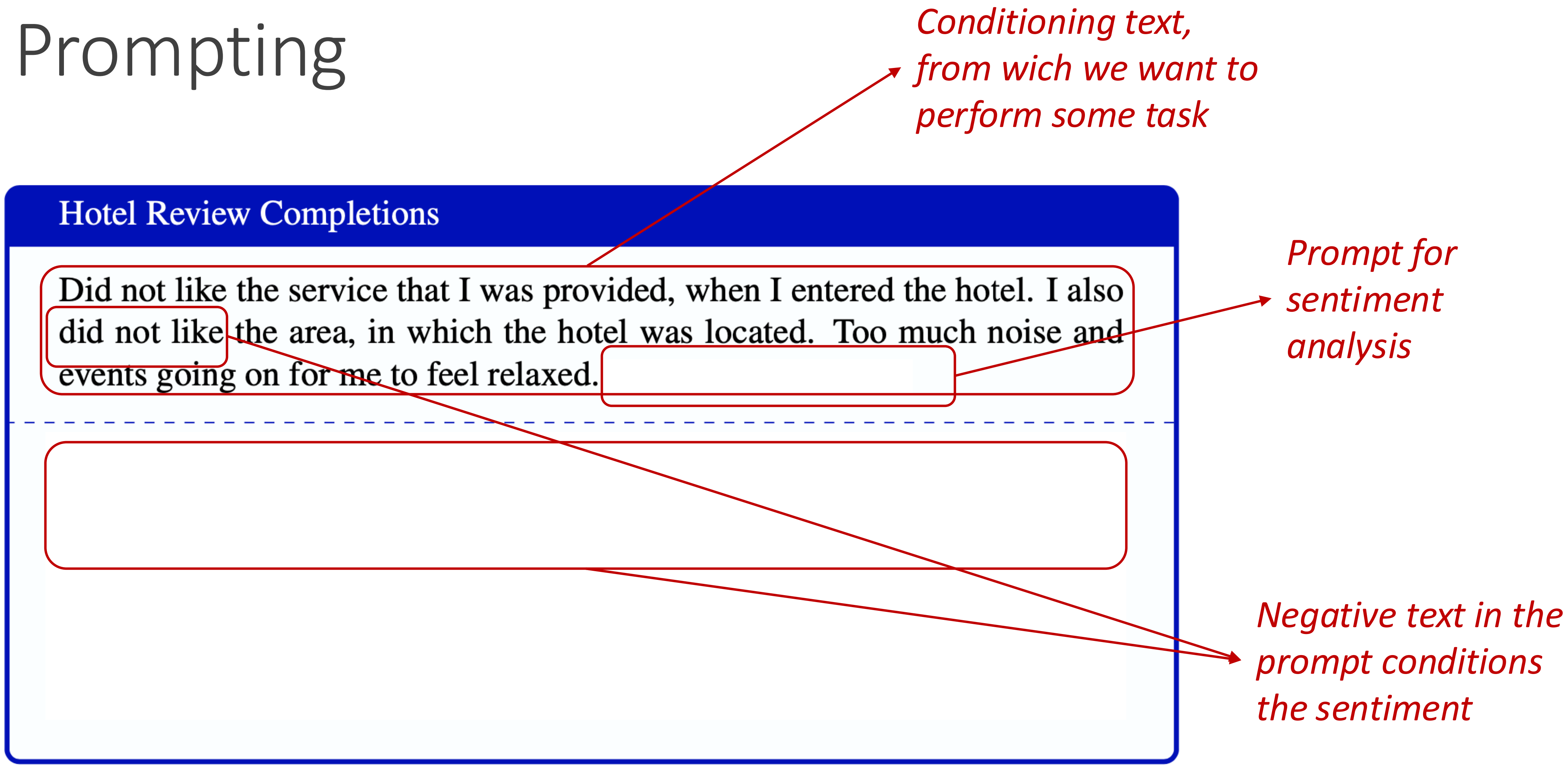
Model Alignment, Prompting, and In-Context Learning

Prompting

Prompting

- *Prompt*: text string issued to a language model to get the model to do something useful
  - The user's prompt string is passed to the LM, which iteratively generates tokens conditioned on the prompt
  - The prompt creates a context that guides LLMs to generate useful outputs to achieve some user goal
- *Prompt Engineering*: the process of finding effective prompts for a task

Prompting


Conditioning text, from wich we want to perform some task
Hotel Review Completions
Did not like the service that I was provided, when I entered the hotel. I also did not like the area, in which the hotel was located. Too much noise and events going on for me to feel relaxed.
Prompt for sentiment analysis
Negative text in the prompt conditions the sentiment

Prompting

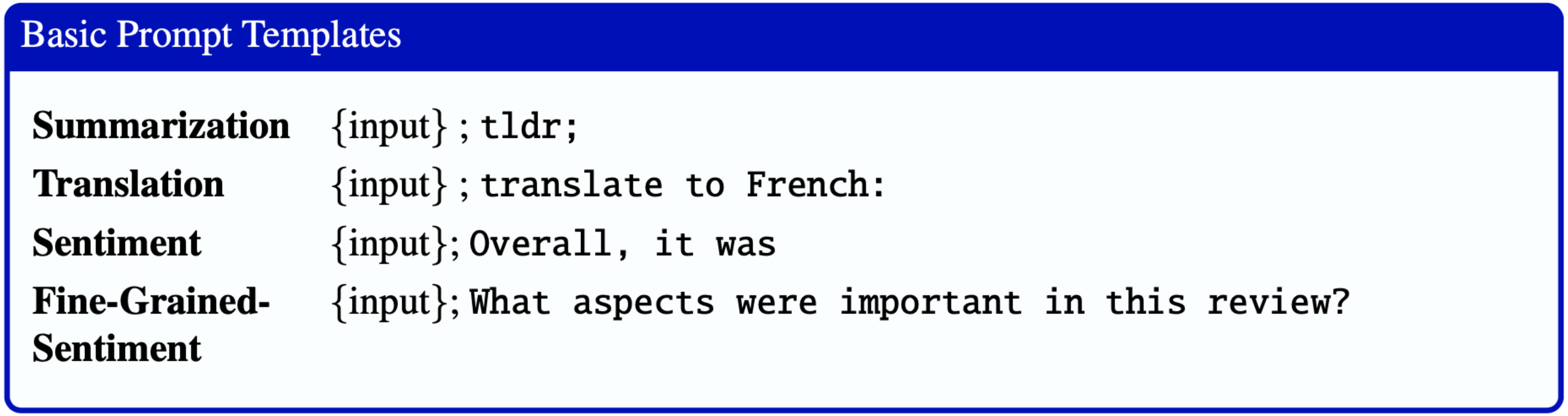
Prompts can be used for a great variety of tasks!!

- Summarization
- Translation
- (Fine-grained) sentiment analysis
- …

Templates

If we want to solve a task, we don’t want to have to create a new prompt each time we do it.

Use *templates*, instead!!

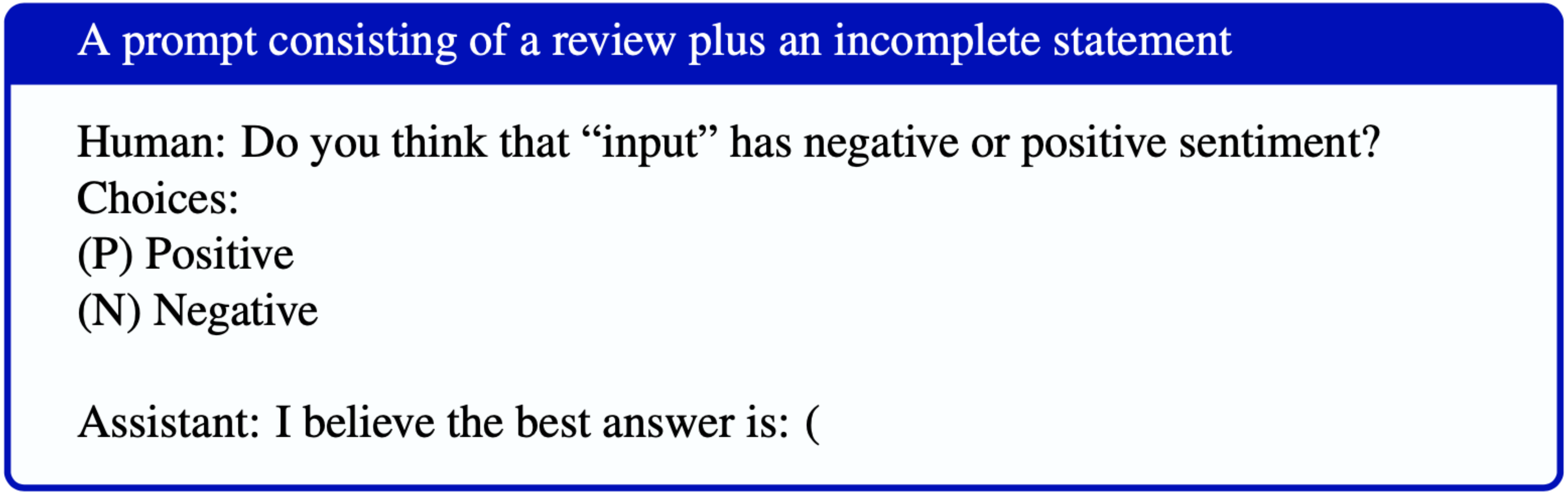


Basic Prompt Templates

Summarization	{input} ; `tldr;`
Translation	{input} ; `translate to French:`
Sentiment	{input}; `Overall, it was`
Fine-Grained-Sentiment	{input}; `What aspects were important in this review?`

Templates

Prompts (and templates also) need to be designed unambiguously, so that any reasonable continuation would accomplish the desired task.

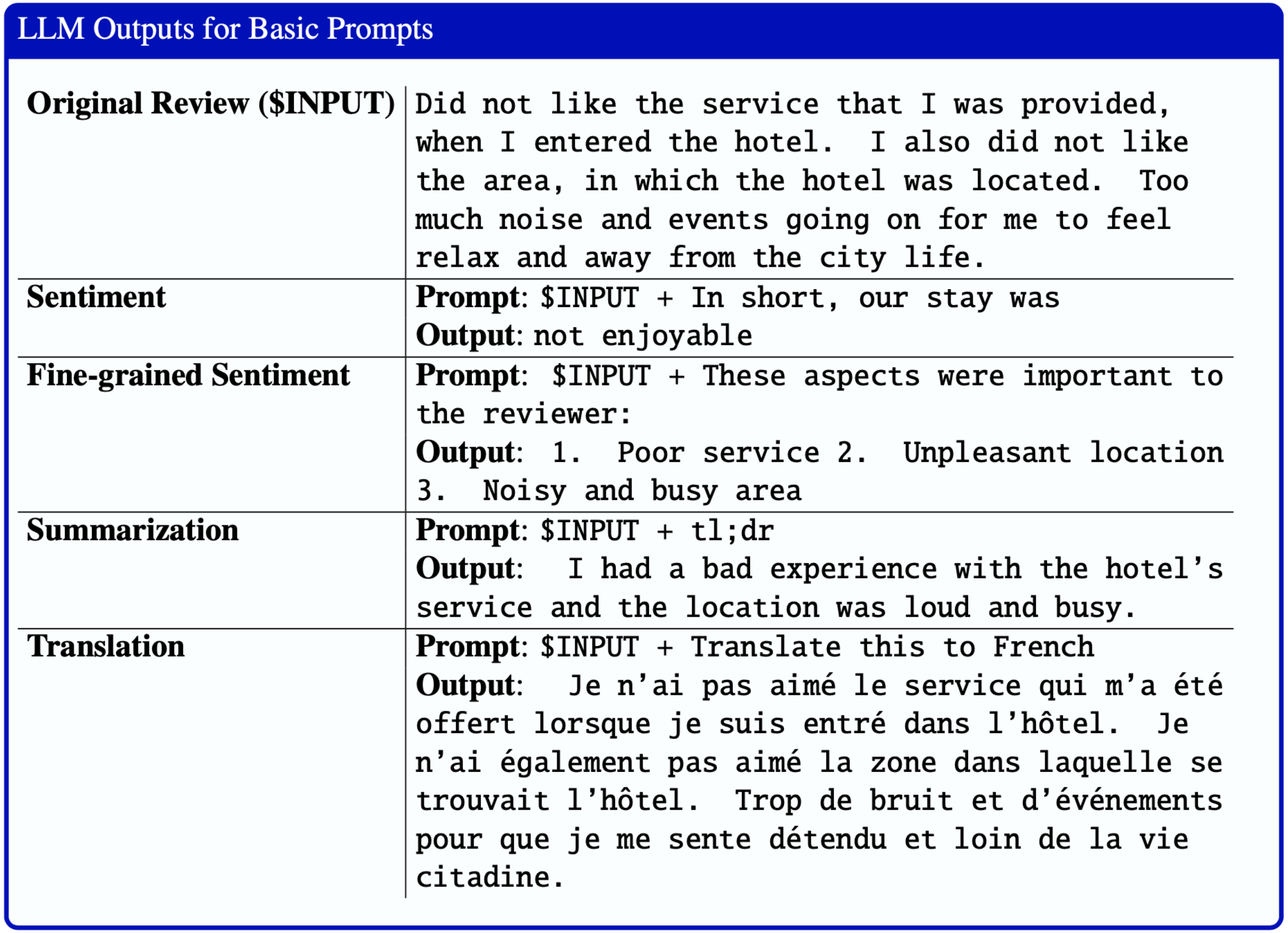


A prompt consisting of a review plus an incomplete statement

Human: Do you think that “input” has negative or positive sentiment?
Choices:
(P) Positive
(N) Negative

Assistant: I believe the best answer is: (

Templates



LLM Outputs for Basic Prompts

Original Review ($INPUT)	Did not like the service that I was provided, when I entered the hotel. I also did not like the area, in which the hotel was located. Too much noise and events going on for me to feel relax and away from the city life.
Sentiment	**Prompt**: $INPUT + In short, our stay was **Output**: not enjoyable
Fine-grained Sentiment	**Prompt**: $INPUT + These aspects were important to the reviewer: **Output**: 1. Poor service 2. Unpleasant location 3. Noisy and busy area
Summarization	**Prompt**: $INPUT + tl;dr **Output**: I had a bad experience with the hotel's service and the location was loud and busy.
Translation	**Prompt**: $INPUT + Translate this to French **Output**: Je n'ai pas aimé le service qui m'a été offert lorsque je suis entré dans l'hôtel. Je n'ai également pas aimé la zone dans laquelle se trouvait l'hôtel. Trop de bruit et d'événements pour que je me sente détendu et loin de la vie citadine.

Prompting with templates process

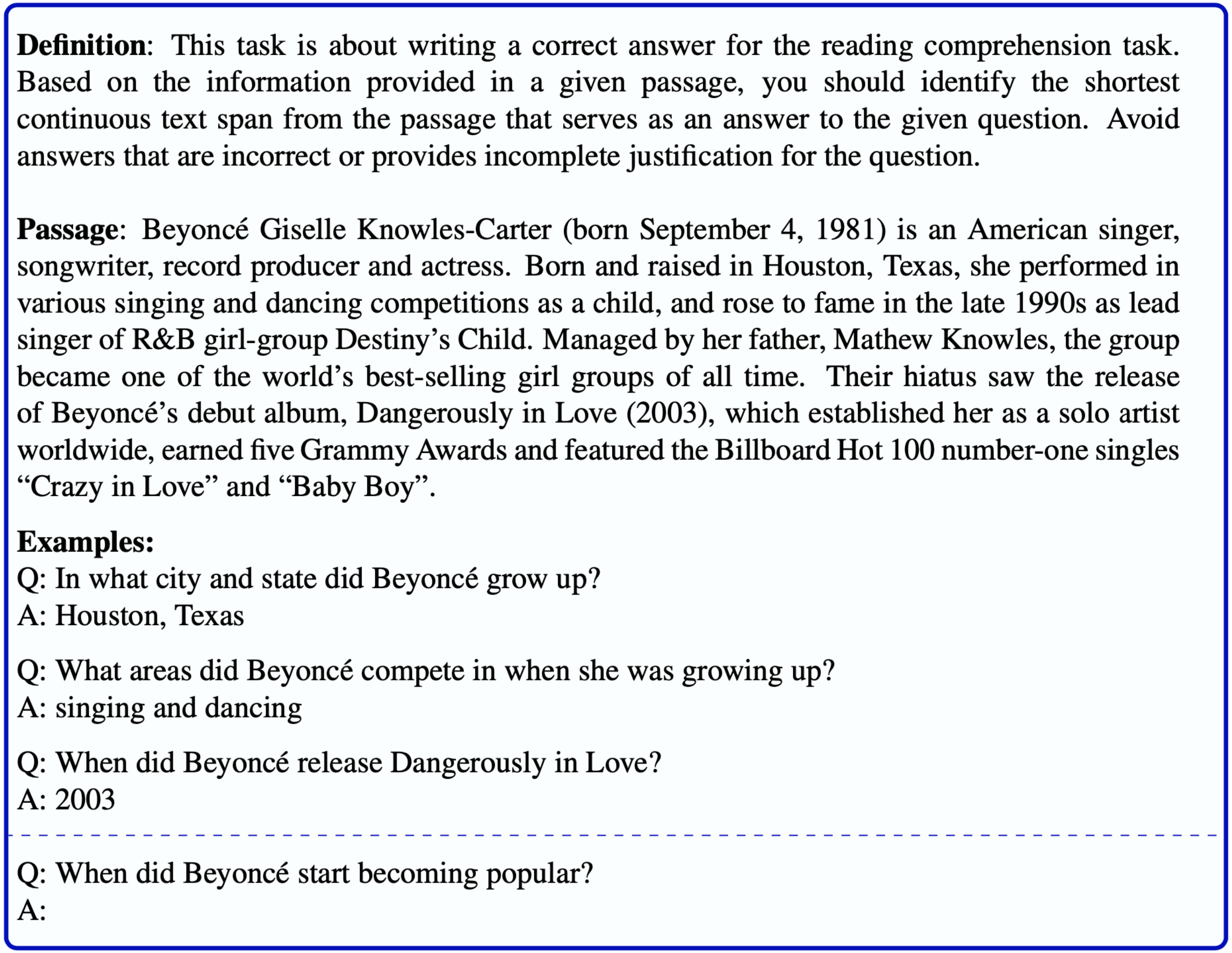
1. For a given task, develop a a task-specific template that has a free parameter for the input text.
2. Given that input and the task-specific template, the input is used to instantiate a filled prompt that is then passed to a pretrained language model.
3. Autoregressive decoding is then used to generate a sequence of token outputs.
4. The output of the model can either be used directly as the desired output or a task-appropriate answer can be extracted from the generated output.

Few-shot prompting

- Improving a prompt by including some labeled examples in the prompt template (the *demonstrations*)

- On the contrary, *zero-shots prompting* uses prompts that do not contain any labeled examples

Few-shot prompitng



Definition: This task is about writing a correct answer for the reading comprehension task. Based on the information provided in a given passage, you should identify the shortest continuous text span from the passage that serves as an answer to the given question. Avoid answers that are incorrect or provides incomplete justification for the question.

Passage: Beyoncé Giselle Knowles-Carter (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".

Examples:
Q: In what city and state did Beyoncé grow up?
A: Houston, Texas

Q: What areas did Beyoncé compete in when she was growing up?
A: singing and dancing

Q: When did Beyoncé release Dangerously in Love?
A: 2003

Q: When did Beyoncé start becoming popular?
A:

Few-shot prompting

- How many demonstrations?
- Not too many: choose a random small number
  - The largest performance gains tend to come from the first training example, and diminish for subsequent ones
  - They are useful at demostrating the task to the LLM along with the format of the sequence

Few-shot prompting

- How to select demonstrations?
- Automatically
  - Select the top-T most similar embeddings in the training set to the embedding of the current example
- Programmatically
  - Use some task performance measure to select the demonstrations that maximize the performance

In-Context Learning

- Prompting, with or without demonstrations, can be regarded as a sort of learning
    - The further a model gets in a prompt, the better it tends to get at predicting the upcoming tokens.
    - The information in the context is helping give the model more predictive power.
- We refer to either kind of learning that language models do from their prompts as *in-context learning*

Induction heads

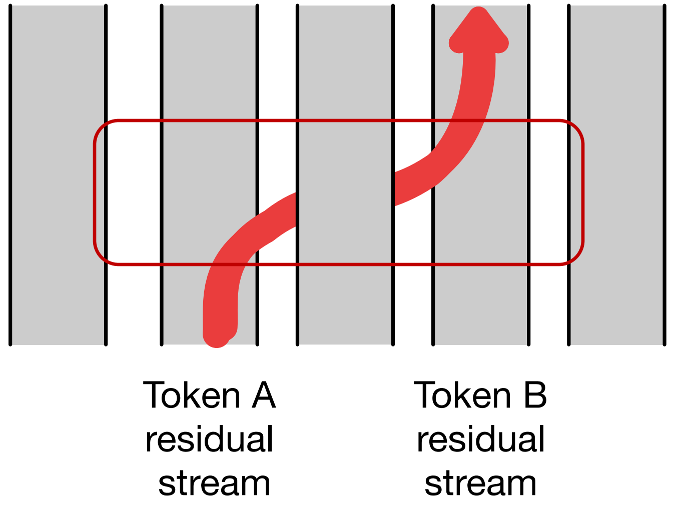
- Several intriguing hypotheses on how in-context learning works

- *Induction heads* have been devised as neural circuits made by some of the attention heads that assume a particular role in helping the model to predict:

$$\{A, B, \ldots, A\} \rightarrow B$$

https://arxiv.org/pdf/2404.07129

Induction heads



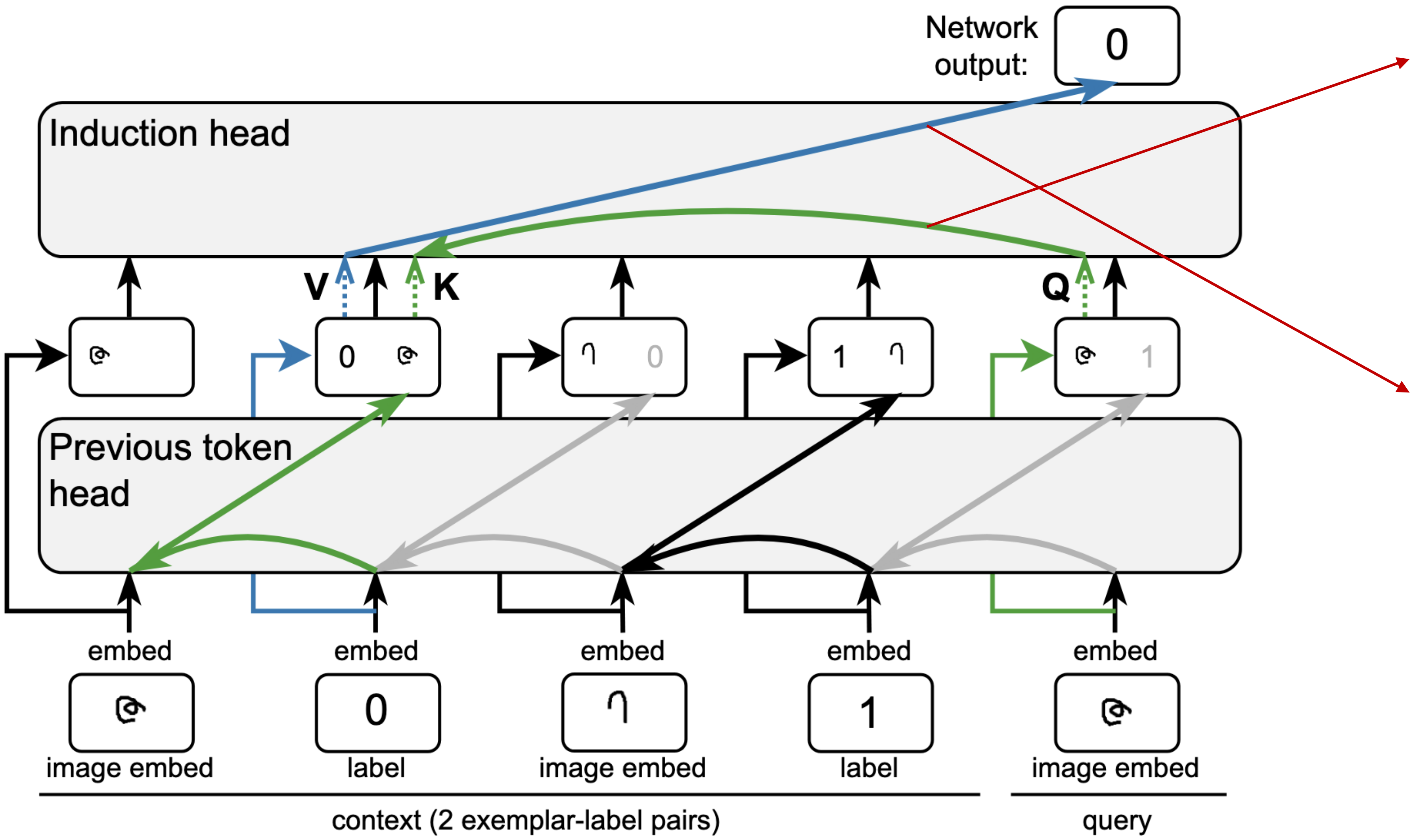
- Remember the notion of *residual stream*

- *Previous token heads* are attention heads responsible for attending to the previous token and copying it into the attending token’s residual stream.

- *Induction heads* Induction heads then perform a match-and-copy operation, looking for a match between a query derived from the current token and key derived from the output of the previous token head

https://arxiv.org/pdf/2404.07129

Induction heads



https://arxiv.org/pdf/2404.07129

Green path: Matching

*Current token's **Q** strongly matches the **K** of the token whose residual stream embeds the current token itself*

Blue path: Copying

*Matched token's **V** is moved (copied) to the output for the current token*

Model Alignment, Prompting, and In-Context Learning

Model Alignment

Model Alignment

- The power of in-context learning is limited
  - Often, the autoregressive tendency of the model pushes it to ignore prompt conditioning.
  - Pretrained LLMs generate texts that can be harmful (false, usafe, toxic, …)
- *Model alignment*: methods designed to adjust LLMs to better align them to human needs to generate helpful and non-harmful texts.

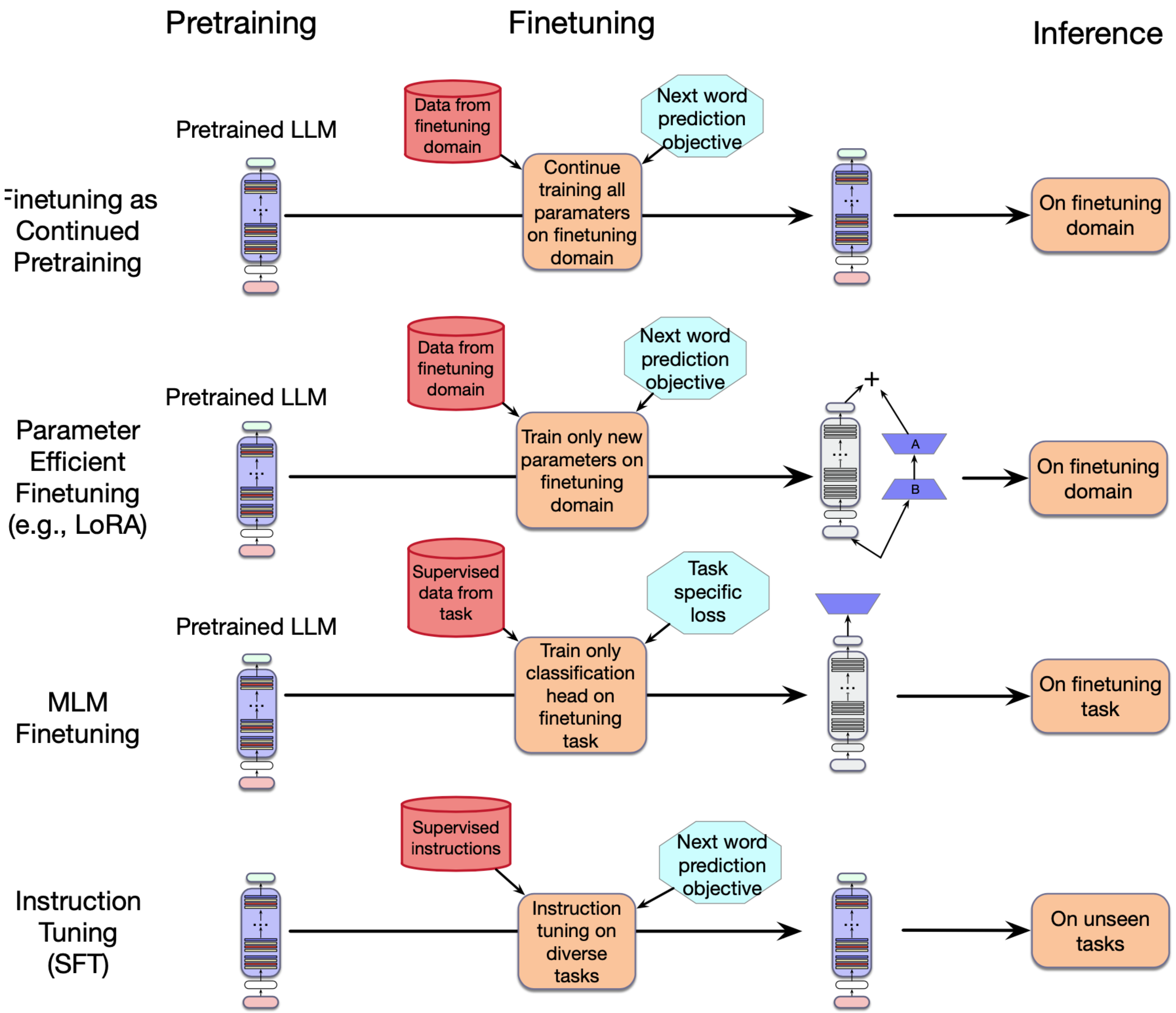
Model Alignment

- Two main strategies for model alignment
  - *Instruction tuning* (supervised finetuning - SFT): models are finetuned on a corpus of instructions and questions with their corresponding responses.
  - *Preference alignment*: a separate model is trained to decide how much a candidate response aligns with human preferences; this model is then used to finetune the base model.
  - RLHF - Reinforcement Learning from Human Feedback, is the most known preference alignment technique

Instruction tuning

- Instruction tuning (instruction finetuning, instruct tuning) is a method for making an LLM better at following instructions for a range of tasks.
- A base pretrained LLM is finetuned on a corpus of instructions and responses using always the language modeling loss.
- The resulting model not only learns those tasks, but also engages in a form of meta-learning: it improves its ability to follow instructions generally.

Instruction tuning


Pretraining
Finetuning
Inference
Finetuning as Continued Pretraining
Pretrained LLM
Data from finetuning domain
Next word prediction objective
Continue training all paramaters on finetuning domain
On finetuning domain
Parameter Efficient Finetuning (e.g., LoRA)
Pretrained LLM
Data from finetuning domain
Next word prediction objective
Train only new parameters on finetuning domain
+
A
B
On finetuning domain
MLM Finetuning
Pretrained LLM
Supervised data from task
Task specific loss
Train only classification head on finetuning task
On finetuning task
Instruction Tuning (SFT)
Supervised instructions
Next word prediction objective
Instruction tuning on diverse tasks
On unseen tasks

Instructions as Training Data

- Instruction tuning is actually a training procedure.

- As a consequence, labeled instructions corpora need to be developed purposely.

- Several corpora exist written by *humans*.
  - A very time consuming task

Instructions as Training Data

- Aya: 503 million instructions, 114 languages, 12 tasks
  - Q/A, summarization, translation, paraphrasing, sentiment analysis, natural language inference and 6 others
  - 3000 fluent speakers in 65 languages

Instructions as Training Data

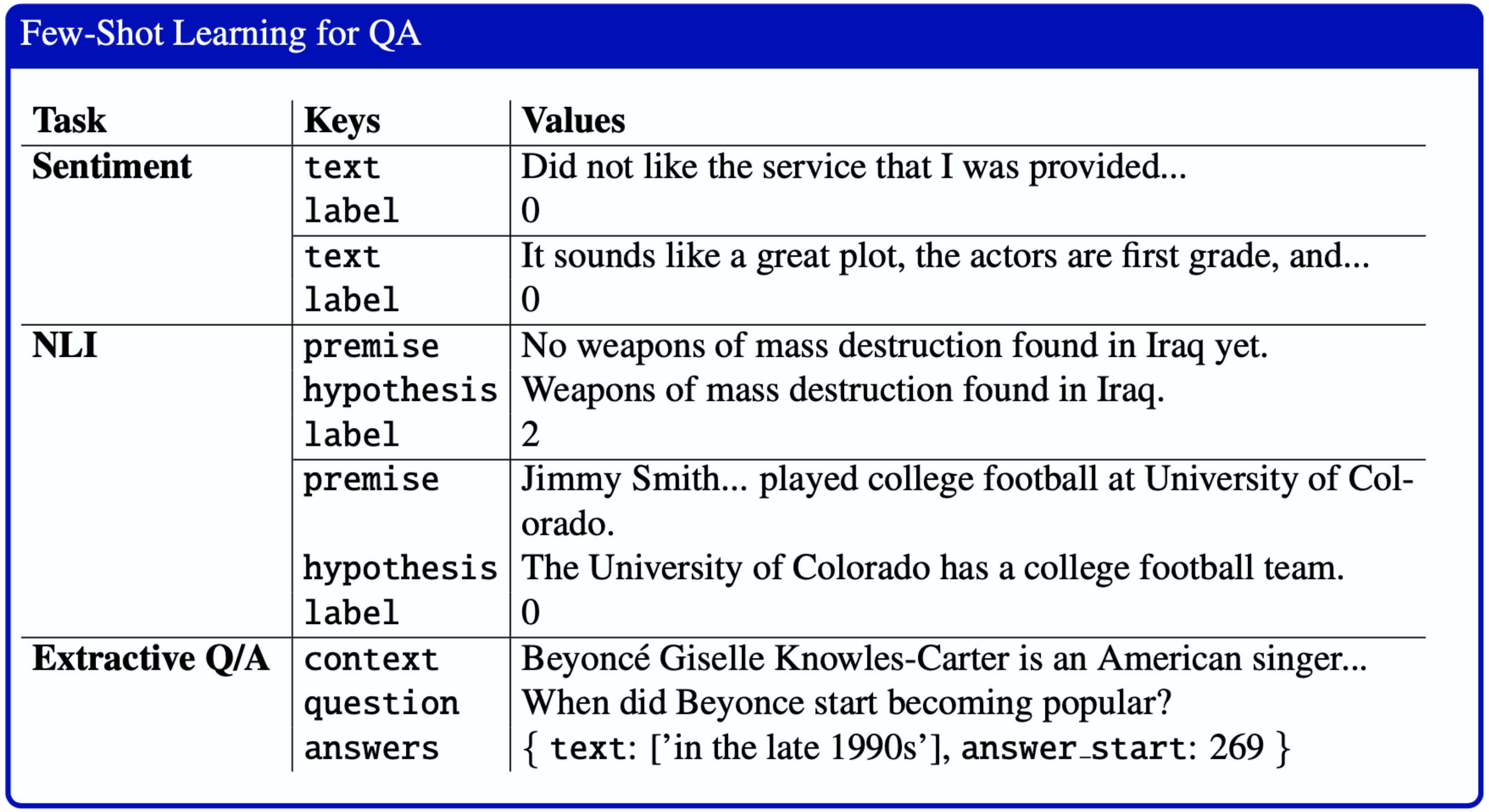
- SuperNatural Instructions: 12 million examples from 1600 tasks.
- Flan 2022: 15 million examples from 1836 tasks.
- OPT-IML: 18 million examples from 2000 tasks

Instructions as Training Data

- How to write high quality supervised instruction data?
- Make use of the very large amount of curated data developed over the years for whatever task
- Transform them into instructions automatically using templates

Instructions as Training Data

- How to write high quality supervised instruction data?

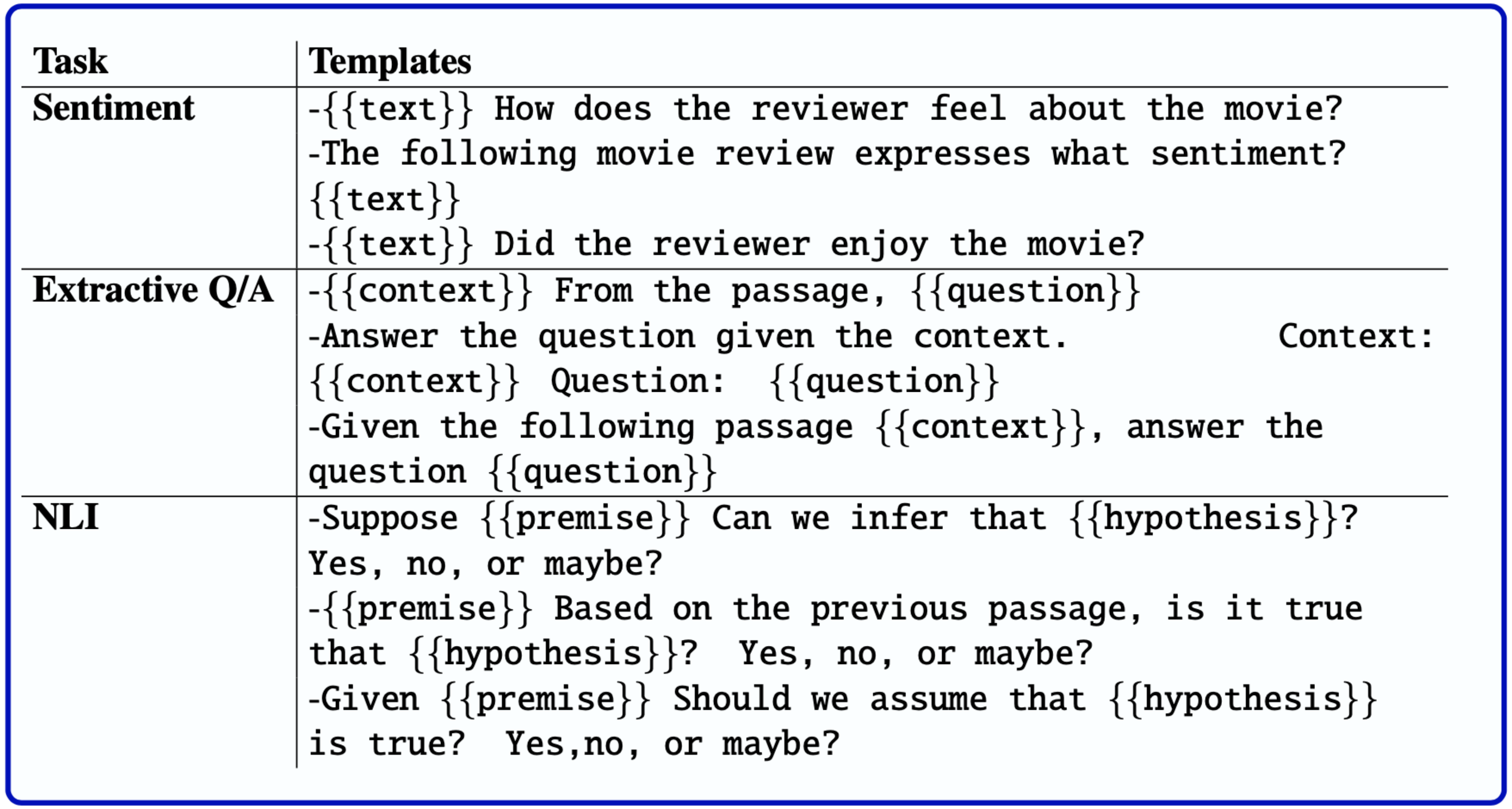


Few-Shot Learning for QA

Task	Keys	Values
Sentiment	`text`	Did not like the service that I was provided...
	`label`	0
	`text`	It sounds like a great plot, the actors are first grade, and...
	`label`	0
NLI	`premise`	No weapons of mass destruction found in Iraq yet.
	`hypothesis`	Weapons of mass destruction found in Iraq.
	`label`	2
	`premise`	Jimmy Smith... played college football at University of Colorado.
	`hypothesis`	The University of Colorado has a college football team.
	`label`	0
Extractive Q/A	`context`	Beyoncé Giselle Knowles-Carter is an American singer...
	`question`	When did Beyonce start becoming popular?
	`answers`	{ `text`: ['in the late 1990s'], `answer_start`: 269 }

Instructions as Training Data

- How to write high quality supervised instruction data?



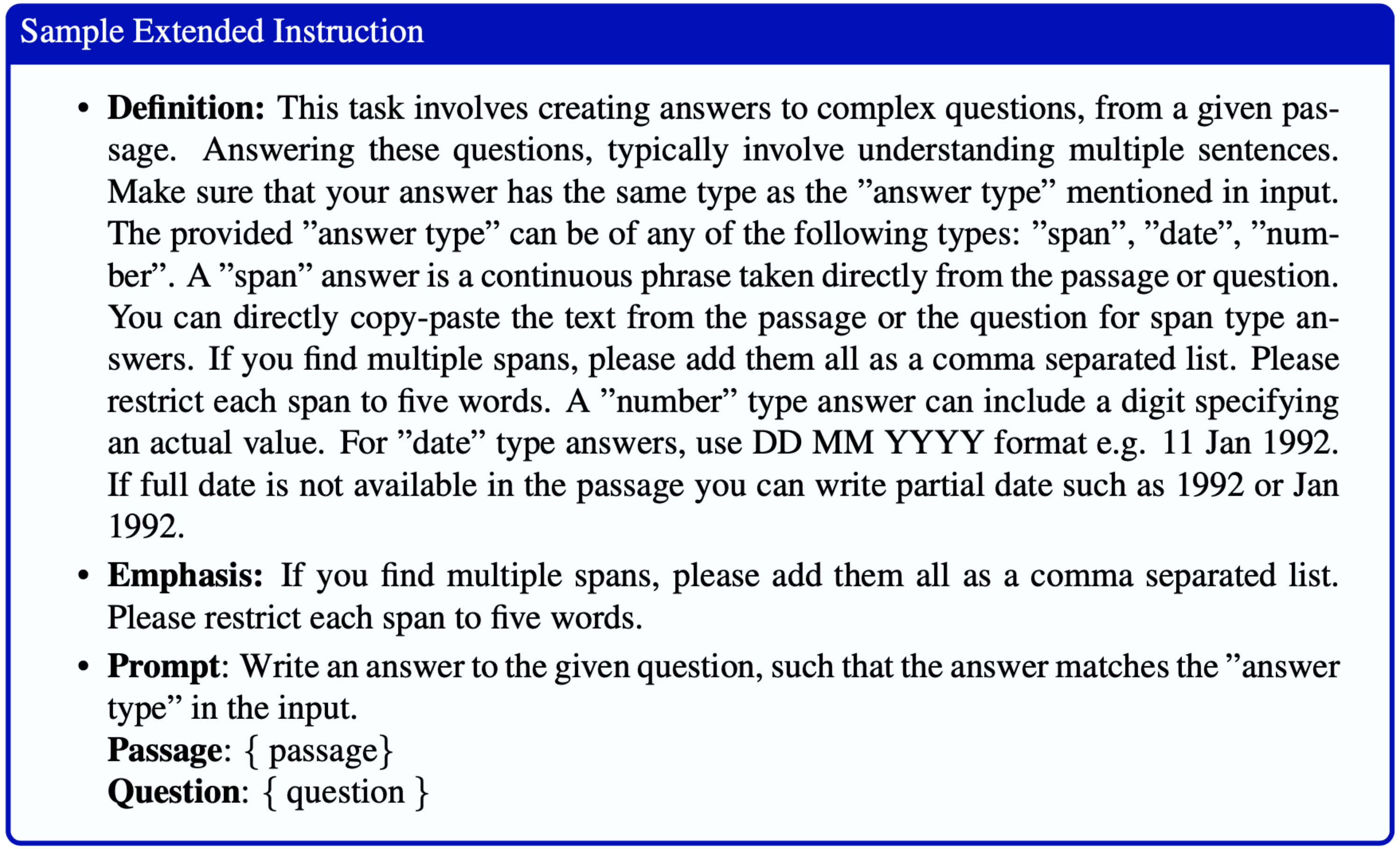
Task	Templates
Sentiment	-{{text}} How does the reviewer feel about the movie? -The following movie review expresses what sentiment? {{text}} -{{text}} Did the reviewer enjoy the movie?
Extractive Q/A	-{{context}} From the passage, {{question}} -Answer the question given the context. Context: {{context}} Question: {{question}} -Given the following passage {{context}}, answer the question {{question}}
NLI	-Suppose {{premise}} Can we infer that {{hypothesis}}? Yes, no, or maybe? -{{premise}} Based on the previous passage, is it true that {{hypothesis}}? Yes, no, or maybe? -Given {{premise}} Should we assume that {{hypothesis}} is true? Yes,no, or maybe?

Instructions as Training Data

- How to write high quality supervised instruction data?
- Many curated datasets are generated by crowdworkers based on carefully written annotation guidelines
- Prompt LLMs with instructions generated directly from these guidelines

Instructions as Training Data

- How to write high quality supervised instruction data?


Sample Extended Instruction

- **Definition:** This task involves creating answers to complex questions, from a given passage. Answering these questions, typically involve understanding multiple sentences. Make sure that your answer has the same type as the ”answer type” mentioned in input. The provided ”answer type” can be of any of the following types: ”span”, ”date”, ”number”. A ”span” answer is a continuous phrase taken directly from the passage or question. You can directly copy-paste the text from the passage or the question for span type answers. If you find multiple spans, please add them all as a comma separated list. Please restrict each span to five words. A ”number” type answer can include a digit specifying an actual value. For ”date” type answers, use DD MM YYYY format e.g. 11 Jan 1992. If full date is not available in the passage you can write partial date such as 1992 or Jan 1992.
- **Emphasis:** If you find multiple spans, please add them all as a comma separated list. Please restrict each span to five words.
- **Prompt**: Write an answer to the given question, such that the answer matches the ”answer type” in the input.
  Passage: { passage}
  Question: { question }

Instructions as Training Data

- How to write high quality supervised instruction data?
- LLMs can be used at each stage of the automatic transformation process
- i.e. they can be used to paraphrase instruction, thus augmenting the corpus size

Evaluation of Instruction-Tuned Models

- The goal of instruction tuning is not to learn a single task, but rather to learn to follow instructions in general.
- Assessing instruction-tuning methods means to assess how well an instruction-trained model performs on novel tasks.

Evaluation of Instruction-Tuned Models

- How to do it? Leave-one-out approach

- We remove an *entire cluster of tasks* from the corpus and use them for testing

  - Many tasks in a corpus are similar, and often overlap

  - The same evaluation metrics can be used for all the tasks in the test set

Model Alignment, Prompting, and In-Context Learning

Chain-of-Thought Prompting

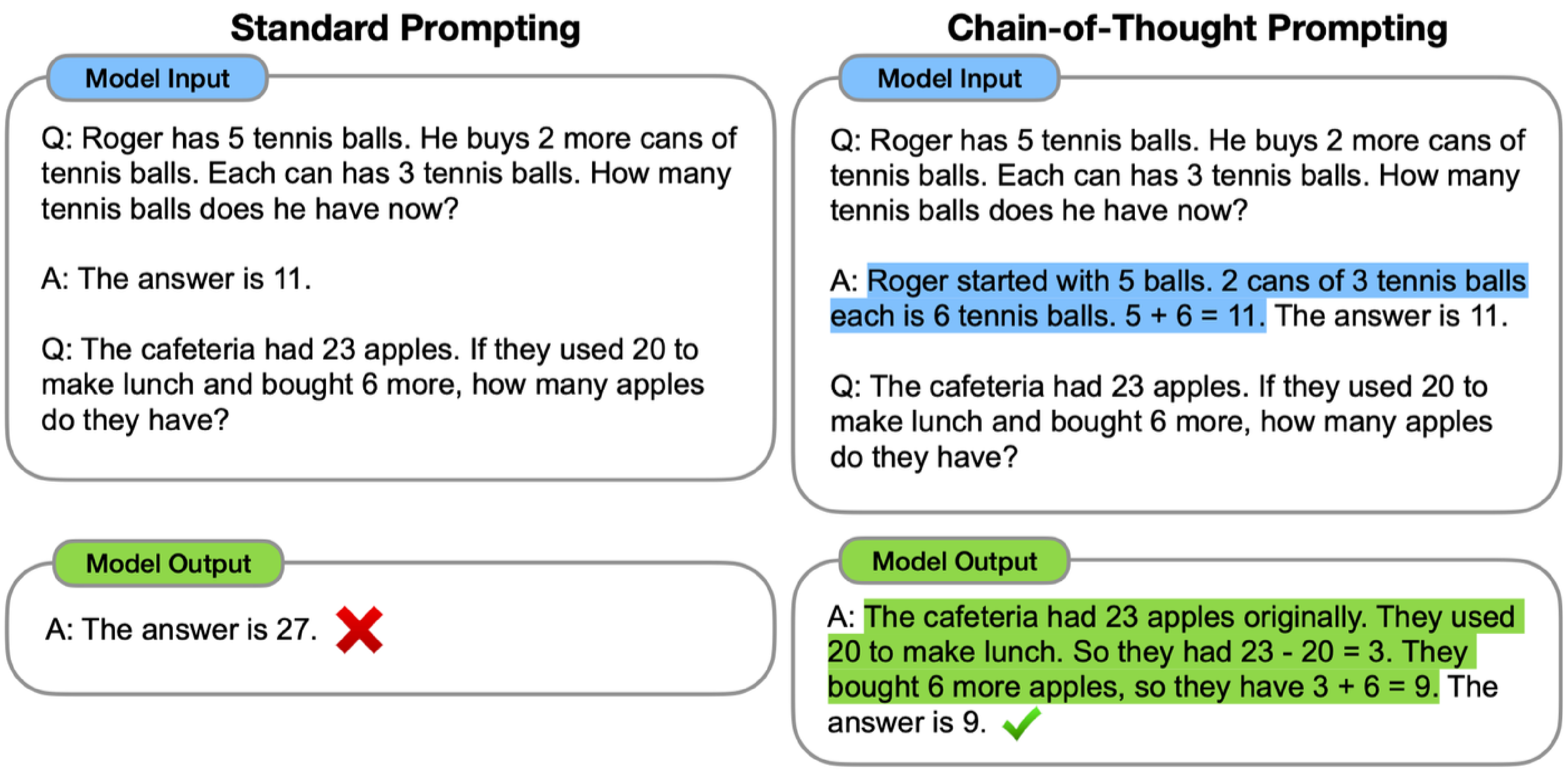
Chain-of-Thought Prompting

- Chain-of-thought prompting is a technique to improve performance on difficult reasoning tasks that language models tend to fail on.
- People solve complex tasks by breaking them down into steps.
- The language in the prompt will encourage LLMs to break them down in the same way.

Chain-of-Thought Prompting

- Each of the demonstrations in the few-shot prompt is augmented with some text explaining some reasoning steps.
- The goal is to cause the language model to output similar kinds of reasoning steps for the problem being solved, and to generate the correct answer as the output.

Chain-of-Thought Prompting



Standard Prompting

Model Input

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 Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

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?

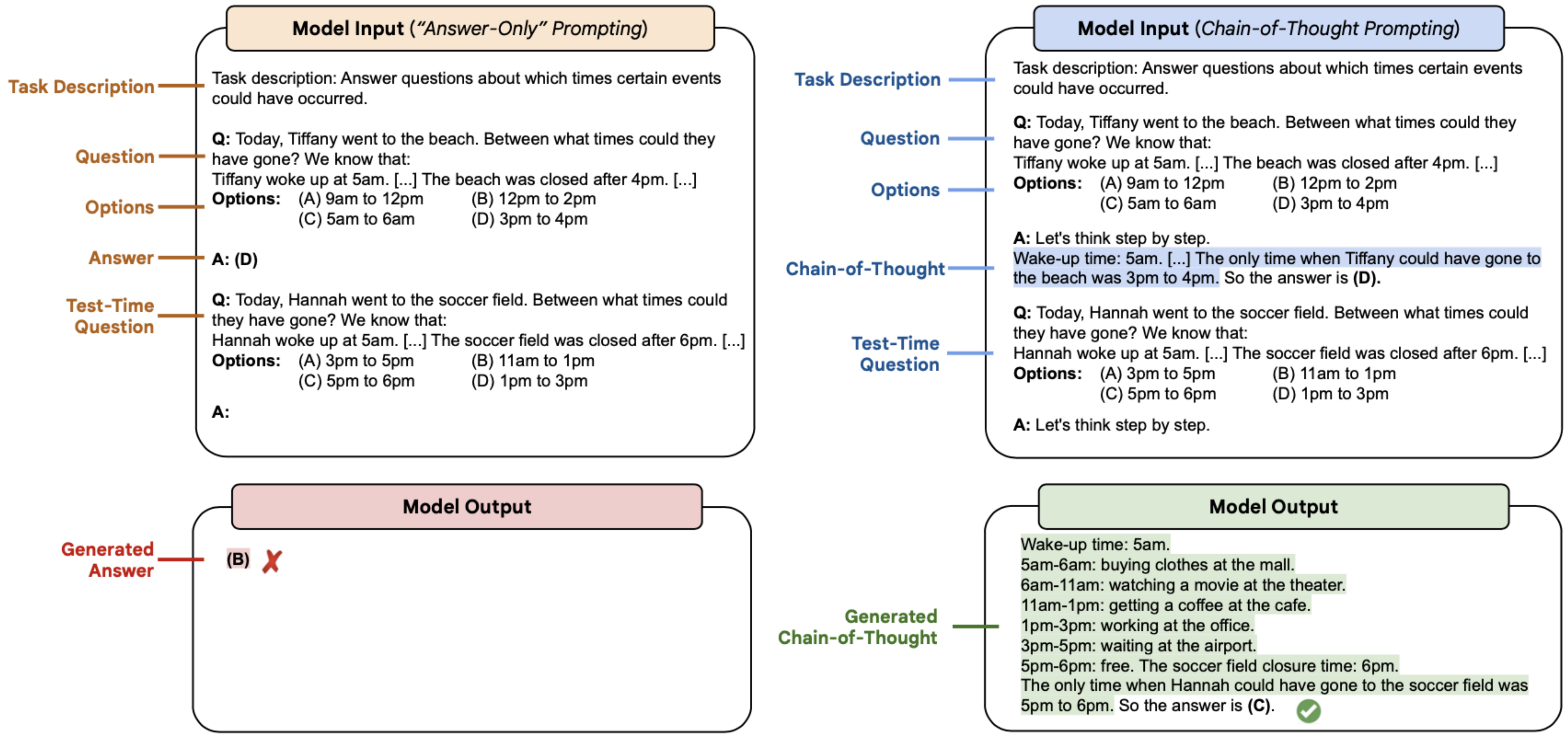
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

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

Model Output

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. ✓

Chain-of-Thought Prompting



Model Input (*"Answer-Only" Prompting*)

Task description: Answer questions about which times certain events could have occurred.

Q: Today, Tiffany went to the beach. Between what times could they have gone? We know that:
Tiffany woke up at 5am. [...] The beach was closed after 4pm. [...]
Options: (A) 9am to 12pm (B) 12pm to 2pm (C) 5am to 6am (D) 3pm to 4pm

A: (D)

Q: Today, Hannah went to the soccer field. Between what times could they have gone? We know that:
Hannah woke up at 5am. [...] The soccer field was closed after 6pm. [...]
Options: (A) 3pm to 5pm (B) 11am to 1pm (C) 5pm to 6pm (D) 1pm to 3pm

A:

Model Output

(B) ✗

Model Input (*Chain-of-Thought Prompting*)

Task description: Answer questions about which times certain events could have occurred.

Q: Today, Tiffany went to the beach. Between what times could they have gone? We know that:
Tiffany woke up at 5am. [...] The beach was closed after 4pm. [...]
Options: (A) 9am to 12pm (B) 12pm to 2pm (C) 5am to 6am (D) 3pm to 4pm

A: Let's think step by step.
Wake-up time: 5am. [...] The only time when Tiffany could have gone to the beach was 3pm to 4pm. So the answer is **(D).**

Q: Today, Hannah went to the soccer field. Between what times could they have gone? We know that:
Hannah woke up at 5am. [...] The soccer field was closed after 6pm. [...]
Options: (A) 3pm to 5pm (B) 11am to 1pm (C) 5pm to 6pm (D) 1pm to 3pm

A: Let's think step by step.

Model Output

Wake-up time: 5am.
5am-6am: buying clothes at the mall.
6am-11am: watching a movie at the theater.
11am-1pm: getting a coffee at the cafe.
1pm-3pm: working at the office.
3pm-5pm: waiting at the airport.
5pm-6pm: free. The soccer field closure time: 6pm.
The only time when Hannah could have gone to the soccer field was 5pm to 6pm. So the answer is **(C)**. ✓

Automatic Prompt Optimization

Automatic Prompt Optimization

- Given a prompt for a task, prompt optimization optimization methods search for prompts with improved performance.

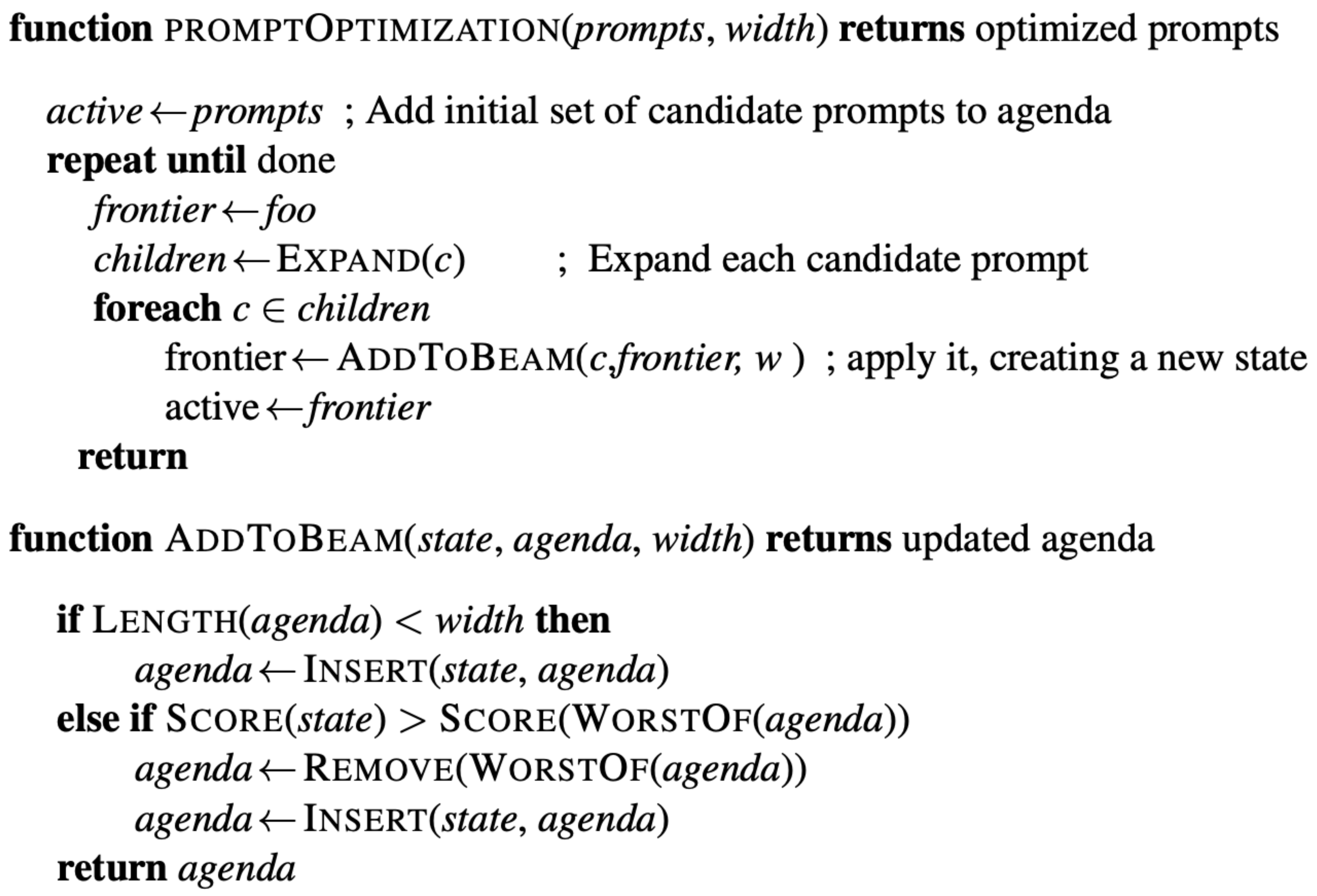
- This is much like iterative improvement search through a space of prompts

Automatic Prompt Optimization

- As a search algorithm we need:
  - A start state – an initial human or machine generated prompt or prompts
  - A scoring metric – a method for assessing how well the prompt performs on the task
  - An expansion method – a method for generating variants of the prompt

Automatic Prompt Optimization

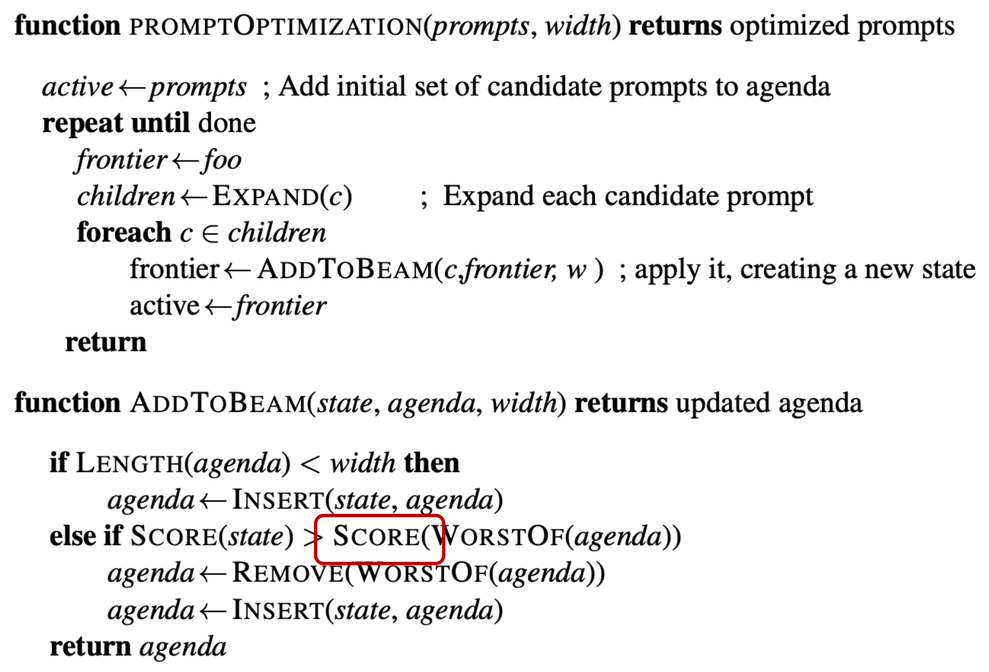
- Beam search



```
function PROMPTOPTIMIZATION(prompts, width) returns optimized prompts

  active ← prompts   ; Add initial set of candidate prompts to agenda
  repeat until done
     frontier ← foo
     children ← EXPAND(c)        ;  Expand each candidate prompt
     foreach c ∈ children
           frontier ← ADDTOBEAM(c,frontier, w )  ; apply it, creating a new state
           active ← frontier
    return

function ADDTOBEAM(state, agenda, width) returns updated agenda

   if LENGTH(agenda) < width then
         agenda ← INSERT(state, agenda)
   else if SCORE(state) > SCORE(WORSTOF(agenda))
         agenda ← REMOVE(WORSTOF(agenda))
         agenda ← INSERT(state, agenda)
   return agenda
```

Automatic Prompt Optimization

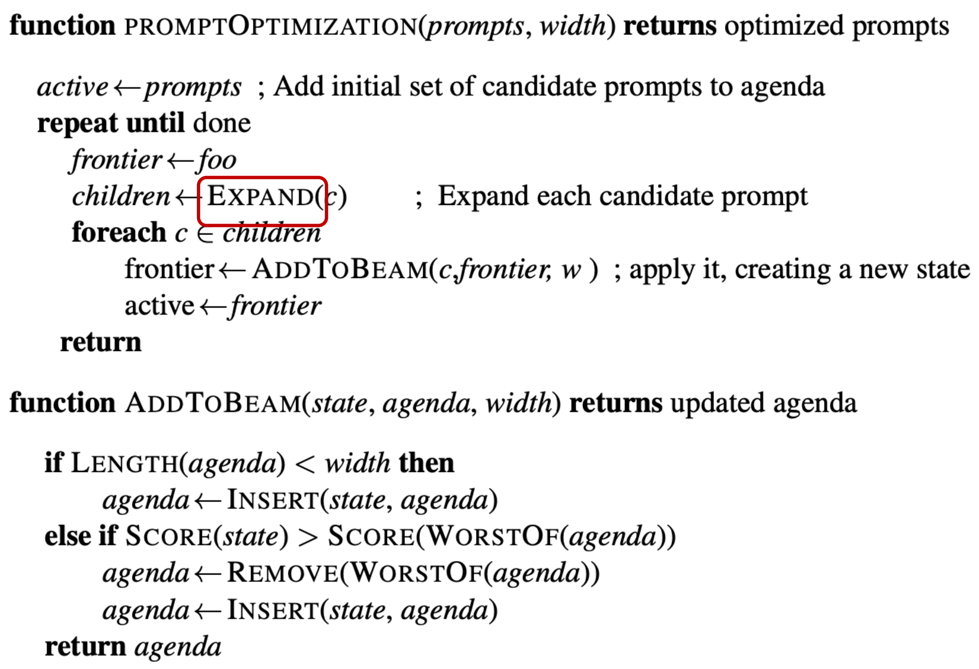
- Candidate scoring
  - Crucial for computational cost
  - *Execution accuracy*
    - Candidate prompts are mixed with samples from the training data
    - The LLM output is evaluated vs the training label using a metric appropriate for the task



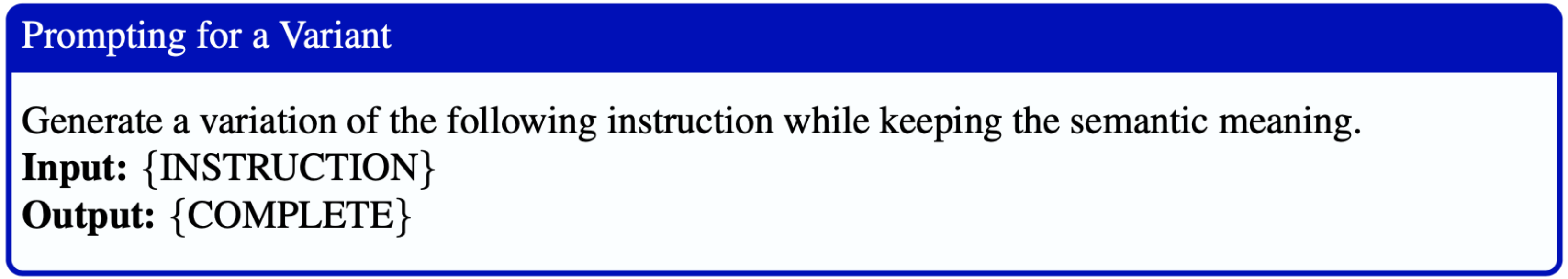
```
function PROMPTOPTIMIZATION(prompts, width) returns optimized prompts

  active ← prompts  ; Add initial set of candidate prompts to agenda
  repeat until done
    frontier ← foo
    children ← EXPAND(c)        ; Expand each candidate prompt
    foreach c ∈ children
        frontier ← ADDTOBEAM(c,frontier, w )  ; apply it, creating a new state
        active ← frontier
   return

function ADDTOBEAM(state, agenda, width) returns updated agenda

  if LENGTH(agenda) < width then
      agenda ← INSERT(state, agenda)
  else if SCORE(state) > SCORE(WORSTOF(agenda))
      agenda ← REMOVE(WORSTOF(agenda))
      agenda ← INSERT(state, agenda)
  return agenda
```

Automatic Prompt Optimization

- Prompt expansion
  - Variants of a given prompt to create a set of neighboring prompts that may improve performance over the original
  - Praphrasing is the usual strategy



```
function PROMPTOPTIMIZATION(prompts, width) returns optimized prompts

 active ← prompts  ; Add initial set of candidate prompts to agenda
 repeat until done
   frontier ← foo
   children ← EXPAND(c)        ; Expand each candidate prompt
   foreach c ∈ children
       frontier ← ADDTOBEAM(c,frontier, w ) ; apply it, creating a new state
       active ← frontier
   return

function ADDTOBEAM(state, agenda, width) returns updated agenda

  if LENGTH(agenda) < width then
      agenda ← INSERT(state, agenda)
  else if SCORE(state) > SCORE(WORSTOF(agenda))
      agenda ← REMOVE(WORSTOF(agenda))
      agenda ← INSERT(state, agenda)
  return agenda
```



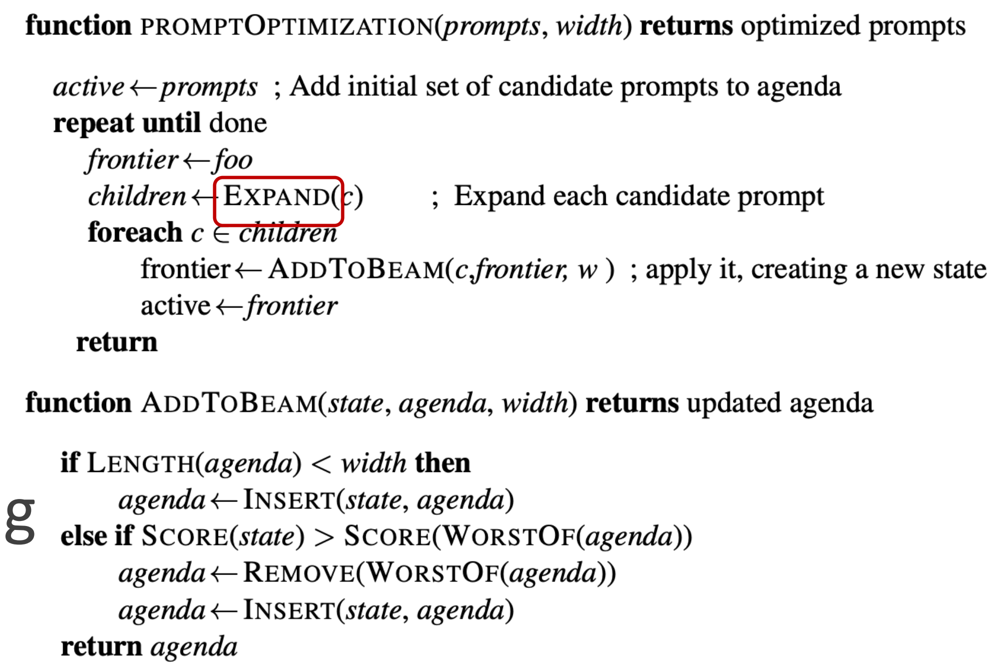
Prompting for a Variant

Generate a variation of the following instruction while keeping the semantic meaning.
Input: {INSTRUCTION}
Output: {COMPLETE}

Automatic Prompt Optimization

- Prompt expansion
  - Let’s use ***informed*** search

1. Run the prompt on a sample of training examples
2. Identify examples where the prompt *fails*
3. Ask a LLM to produce a critique of the prompt in light of the failed examples
4. Provide the critique to a LLM, and ask it to generate improved prompts



```
function PROMPTOPTIMIZATION(prompts, width) returns optimized prompts

  active ← prompts  ; Add initial set of candidate prompts to agenda
  repeat until done
     frontier ← foo
     children ← EXPAND(c)        ; Expand each candidate prompt
     foreach c ∈ children
          frontier ← ADDTOBEAM(c,frontier, w )  ; apply it, creating a new state
          active ← frontier
    return

function ADDTOBEAM(state, agenda, width) returns updated agenda

   if LENGTH(agenda) < width then
        agenda ← INSERT(state, agenda)
   else if SCORE(state) > SCORE(WORSTOF(agenda))
        agenda ← REMOVE(WORSTOF(agenda))
        agenda ← INSERT(state, agenda)
   return agenda
```

Automatic Prompt Optimization

- Prompt expansion

Critiquing Prompt

```
I'm trying to write a zero-shot classifier prompt.
My current prompt is:  {prompt}
But this prompt gets the following examples wrong:
{error_string}
Give {num_feedbacks} reasons why the prompt could have
gotten these examples wrong.
```

Prompt Improvement Prompt

```
I'm trying to write a zero-shot classifier.  My current prompt is:
{prompt}
But it gets the following examples wrong:  {error_str}

Based on these examples the problem with this prompt is that {gradient}.
Based on the above information, I wrote {steps_per_gradient} different
improved prompts.  Each prompt is wrapped with <START> and <END>.

The {steps_per_gradient} new prompts are:
```

```
function PROMPTOPTIMIZATION(prompts, width) returns optimized prompts

  active ← prompts  ; Add initial set of candidate prompts to agenda
  repeat until done
    frontier ← foo
    children ← EXPAND(c)        ; Expand each candidate prompt
    foreach c ∈ children
        frontier ← ADDTOBEAM(c,frontier, w )  ; apply it, creating a new state
        active ← frontier
   return

function ADDTOBEAM(state, agenda, width) returns updated agenda

  if LENGTH(agenda) < width then
      agenda ← INSERT(state, agenda)
  else if SCORE(state) > SCORE(WORSTOF(agenda))
      agenda ← REMOVE(WORSTOF(agenda))
      agenda ← INSERT(state, agenda)
  return agenda
```

Model Alignment, Prompting, and In-Context Learning

Model Alignment with Human Preferences: RLHF and DPO

RLHF

- Makes use of human trainers to improve model performance.
    - Human trainers rank the response provided by the model in a previous conversation
    - Ranks are then used to create a reward model used in the iterations of the *Proximal Policy Optimization* (PPO) reinforcement learning algorithm

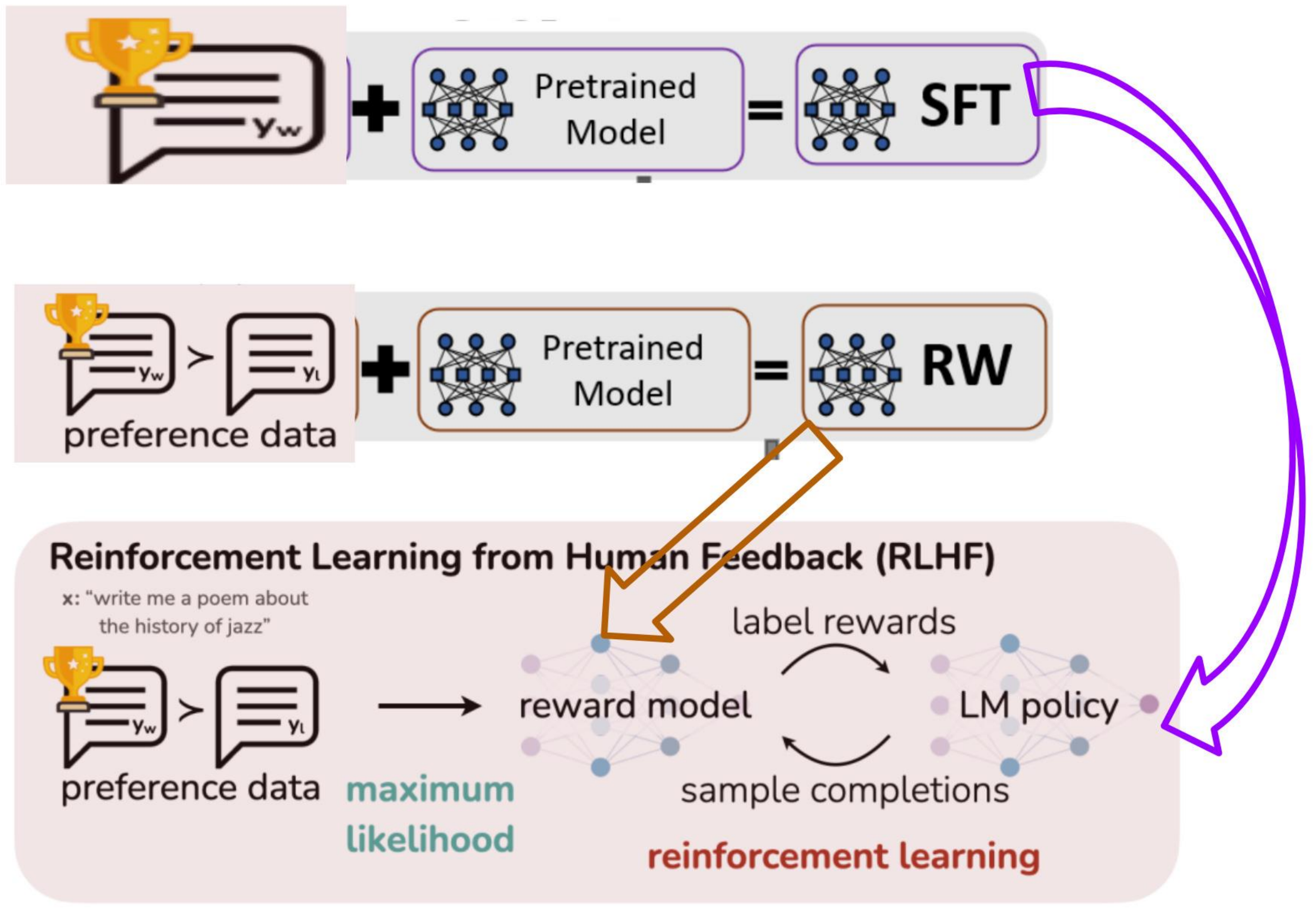
RLHF

Reinforcement Learning from Human Feedback(RLHF)

STEP1:
Supervised Fine Tune

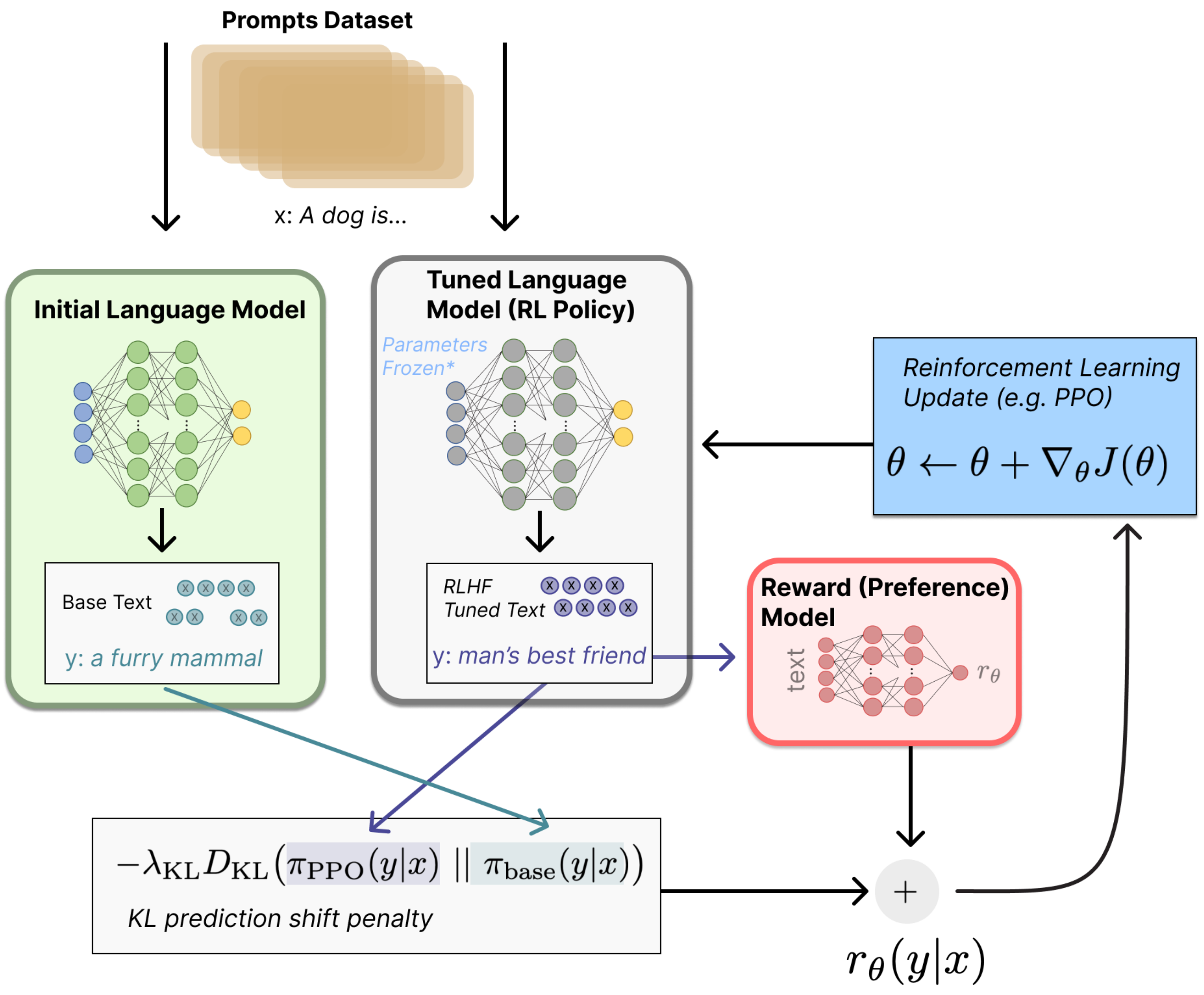
STEP2:
Reward Model Training

STEP3:
Reinforcement Learning



https://bit.ly3Ou9thQ/

RLHF



Prompts Dataset
x: A dog is...
Initial Language Model
Base Text
y: a furry mammal
Tuned Language Model (RL Policy)
Parameters Frozen*
RLHF Tuned Text
y: man's best friend
Reinforcement Learning Update (e.g. PPO)
$\theta \leftarrow \theta + \nabla_\theta J(\theta)$
Reward (Preference) Model
text
r_θ
$-\lambda_{\text{KL}} D_{\text{KL}} (\pi_{\text{PPO}}(y|x) \,||\, \pi_{\text{base}}(y|x))$
KL prediction shift penalty
+
$r_\theta(y|x)$

https://huggingface.co/blog/rlhf

Proximal Policy Optimization - PPO

- PPO is a «policy gradient» RL algorithm.

1. **Policy Initialization**: Initialize a policy (a function that maps states to actions).
2. *Repeat*
    1. **Data Collection**: Collect data by interacting with the environment using the current policy.
    2. **Estimate Advantage**: Calculate the advantage function, which measures how much better an action is compared to the average action.
    3. **Update Policy**: Update the policy using gradient ascent, using a clipping function to make the new policy not too different from the old one.
3. *Until convergence*

https://bit.ly/41aUCjW

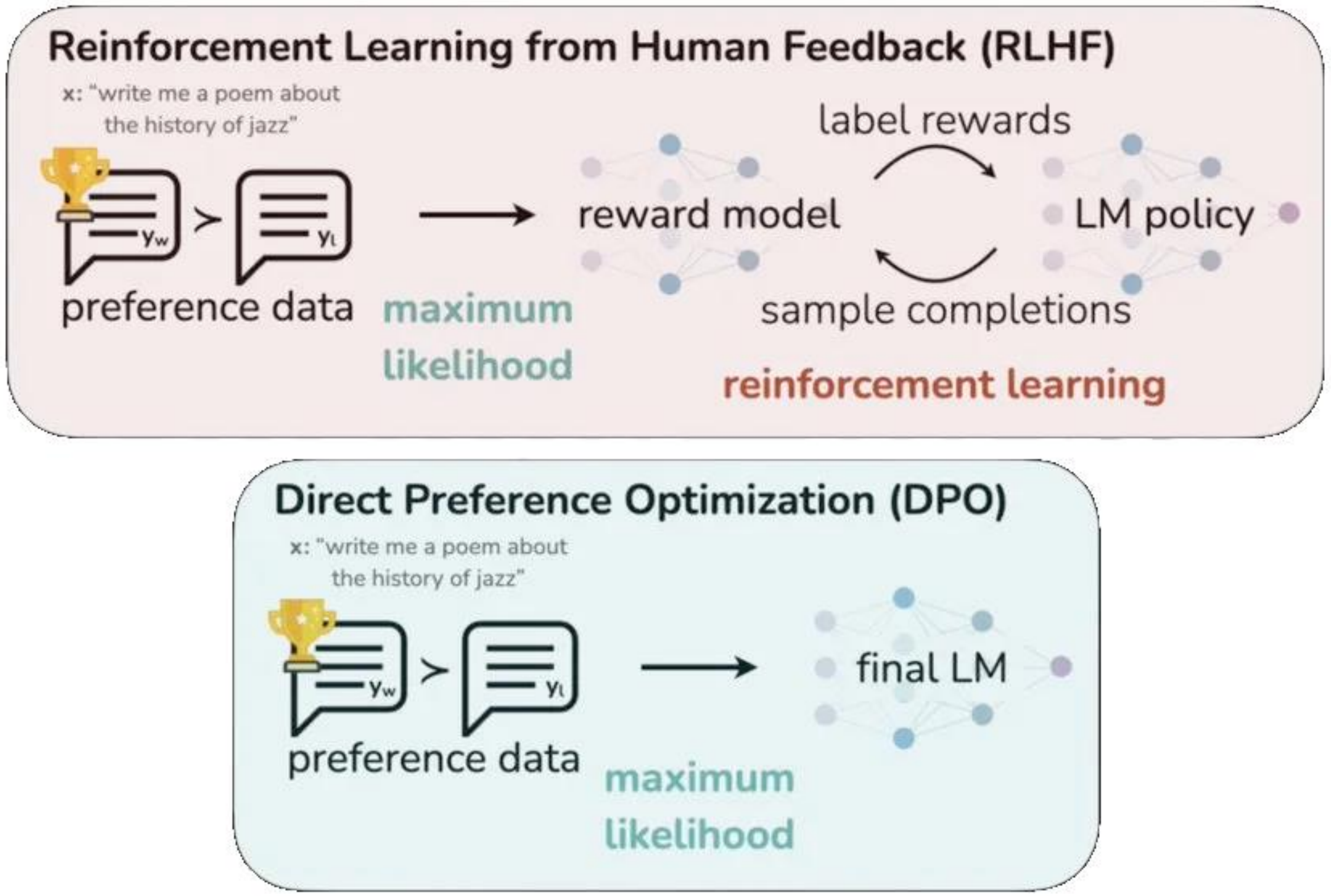
Proximal Policy Optimization - PPO

- PPO is stable (policy changes are controlled), robust, and computationally efficient
  - Deals with complex reward landscapes
  - Good for complex tasks requiring Iterative Learning, like code generation and long term planning
  - Good for model deployment at scale (i.e. GPT models)

https://bit.ly/41aUCjW

Direct Preference Optimization - DPO

DPO is an improved approach for preference alignent that eliminates the need for reward model fitting and extensive hyperparameter tuning.


Reinforcement Learning from Human Feedback (RLHF)
x: "write me a poem about the history of jazz"
y_w
>
y_l
preference data
maximum likelihood
reward model
label rewards
LM policy
sample completions
reinforcement learning
Direct Preference Optimization (DPO)
x: "write me a poem about the history of jazz"
y_w
>
y_l
preference data
maximum likelihood
final LM

https://bit.ly/49enn0V

Direct Preference Optimization - DPO

- PPO is a finetuning technique for LLMs

1. **Data Collection:** Gather a dataset of model outputs and corresponding human preferences. These preferences can be of rankings, ratings, or binary comparisons.
2. **Model Initialization:** Initialize the model with random parameters.
3. *Repeat*
   1. **Preference Modeling**: Develop a loss function that measures the discrepancy between the model's outputs and human preferences.
   2. **Parameter Update:** To minimize the loss function, update the model's parameters using gradient descent or similar optimization algorithms.
4. *Until convergence*

https://bit.ly/41aUCjW

Direct Preference Optimization - DPO

- DPO performs faster and more effective alignment, it is very lightweight, and can mitigate the risk of inheriting biases from the training data.
  - Good for simpler tasks
  - Requires very low computational load
  - Allows for frequent update of the model to users' preferences

https://bit.ly/41aUCjW