Natural Language Processing

Week 9

Agenda

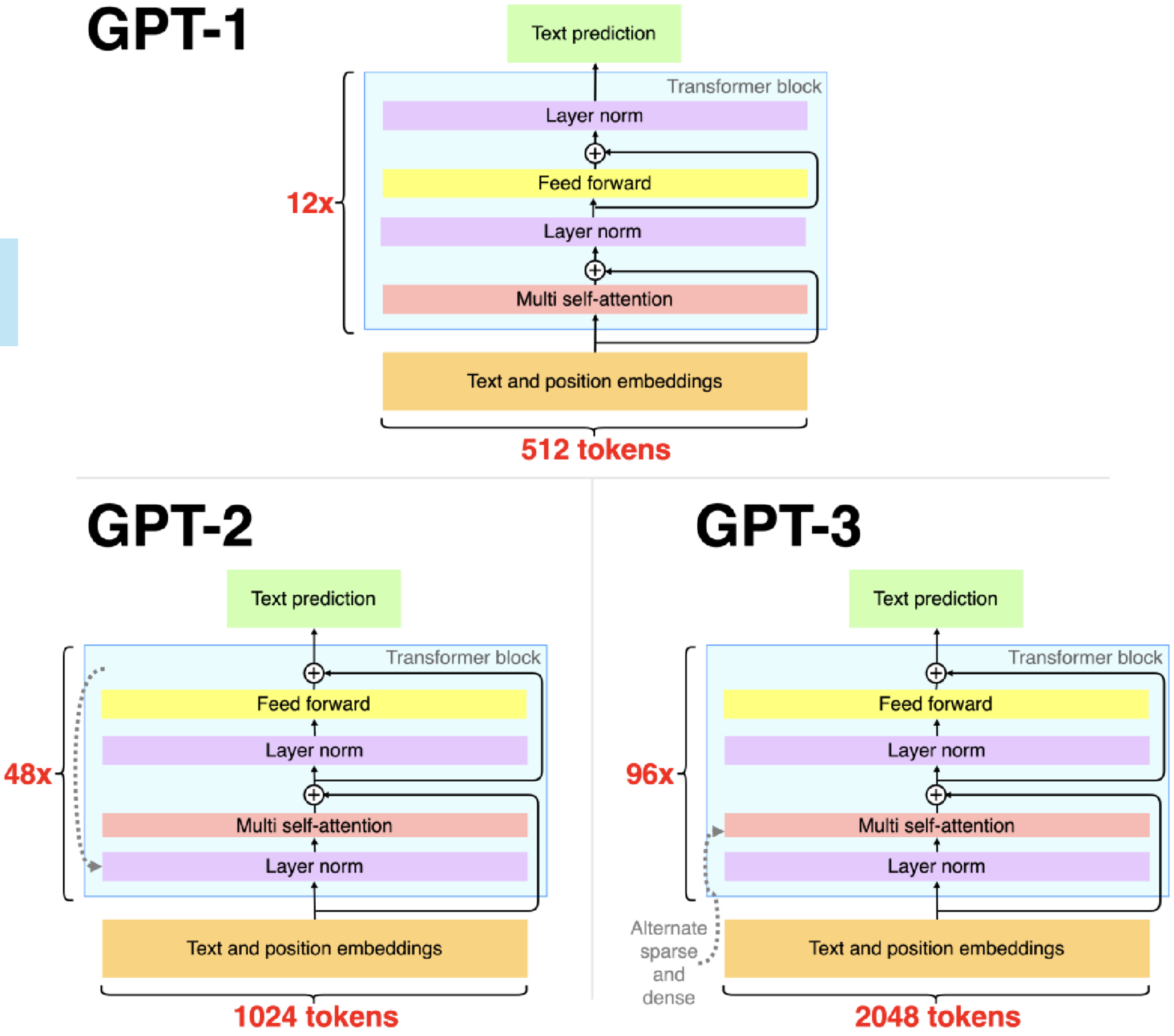
- Recap
- Finish Week 8
- GPT3 ---> InstructGPT (RLHF)
- Prompt Engineering
- Retrieval Augmented Generation (RAG)

From GPT3 to InstructGPT

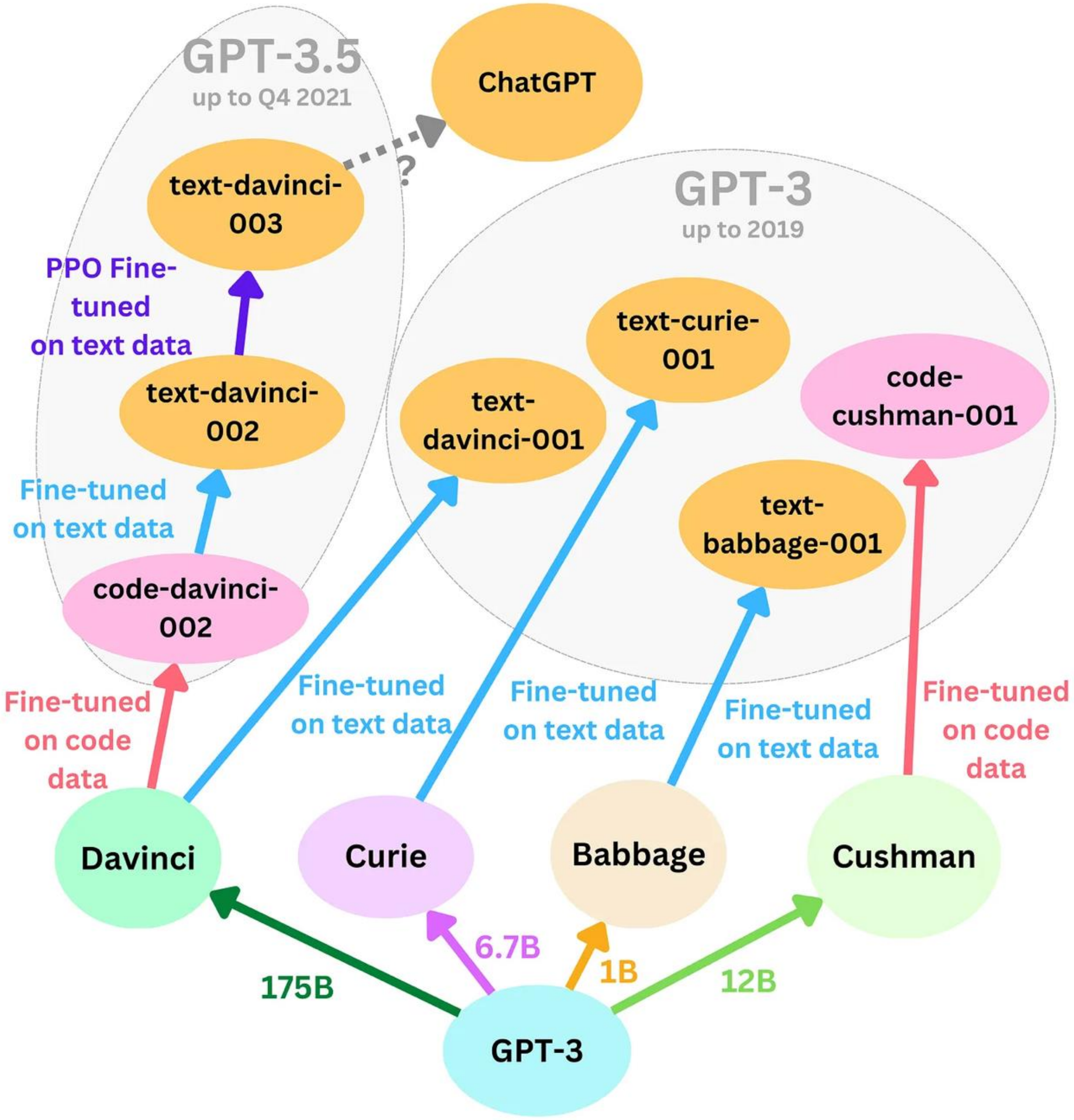
Reminder: GPT 1 – GPT 3

- Very similar in terms of models in terms of architecture

Model	Parameters	Layers	Context Window
GPT-1	117 million	12	512
GPT-2 - Small	124 million	12	1024
GPT-2 - Medium	355 million	24	1024
GPT-2 - Large	774 million	36	1024
GPT-2 - XL	1.5 billion	48	1024
GPT-3 – 2.7B	2.7 billion	32	2048
GPT-3 – 6.7B	6.7 billion	40	2048
GPT-3 – 13B	13 billion	48	2048
GPT-3 – 175B	175 billion	96	2048



Reminder: GPT 1 – GPT 3



https://newsletter.theaiedge.io/p/the-chatgpt-models-family

https://platform.openai.com/docs/models

InstructGPT

Prompt	Explain the moon landing to a 6 year old in a few sentences.
GPT 3 Completion	Explain the theory of gravity to a 6 year old. Explain the theory of relativity to a 6 year old in a few sentences. Explain the big bang theory to a 6 year old. Explain evolution to a 6 year old.
InstructGPT Completion	People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them

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

Misalignment

"… the language modeling objective used for many recent large LMs – predicting the next token on a webpage from the internet – is different from the object 'follow the user's instructions helpfully and safely'."

https://arxiv.org/abs/2203.02155

Misalignment

- **Failure to assist users:** model disregards explicit user instructions
- **Hallucinations:** the model generates nonexistent or incorrect information
- **Lack of Transparency:** the model's decision-making process is difficult for users to understand
- **Toxic / Biased Output:** a model trained on internet-scale data may produce such an output even if not explicitly instructed to do so

InstructGPT

"Training language models to follow instructions with human feedback"

Making language models bigger does not inherently make them better at following a user's intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these models are not *aligned* with their users. In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through the OpenAI API, we collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning. We then collect a dataset of rankings of model outputs, which we use to further fine-tune this supervised model using reinforcement learning from human feedback. We call the resulting models *InstructGPT*. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters. Moreover, InstructGPT models show improvements in truthfulness and reductions in toxic output generation while having minimal performance regressions on public NLP datasets. Even though InstructGPT still makes simple mistakes, our results show that fine-tuning with human feedback is a promising direction for aligning language models with human intent.

Training InstructGPT

- **Supervised Finetuning**
  - Human-written responses to 13k prompts submitted through the OpenAI API
- **Reward Modeling**
  - Creating a **preference dataset** 33k prompts
  - Training a rewards model on preference dataset
- **Reinforcement Learning**
  - Further fine-tuning with the reward model using reinforcement learning

https://arxiv.org/abs/2203.02155

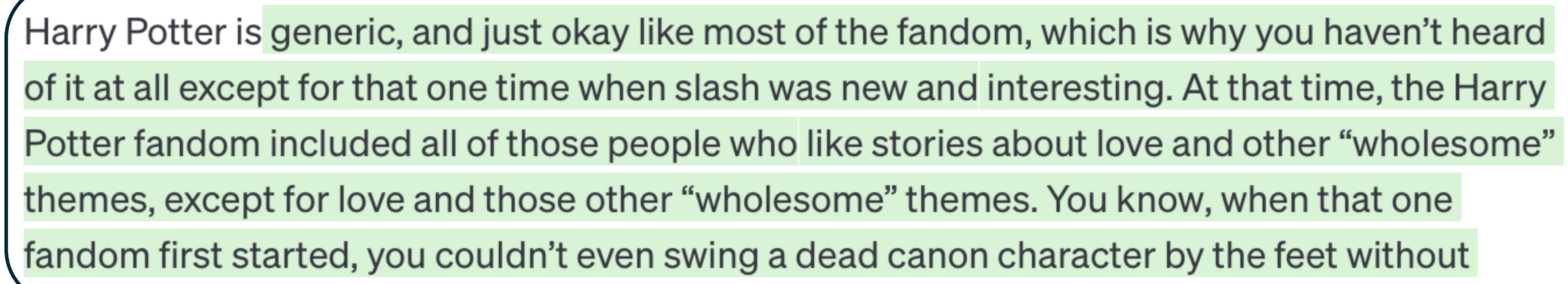
Supervised Fine-Tuning

Supervised Fine-tuning (SFT)

- During SFT, the model is fine-tuned on “ideal” text generations for different tasks
- If we give a pre-trained model a question: “Why does the apple fall from the tree”, any of the following would be valid completions:
    - “… and hit Isaac Newton”
    - “? Is it thrown? What knocked it off?”
- The goal of the SFT is to optimize the model to generate responses users are looking for
- This is just a regular, causal fine-tuning that can be done for any model

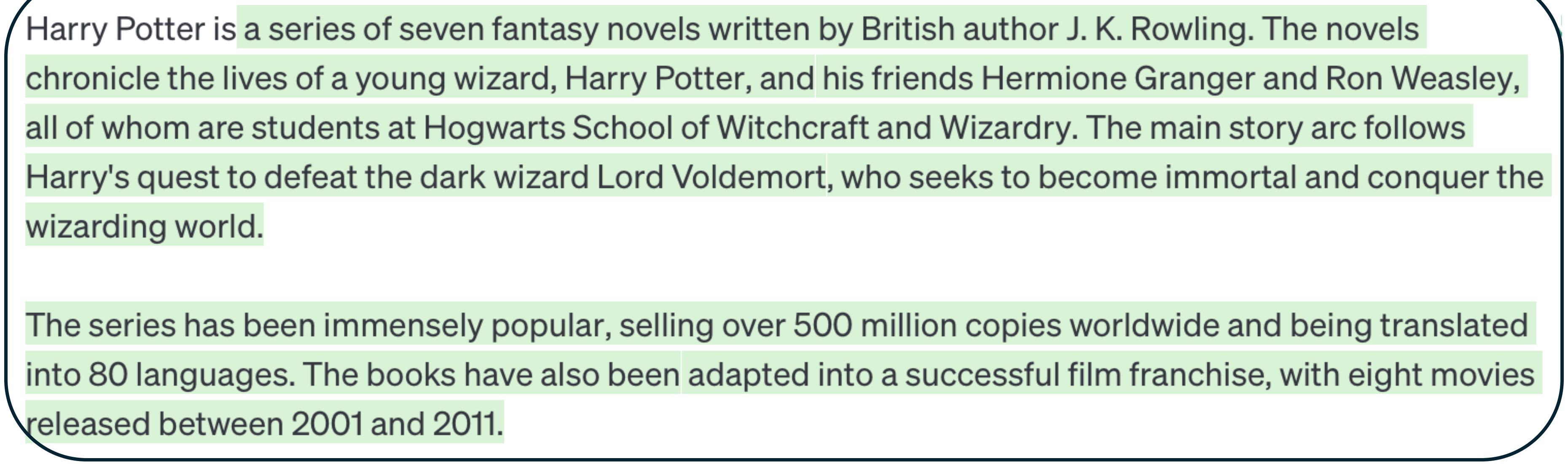
Before vs. After SFT

GPT-3 without instruction fine-tuning



Harry Potter is generic, and just okay like most of the fandom, which is why you haven't heard of it at all except for that one time when slash was new and interesting. At that time, the Harry Potter fandom included all of those people who like stories about love and other "wholesome" themes, except for love and those other "wholesome" themes. You know, when that one fandom first started, you couldn't even swing a dead canon character by the feet without

GPT-3.5 with instruction fine-tuning



Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc follows Harry's quest to defeat the dark wizard Lord Voldemort, who seeks to become immortal and conquer the wizarding world.

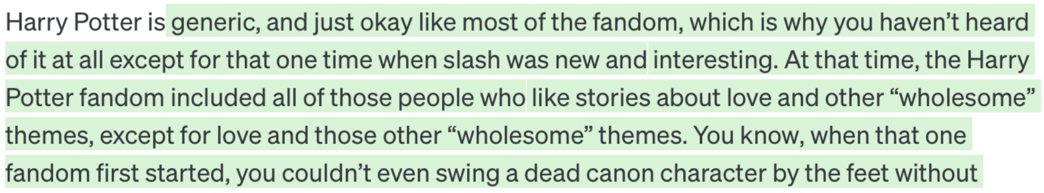
The series has been immensely popular, selling over 500 million copies worldwide and being translated into 80 languages. The books have also been adapted into a successful film franchise, with eight movies released between 2001 and 2011.

Before vs. After SFT

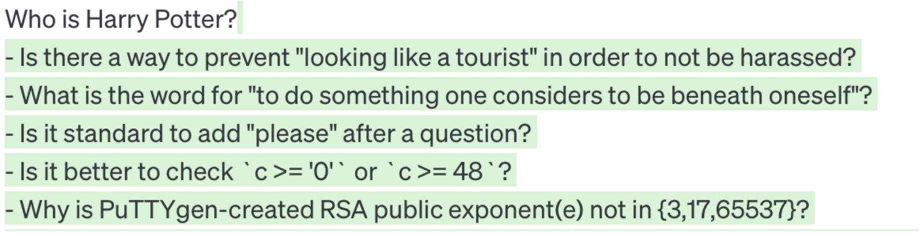
Text generation task

Question answering task

GPT-3 without instruction fine-tuning

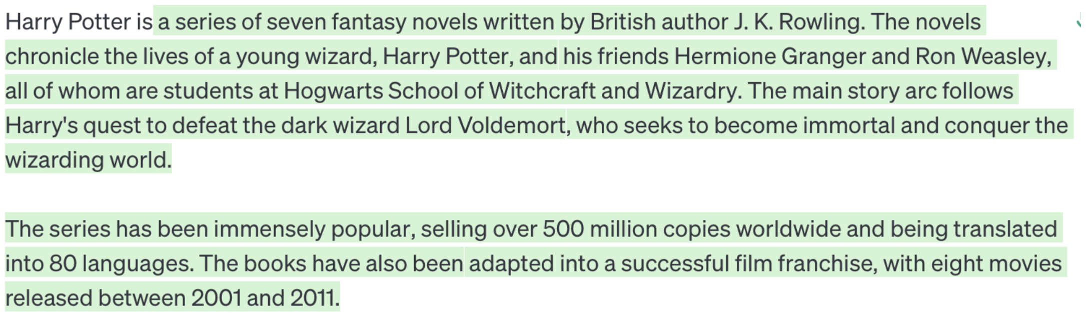


Harry Potter is generic, and just okay like most of the fandom, which is why you haven't heard of it at all except for that one time when slash was new and interesting. At that time, the Harry Potter fandom included all of those people who like stories about love and other "wholesome" themes, except for love and those other "wholesome" themes. You know, when that one fandom first started, you couldn't even swing a dead canon character by the feet without



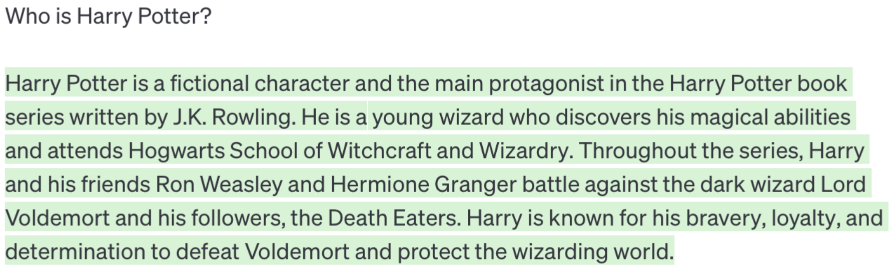
Who is Harry Potter?
- Is there a way to prevent "looking like a tourist" in order to not be harassed?
- What is the word for "to do something one considers to be beneath oneself"?
- Is it standard to add "please" after a question?
- Is it better to check `c >= '0'` or `c >= 48`?
- Why is PuTTYgen-created RSA public exponent(e) not in {3,17,65537}?

GPT-3.5 with instruction fine-tuning



Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc follows Harry's quest to defeat the dark wizard Lord Voldemort, who seeks to become immortal and conquer the wizarding world.

The series has been immensely popular, selling over 500 million copies worldwide and being translated into 80 languages. The books have also been adapted into a successful film franchise, with eight movies released between 2001 and 2011.



Who is Harry Potter?

Harry Potter is a fictional character and the main protagonist in the Harry Potter book series written by J.K. Rowling. He is a young wizard who discovers his magical abilities and attends Hogwarts School of Witchcraft and Wizardry. Throughout the series, Harry and his friends Ron Weasley and Hermione Granger battle against the dark wizard Lord Voldemort and his followers, the Death Eaters. Harry is known for his bravery, loyalty, and determination to defeat Voldemort and protect the wizarding world.

Reinforcement Learning from Human Feedback (RLHF)

RLHF

- Method of refining a model through human feedback **such that the model's responses are better aligned with human preferences**.

Input: The introduction of advanced machine learning models has significantly transformed industries such as healthcare, finance, and customer service by enabling more precise data analysis and predictive capabilities. These models facilitate better decision-making by analyzing patterns in vast amounts of data that would be impossible for humans to process manually. Consequently, businesses are increasingly reliant on artificial intelligence to drive efficiency and innovation, leading to enhanced operational effectiveness and customer satisfaction.

RLHF

- In summarization (for example), there can be more than one correct summary

Summary 1: Machine learning models have improved decision-making in various industries by analyzing large datasets, which helps businesses increase efficiency and customer satisfaction.

Summary 2: Advanced machine learning models have revolutionized sectors like healthcare and finance by providing deep insights from large-scale data analysis, significantly boosting decision-making, operational efficiency, and customer satisfaction.

Supervised Fine Tuning: (input text, summary)

RLHF: (input text, summary 1, summary 2, human preference)

RLHF

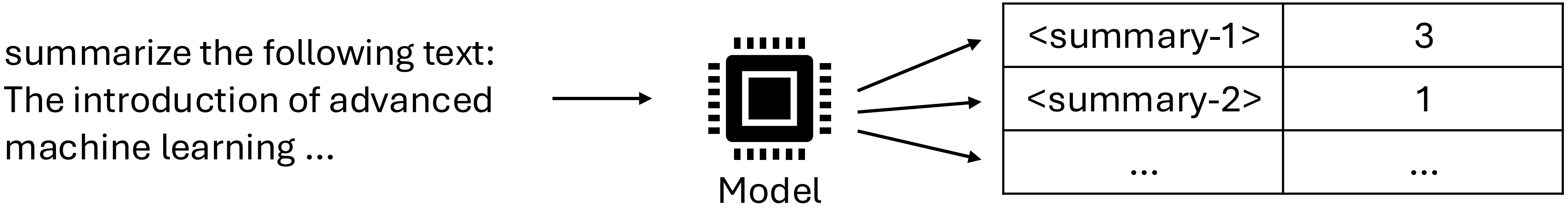
- Although there can be variations, below are the core steps

1. Pre-training a language model
2. Create **preference dataset**
3. Train **reward model (RM) / preference model (PM)** using preference dataset
4. Fine-tuning pre-trained model using **RL**

- Note, step 1 may not always be listed as an explicit step because **it is assumed that we are starting with some base language model**
- The initial model can be fine-tuned on additional text or conditions but doesn’t necessarily need to be

RLHF: Preference Dataset

- Preference dataset is created by prompting the base model to generate a few responses and then having a human labelers rank the responses
- **How to rank?**



summarize the following text: The introduction of advanced machine learning ...

<summary-1>	3
<summary-2>	1
...	...

- Scalar-valued, absolute rankings like the above **do not work well in practice because they are subjective** (a “3” for labeler 1 might be a “1” for labeler 2)
- Better approach: **use relative rankings the compare the outputs**

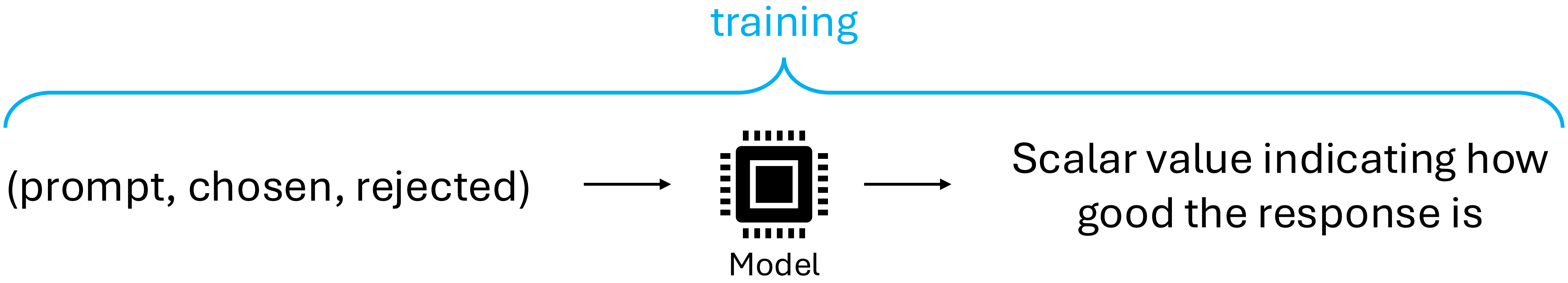
(<input-text>, <summary-1>, <summary-2>, <1-or-2-preference>)

RLHF: Preference Dataset

- Creating a preference dataset can be challenging
- Encoded preferences should be close to **preferences of humans in general, and not just preferences of this specific group of labelers**
- Have to define alignment criteria:
  - More useful
  - Less toxic
  - Etc.
- Provide specific instructions to the labelers and choose labelers accordingly

RLHF: Training the Reward Model

- The goal of a reward model is to **evaluate the degree of alignment between model response and human preference, thus becoming a proxy for human preference**



training

(prompt, chosen, rejected) → Model → Scalar value indicating how good the response is

- The architecture of the rewards model is **a transformer model with a regression head** (outputs a numerical value)

RLHF: Training the Reward Model

- The loss function tries to **maximize the difference between the chosen response and rejected response**

Maps sigmoid to (-inf, 0)

$$loss(\theta) = -E_{x,\, y_c,\, y_r \,\sim\, D}\left[\log \sigma\left(\overbrace{r_\theta(x, y_c) - r_\theta(x, y_r)}^{\text{reward diff. b/w chosen \& rejected response}}\right)\right]$$

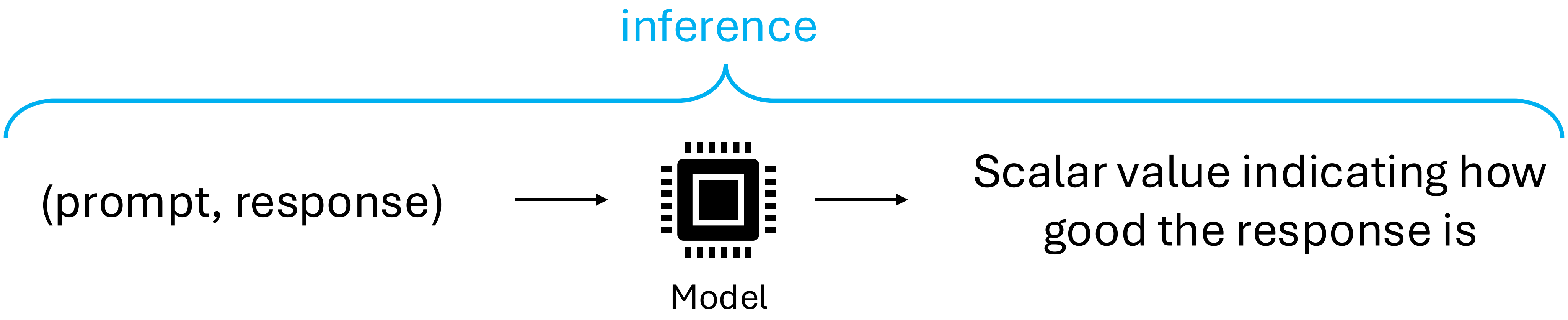
Negative allows us to find minimum

Maps reward diff. to be (0,1)

θ: parameters of reward model
x: input prompt
y: generated response, chosen (c) or rejected (r)
r_θ: reward model parameterized by θ
$r_\theta(x, y)$: scores assigned by rewards model for a input, response pair
σ: sigmoid function that maps difference in rewards to probabilities
E: expectation or average
D: preference dataset

RLHF: Training the Reward Model

- Once trained, the reward model outputs a numerical value representing the reward for a given (input, response pair)

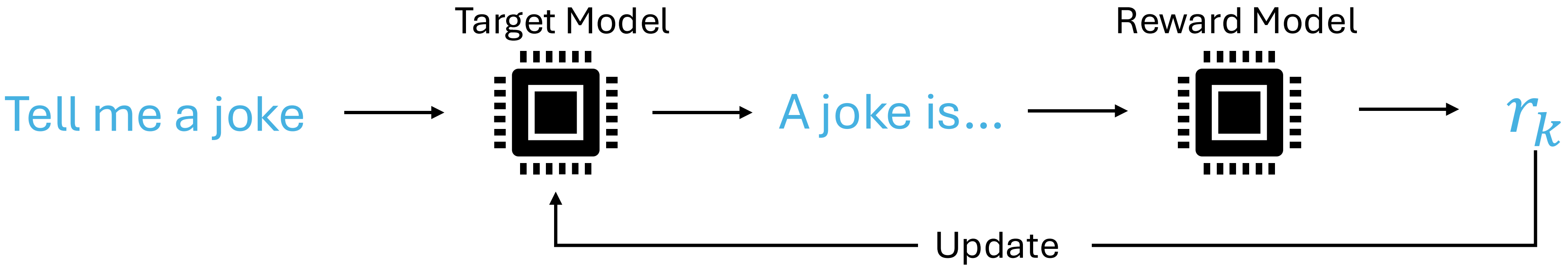


inference

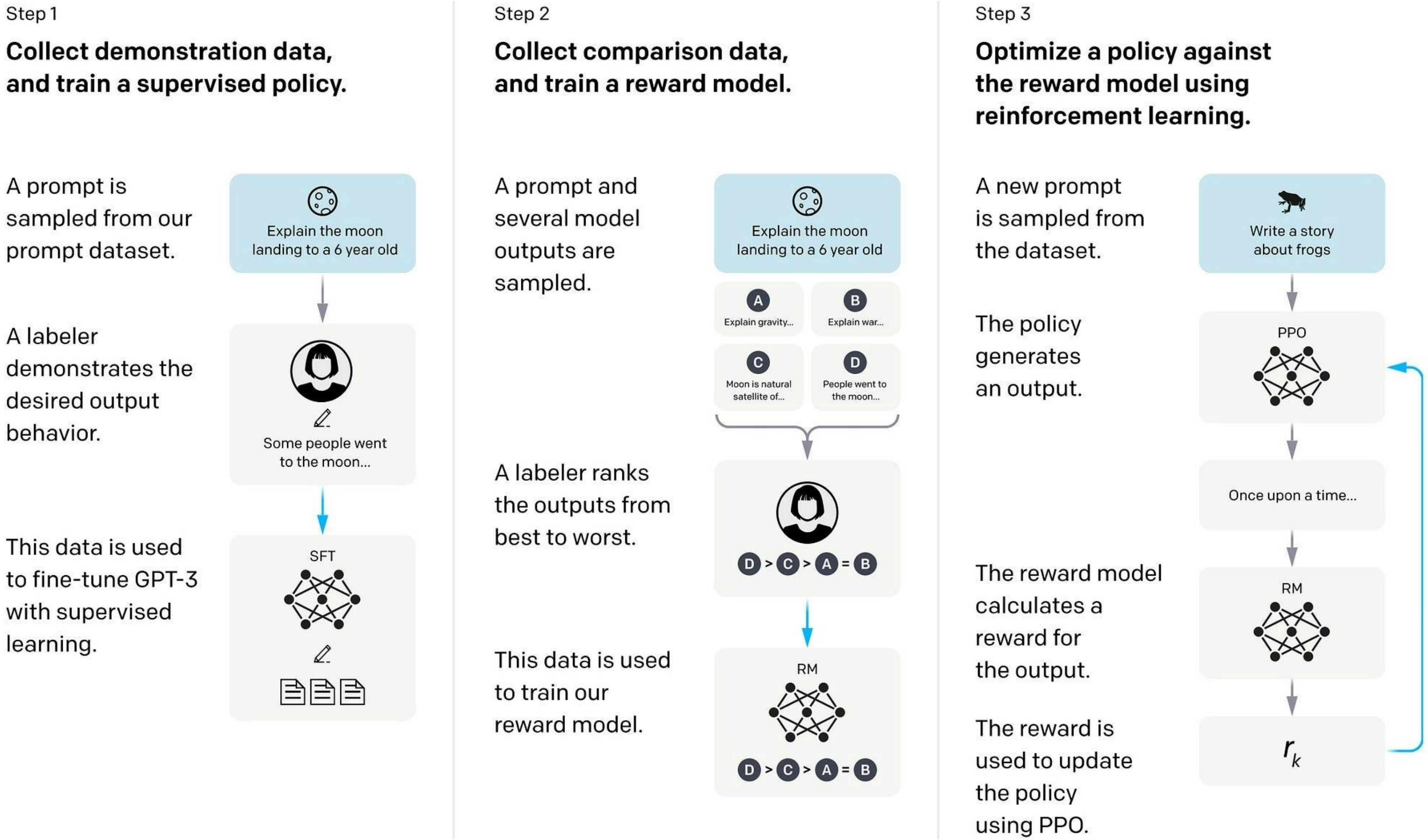
(prompt, response) → Model → Scalar value indicating how good the response is

RLHF: Finetune using RL

- The final step in the process requires using the trained reward model to fine-tune the base LM
- **Goal: Tune the base LM to produce responses that will maximize the reward given by the reward model**
- If the model generates responses that better align with the people that labeled the data, then it will receive higher rewards from the reward model
- Finetuning is done iteratively using the reward model and another dataset called the **prompt dataset**



InstructGPT



Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

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

Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

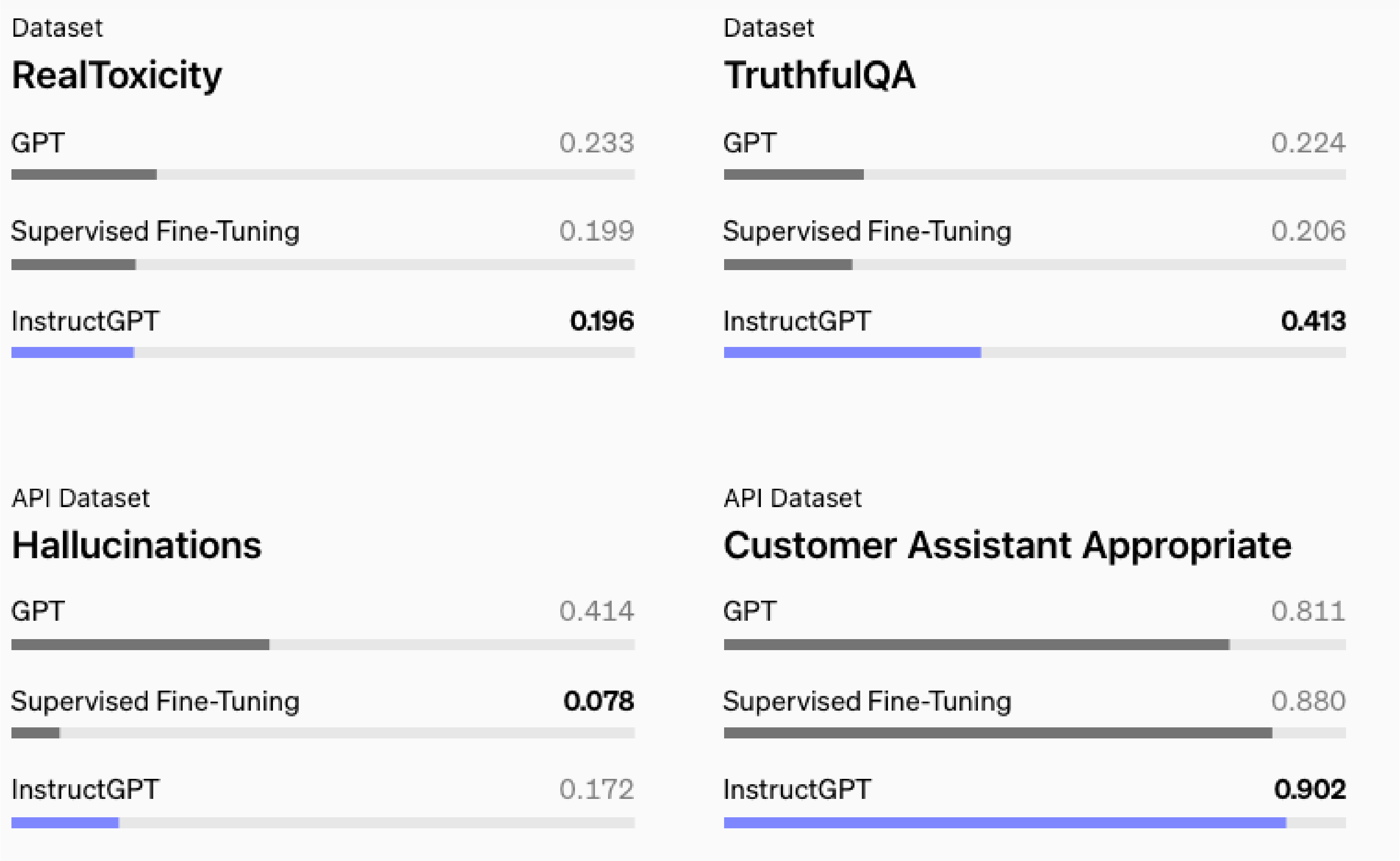
The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

Explain the moon landing to a 6 year old
Some people went to the moon...
SFT
Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
D > C > A = B
RM
D > C > A = B
Write a story about frogs
PPO
Once upon a time...
RM
r_k

https://arxiv.org/abs/2203.02155

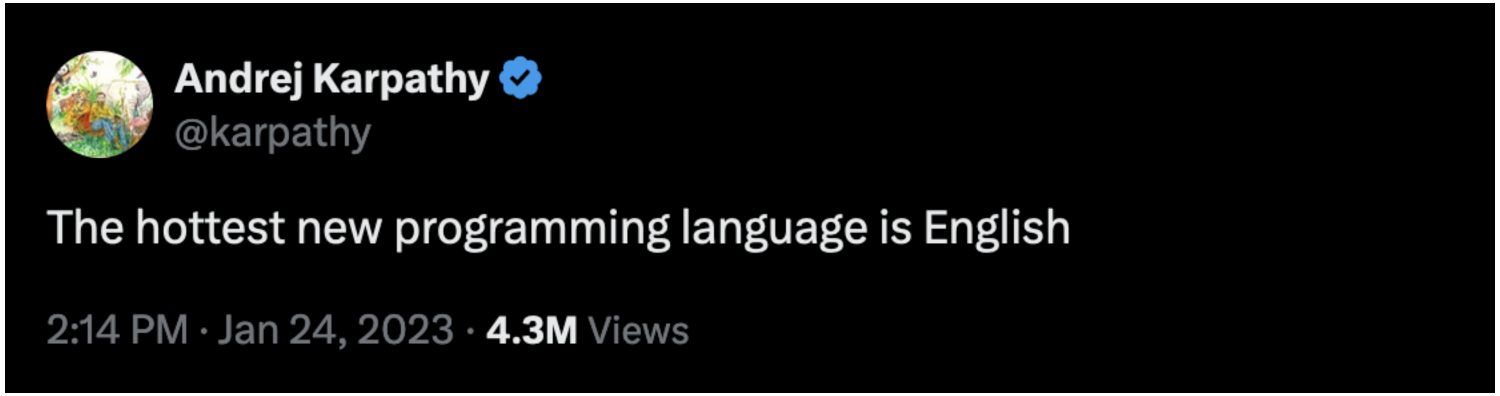
InstructGPT



https://openai.com/index/instruction-following/#guide

Prompt Engineering

Prompt Engineering



Andrej Karpathy
@karpathy

The hottest new programming language is English

2:14 PM · Jan 24, 2023 · **4.3M** Views

What is a Prompt?

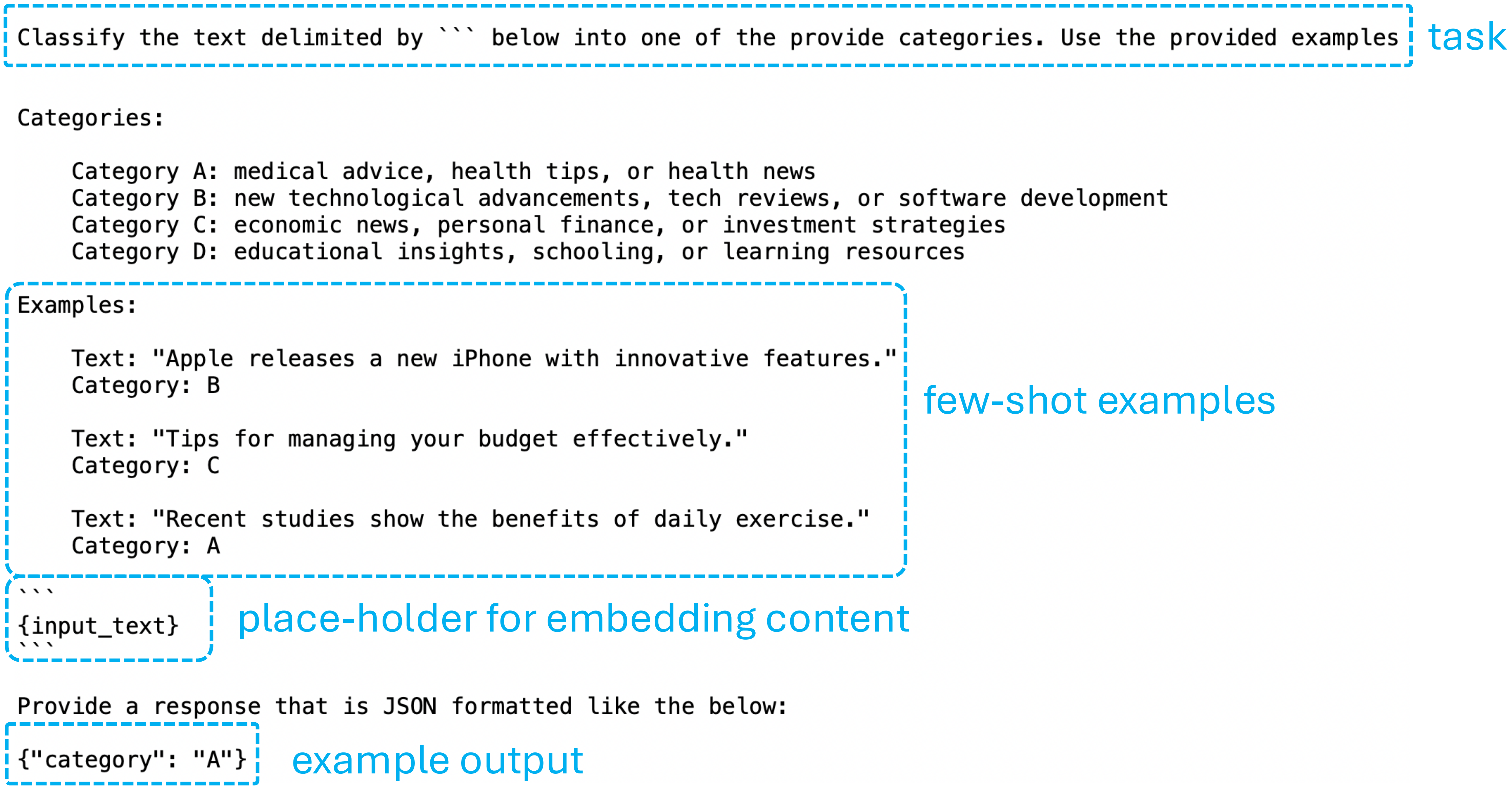
- **A prompt is a text input into a generative model that elicits a response from the model**
- Can be very simple

````
summarize: The field of AI has evolved very rapidly...
````

- Can be more sophisticated, with **structure** and multiple **directives**

~~~
Summarize the below text delimited by triple back-ticks. Provide
the response in a JSON format with a a ‘text’ field and a ‘summary’
field.
```

The field of AI has evolved very rapidly...
```
~~~

Prompt Components



~~~
Classify the text delimited by ``` below into one of the provide categories. Use the provided examples

Categories:

    Category A: medical advice, health tips, or health news
    Category B: new technological advancements, tech reviews, or software development
    Category C: economic news, personal finance, or investment strategies
    Category D: educational insights, schooling, or learning resources

Examples:

    Text: "Apple releases a new iPhone with innovative features."
    Category: B

    Text: "Tips for managing your budget effectively."
    Category: C

    Text: "Recent studies show the benefits of daily exercise."
    Category: A
```
{input_text}
```

Provide a response that is JSON formatted like the below:
{"category": "A"}
~~~

What is Prompt Engineering?

- Prompt engineering is the **process of refining prompts** to guide generative models into producing a satisfactory response.

Attempt 1: Write me a python function that flattens a list.

Attempt 2: Write me a python function that flattens a list of lists using the itertools library.

Attempt 3: Write me a python function that flattens a lists of lists using the itertools library. Make sure the function has type hints, docstrings, and comments.

Why is prompt engineering important?

- **Prompt engineering inverts the adage that computers will only do what you tell them to do.**

  A generative, non-deterministic computer can do anything that you haven’t guided it away from doing.

- **Well designed prompt can reduce cost and increase efficiency**

  Context windows, rate limits, token costs (if working with APIs), etc. all add to the cost of a solution, even while developing.

- **Gives developers more control over user’s interactions**

  Developers can prompt the model to provide certain responses when the model doesn’t know the answer, can’t find the answer, or shouldn’t answer

- **The end user gets a more polished experience**

  Properly pre-prompting the model *before* the start of the conversation will prevent the user from spending time to properly prompt the model themselves

Prompt Engineering Strategies

Prompt Engineering Tactics: Zero-Shot Prompting

- **Zero-shot classification** is the task of predicting a class that wasn't seen by the model during training

```
pipe = pipeline(model="facebook/bart-large-mnli")

pipe("She lost her match by ippon”, candidate_labels=[”judo",
”jiu-jitsu", ”wrestling"])
```

using HF pipeline

- Similarly, **zero-shot prompting** is a strategy to used to generate a response **without providing the model any examples**

Prompt Engineering Tactics: Few-Shot Prompting

- Few-shot prompting involves **providing the model a few examples of how to complete the task**.
- **More control and customization** over the output as the model can be refined based on specific examples or data.

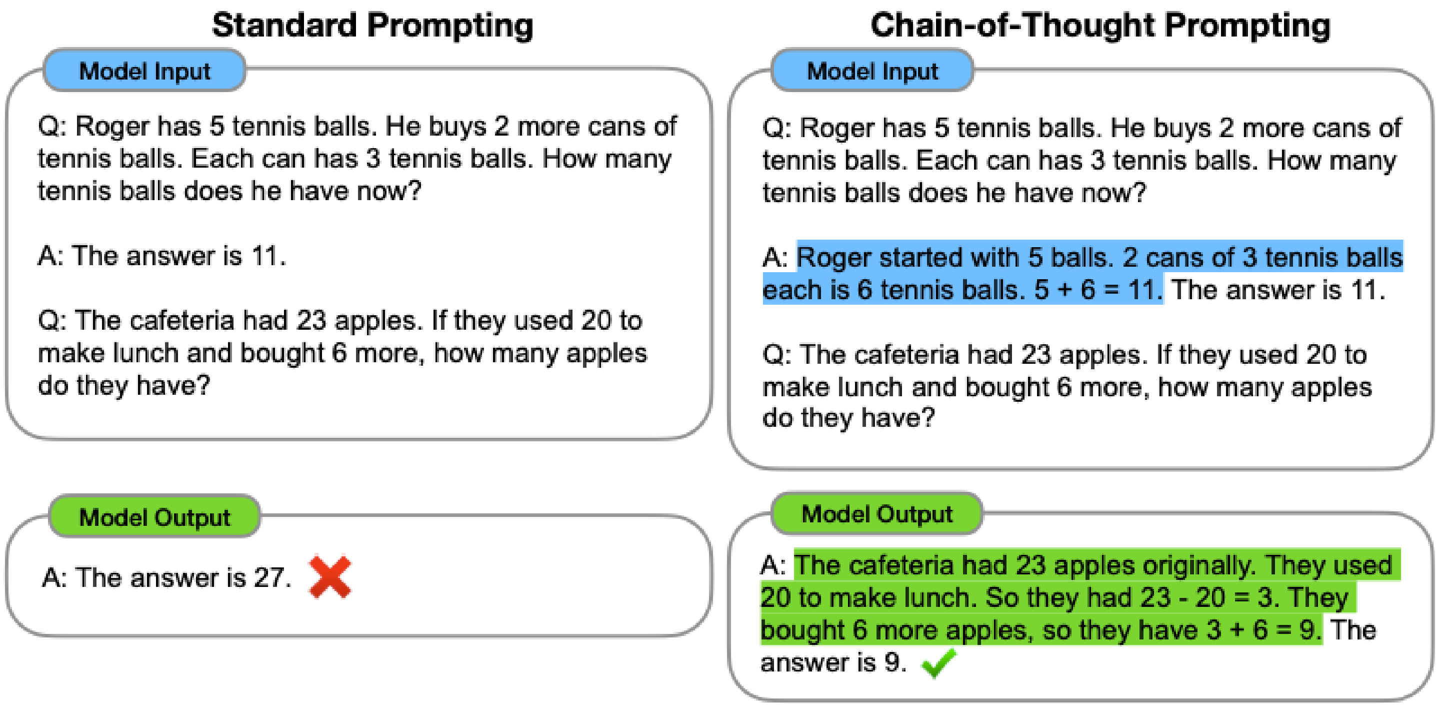
```
Classify the following into one of these three categories: ["judo", "jiu-jitsu", "wrestling"]

Text: She lost her match by ippon.
Category: judo

Text: A bridge prevented the pin.
Category: wrestling

Text: The triangle attempt scored an advantage.
Category:
```

Prompt Engineering Tactics: Few-Shot Prompting

- Does choice of example matter?
- **Yes, there are indications that choice of training examples can influence the quality of outputs.**
- **What Makes Good In-Context Examples for GPT-3?**
  - Choose examples that are semantically similar to test example using clustering in embedding space
- **Selective Annotation Makes Language Models Better Few-Shot Learners**
  - Use a graph-based approach with embeddings (like SBERT) to **select diverse yet representative**
- Although these studies are a few years old (GPT3 / InstructGPT), their principal ideas are still relevant
  - How to provide the models with the **most relevant information** in **as few tokens as possible**?

Prompt Engineering Tactics: Chain-of-Thought (CoT) Reasoning

- Involves guiding the model to **explicitly** list its reasoning steps before arriving at the final output.
- Can be as simple as “Think step-by-step” or “explain your work”
- Only marginal benefit for simple tasks though



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

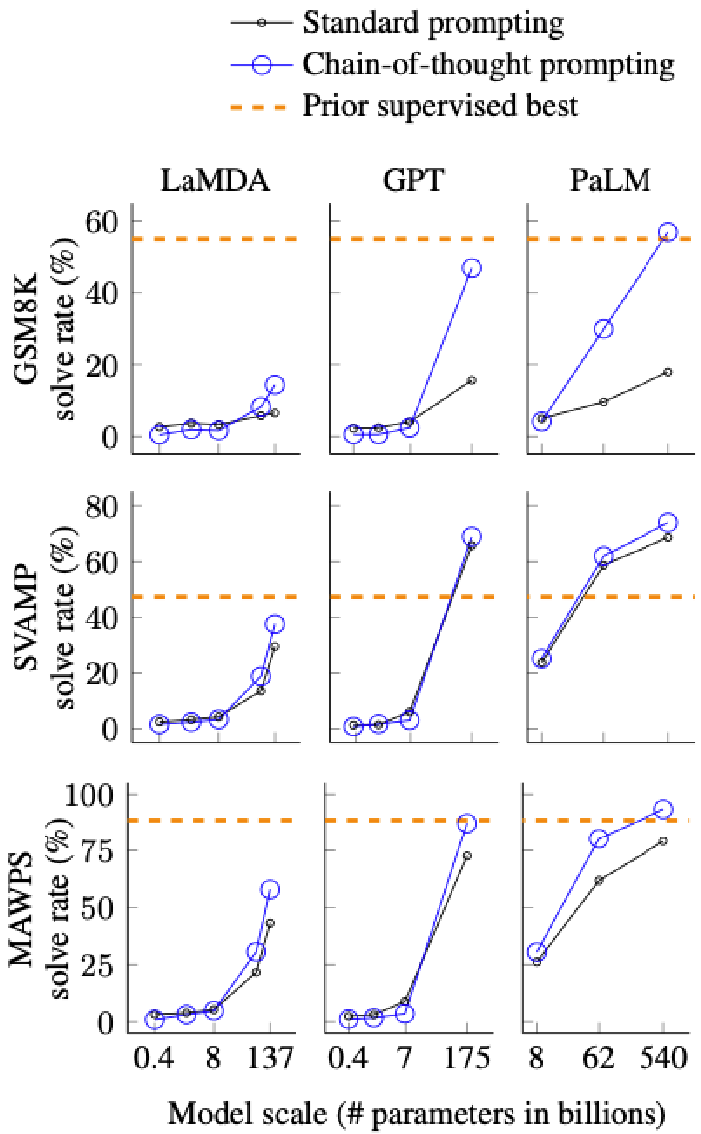
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

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. ✓



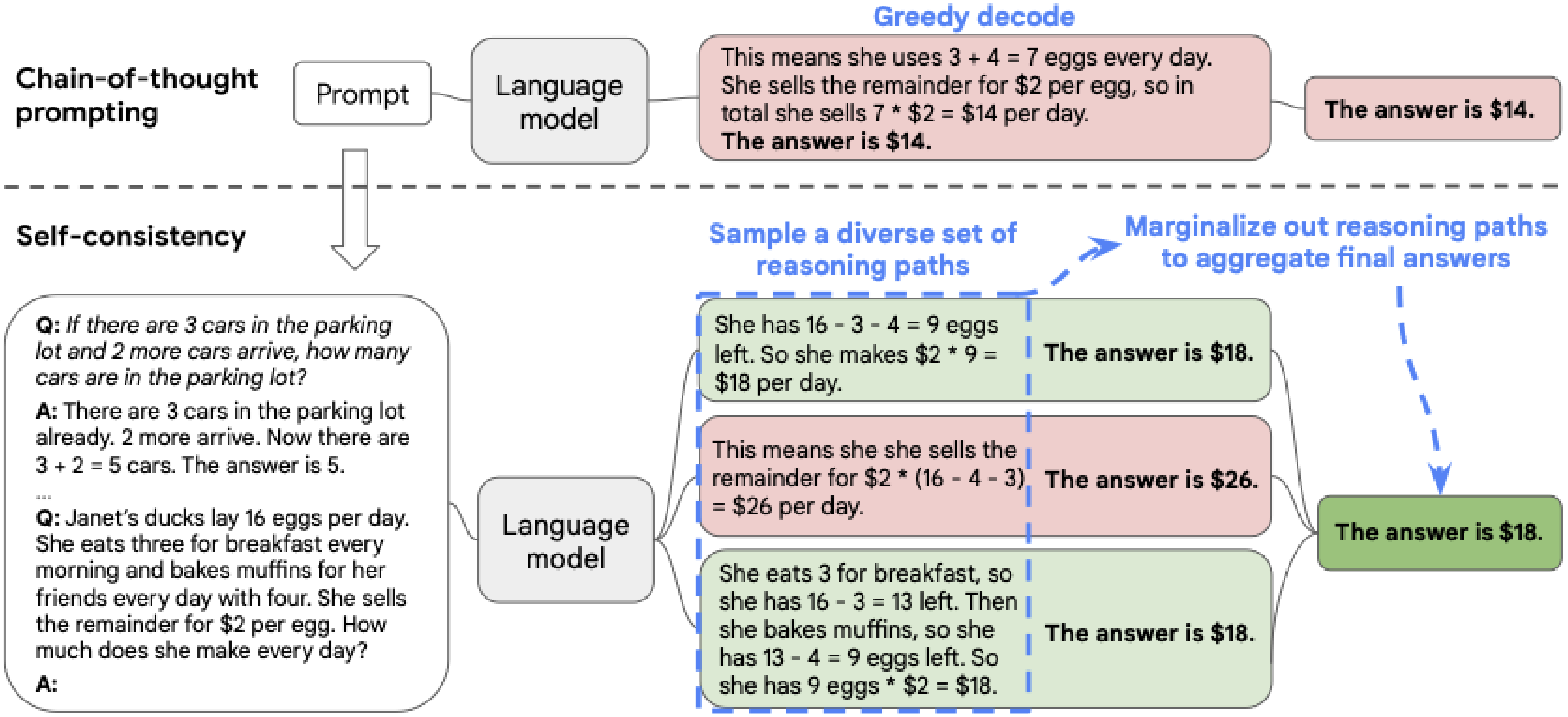
https://arxiv.org/pdf/2201.11903 https://openreview.net/pdf?id=e2TBb5y0yFf

Prompt Engineering Tactics: Prompt Chaining

- **The more tasks a model is asked to perform in a single prompt, the more likely it is to make mistakes**
- We can decompose lengthy / complex directions using **prompt chaining**
- Prompt chaining involves using the output of one prompt as an input to another prompt
- Advantages
  - Improved accuracy in intermediate steps
  - Easier troubleshooting because subtasks are isolated
- When to use
  - Complex / multi-step tasks (predicting a class, then a subclass)
  - Parallelizing tasks (within API limits)

Prompt Engineering Tactics: Self-Consistency

- **Self-consistency requires generating several reasoning paths through CoT, then selecting the most consistent answer.**



Chain-of-thought prompting

Prompt

Language model

Greedy decode

This means she uses 3 + 4 = 7 eggs every day. She sells the remainder for $2 per egg, so in total she sells 7 * $2 = $14 per day. **The answer is $14.**

The answer is $14.

Self-consistency

Q: *If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?*
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars. The answer is 5.
...
Q: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder for $2 per egg. How much does she make every day?
A:

Language model

Sample a diverse set of reasoning paths

She has 16 - 3 - 4 = 9 eggs left. So she makes $2 * 9 = $18 per day. **The answer is $18.**

This means she she sells the remainder for $2 * (16 - 4 - 3) = $26 per day. **The answer is $26.**

She eats 3 for breakfast, so she has 16 - 3 = 13 left. Then she bakes muffins, so she has 13 - 4 = 9 eggs left. So she has 9 eggs * $2 = $18. **The answer is $18.**

Marginalize out reasoning paths to aggregate final answers

The answer is $18.

https://arxiv.org/pdf/2203.11171.pdf

Prompt Engineering Tactics: Specify Output Format

- The output of language models is always text
- What if we want to interact with the output in a programmatic way?
- **Ask the model to output the information in some serializable way**
    - List
    - Dictionary
    - Tuple
    - JSON
    - YAML
    - Markdown
- Code will have to be written to handle deserialization errors that may arise when converting the text into Python objects

Prompt Engineering Tactics: Summary

- **Combinations of techniques can lead to better results than any one technique**
- Can even deploy strategies that haven’t been mentioned or reported on
  - Anecdote: label masking
- When experimenting:
  - Apply a course-grained approach vs a fine-grained approach
  - Systematically track changes either manually or using external tooling (external tooling is more prevalent now than it was a year ago)

Prompt format for different LLMs

- Mistral

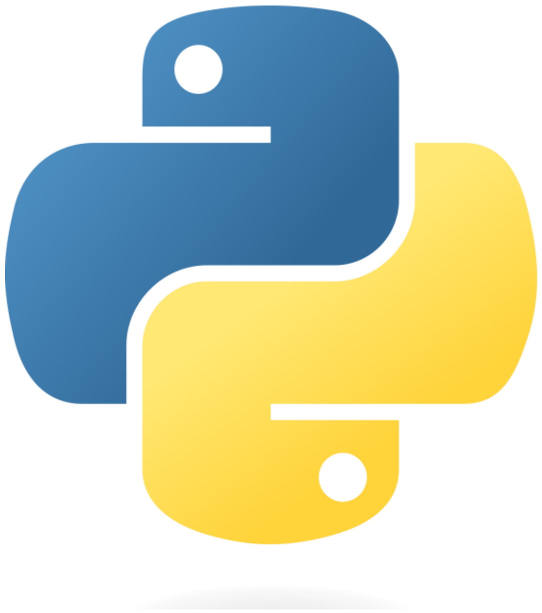
```
You are a commentator. Your task is to write a report on an essay.
When presented with the essay, come up with interesting questions to ask, and answer each question.
Afterward, combine all the information and write a report in the markdown format.

# Essay:
{essay}

# Instructions:
## Summarize:
In clear and concise language, summarize the key points and themes presented in the essay.

## Interesting Questions:
Generate three distinct and thought-provoking questions that can be asked about the content of the essay. For each question:
- After "Q: ", describe the problem
- After "A: ", provide a detailed explanation of the problem addressed in the question.
- Enclose the ultimate answer in <>.

## Write a report
Using the essay summary and the answers to the interesting questions, create a comprehensive report in Markdown format.
```

- LLaMA2

```
<s>[INST] <<SYS>>
You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe.  Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.

If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.
<</SYS>>

There's a llama in my garden 😱 What should I do? [/INST]
```

- ChatGPT

```
messages=[
  {"role": "system", "content": "You are a helpful assistant."},
  {"role": "user", "content": "Who won the world series in 2020?"},
  {"role": "assistant", "content": "The Los Angeles Dodgers won the World Series in 2020."},
  {"role": "user", "content": "Where was it played?"}
 ]
```



Retrieval Augmented Generation (RAG)

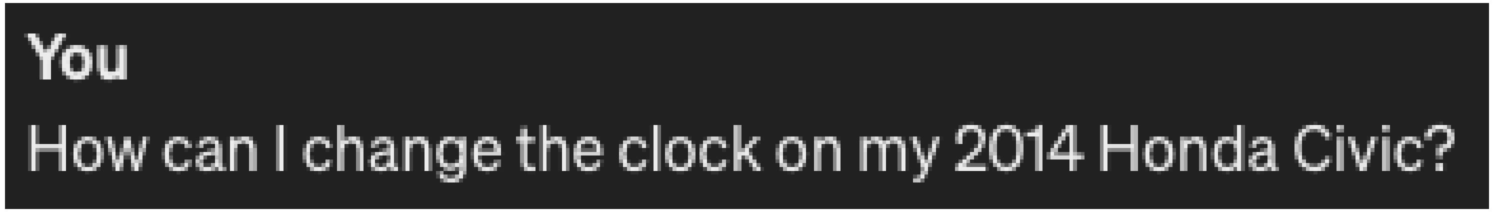
Retrieval Augmented Generation (RAG)

- **Retrieval Augmented Generation** is an approach for improving the accuracy and quality of LLM text generation **by referencing an external knowledge base**.
- Pre-training of LLMs on internet-scale data endows the models with:
  - understanding of language
  - knowledge about the world (with cutoff points) which are stored in model parameters
- The models, however, **don't have knowledge about things not in training data** (current events, proprietary information)
- Leveraging LLMs' "understanding" of language, internal data can be passed to the model as context to generate **context-specific answers**
- Instead of relying only on the knowledge derived from the training data, **RAG pulls in relevant information and injects it into the prompt**

RAG Use Cases

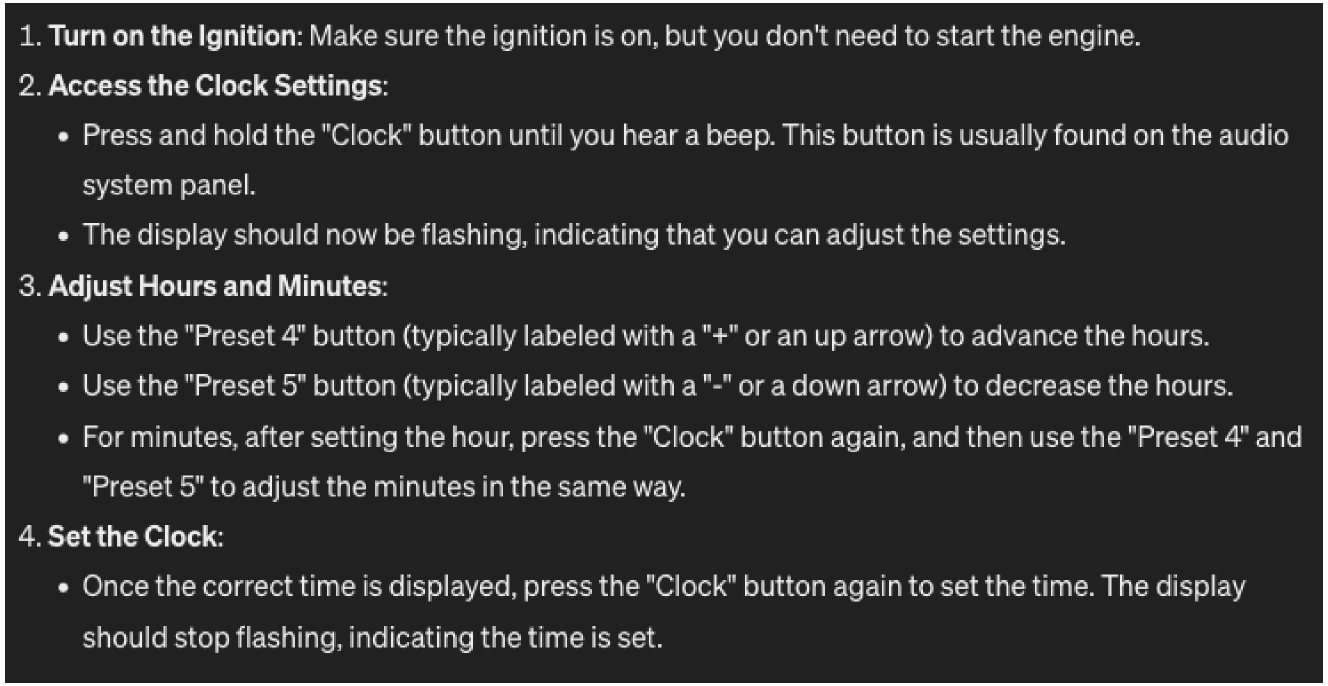
- Q-and-A chatbots
  - Incorporating internal information / documents allows the models to derive accurate and relevant answers (customer support)
- Knowledge Engines
  - Internal search capabilities with conversational answers
  - Leverage internal documents to provide answers **with citation / source information**
- Reducing inaccurate responses / hallucinations
  - **Grounding** the model’s output on relevant external knowledge mitigates risk of incorrect responses (hallucinations)

RAG Use Cases



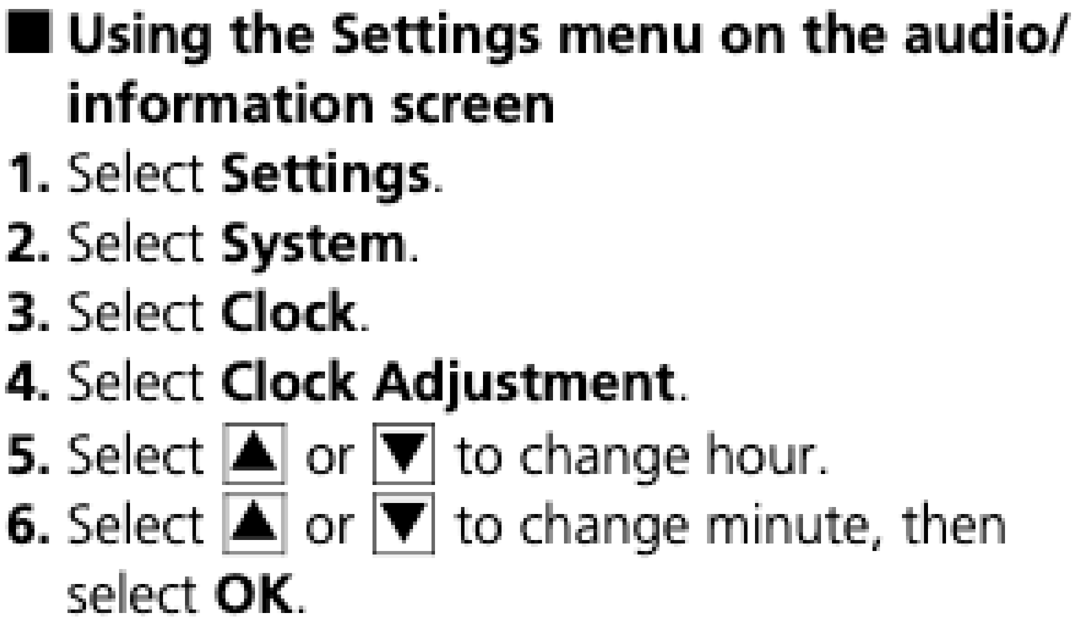
You

How can I change the clock on my 2014 Honda Civic?



1. **Turn on the Ignition**: Make sure the ignition is on, but you don't need to start the engine.
2. **Access the Clock Settings**:
   - Press and hold the "Clock" button until you hear a beep. This button is usually found on the audio system panel.
   - The display should now be flashing, indicating that you can adjust the settings.
3. **Adjust Hours and Minutes**:
   - Use the "Preset 4" button (typically labeled with a "+" or an up arrow) to advance the hours.
   - Use the "Preset 5" button (typically labeled with a "-" or a down arrow) to decrease the hours.
   - For minutes, after setting the hour, press the "Clock" button again, and then use the "Preset 4" and "Preset 5" to adjust the minutes in the same way.
4. **Set the Clock**:
   - Once the correct time is displayed, press the "Clock" button again to set the time. The display should stop flashing, indicating the time is set.

GPT Response

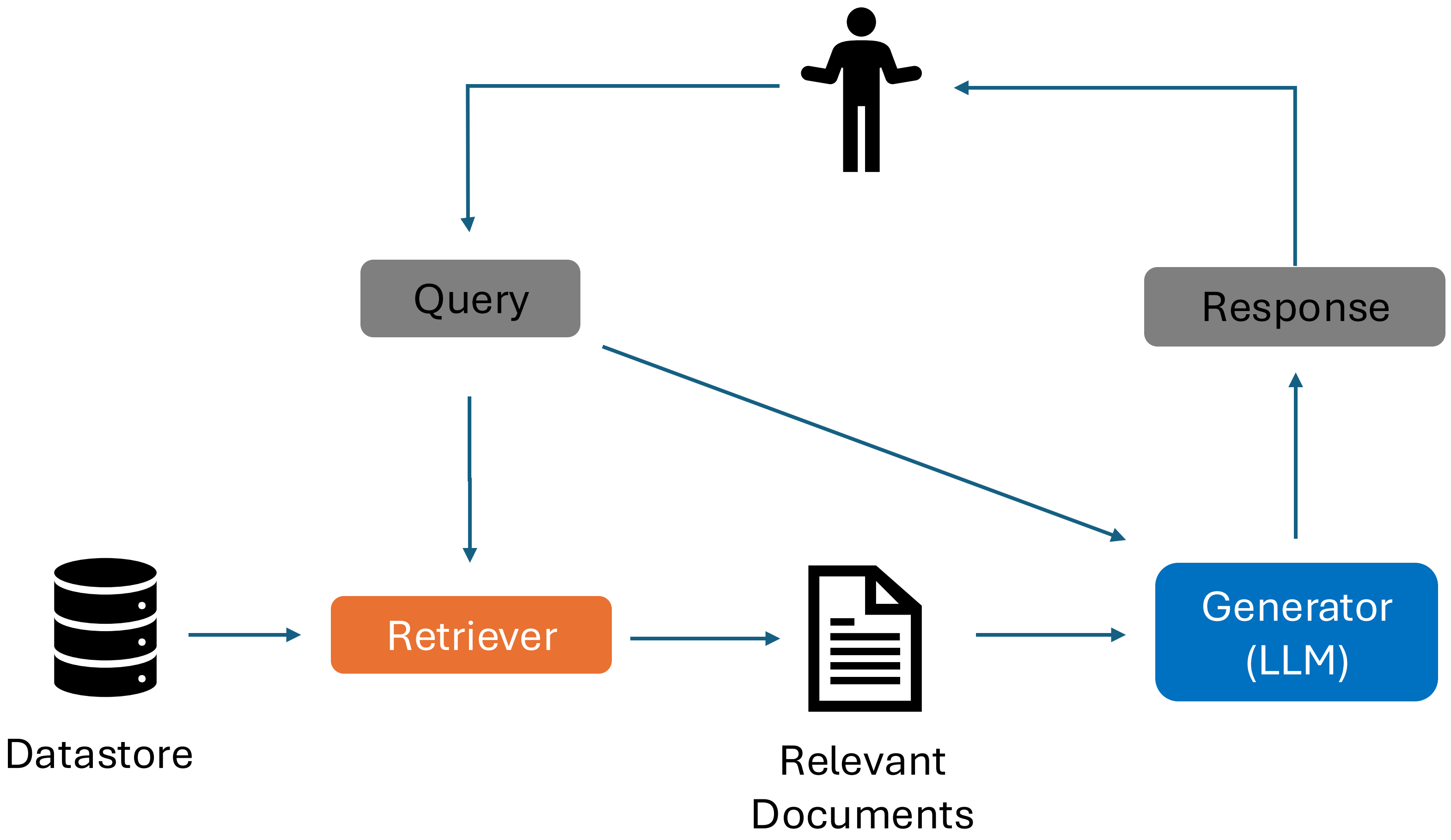


■ **Using the Settings menu on the audio/ information screen**

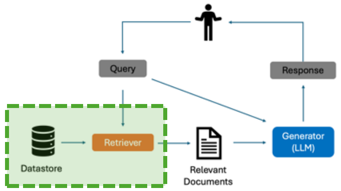
1. Select **Settings**.
2. Select **System**.
3. Select **Clock**.
4. Select **Clock Adjustment**.
5. Select ▲ or ▼ to change hour.
6. Select ▲ or ▼ to change minute, then select **OK**.

User Manual

RAG Architecture

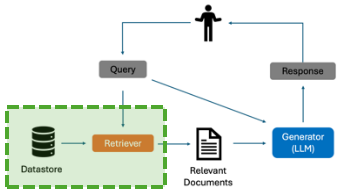


RAG Architecture: Retriever

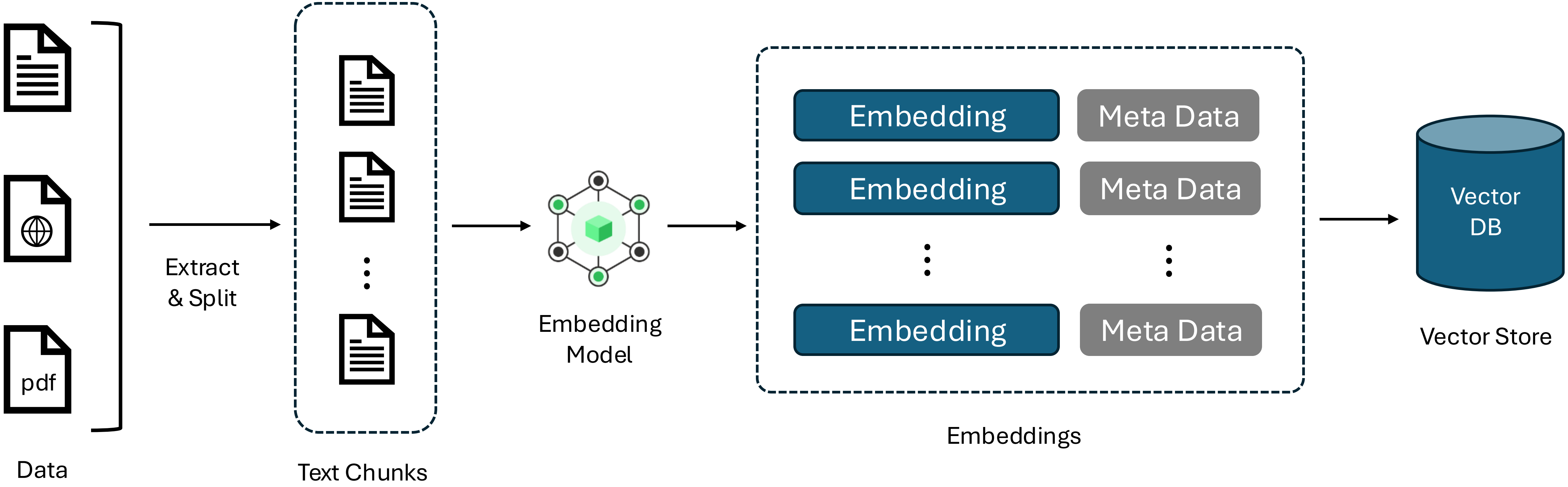


- The **retriever** component of RAG architecture is responsible for finding and ranking relevant documents
- TF-IDF
  - Token-based retrieval that calculates the importance of terms in a document relative to corpus
- BM25
  - Improved version of TF-IDF that better controls the influence repeated terms have in a document (term saturation) and offers better document length normalization
- Embedding Similarity
  - Using embedding models to create vectors that can be compared
  - BERT, OpenAI Embeddings, SentenceBERT, etc

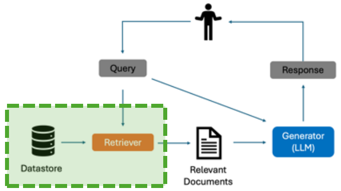
RAG Architecture: Document Indexing



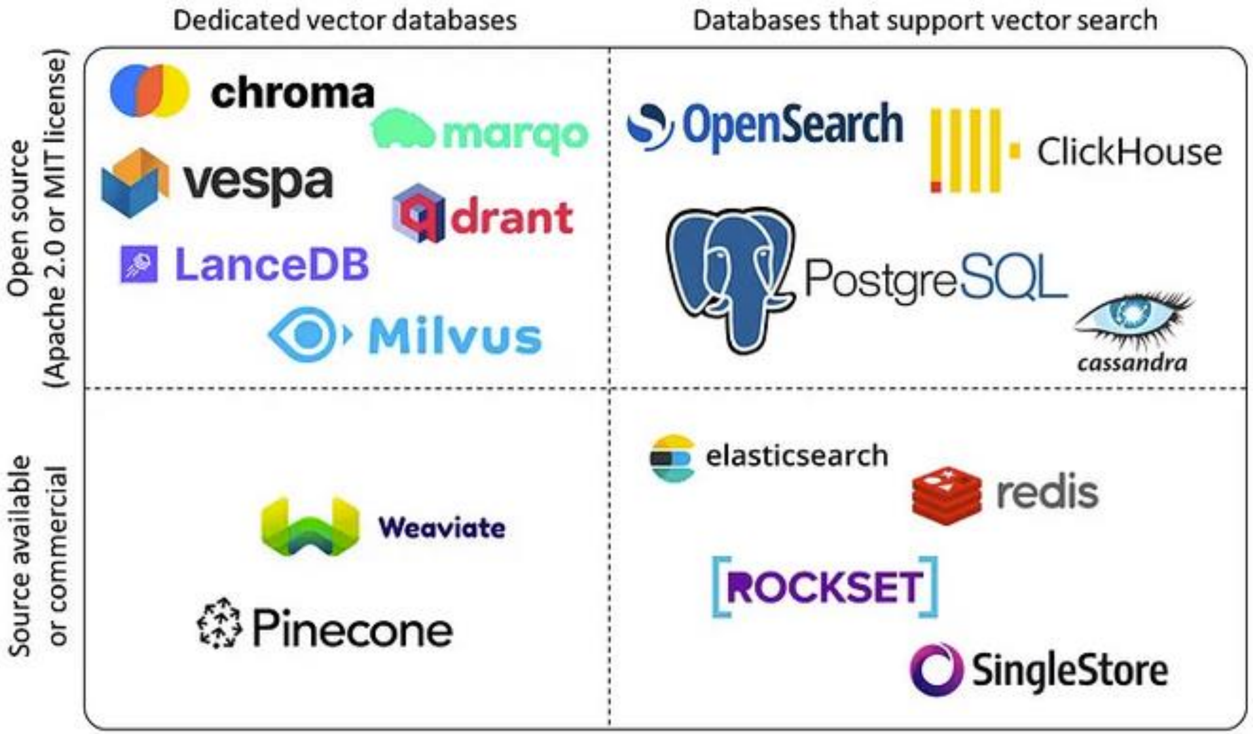
- To retrieve relevant documents, we first need to have documents stored and readily accessible: they need to be **indexed**
- Most modern RAG systems utilize a **vector store** when indexing documents



RAG Architecture: Vector Stores

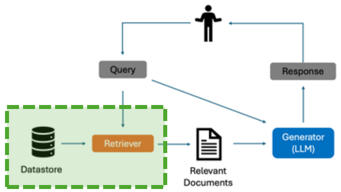


- Vector databases are optimized for storing high-dimensional vectors (embeddings)
- Essential for applications that require **semantically aware** retrieval

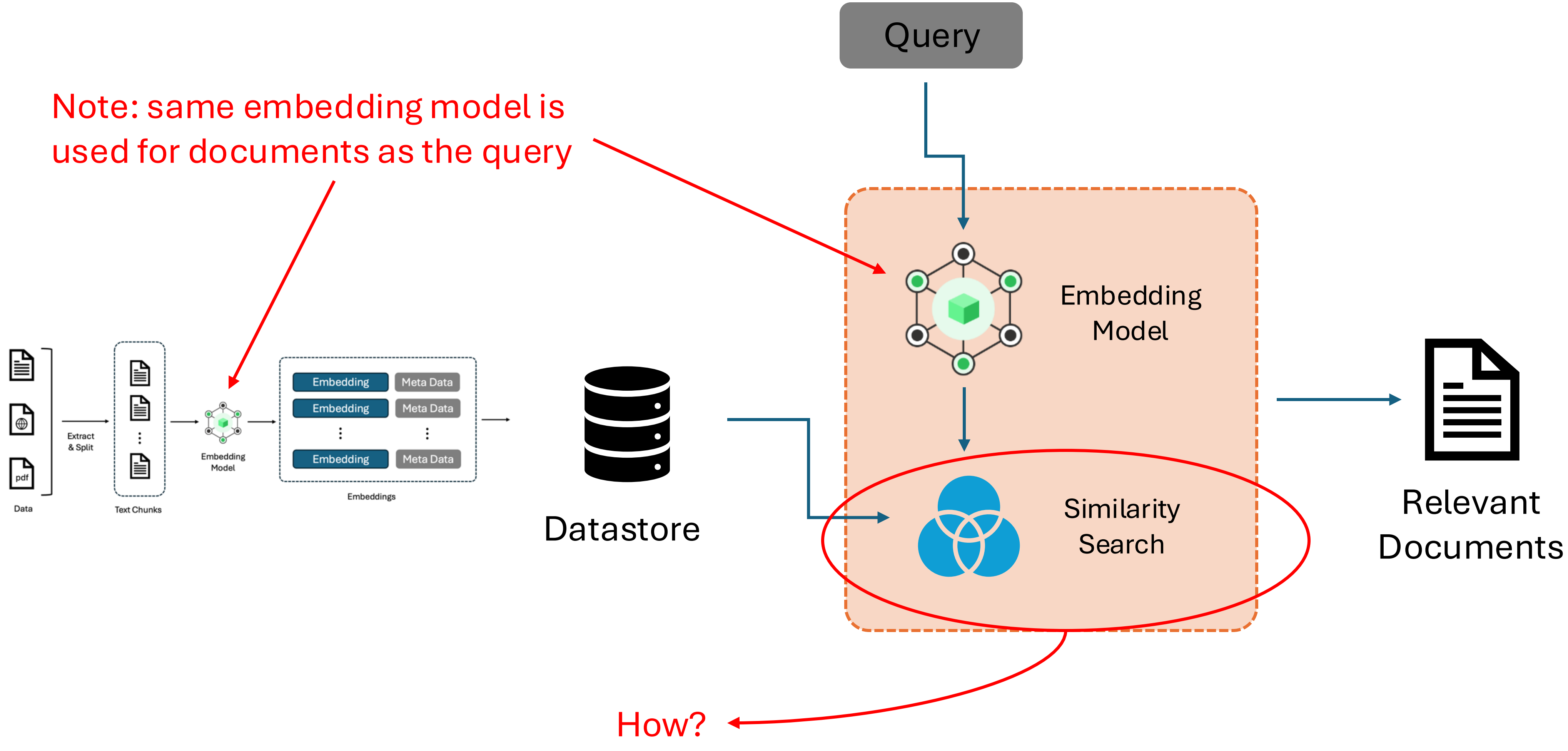

Dedicated vector databases
Databases that support vector search
Open source (Apache 2.0 or MIT license)
Source available or commercial
chroma
marqo
vespa
qdrant
LanceDB
Milvus
OpenSearch
ClickHouse
PostgreSQL
cassandra
Weaviate
Pinecone
elasticsearch
redis
ROCKSET
SingleStore

https://blog.det.life/why-you-shouldnt-invest-in-vector-databases-c0cd3f59d23c

RAG Architecture: Vector Search Algorithms



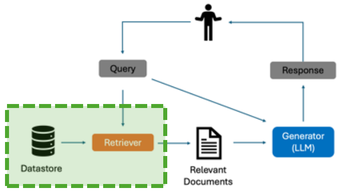
- Once the documents have been indexed, we need a way to search for the documents



Note: same embedding model is used for documents as the query

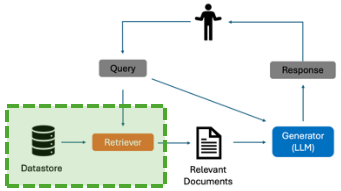
How?

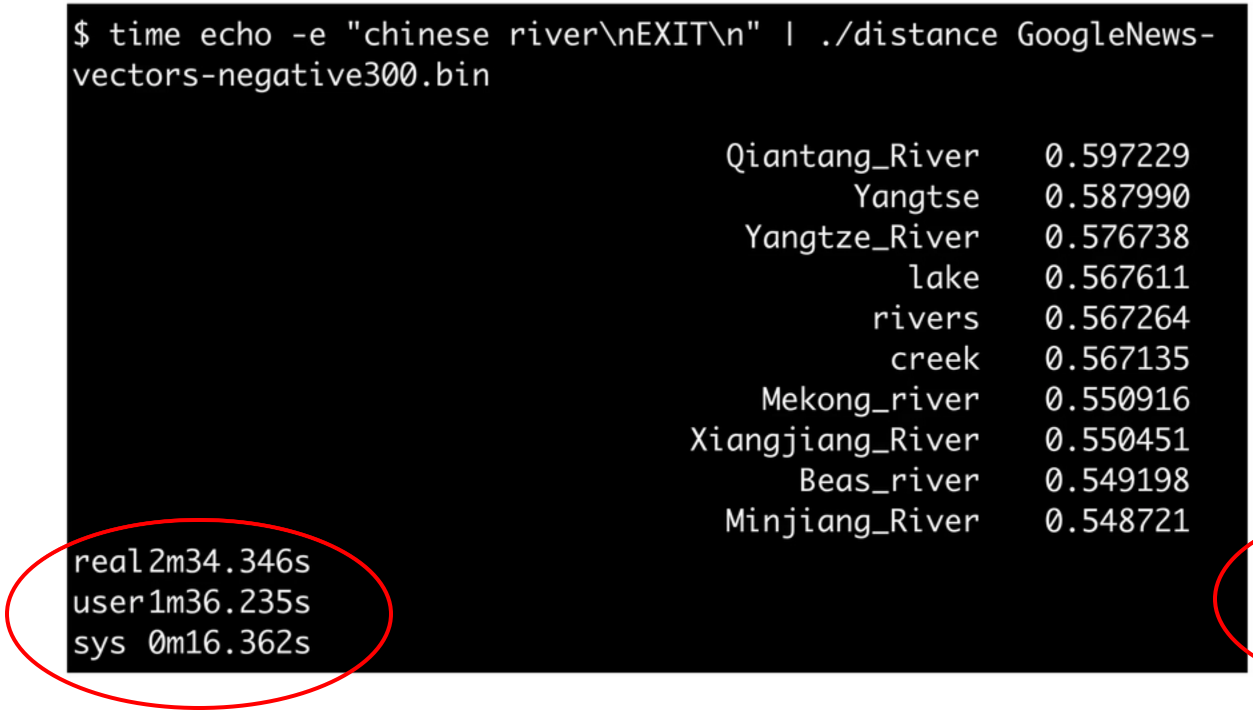
RAG Architecture: Vector Search Algorithms



- The naïve approach would be to compare the query vector to every vector in the database
- Such a comparison is $O(n)$ time complexity, however it is still too slow
  - Modern architectures have to handle billions of records
  - The larger the embedding space, the more computationally intensive the similarity calculation is
- Instead, we can use **Approximate Nearest Neighbor (ANN)** methods to speed up our search

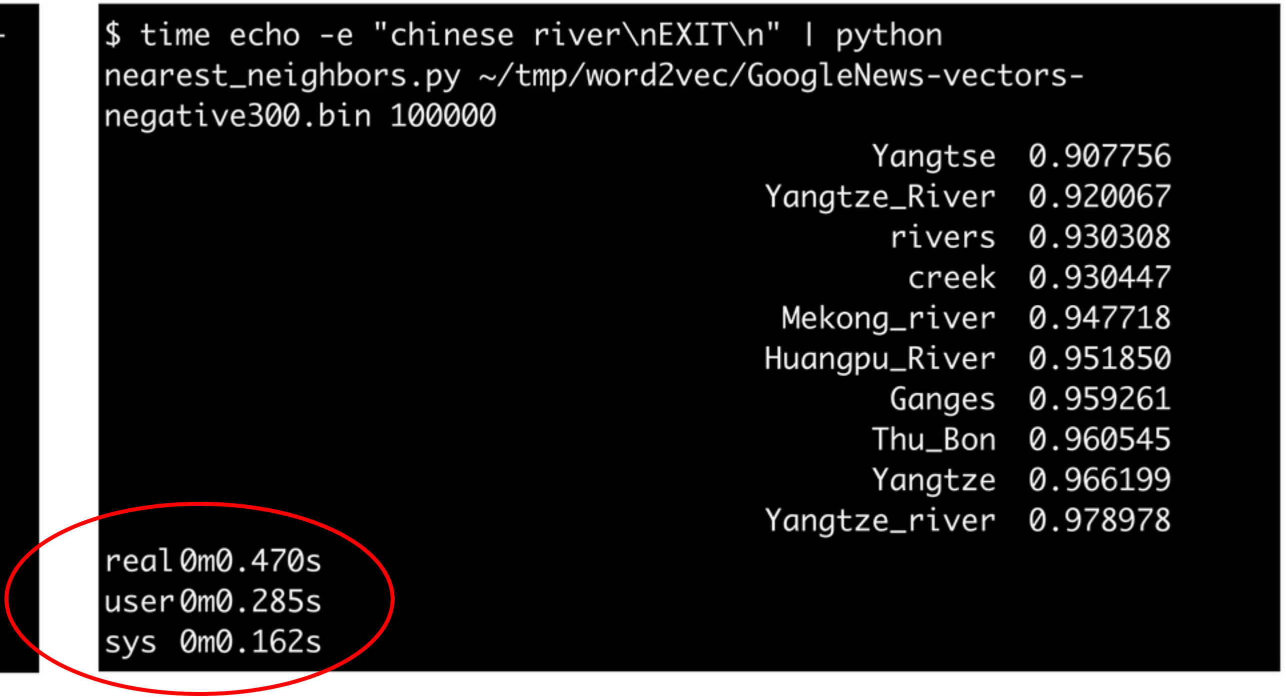
RAG Architecture: Vector Search Algorithms





```
$ time echo -e "chinese river\nEXIT\n" | ./distance GoogleNews-
vectors-negative300.bin

                                  Qiantang_River    0.597229
                                         Yangtse    0.587990
                                    Yangtze_River    0.576738
                                            lake    0.567611
                                          rivers    0.567264
                                           creek    0.567135
                                     Mekong_river    0.550916
                                  Xiangjiang_River    0.550451
                                       Beas_river    0.549198
                                   Minjiang_River    0.548721
real 2m34.346s
user 1m36.235s
sys  0m16.362s
```

Brute Force

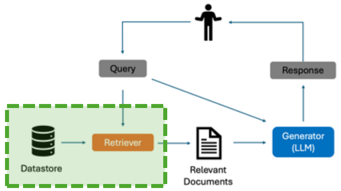


```
$ time echo -e "chinese river\nEXIT\n" | python
nearest_neighbors.py ~/tmp/word2vec/GoogleNews-vectors-
negative300.bin 100000
                                         Yangtse  0.907756
                                   Yangtze_River  0.920067
                                          rivers  0.930308
                                           creek  0.930447
                                    Mekong_river  0.947718
                                  Huangpu_River  0.951850
                                          Ganges  0.959261
                                         Thu_Bon  0.960545
                                         Yangtze  0.966199
                                   Yangtze_river  0.978978
real 0m0.470s
user 0m0.285s
sys  0m0.162s
```

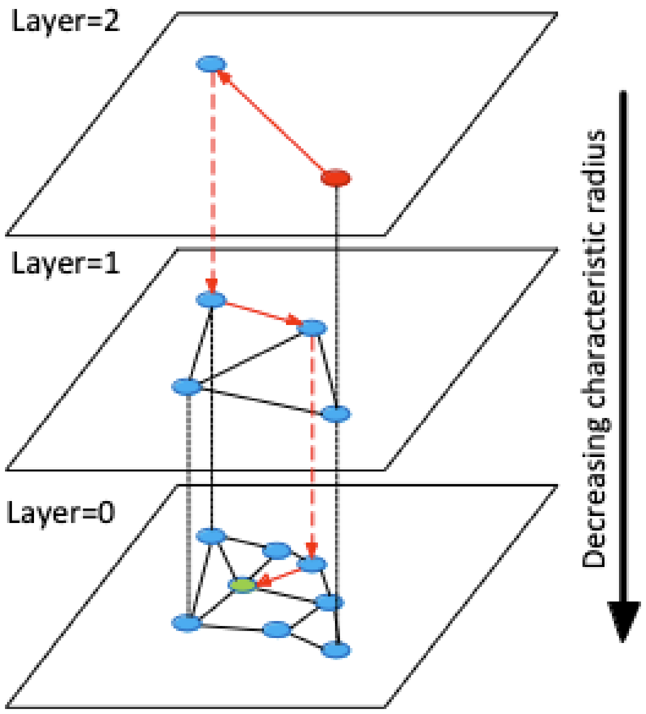
Annoy

https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html

Hierarchical Navigable Small World (HNSW)



- HNSW is **graph-based** ANN algorithm that relies on a layered search space
- **During indexing…**
  - Algorithm creates a layered structure, with the bottom layer containing all the nodes (data points)
  - Each layer above is a random sample from the layer below
- **During search…**
  - We start at the top layer
  - At each layer, we find the closest vector before moving to the lower, more granular layers

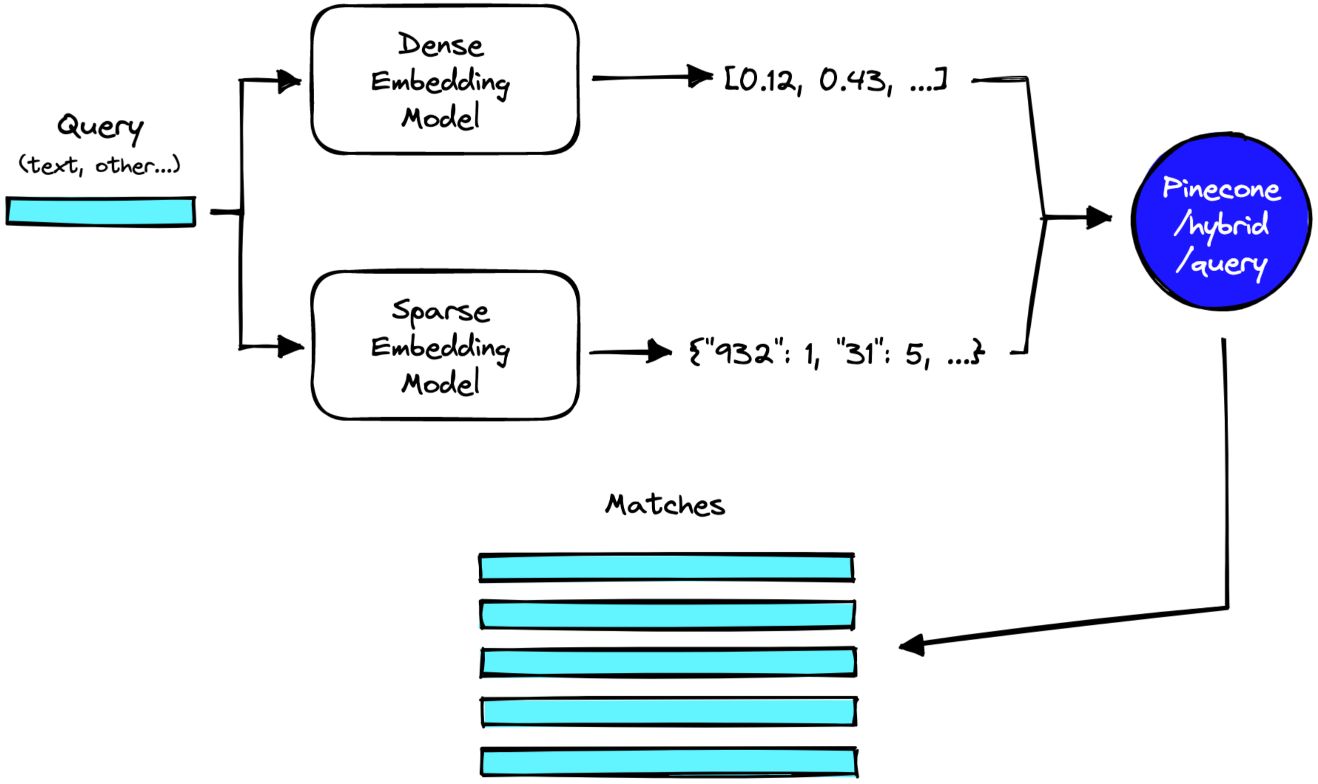


https://arxiv.org/pdf/1603.09320

Hybrid Search

- Although powerful, pure semantic search can be improved
- Hybrid search balances the retrieval of documents that contain exact query terms (BM25) with the need to find documents that are semantically related (but don't contain exact search terms)
- **Reciprocal Rank Fusion** is a method to combine the results from multiple ranking systems into a single, unified list

$$RRF_d = \sum_{r \in R} \frac{1}{r_d + k}$$

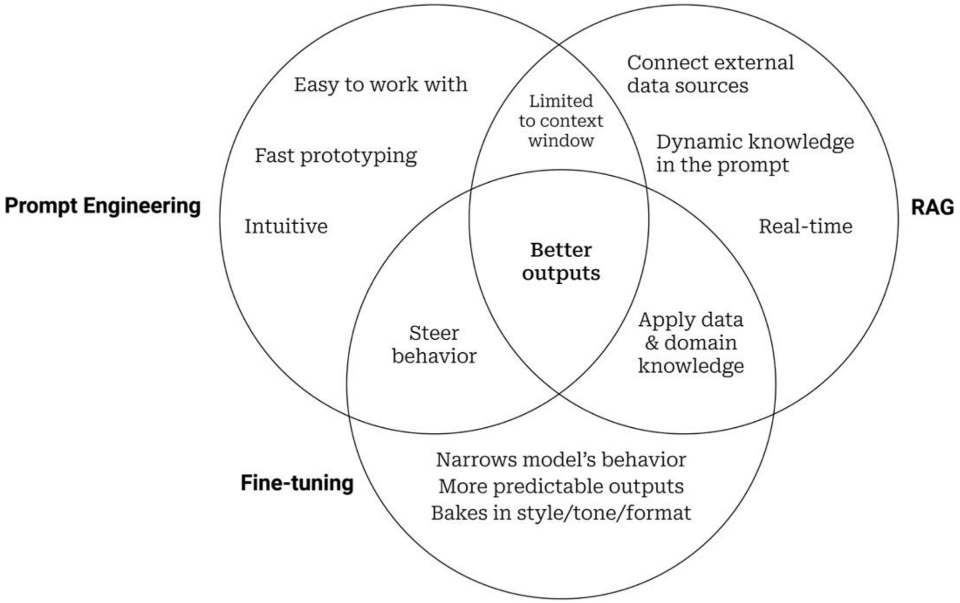


https://dl.acm.org/doi/10.1145/1571941.1572114

https://www.pinecone.io/learn/hybrid-search-intro/

When to use which?

	Data availability	Resources	Use case	Inferencing time
Prompt Engineering	A few or no labels	LLM	Quickly reasoning to complete tasks (e.g. Proof of Concept)	Fast
RAG	An external knowledge database (relevant data)	VectorScore, LLM	Enable internal data access to LLM (e.g. Specific knowledge Chatbot)	latency when inferencing
Fine-tuning	Requires labeled data (representative data)	Model weights and architecture, GPU/TPU	Repetitive task to output specific style/tone/format (e.g. English to SQL)	Moderate changes


Prompt Engineering
Easy to work with
Fast prototyping
Intuitive
Limited to context window
Connect external data sources
Dynamic knowledge in the prompt
Real-time
RAG
Better outputs
Steer behavior
Apply data & domain knowledge
Fine-tuning
Narrows model's behavior
More predictable outputs
Bakes in style/tone/format