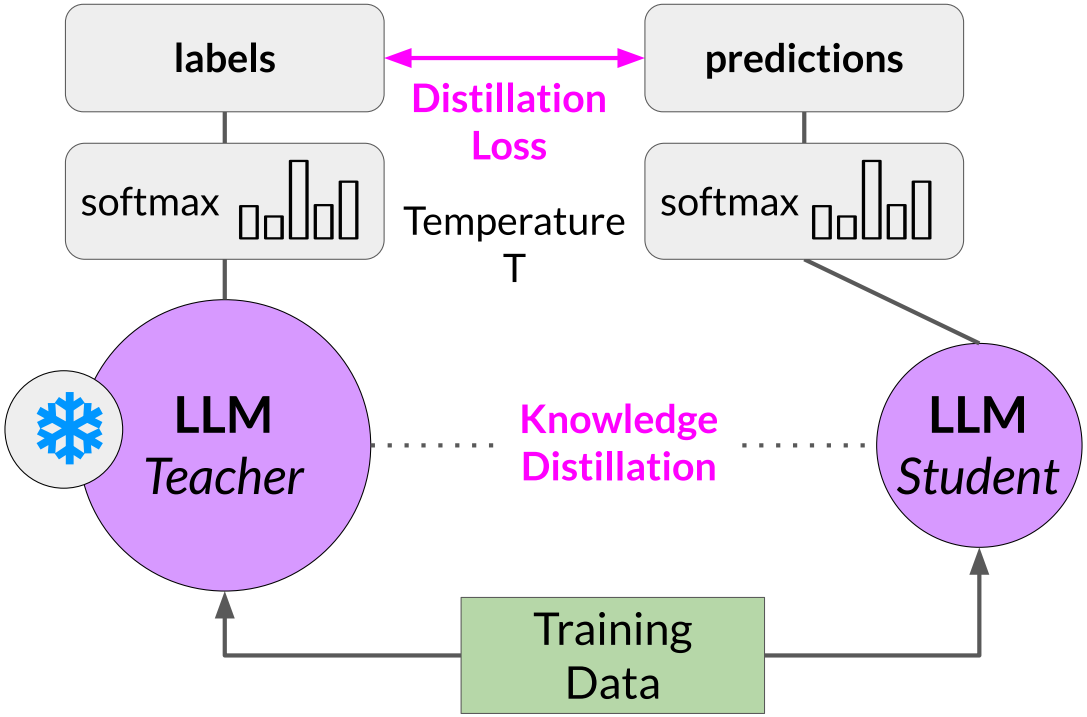
Distillation

Train a smaller student model from a larger teacher model



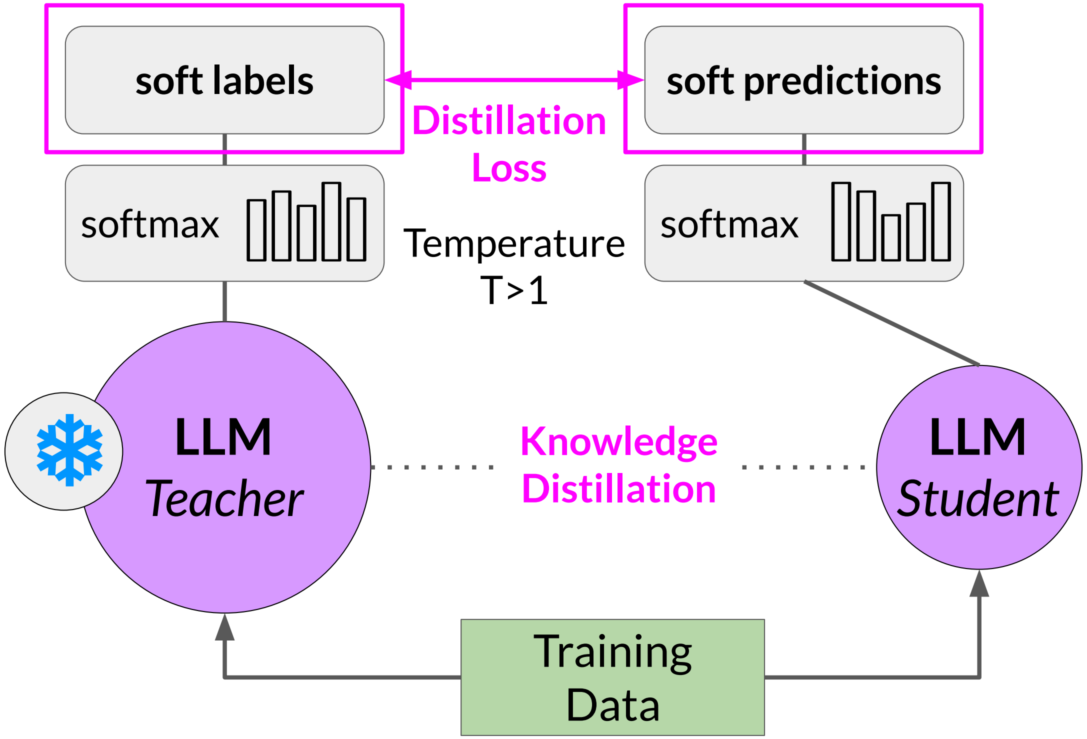
Distillation

Train a smaller student model from a larger teacher model



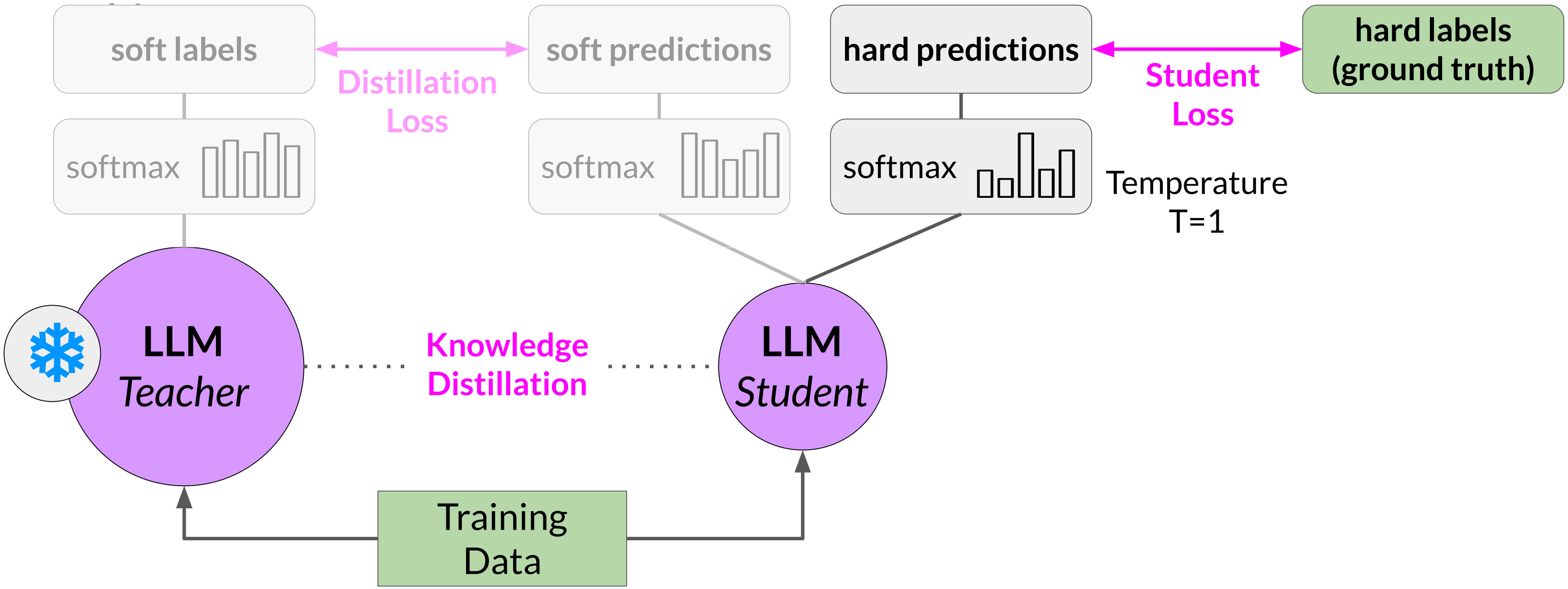
Distillation

Train a smaller student model from a larger teacher model



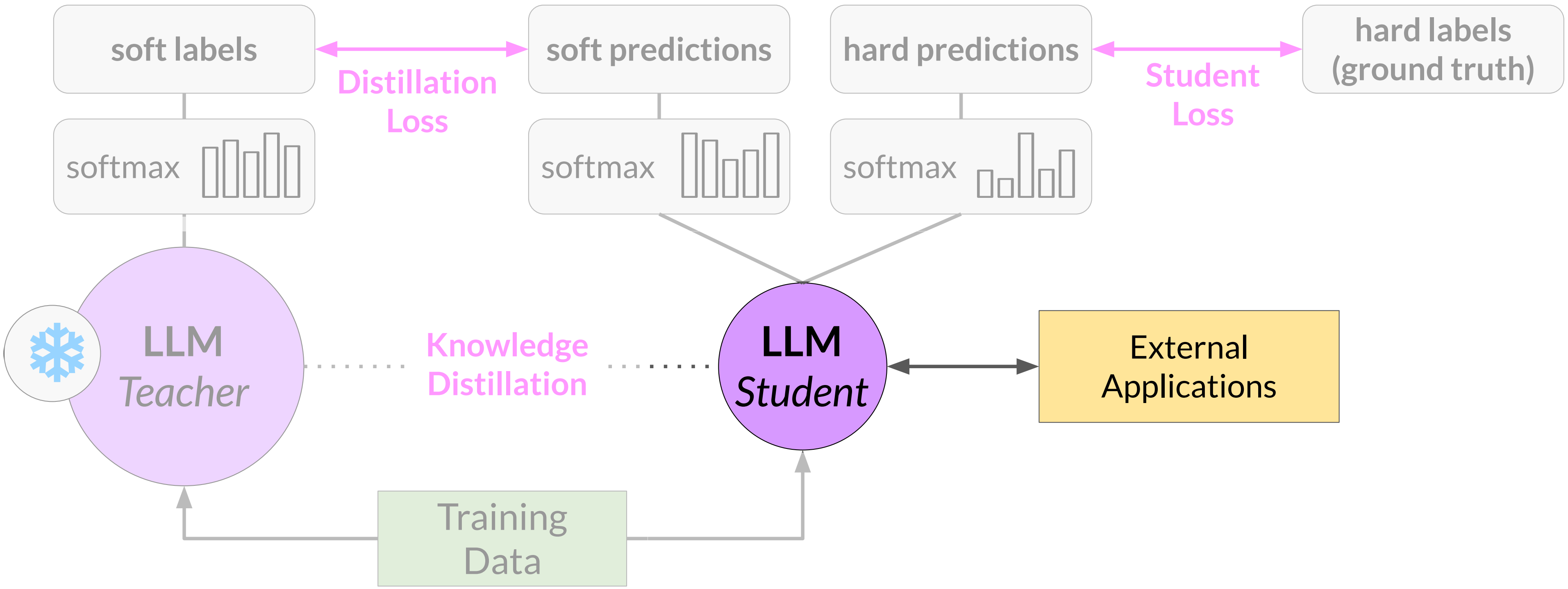
Distillation

Train a smaller student model from a larger teacher model



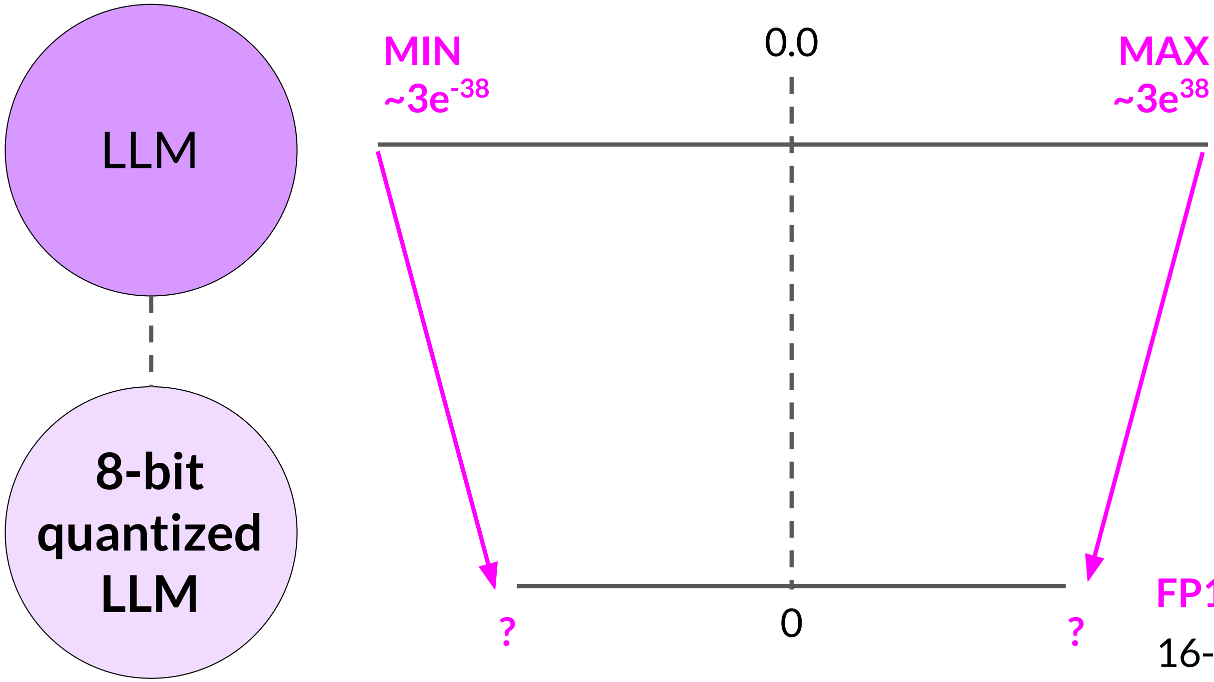
Distillation

Train a smaller student model from a larger teacher model



Post-Training Quantization (PTQ)

Reduce precision of model weights



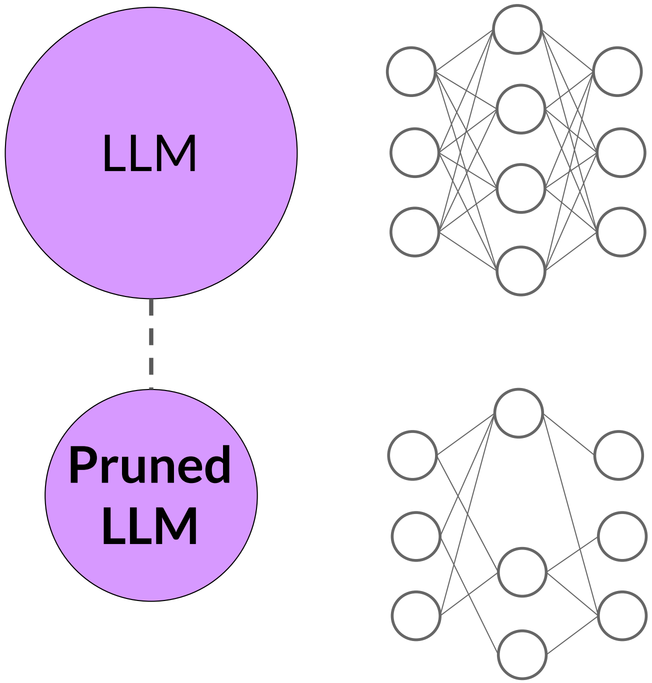
FP32
32-bit floating point

- **Applied to model weights** (and/or activations)
- **Requires calibration** to capture dynamic range

FP16 | BFLOAT16 | INT8
16-bit floating point | 8-bit integer

Pruning

Remove model weights with values close or equal to zero



- Pruning methods
  - Full model re-training
  - PEFT/LoRA
  - Post-training

- In theory, reduces model size and improves performance

- In practice, only small % in LLMs are zero-weights

Cheat Sheet - Time and effort in the lifecycle

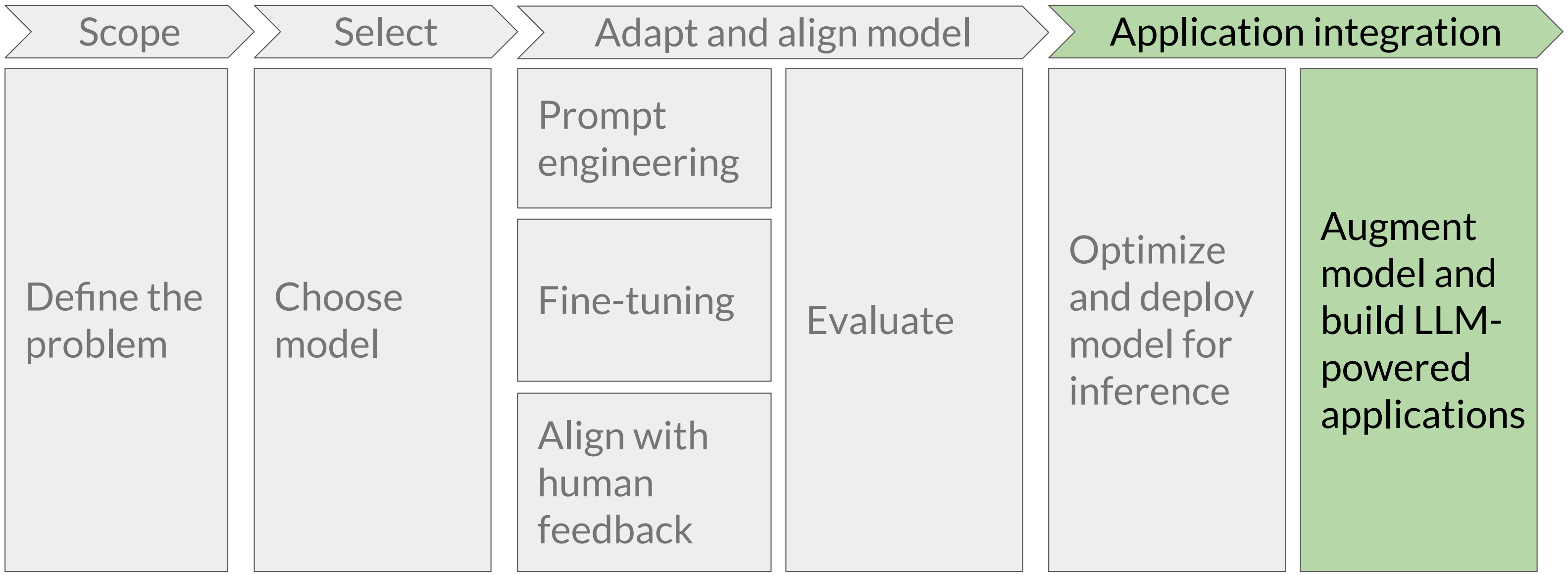
	Pre-training	Prompt engineering	Prompt tuning and fine-tuning	Reinforcement learning/human feedback	Compression/ optimization/ deployment
Training duration	Days to weeks to months	Not required	Minutes to hours	Minutes to hours similar to fine-tuning	Minutes to hours
Customization	Determine model architecture, size and tokenizer. Choose vocabulary size and # of tokens for input/context Large amount of domain training data	No model weights Only prompt customization	Tune for specific tasks Add domain-specific data Update LLM model or adapter weights	Need separate reward model to align with human goals (helpful, honest, harmless) Update LLM model or adapter weights	Reduce model size through model pruning, weight quantization, distillation Smaller size, faster inference
Objective	Next-token prediction	Increase task performance	Increase task performance	Increase alignment with human preferences	Increase inference performance
Expertise	High	Low	Medium	Medium-High	Medium

Using the LLM in applications

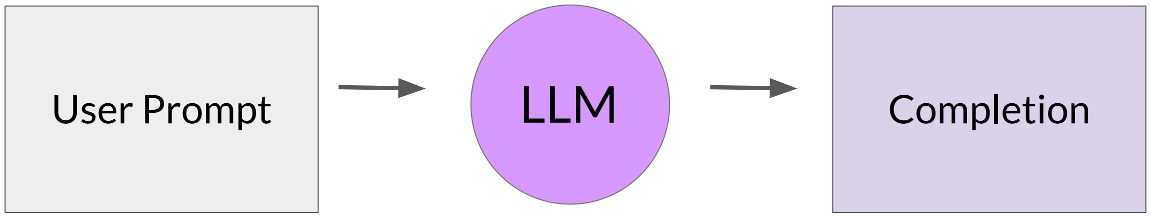
Models having difficulty

Prompt	Model	Completion	
`Who is the Prime Minister of the UK?`	LLM	`Who is the Prime Minister of the UK?` `Boris Johnson`	**Out of date**
`What is 40366 / 439?`	LLM	`What is 40366 / 439?` `92.549`	**Wrong (91.949)**
`What is a Martian Dunetree?`	LLM	`What is a Martian Dunetree?` `A Martian Dunetree is a type of extraterrestrial plant found on Mars.`	**Hallucination**

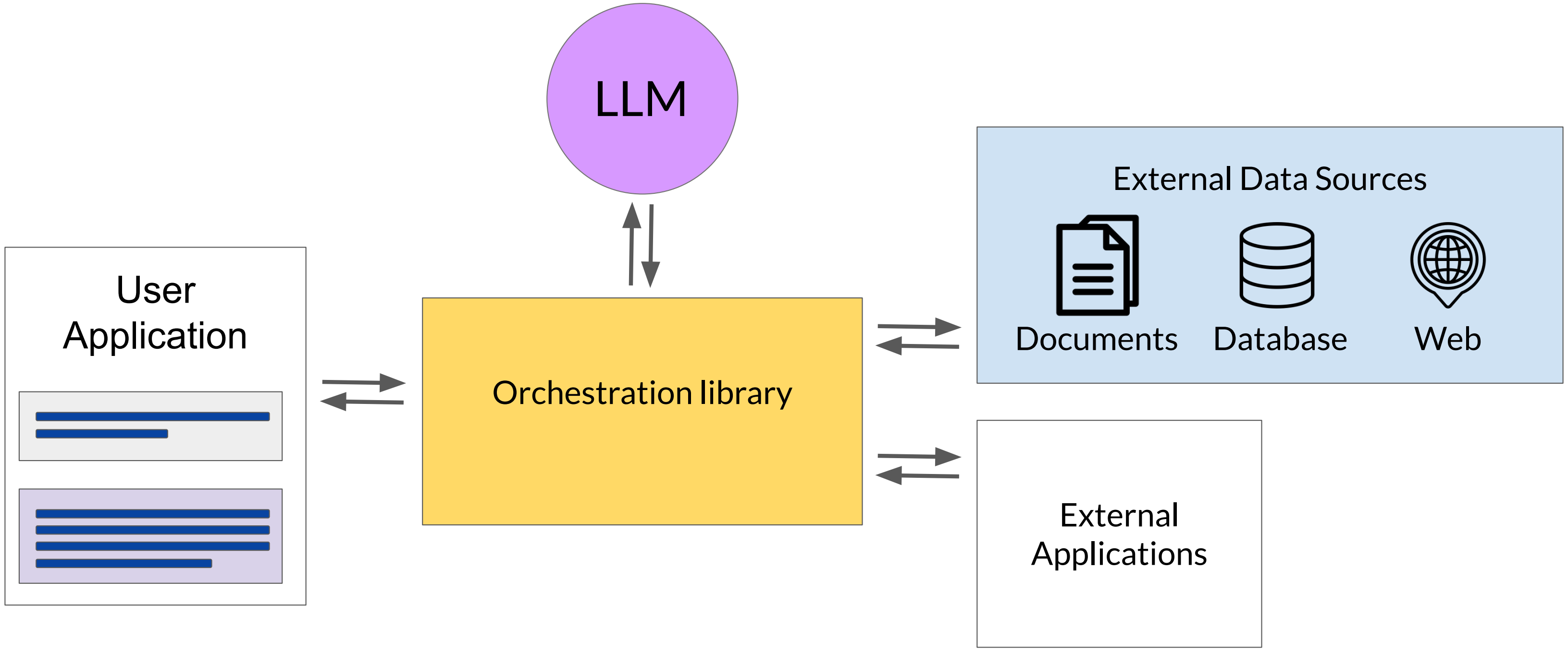
Generative AI project lifecycle



LLM-powered applications

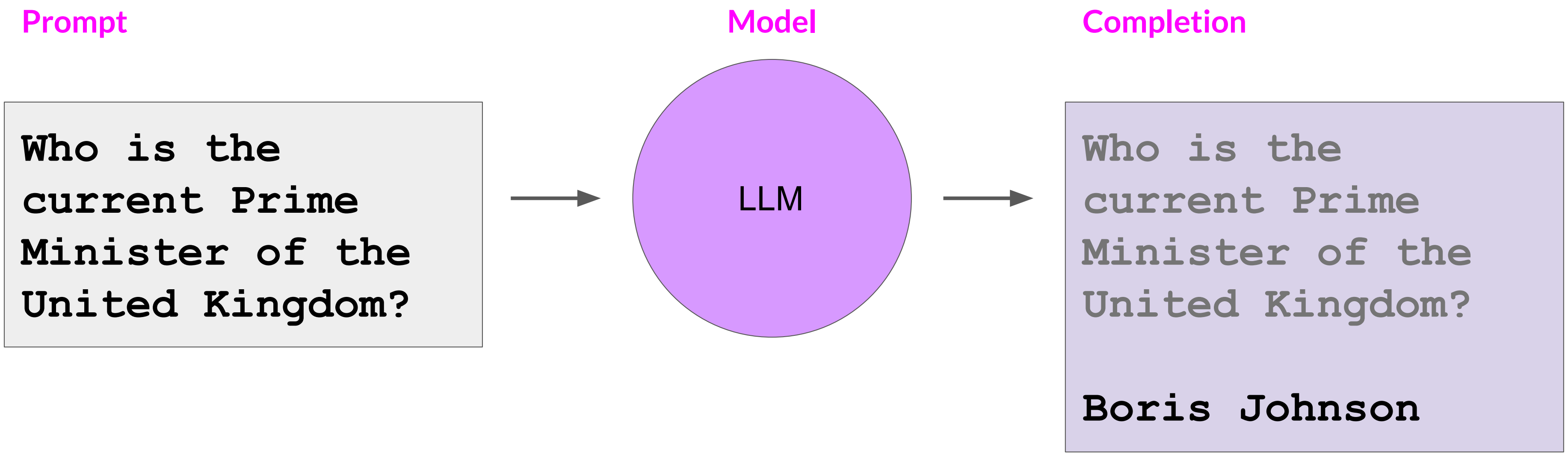


LLM-powered applications

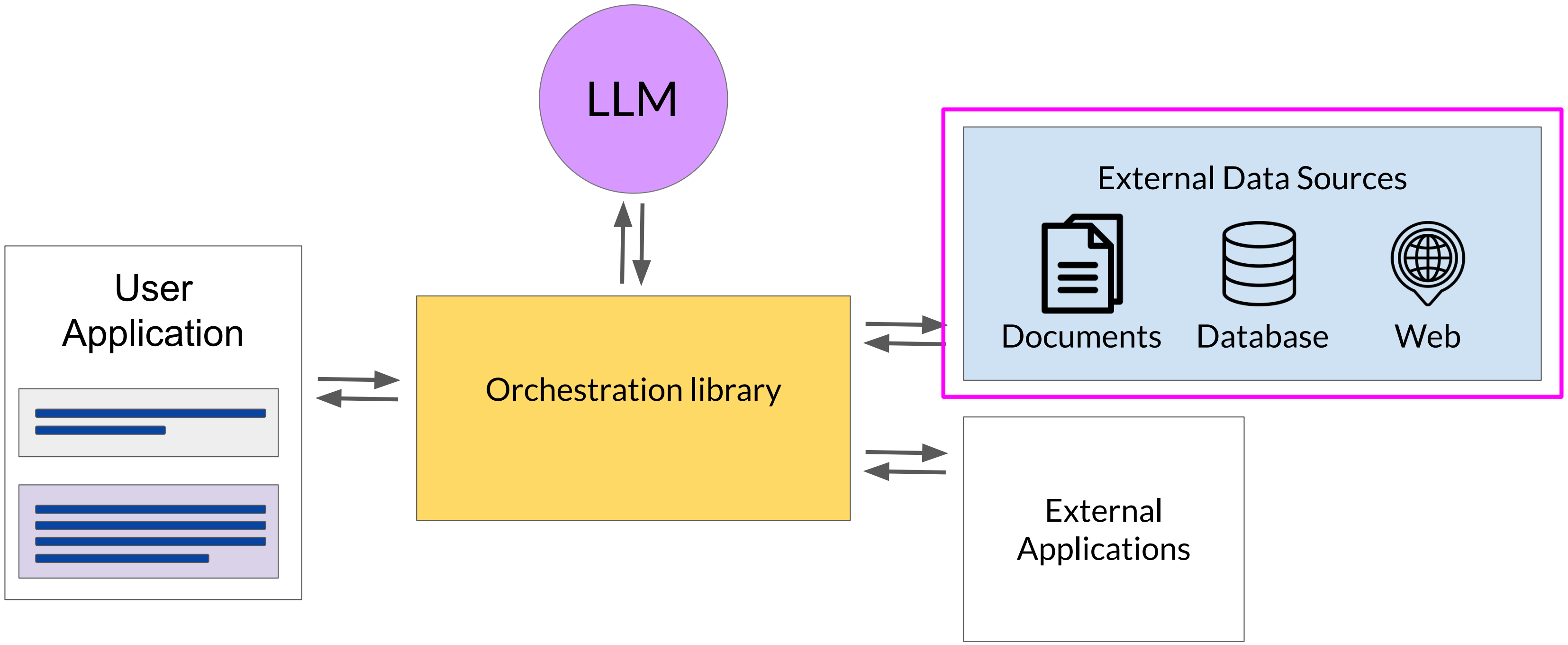


Retrieval augmented generation (RAG)

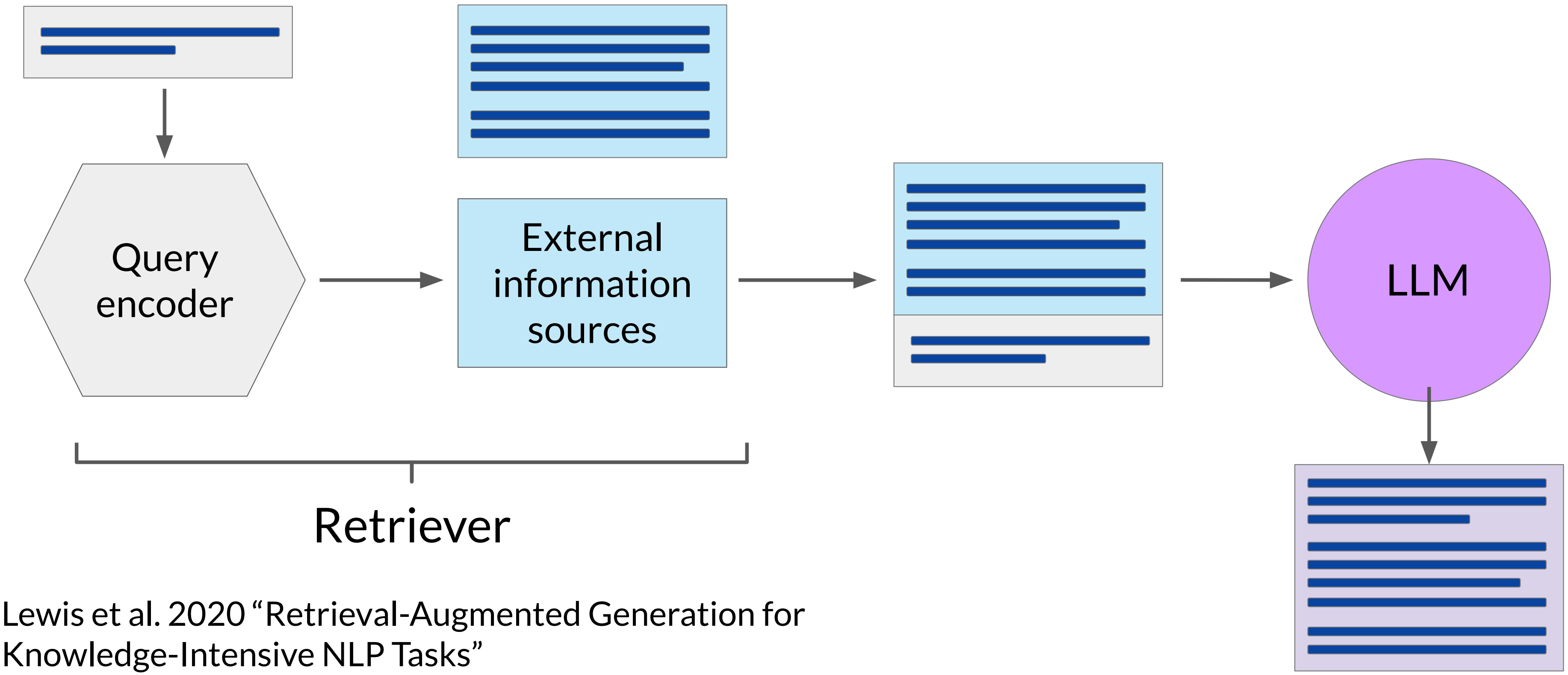
Knowledge cut-offs in LLMs



LLM-powered applications

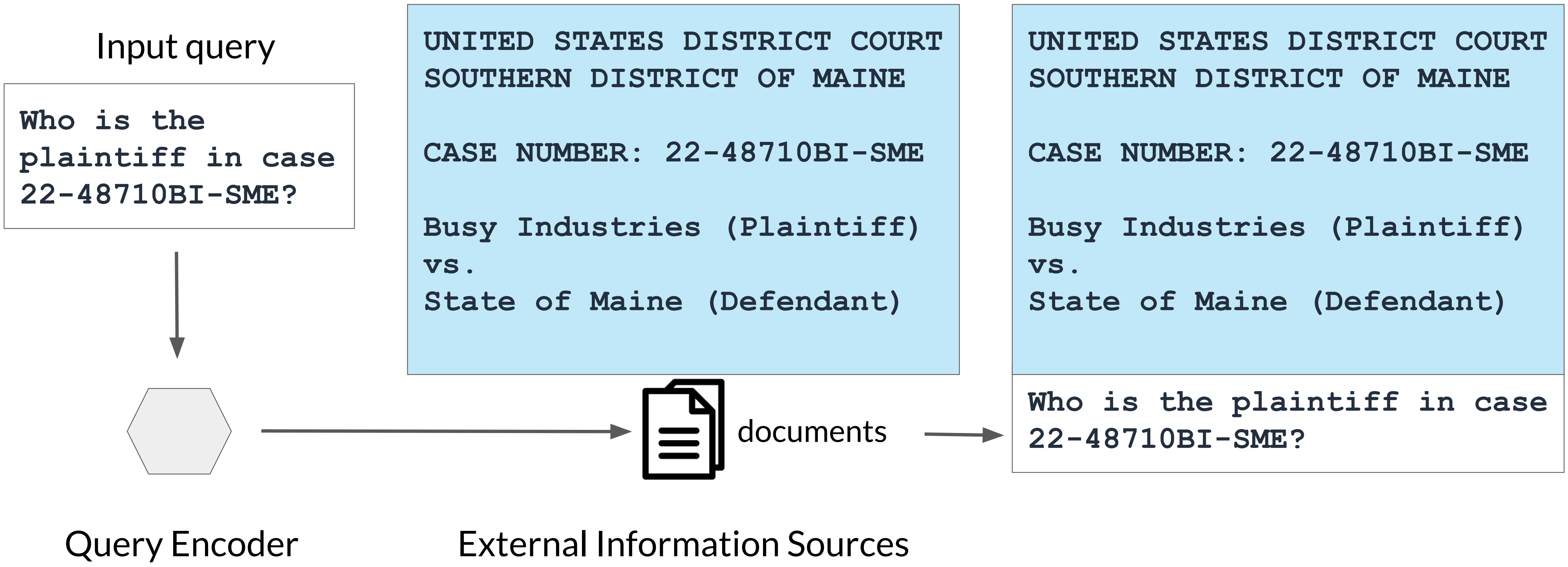


Retrieval Augmented Generation (RAG)

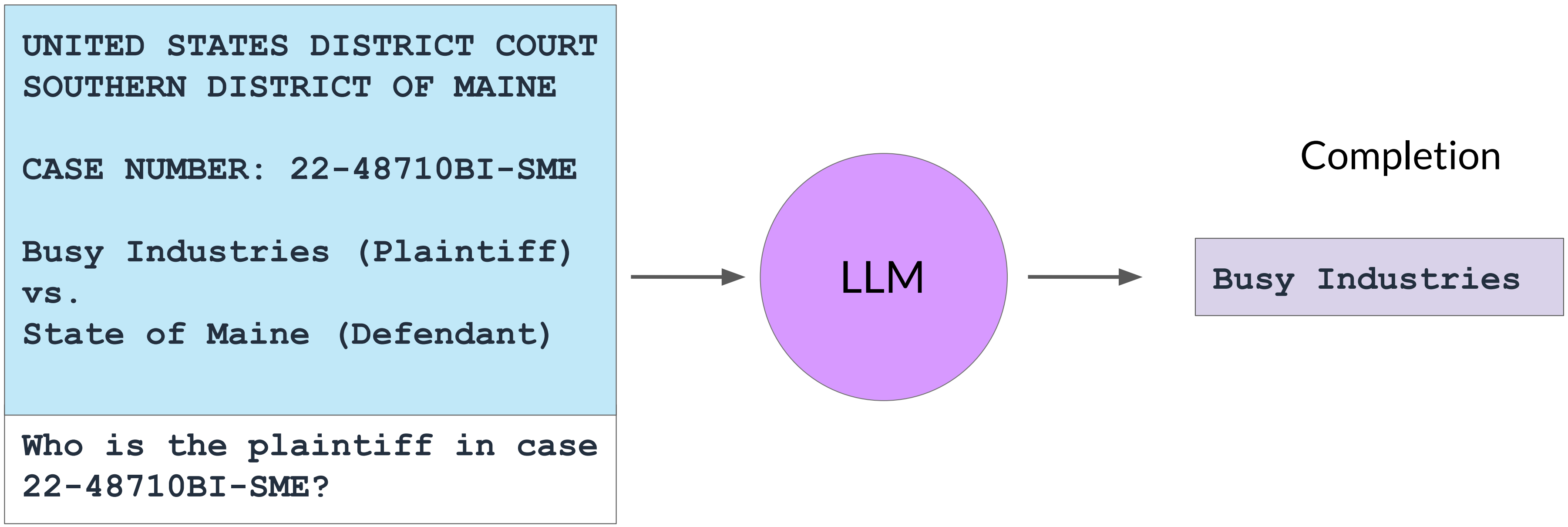


Lewis et al. 2020 "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks"

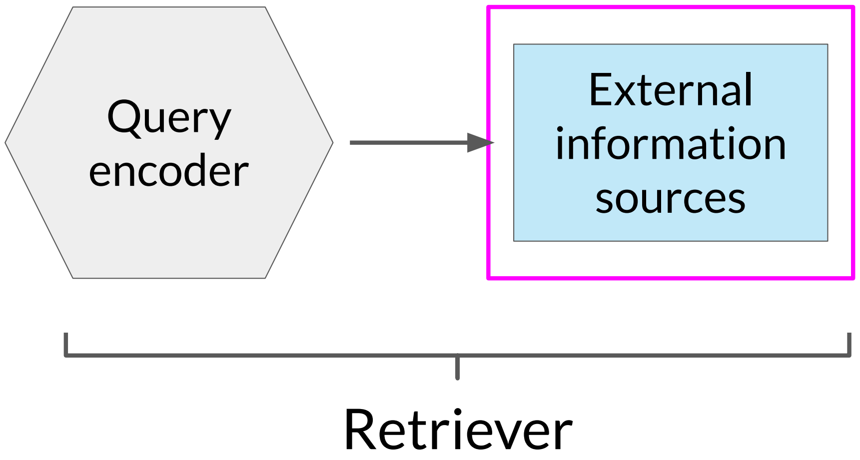
Example: Searching legal documents



Example: Searching legal documents



RAG integrates with many types of data sources



External Information Sources

- Documents
- Wikis
- Expert Systems
- Web pages
- Databases
- Vector Store

Data preparation for vector store for RAG

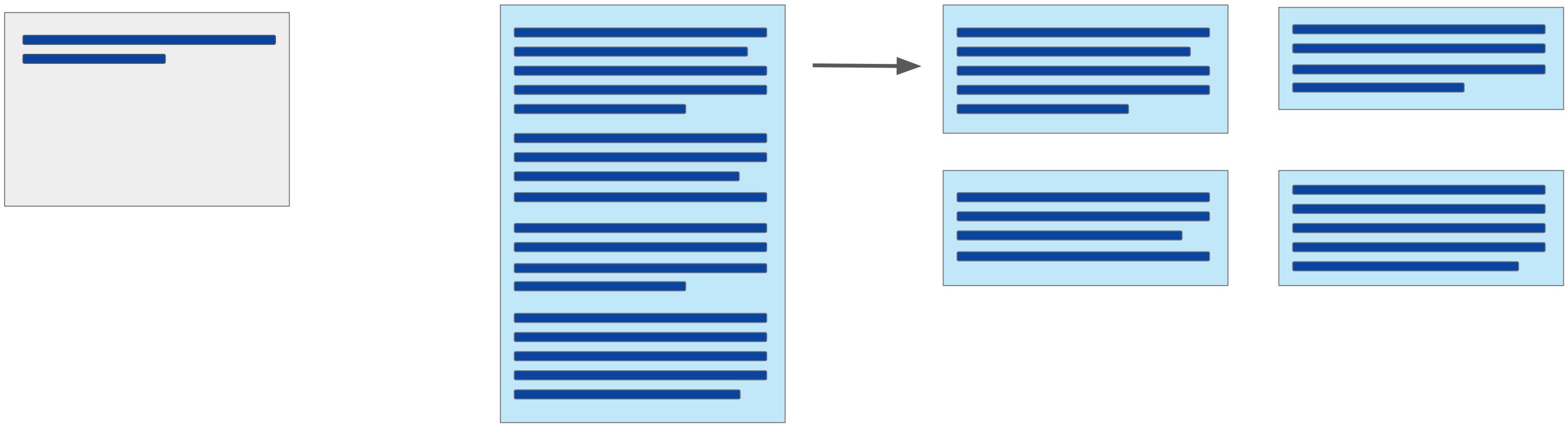
Two considerations for using external data in RAG:

1. Data must fit inside context window

Prompt context limit few 1000 tokens

Single document too large to fit in window

Split long sources into short chunks

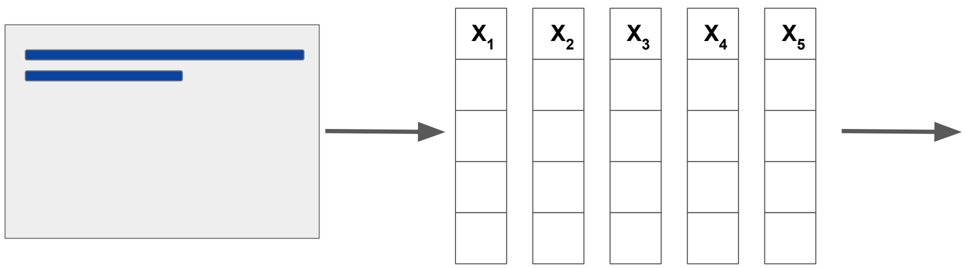


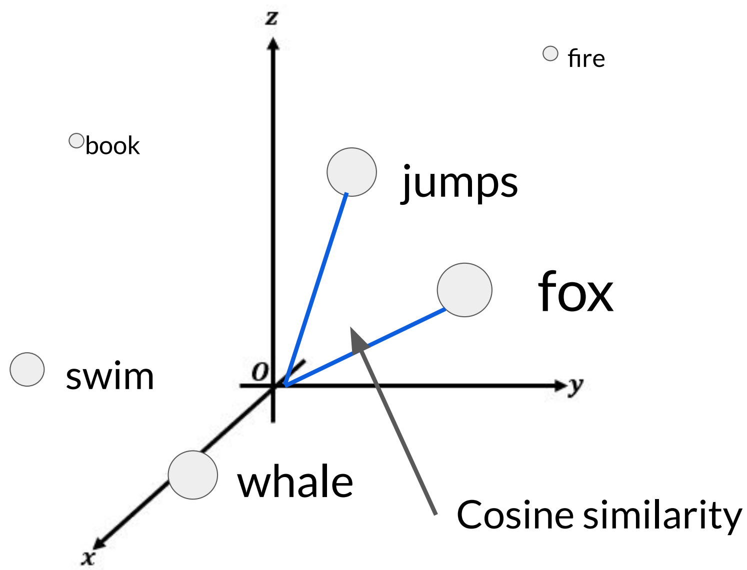
Data preparation for RAG

Two considerations for using external data in RAG:

1. Data must fit inside context window
2. Data must be in format that allows its relevance to be assessed at inference time: **Embedding vectors**

Prompt text converted to embedding vectors



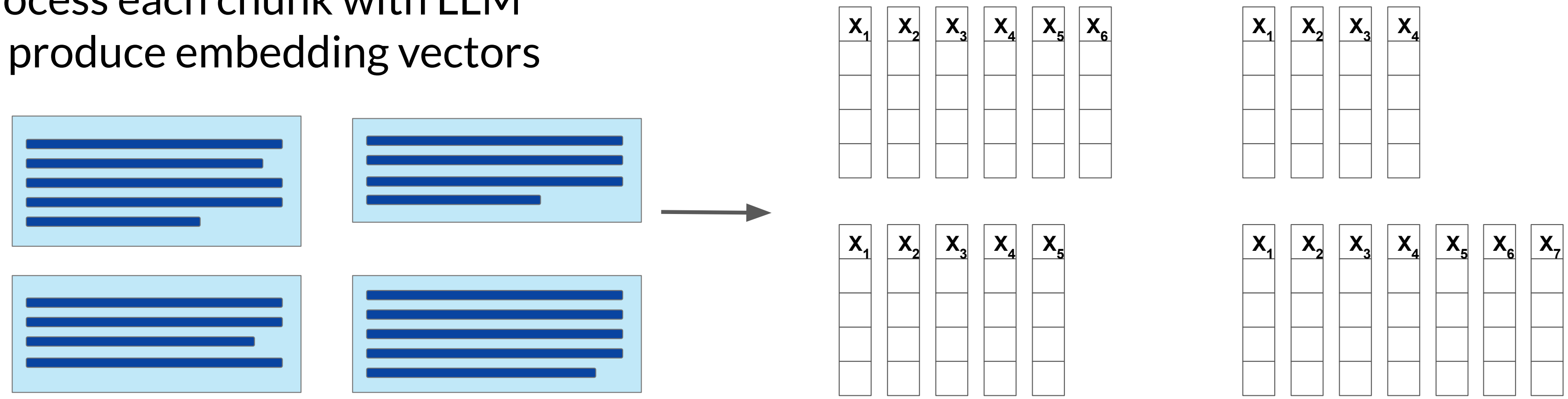


Data preparation for RAG

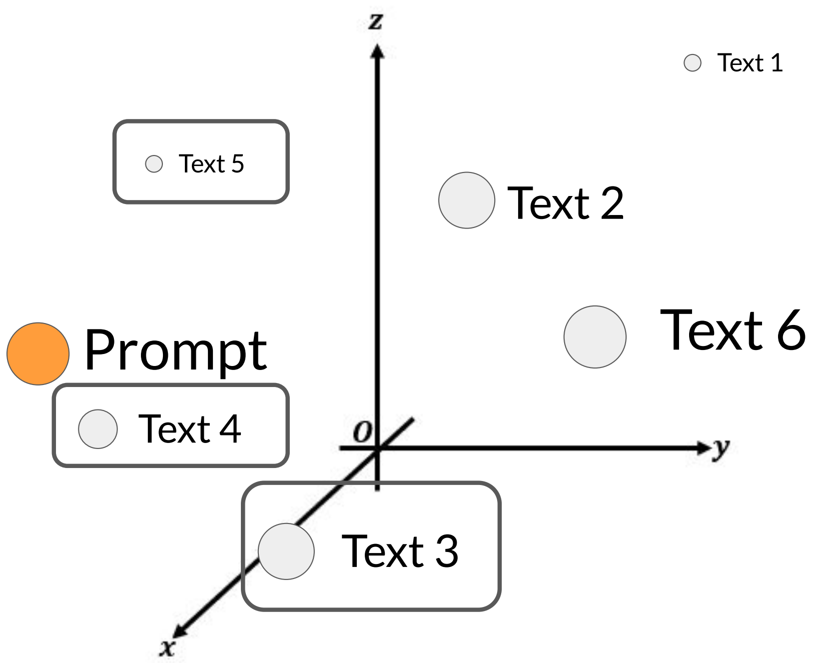
Two considerations for using external data in RAG:

1. Data must fit inside context window
2. Data must be in format that allows its relevance to be assessed at inference time: **Embedding vectors**

Process each chunk with LLM
to produce embedding vectors



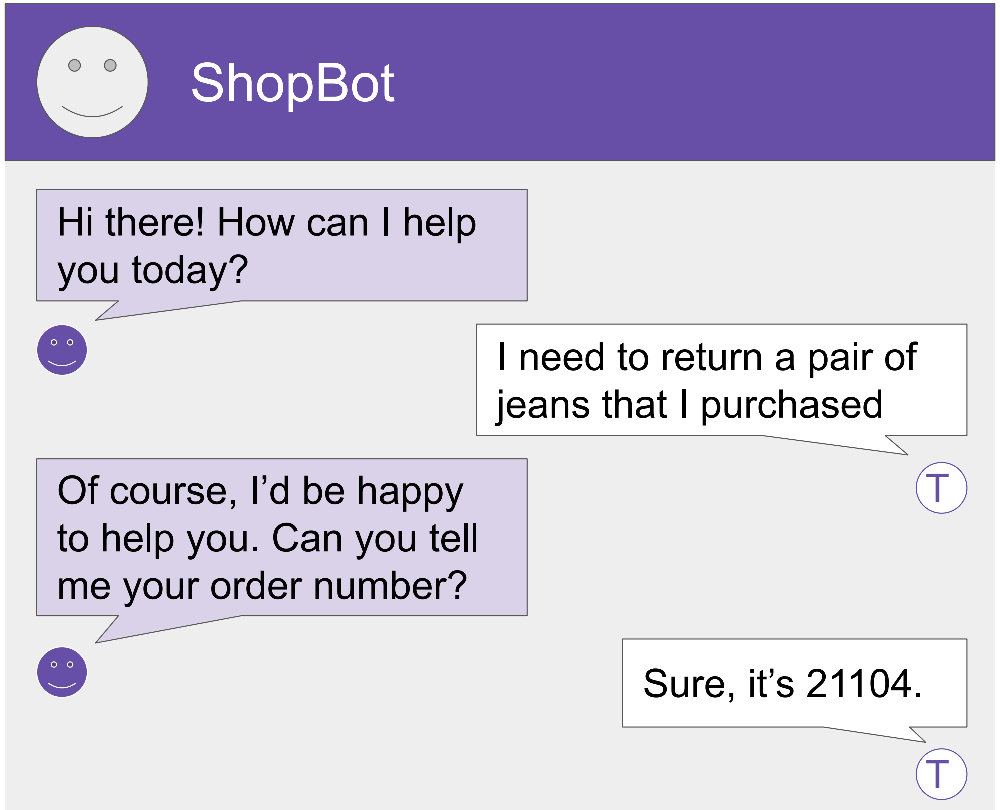
Vector database search



- Each text in vector store is identified by a key
- Enables a **citation** to be included in completion

Enabling interactions with external applications

Having an LLM initiate a clothing return



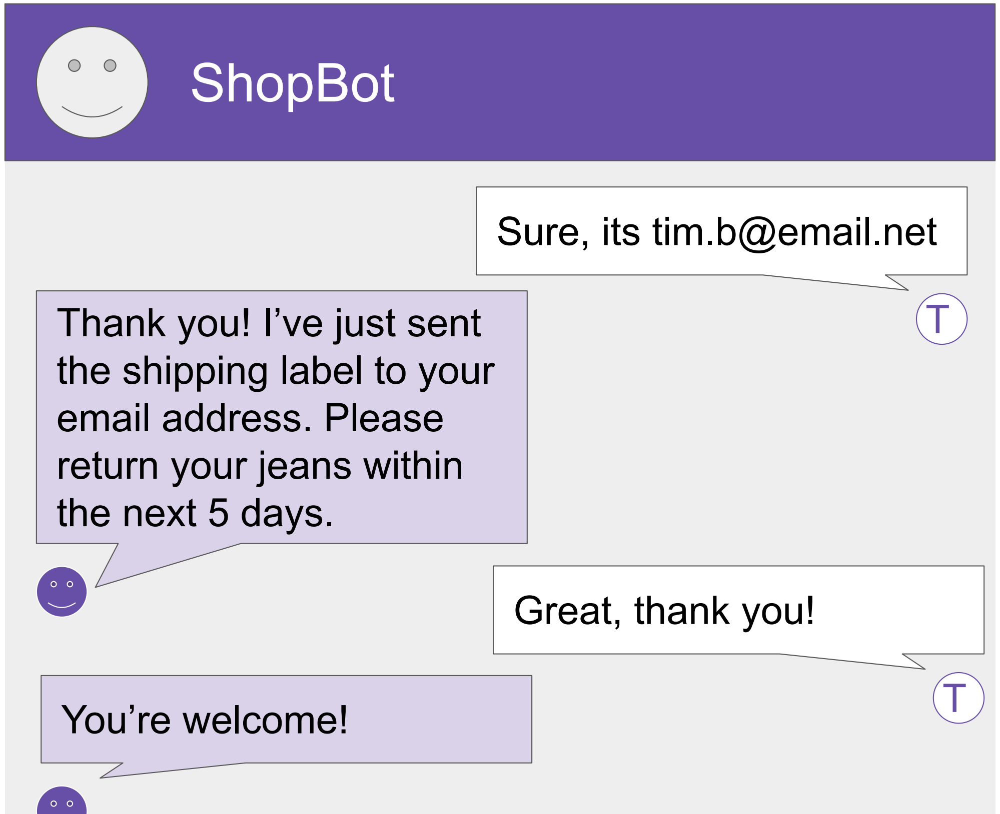
Having an LLM initiate a clothing return

Lookup with RAG

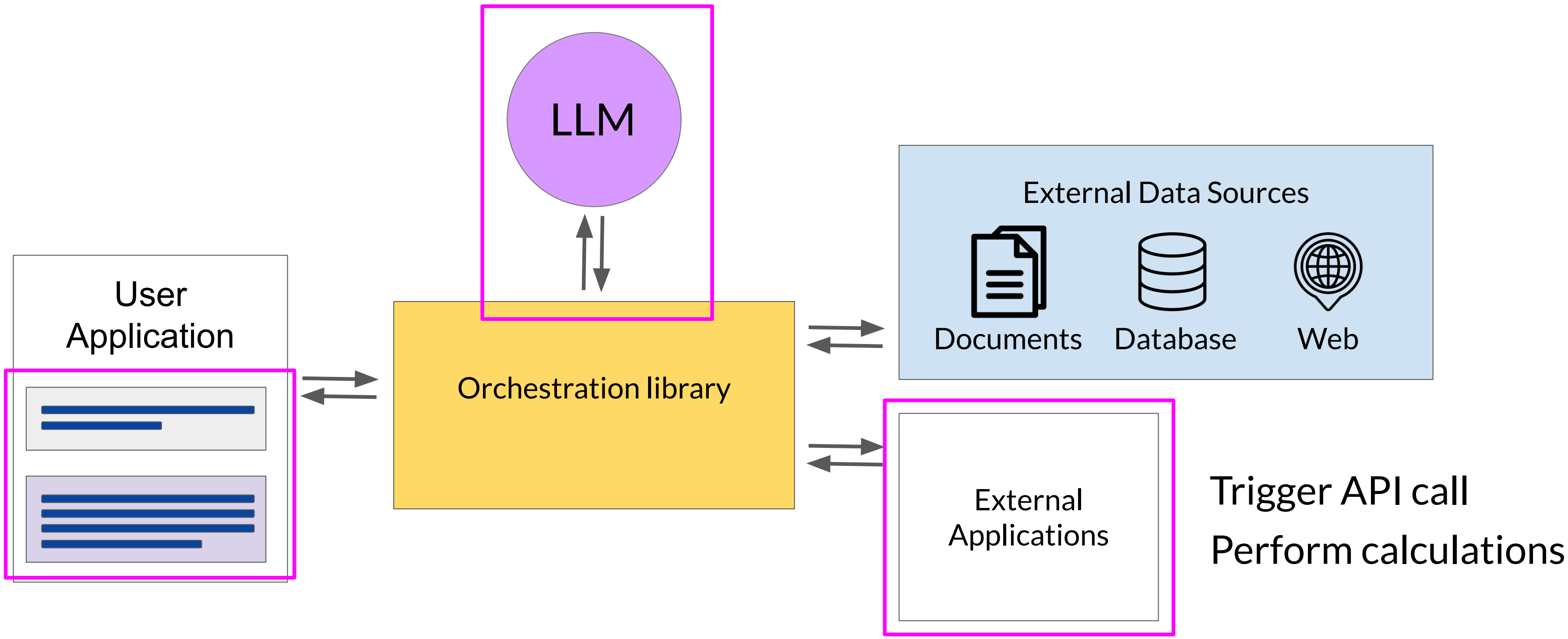
API call

Having an LLM initiate a clothing return

API call to the shipper



LLM-powered applications



Requirements for using LLMs to power applications

Plan actions

Steps to process return:

Step 1: Check order ID

Step 2: Request label

Step 3: Verify user email

Step 4: Email user label

Format outputs

SQL Query:

SELECT COUNT(*)
FROM orders
WHERE order_id = 21104

Validate actions

Collect required user information and make sure it is in the completion

User email:
tim.b@email.net

Prompt structure is important!

Helping LLMs reason and plan with Chain-of-Thought Prompting

LLMs can struggle with complex reasoning problems

Prompt

```
Q: Roger has 5 tennis balls.
He buys 2 more cans of tennis
balls. Each can has 3 tennis
balls. How many tennis balls
does he have now?

A: The answer is 11

Q: The cafeteria had 23
apples. If they used 20 to
make lunch and bought 6 more,
how many apples do they have?
```

Model

LLM

Completion

```
Q: Roger has 5 tennis balls.
He buys 2 more cans of tennis
balls. Each can has 3 tennis
balls. How many tennis balls
does he have now?

A: The answer is 11

Q: The cafeteria had 23
apples. If they used 20 to
make lunch and bought 6 more,
how many apples do they have?

A: The answer is 27.
```

Humans take a step-by-step approach to solving complex problems

Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

Start: Roger started with 5 balls.
Step 1: 2 cans of 3 tennis balls each is 6 tennis balls.
Step 2: 5 + 6 = 11
End: The answer is 11

Reasoning steps

"Chain of thought"

Chain-of-Thought Prompting can help LLMs reason

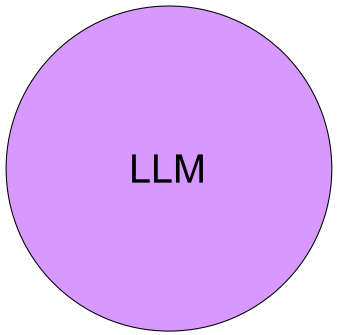
Prompt

```
Q: Roger has 5 tennis balls.
He buys 2 more cans of tennis
balls. Each can has 3 tennis
balls. How many tennis balls
does he have now?

A: Roger started with 5
balls. 2 cans of 3 tennis
balls each is 6 tennis balls.
5 + 6 = 11.The answer is 11.

Q: The cafeteria had 23
apples. If they used 20 to
make lunch and bought 6 more,
how many apples do they have?
```

Model



Completion

```
Q: Roger has 5 tennis balls.
...
...
...
how many apples do they have?

A: The cafeteria had 23
apples originally. They used
20 to make lunch. So they had
23-20 = 3. They bought 6 more
apples, so they have 3 + 6 =
9. The answer is 9. ✔
```

Source: Wei et al. 2022, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models"

Chain-of-Thought Prompting can help LLMs reason

Prompt

```
Q: Yes or no: Would a pear
sink in water?

A: The density of a pear is
about 0.6 g/cm^3, which is
less than water. Thus, a pear
would float. So the answer is
no.

Q. Yes or no: Would a gold
ring sink to the bottom of a
swimming pool?
```

Model

LLM

Completion

```
Q: Yes or no: Would a pear
sink in water?
...
...
So the answer is no.

The density of gold is 19.3
g/cm^3, which is much greater
than the density of water.
Therefore, a gold ring would
sink to the bottom of a
swimming pool. So the answer
is yes.
```

Source: Wei et al. 2022, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”

Program-aided Language Models

LLMs can struggle with mathematics

Prompt

```
What is 40366 / 439?
```

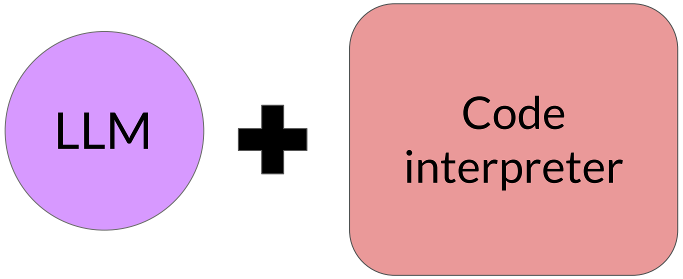
Model

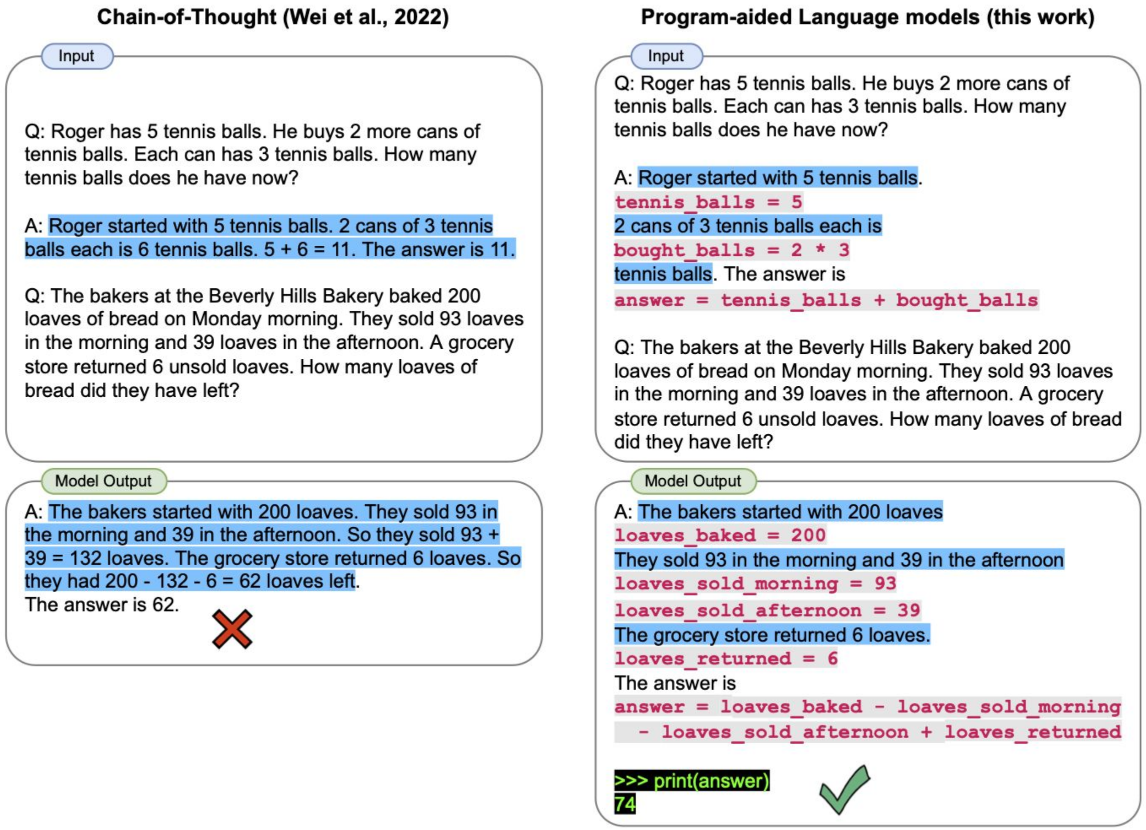
LLM

Completion

```
What is 40366 / 439?
92.549
```

Program-aided language (PAL) models





Source: Gao et al. 2022, “PAL: Program-aided Language Models”

PAL example

Prompt with one-shot example

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

Answer:

```
# Roger started with 5 tennis balls
tennis_balls = 5
# 2 cans of tennis balls each is
bought_balls = 2 * 3
# tennis balls. The answer is
answer = tennis_balls + bought_balls
```

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves did they have left?

PAL example

Prompt with one-shot example

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

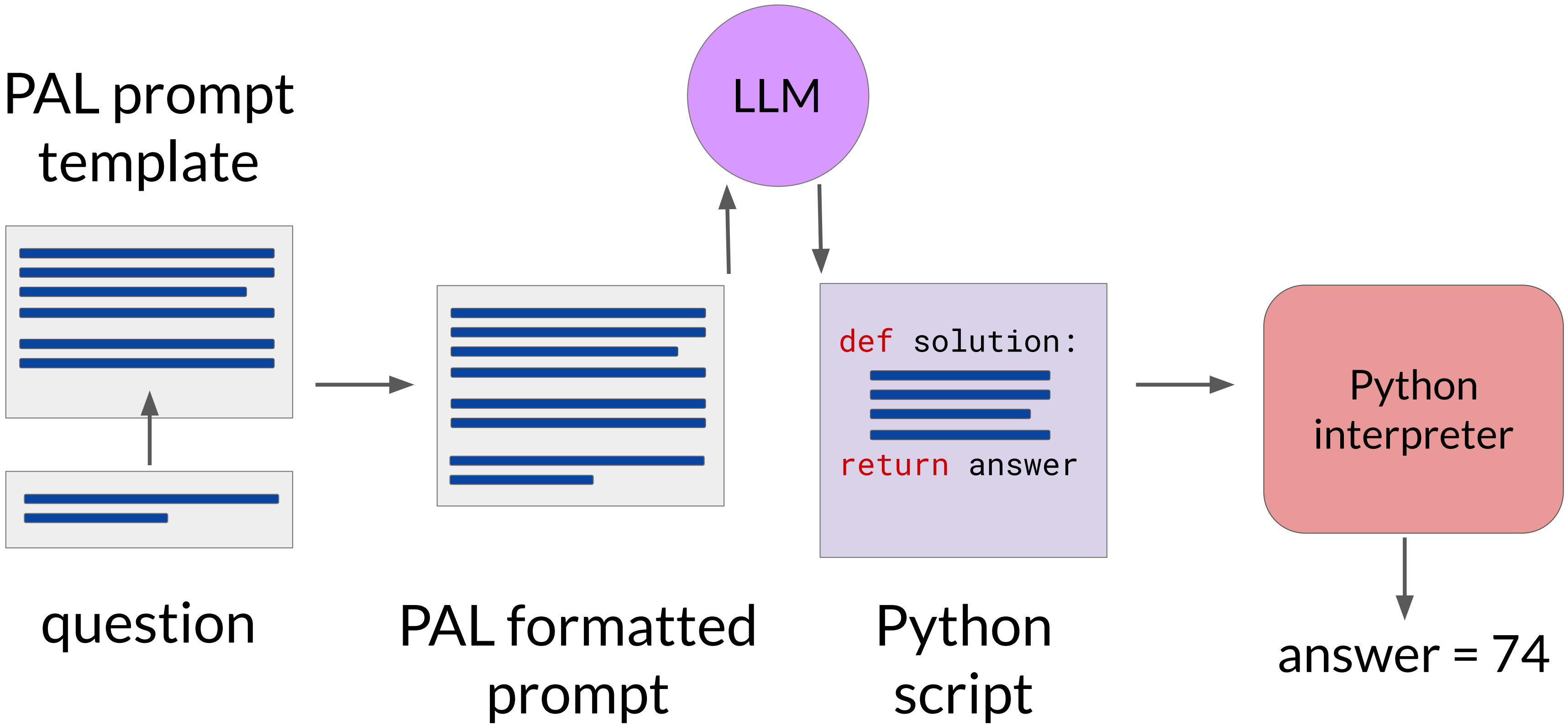
Answer:
```
# Roger started with 5 tennis balls
tennis_balls = 5
# 2 cans of tennis balls each is
bought_balls = 2 * 3
# tennis balls. The answer is
answer = tennis_balls + bought_balls
```

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves did they have left?

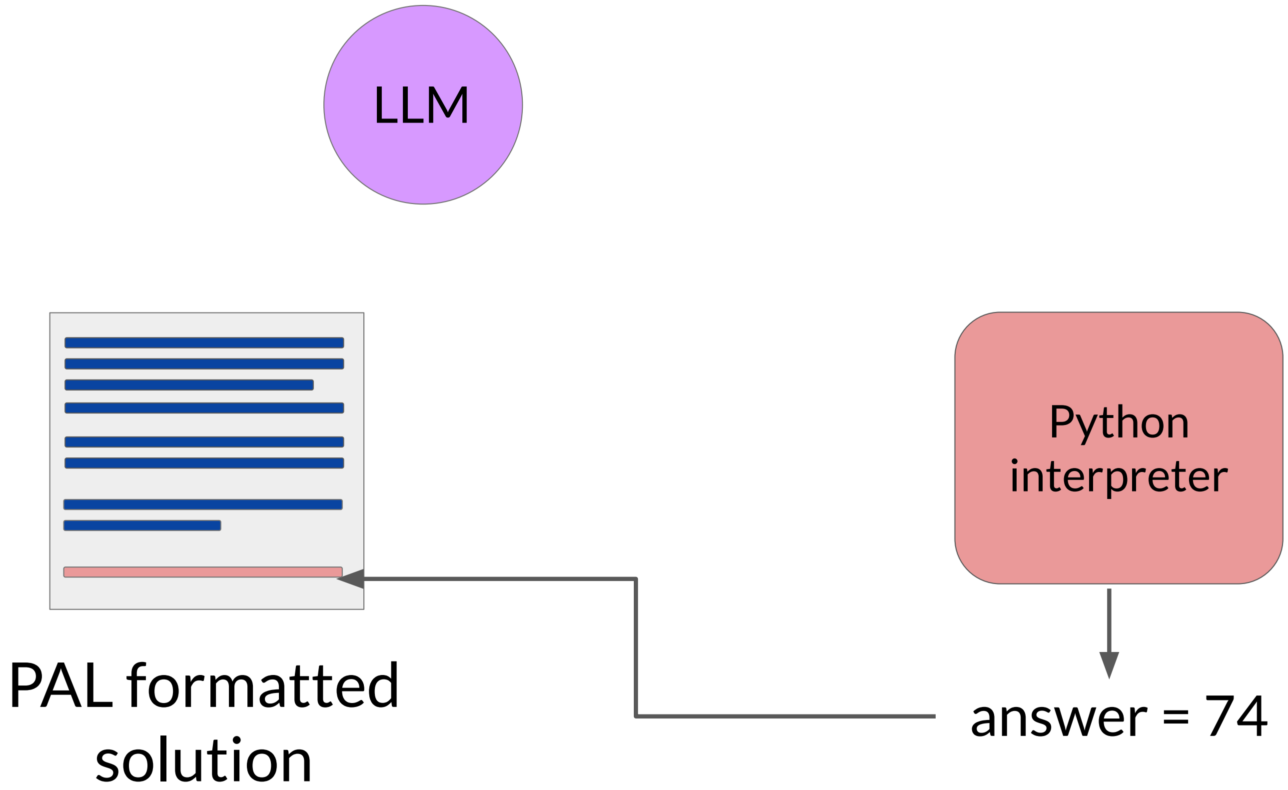
Completion, CoT reasoning (blue) , and PAL execution (pink)

Answer:
```
# The bakers started with 200 loaves
loaves_baked = 200
# They sold 93 in the morning and 39 in the afternoon
loaves_sold_morning = 93
loaves_sold_afternoon = 39
# The grocery store returned 6 loaves.
loaves_returned = 6
# The answer is
answer = loaves_baked
  - loaves_sold_morning
  - loaves_sold_afternoon
  + loaves_returned
```

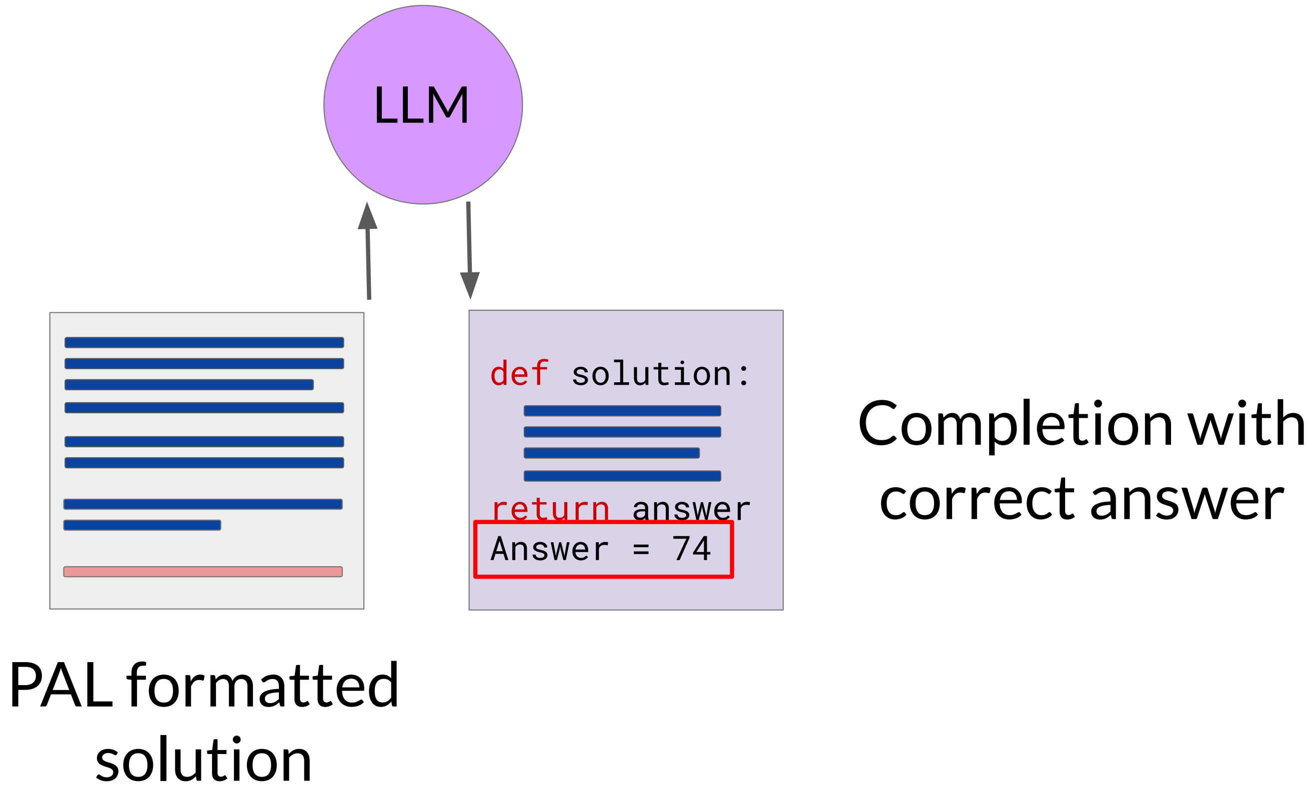
Program-aided language (PAL) models



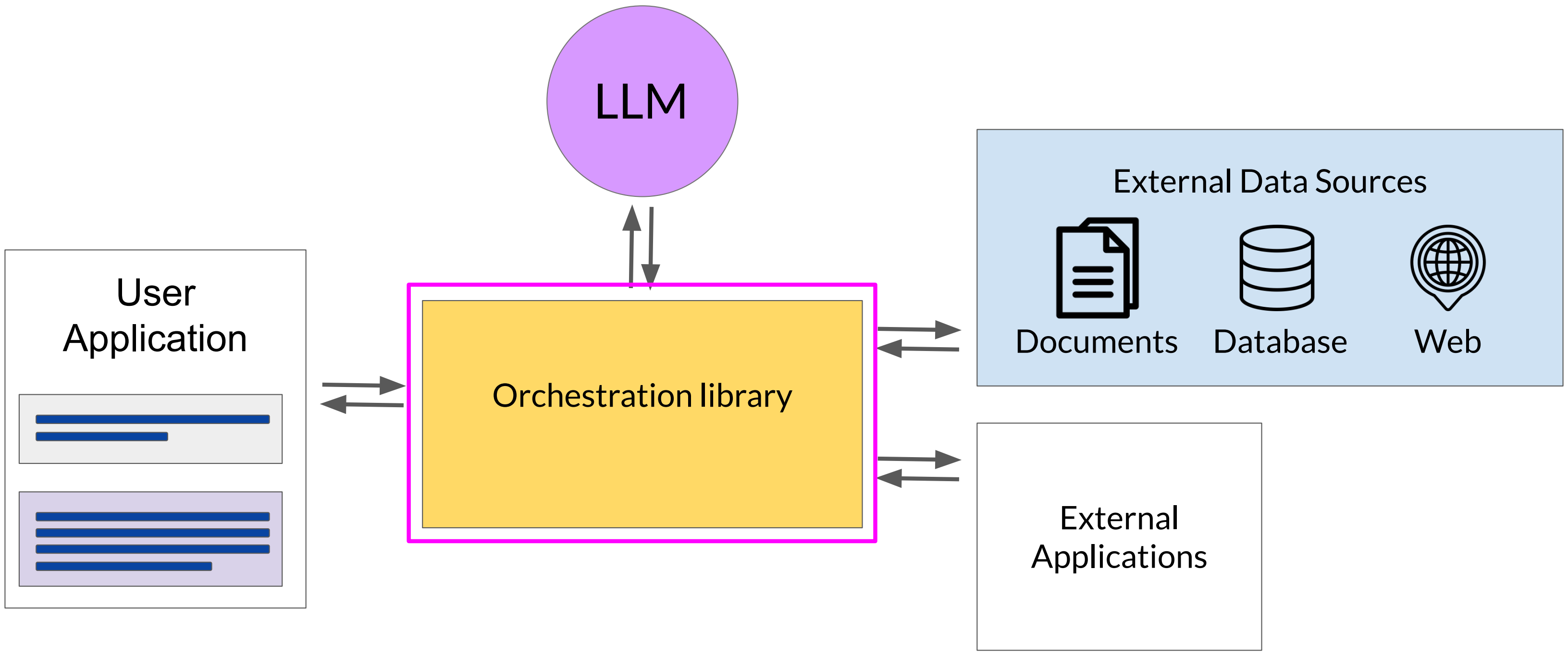
Program-aided language (PAL) models



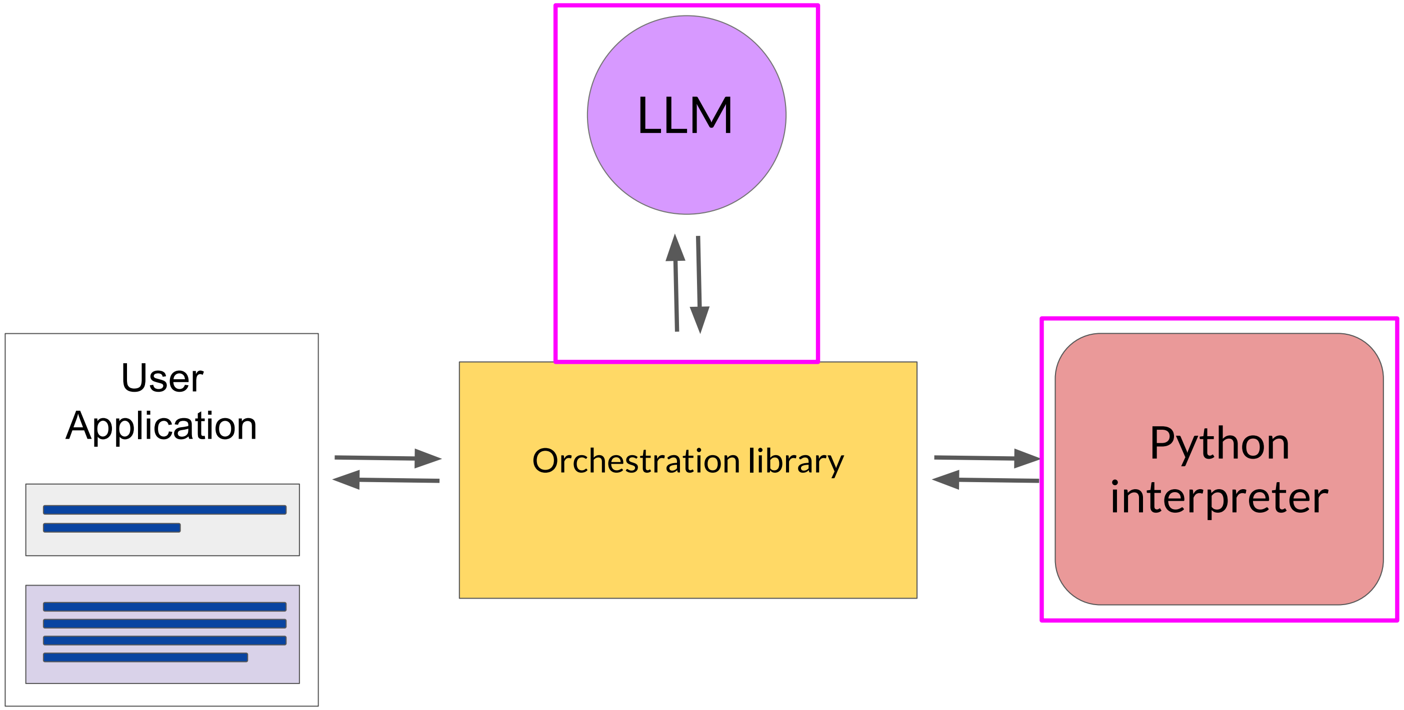
Program-aided language (PAL) models



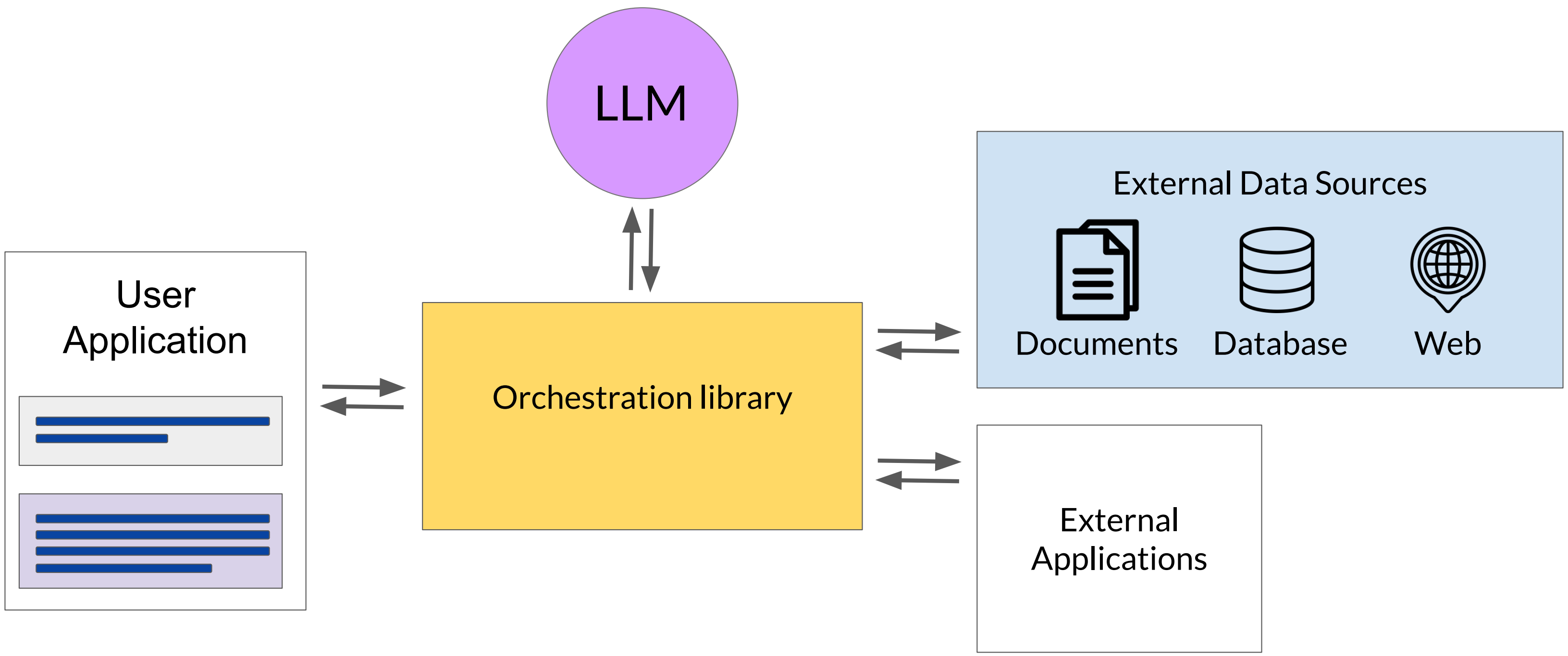
LLM-powered applications



PAL architecture

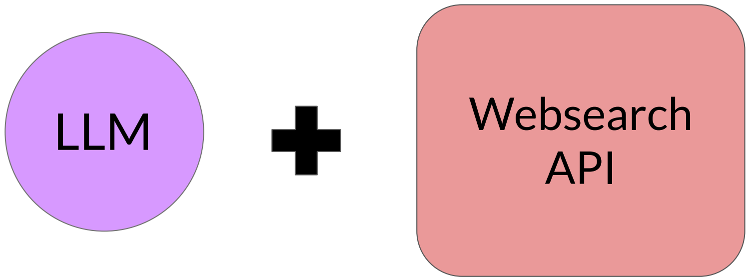


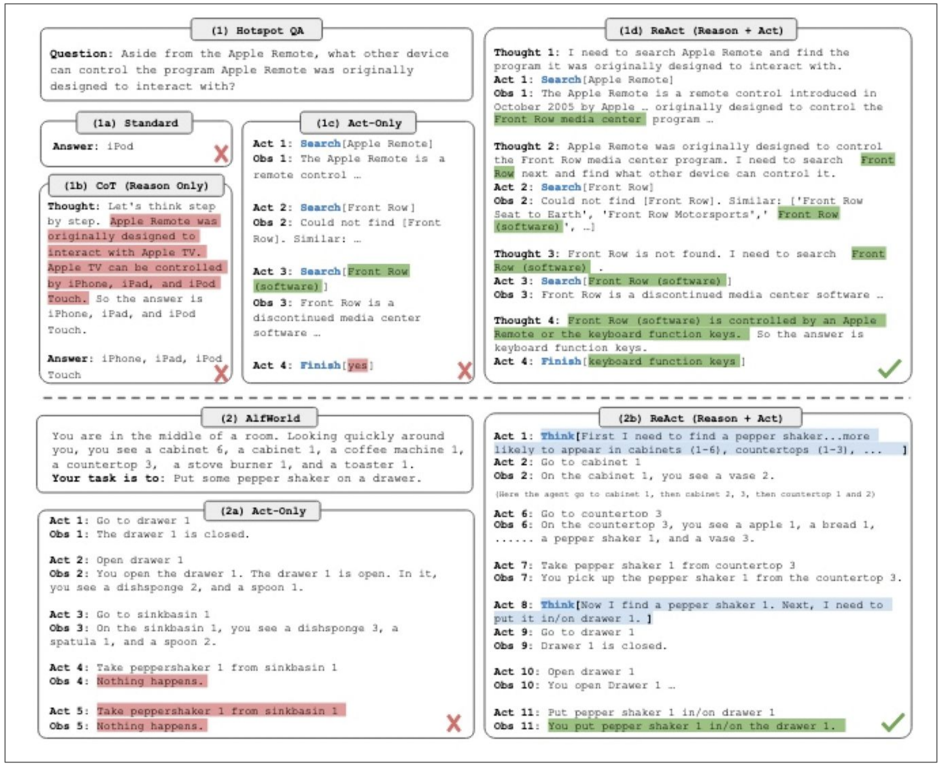
LLM-powered applications



ReAct: Combining reasoning and action in LLMs

ReAct: Synergizing Reasoning and Action in LLMs

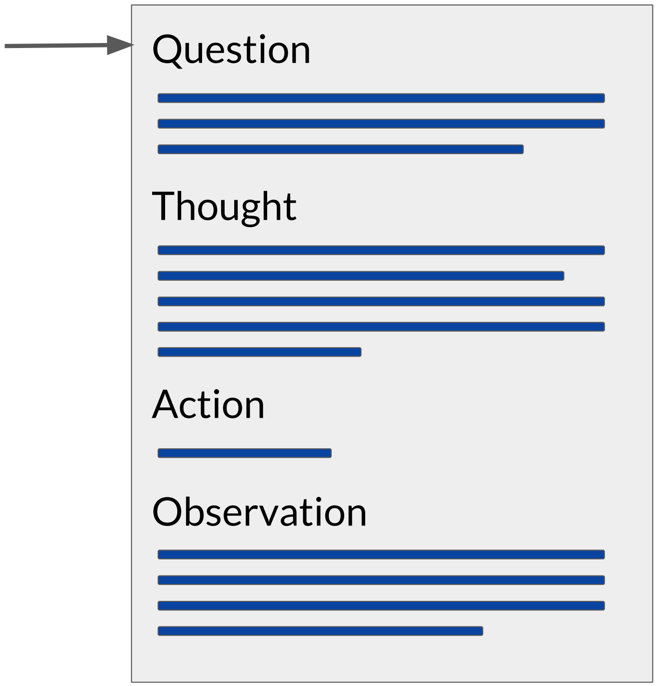




HotPot QA: multi-step question answering
Fever: Fact verification

Source: Yao et al. 2022, "ReAct: Synergizing Reasoning and Acting in Language Models"

ReAct: Synergizing Reasoning and Action in LLMs

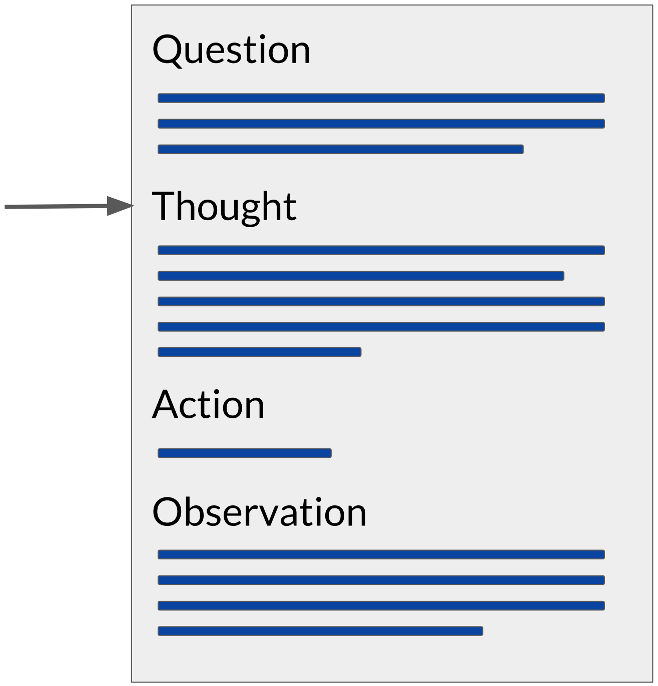


Question: Problem that requires advanced reasoning and multiple steps to solve.

E.g.
“Which magazine was started first, *Arthur’s Magazine* or *First for Women*?”

Source: Yao et al. 2022, “ReAct: Synergizing Reasoning and Acting in Language Models”

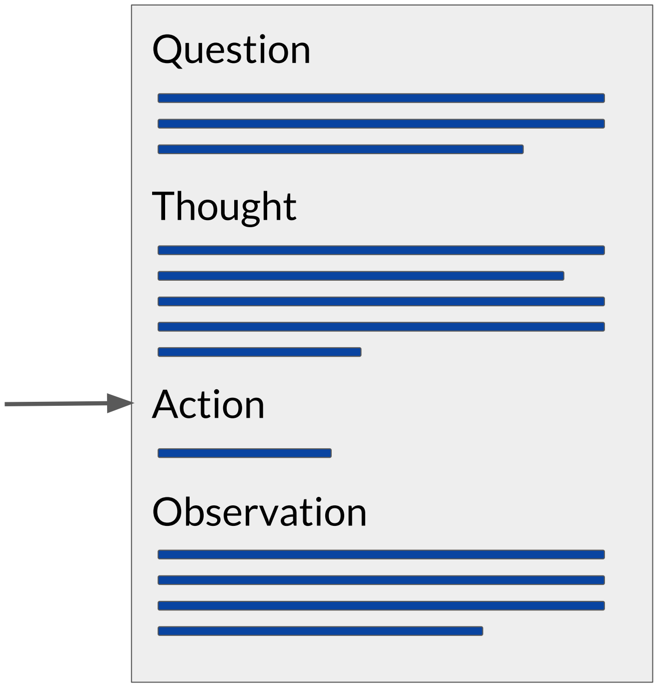
ReAct: Synergizing Reasoning and Action in LLMs



Thought: A reasoning step that identifies how the model will tackle the problem and identify an action to take.

"I need to search Arthur's Magazine and First for Women, and find which one was started first."

ReAct: Synergizing Reasoning and Action in LLMs



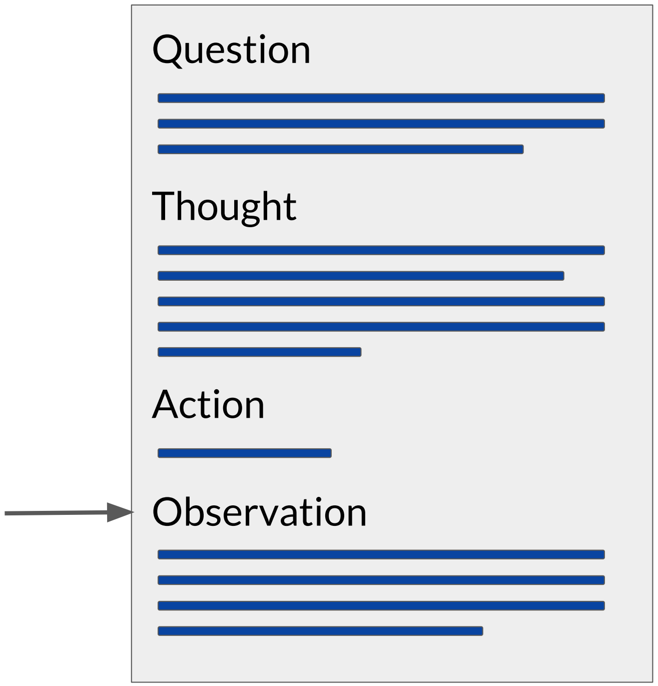
Action: An external task that the model can carry out from an allowed set of actions.

E.g.
`search[entity]`
`lookup[string]`
`finish[answer]`

Which one to choose is determined by the information in the preceding thought.

`search[Arthur's Magazine]`

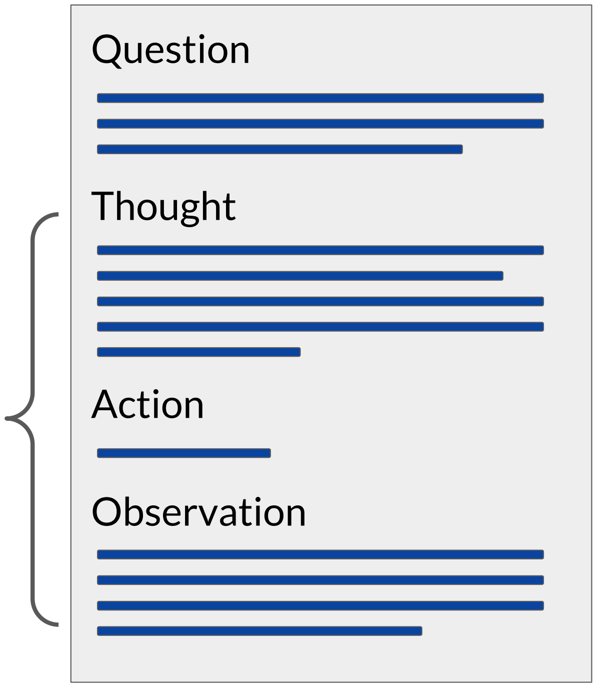
ReAct: Synergizing Reasoning and Action in LLMs



Observation: the result of carrying out the action

E.g.
"Arthur's Magazine (1844-1846) was an American literary periodical published in Philadelphia in the 19th century."

ReAct: Synergizing Reasoning and Action in LLMs

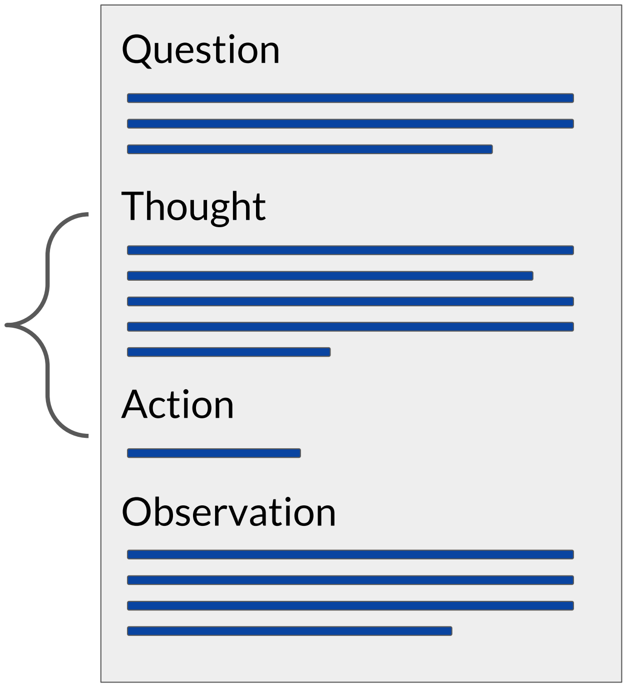


Thought 2:
"Arthur's magazine was started in 1844. I need to search First for Women next."

Action 2:
`search[First for Women]`

Observation 2:
"First for Women is a woman's magazine published by Bauer Media Group in the USA.[1] The magazine was started in 1989."

ReAct: Synergizing Reasoning and Action in LLMs



Thought 3:
"First for Women was started in 1989. 1844 (Arthur's Magazine) < 1989 (First for Women), so Arthur's Magazine as started first"

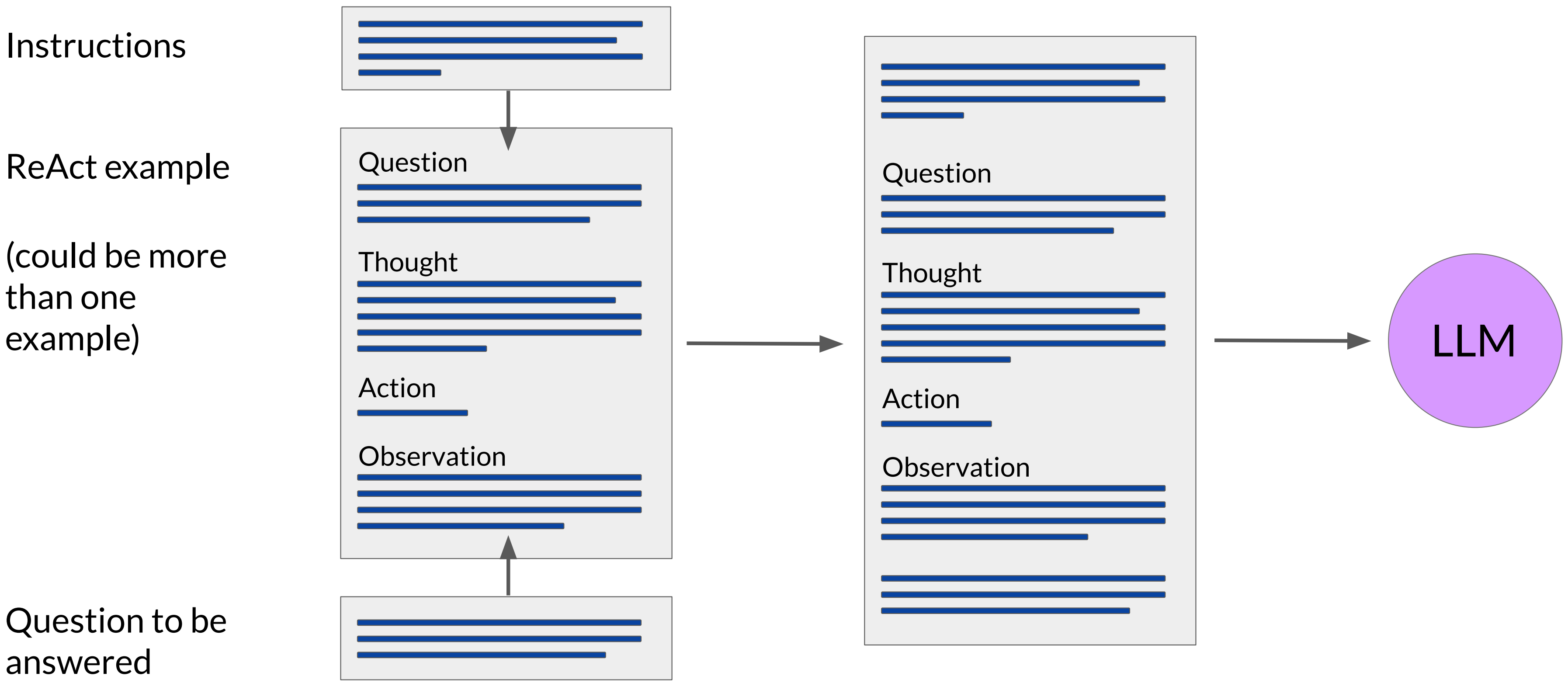
Action 2:
`finish[Arthur's Magazine]`

ReAct instructions define the action space

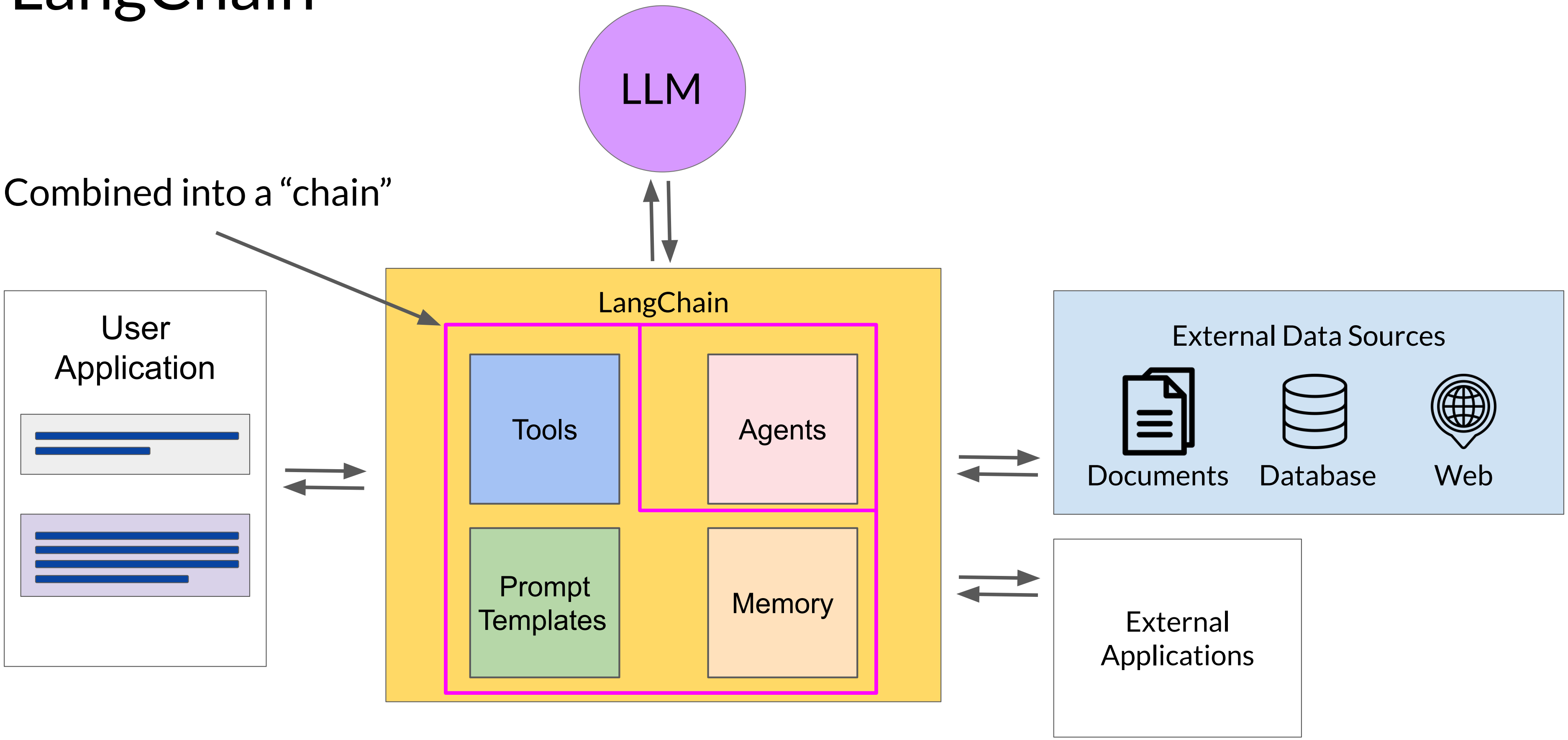
```
Solve a question answering task with interleaving Thought, Action,
Observation steps.

Thought can reason about the current situation, and Action can be
three types:
(1) Search[entity], which searches the exact entity on Wikipedia and
returns the first paragraph if it exists. If not, it will return
some similar entities to search.
(2) Lookup[keyword], which returns the next sentence containing
keyword in the current passage.
(3) Finish[answer], which returns the answer and finishes the task.
Here are some examples.
```

Building up the ReAct prompt

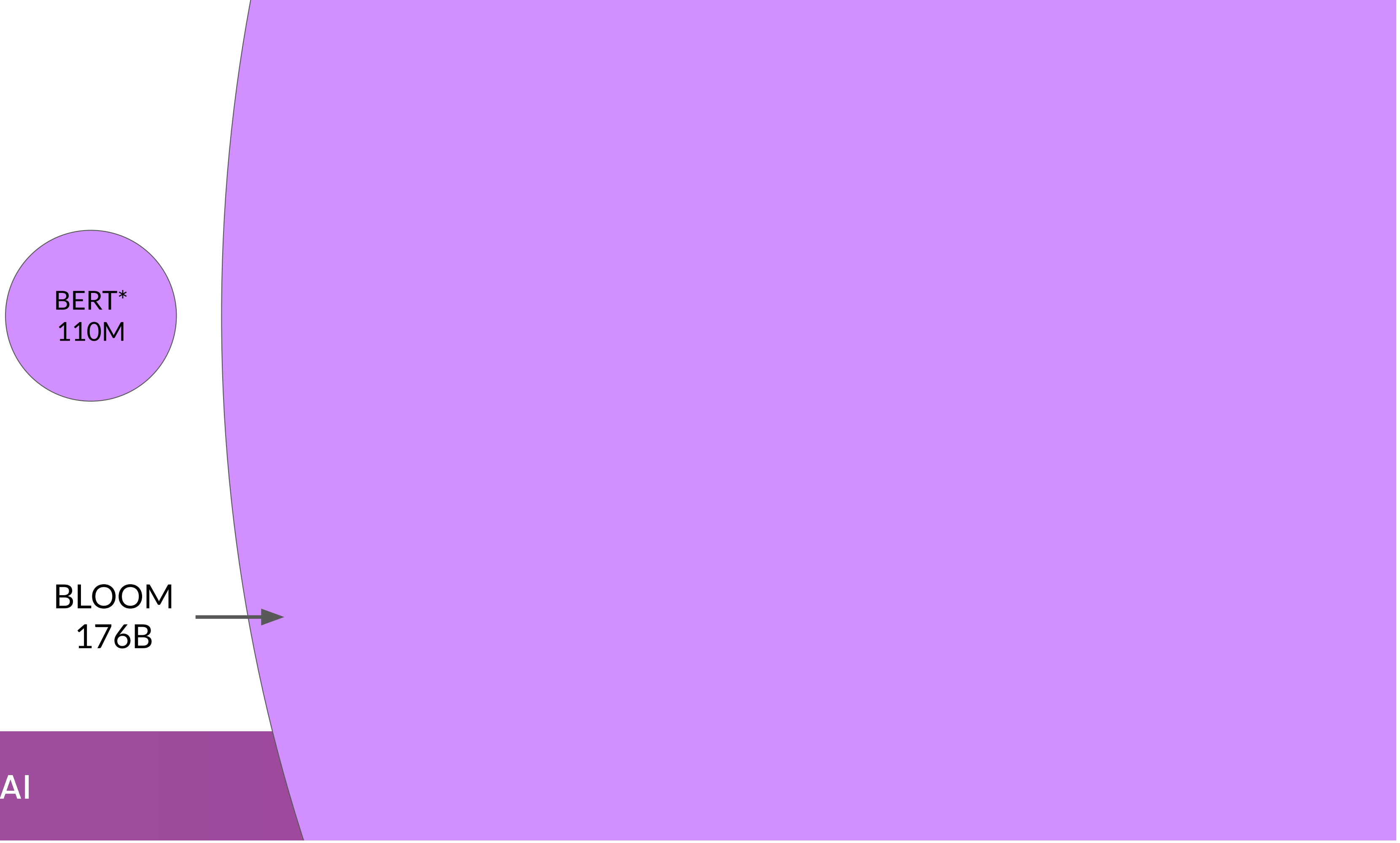


LangChain



LLM
Combined into a "chain"
User
Application
LangChain
Tools
Agents
Prompt
Templates
Memory
External Data Sources
Documents
Database
Web
External
Applications

The significance of scale: application building



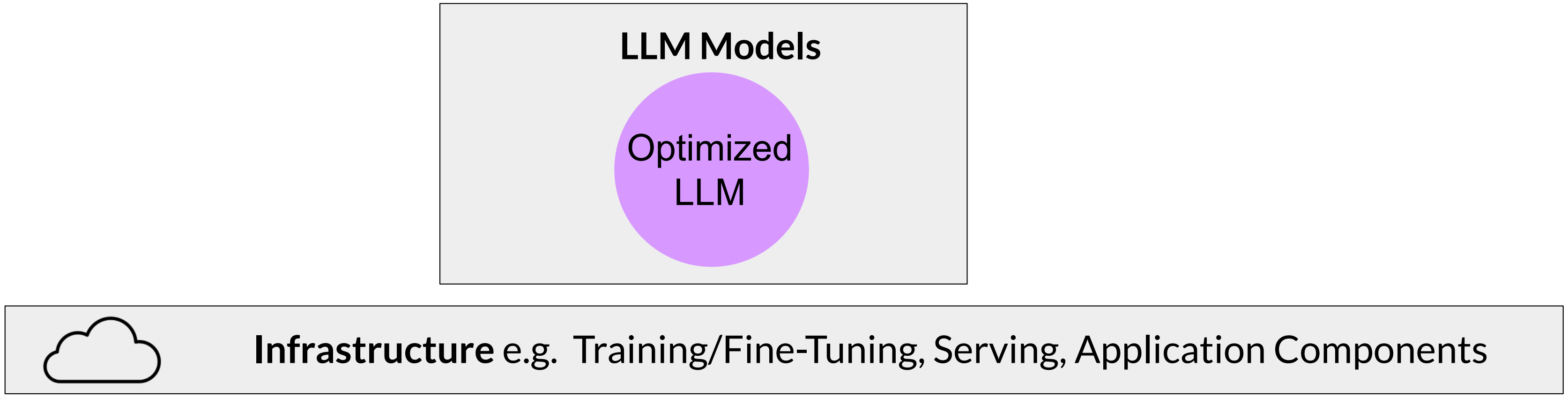
*Bert-base

LLM powered application architectures

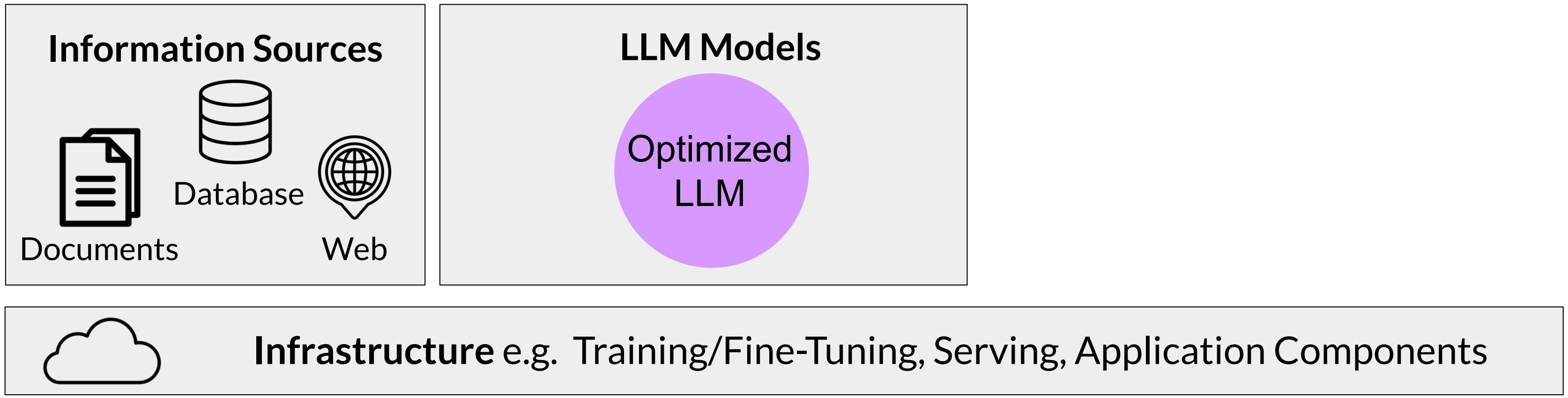
Building generative applications

Infrastructure e.g. Training/Fine-Tuning, Serving, Application Components

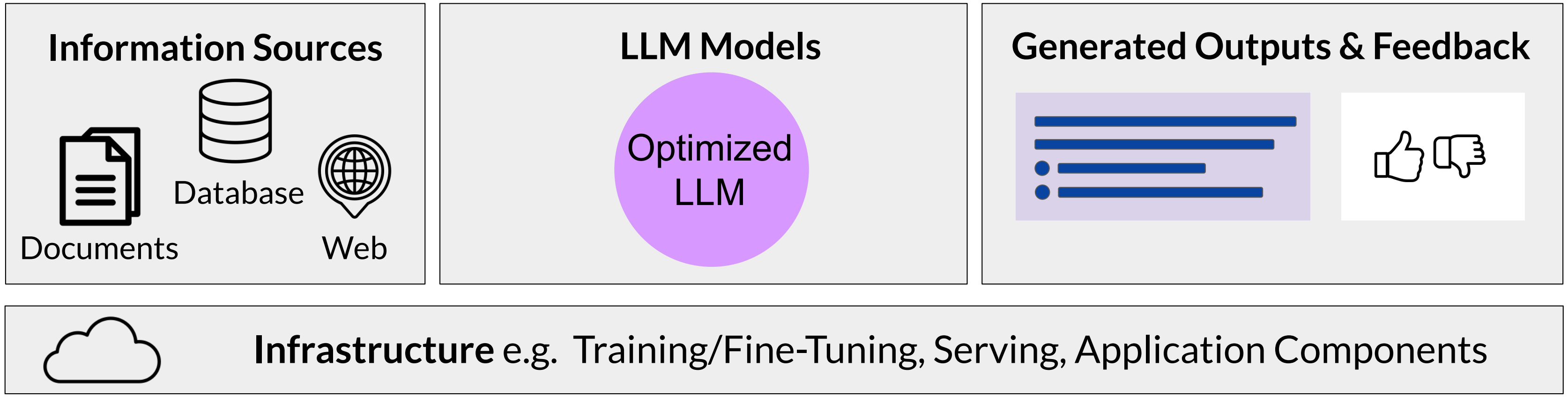
Building generative applications



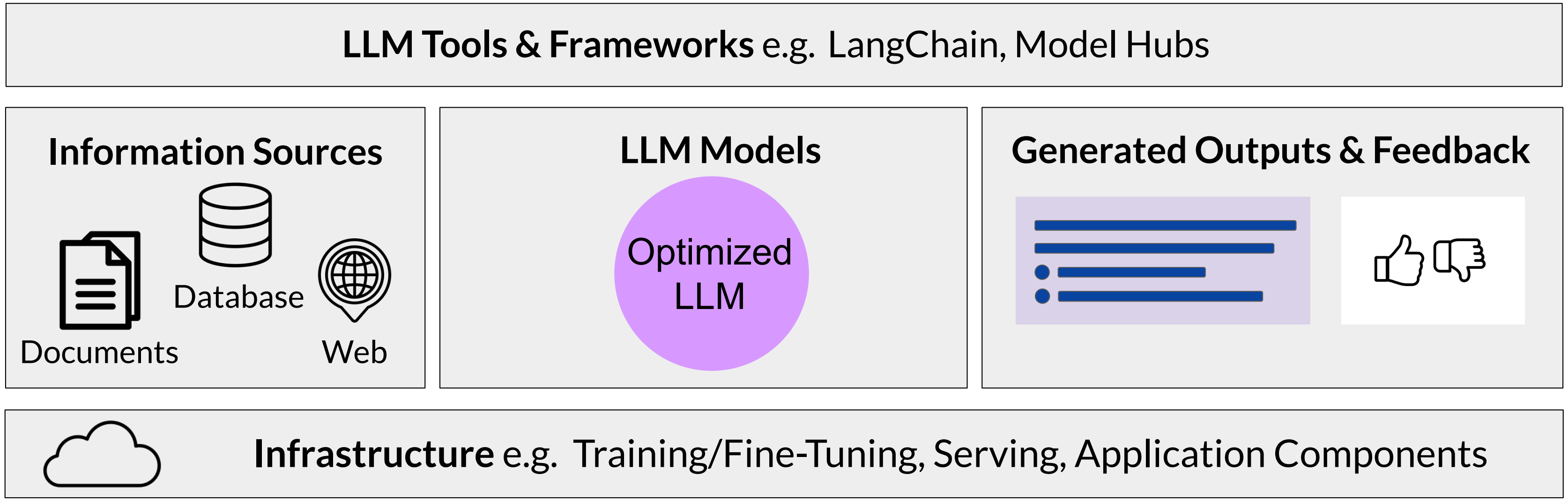
Building generative applications



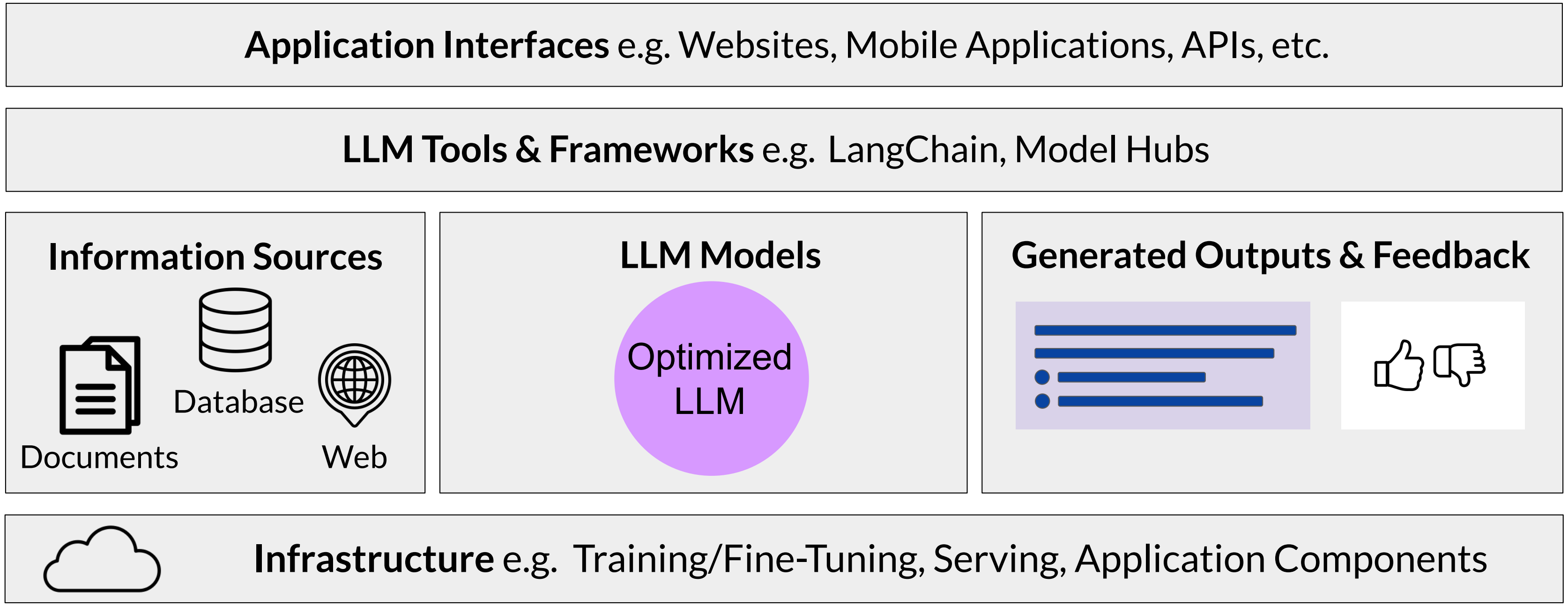
Building generative applications



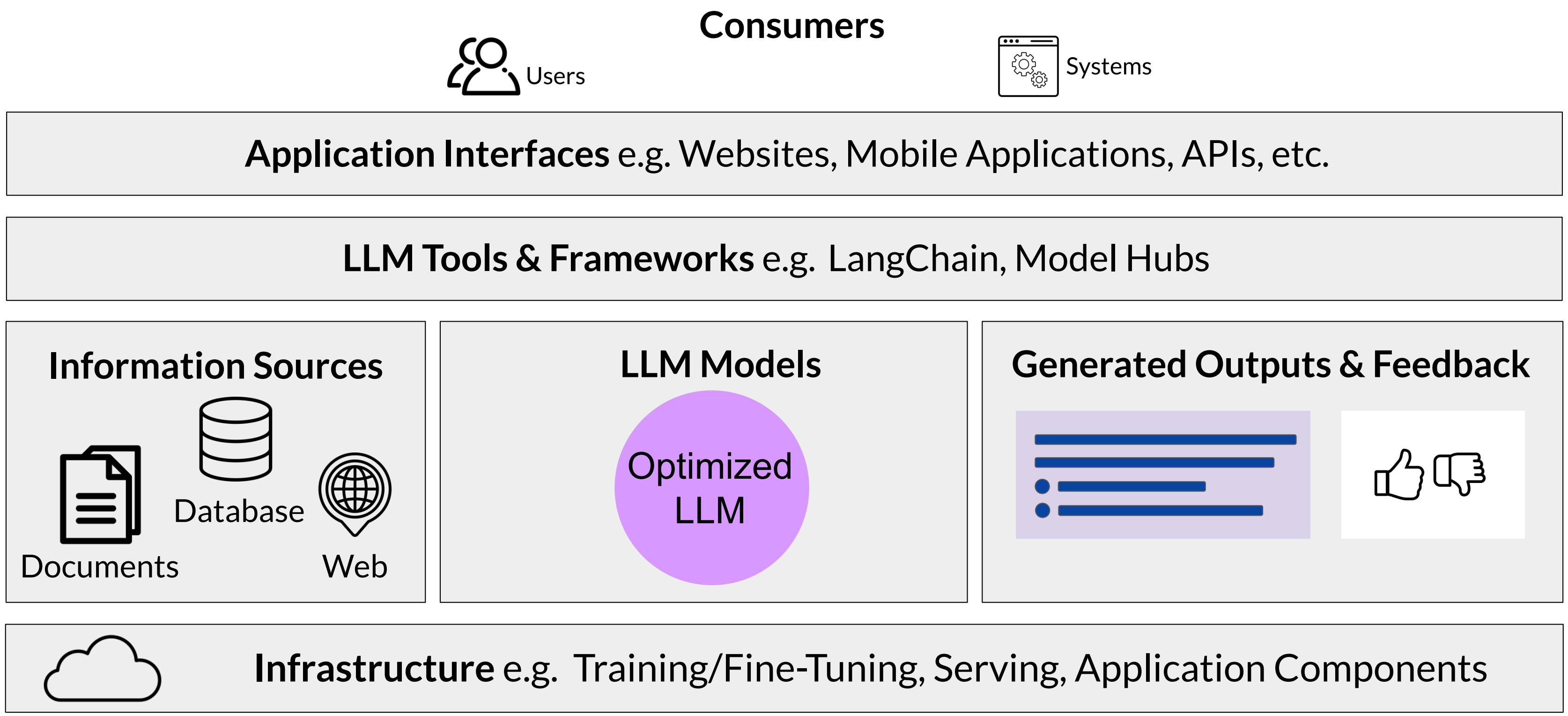
Building generative applications



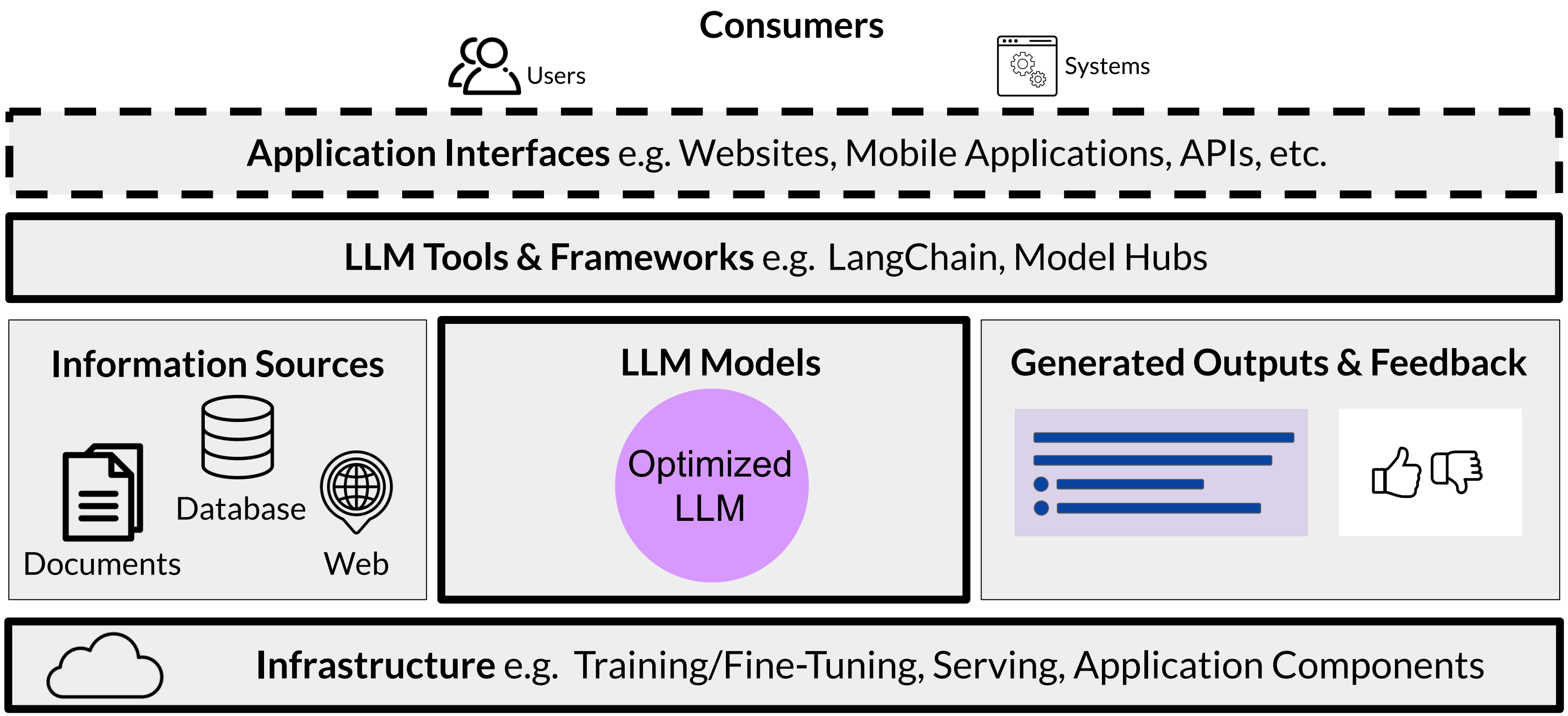
Building generative applications



Building generative applications



Building generative applications



Conclusion, Responsible AI, and on-going research

Special challenges of responsible generative AI

- Toxicity
- Hallucinations
- Intellectual Property

Toxicity

LLM returns responses that can be potentially harmful or discriminatory towards protected groups or protected attributes

How to mitigate?

- Careful curation of training data
- Train guardrail models to filter out unwanted content
- Diverse group of human annotators

Hallucinations

LLM generates factually incorrect content

How to mitigate?

- Educate users about how generative AI works
- Add disclaimers
- Augment LLMs with independent, verified citation databases
- Define intended/unintended use cases

Intellectual Property

Ensure people aren't plagiarizing, make sure there aren't any copyright issues

How to mitigate?

- Mix of technology, policy, and legal mechanisms
- Machine "unlearning"
- Filtering and blocking approaches

Responsibly build and use generative AI models

- Define use cases: the more specific/narrow, the better
- Assess risks for each use case
- Evaluate performance for each use case
- Iterate over entire AI lifecycle

On-going research

- Responsible AI
- Scale models and predict performance
- More efficiencies across model development lifecycle
- Increased and emergent LLM capabilities