Language models

Large Language Models

Text classification with LLMs

Integrating knowledge with language models

This class should be complemented by the following page with resources:
https://amazing-output-ccd.notion.site/Resources-for-LLMs-53c88cc0e3684968a0020534b503594f

LLMs – main architectures

- **Encoder-decoder** – similar to the original transformers (e.g. T5)
- **Causal decoder** - unidirectional attention, allowing input tokens to only interact with those preceding them; the model generates the output sequence one token at a time, and at each step, it considers only the tokens that precede it (e.g. Llama, GPT)
- **Mixture of experts** - the model is composed of multiple models, referred to as ”experts”, each of which specializes in some aspect of the data to enhance the overall model’s performance. Based on the idea of model ensembles in ML, is gaining traction (GPT4 ? , Mistral)

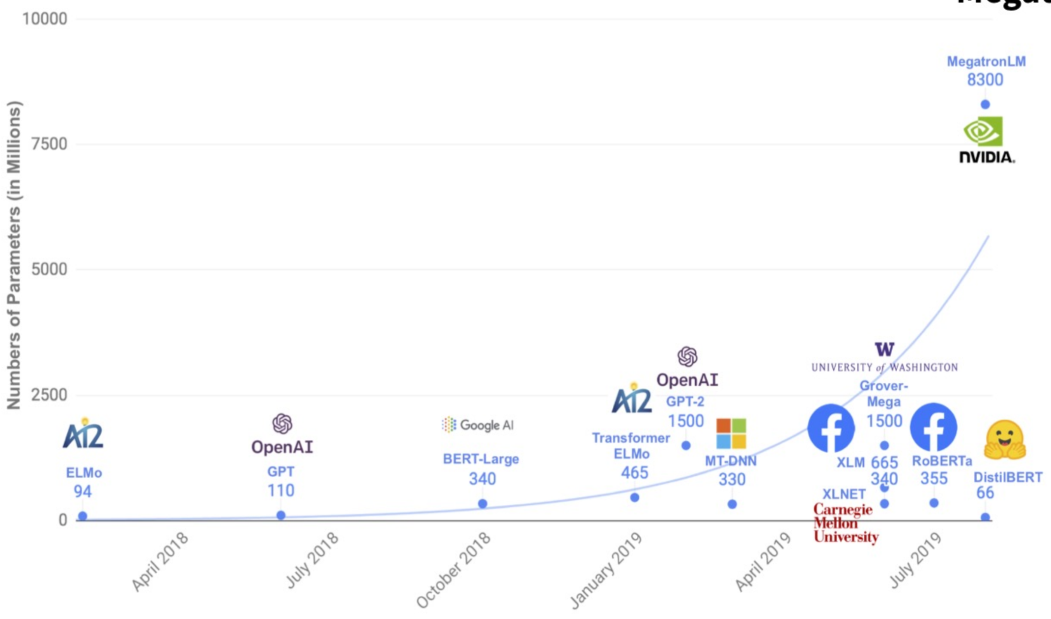
LLMs constantly evolving

Number of model parameters

PaLM (Google; 540B)
Megatron-Turing NLG (MS; NVIDIA; 530B)

GPT-3 (OpenAI; 175B)

Megatron-11B (FB)



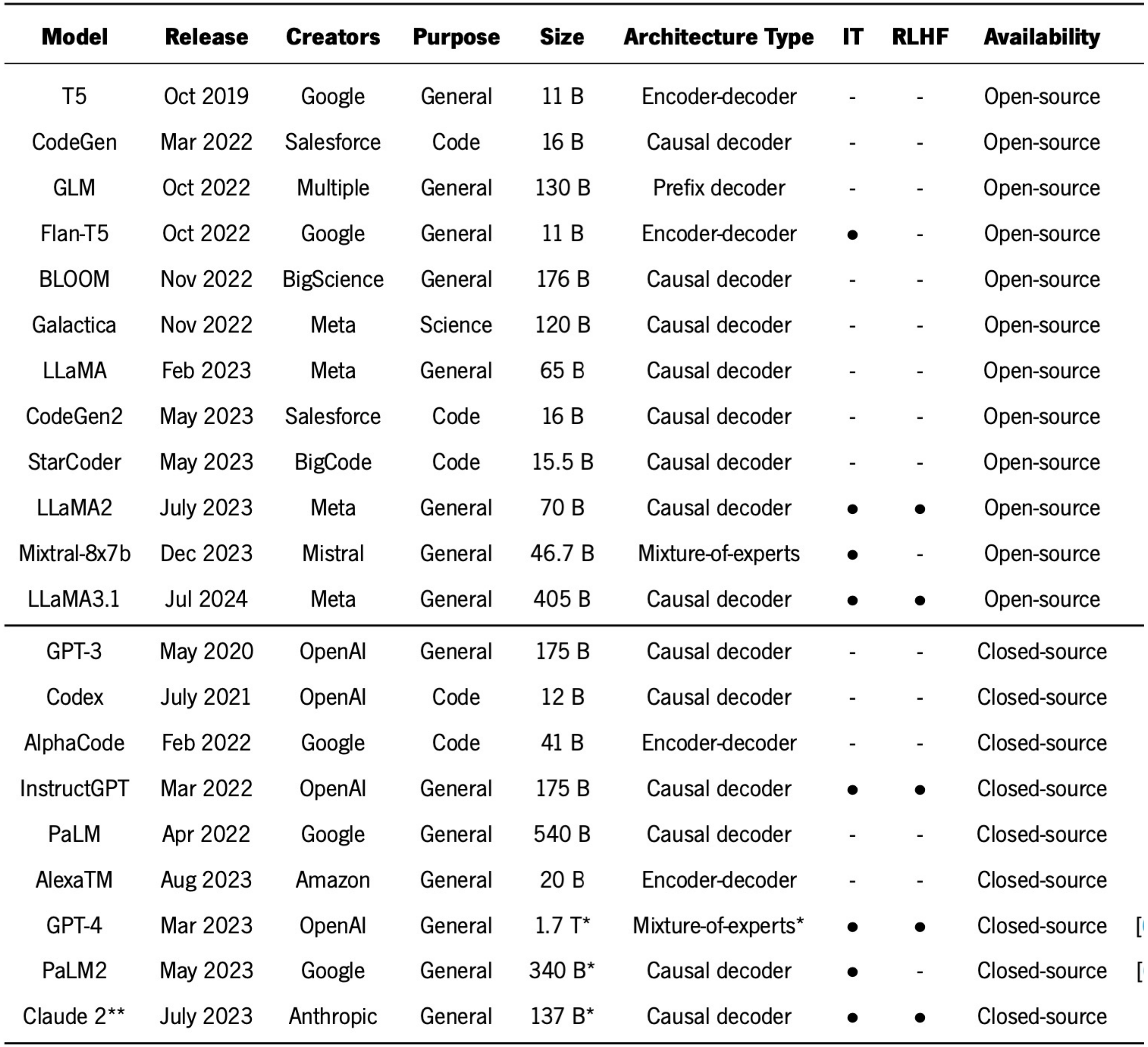
We would need more than 50 slides to represent these models...
More than 5000x larger than the first ones on the chart and don't have the latest ones yet ...!
Training LLMs is something only within the reach of organizations with many resources

An interesting example if you want to train your (mini) LLM (for Esperanto): https://huggingface.co/blog/how-to-train

Adapted from a presentation of Christopher Potts – Stanford; paper with original graphic- https://arxiv.org/abs/1910.01108

LLMs – existing models

- **IT** – Instruction tuning
- **RLHF** – Reinforcement Learning with Human Feedback



Model	Release	Creators	Purpose	Size	Architecture Type	IT	RLHF	Availability
T5	Oct 2019	Google	General	11 B	Encoder-decoder	-	-	Open-source
CodeGen	Mar 2022	Salesforce	Code	16 B	Causal decoder	-	-	Open-source
GLM	Oct 2022	Multiple	General	130 B	Prefix decoder	-	-	Open-source
Flan-T5	Oct 2022	Google	General	11 B	Encoder-decoder	●	-	Open-source
BLOOM	Nov 2022	BigScience	General	176 B	Causal decoder	-	-	Open-source
Galactica	Nov 2022	Meta	Science	120 B	Causal decoder	-	-	Open-source
LLaMA	Feb 2023	Meta	General	65 B	Causal decoder	-	-	Open-source
CodeGen2	May 2023	Salesforce	Code	16 B	Causal decoder	-	-	Open-source
StarCoder	May 2023	BigCode	Code	15.5 B	Causal decoder	-	-	Open-source
LLaMA2	July 2023	Meta	General	70 B	Causal decoder	●	●	Open-source
Mixtral-8x7b	Dec 2023	Mistral	General	46.7 B	Mixture-of-experts	●	-	Open-source
LLaMA3.1	Jul 2024	Meta	General	405 B	Causal decoder	●	●	Open-source
GPT-3	May 2020	OpenAI	General	175 B	Causal decoder	-	-	Closed-source
Codex	July 2021	OpenAI	Code	12 B	Causal decoder	-	-	Closed-source
AlphaCode	Feb 2022	Google	Code	41 B	Encoder-decoder	-	-	Closed-source
InstructGPT	Mar 2022	OpenAI	General	175 B	Causal decoder	●	●	Closed-source
PaLM	Apr 2022	Google	General	540 B	Causal decoder	-	-	Closed-source
AlexaTM	Aug 2023	Amazon	General	20 B	Encoder-decoder	-	-	Closed-source
GPT-4	Mar 2023	OpenAI	General	1.7 T*	Mixture-of-experts*	●	●	Closed-source
PaLM2	May 2023	Google	General	340 B*	Causal decoder	●	-	Closed-source
Claude 2**	July 2023	Anthropic	General	137 B*	Causal decoder	●	●	Closed-source

List of some of the available LLMs
September 2024 (not updated)
From Paulo Seixal MSc thesis

LLMs – open-source - examples

- **BERT** (Google, 2018) - quite used in NLP; improved in last years; currently thousands of open-source, free, and pre-trained Bert models available for specific use cases
- **BLOOM** (2022) – 176B; 46 languages; 13 programming languages; access to source code and data
- **Llama3.1** (Meta, July 2024) – 8B, 70B, 405B; several languages; RLHF; chatbots + NLP
- **Mistral** (Sep 2023) – 7B; French company providing competitive models, some of reduced size
- **OPT** (Meta, 2023) - 125 M to 175 B; similar performance to GPT-3
- **XGen-7B** (Salesforce, 2023) - 7B, 8x7B (mixture experts); supports longer context windows
- **Falcon** (TII-UAE, 2023) – 40B, 180B
- **DeepSeek** (2025)– several versions (R1, V3, etc)
- **Gemma 2** (Google, 2024)
- **Grok** (xAI)

https://github.com/Hannibal046/Awesome-LLM?tab=readme-ov-file

Model repo: https://huggingface.co/models

Training LLMs - Pre-training and fine-tuning

Most LLMs go through a pre-training stage to reach the foundational (general-purpose) models (e.g. GPT, Llama, etc)

Datasets include many different sources, as books, news, social media, scientific papers, code repositories, and several other websites

Ensures that the model gains exposure to various writing styles, terminologies, and contexts, enabling a good understanding of language.

Once the pre-training phase is completed, the model enters the **fine-tuning** phase - training on specific datasets tailored to the desired application or a specific domain

LLM tuning

Low-rank adaptation (**LORA**) - tuning only a fraction of parameters, freezing the rest of the model

Knowledge distilation – transfer the learning of a pretrained model (teacher) to a smaller student model. This process allows smaller models to retain much of the accuracy of bigger models while being less resource-intensive in terms of computation and memory and can be used in different contexts

Instruction tuning

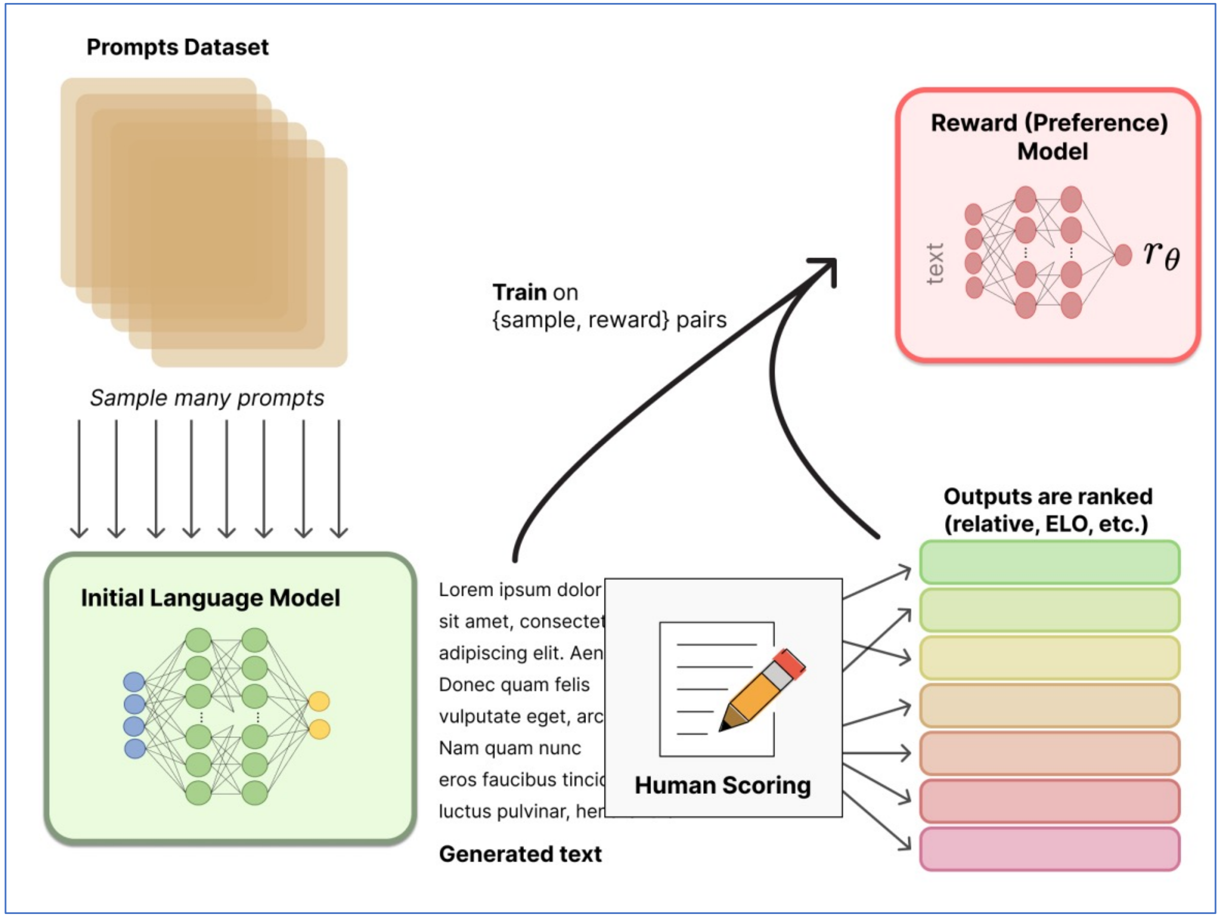
- Method for enhancing the performance of LLMs in tasks aligned with specific objectives
- The LLM is trained on a dataset comprising input-output pairs, where the input constitutes a natural language instruction, and the output corresponds to the desired response
- This approach guides the LLM to generate outputs that not only resemble the target responses, but also take into account the provided instructions, aligning the model more closely with specific task requirements

Reinforcement learning with human feedback

- RLHF was a recent addition to the AI methods toolkit, introduced from 2017 (Christiano et al, 2017), and widely used to refine LLMs for applications involving interactions with humans
- The idea is simple: after pre-training an LLM (decoder), it can generate different responses given a dialog (prompt). These can be evaluated by humans who assign numerical scores to the various options
- These scores can be used to create rewards in the context of reinforcement learning algorithms

Paper: https://arxiv.org/abs/1706.03741

Reinforcement learning with human feedback



Reward (preference) model – trained to return a numeric value given a prompt and a response to this prompt

These models can be based on LLMs or trained from scratch

These models, once trained, can be introduced into the automatic model refinement pipelines

To generate data to train the reward model, prompts are provided to the original LLM that generates responses

Human annotators are used to sort these answers and create rankings.

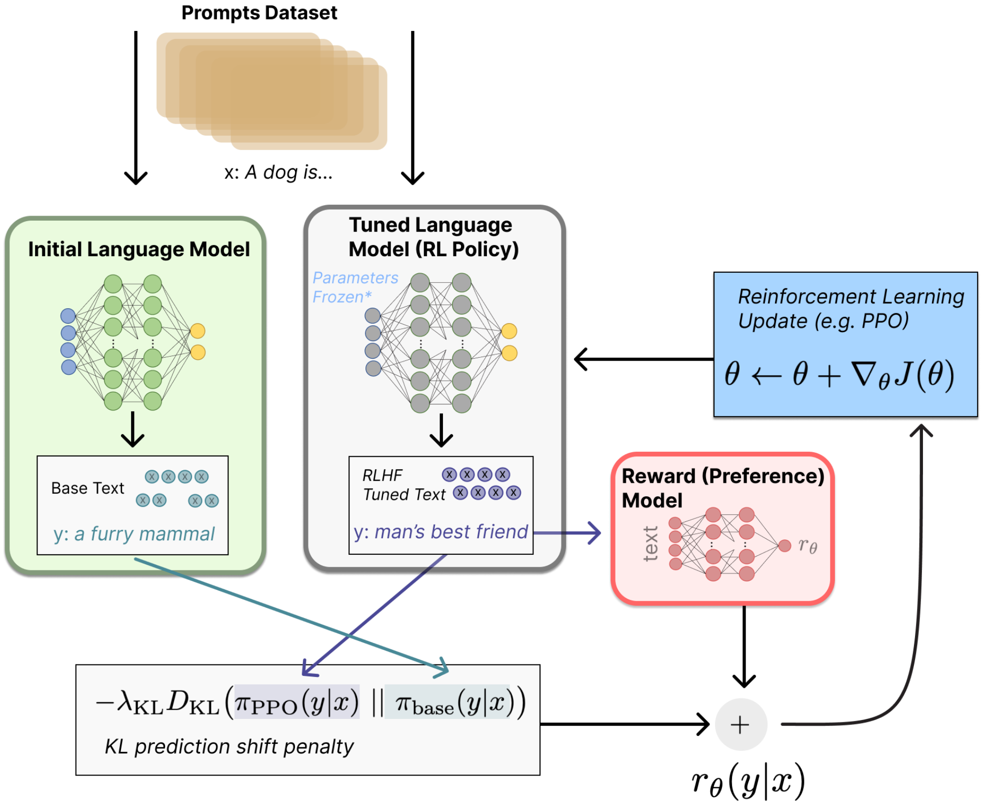
These rankings are normalized and used to create numerical scores (e.g. using Elo systems)

Process starts with a pre-trained model (e.g. GPT) that can be refined with human text (not shown in the image)

Image and details: https://huggingface.co/blog/rlhf

RLHF - optimization

- Once we have the reward model (RM), we can define the task as a reinforcement learning problem:
  - the policy will be implemented by the LLM that we want to refine
  - the reward will be given by the RM (loss will be the output of the RM for the outputs provided by the current LLM, subtracted of a penalty for outputs very different from the original LLM)
  - states given by the possible tokens and their responses.
- Multiple algorithms of Reinforcement Learning can be used, being the most common today Proximal Policy Optimization (PPO)


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
$-\lambda_{\mathrm{KL}} D_{\mathrm{KL}}(\pi_{\mathrm{PPO}}(y|x) \,||\, \pi_{\mathrm{base}}(y|x))$
KL prediction shift penalty
$+$
$r_\theta(y|x)$

https://huggingface.co/blog/rlhf

Using LLMs: applications

- Used directly in dialog applications (e.g. ChatGPT, Bard, ...) or by using the APIs available to provide:
  - Direct answers to questions (although factual answers are not the strength of today's LLMs …)
  - Generation of texts given a prompt– the most direct use given LLMs are generative models, including Code Generation
  - Text enhancement: grammar correction, style, …
  - Language processing tasks (zero-shot learning) – e.g. translation, summarization of texts, recognition of entities/relationships, sentiment analysis, topic modeling, etc.

LLM assessment

- To assess the performance of LLMs, many methods have been proposed
- Two key components: **metrics** and **benchmarks**
- **Metrics** are quantitative measurements or standards for assessing the performance of LLMs providing specific numerical values, enabling the evaluation of different dimensions of a model's effectiveness, efficiency, or quality.
  - ML derived metrics - accuracy, precision, recall, F1 score
  - Perplexity – how well a training set has been learned
  - Recall-Oriented Understudy for Gisting Evaluation (ROUGE) or Bilingual Evaluation Understudy (BLEU) scores - more used in test sets

https://medium.com/@priyankads/perplexity-of-language-models-41160427ed72
https://medium.com/@priyankads/evaluation-metrics-in-natural-language-processing-bleu-dc3cfa8faaa5
https://medium.com/@priyankads/rouge-your-nlp-results-b2feba61053a

LLM assessment - benchmarks

- Benchmarks serve as standardized tasks, datasets, or challenges against which LLM performance is evaluated.
- Provide a common reference point for comparing different models or approaches, contributing to a consistent and reproducible evaluation of LLMs.
- LLM benchmarks encompass a wide range of tasks:
  - sentiment analysis, text summarization, question-answering, and more
- Datasets specifically designed to assess the model’s ability to transfer knowledge across diverse linguistic tasks:
  - General Language Understanding Evaluation (GLUE), SuperGLUE – linguistic tasks
  - Massive Multitask Language Understanding (MMLU) – multi-choice questions
  - AI2 Reasoning challenge (ARC) – grade-school questions
  - Winogrande and HellaSwag – common sense reasoning
  - Human Eval – python coding
  - DROP – reading comprehension and arithmetic
  - GSM-8K – grede-school Maths

Domain specific models

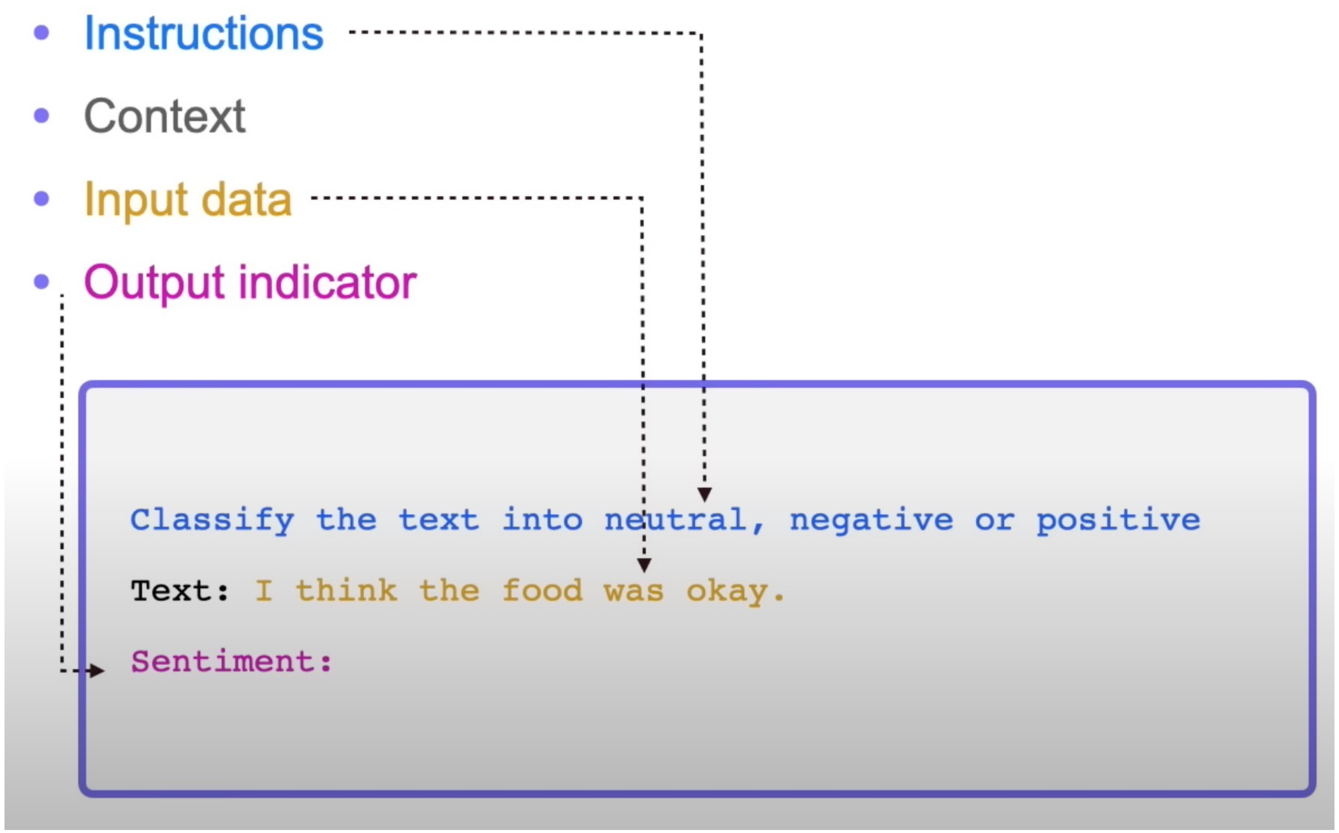
- LLM fine-tuning lead to the development of domain-specific models designed to understand specific nuances within a certain field using corpora of data specific to a given domain
- Advantage against general **LLMs** that cannot comprehend the same intricacies and technical terms specific to that particular domain, facilitating more accurate and context-aware results
- Domains
  - Healthcare – e.g. Med-Palm-2 , Almanac
  - Finance – e.g. Bloomberg GPT
  - Biology/ Biomedical – PubMedGPT, BioGPT
  - Scientific research – PaperQA, SciBert, Darwin
  - And many many others…

Prompt Engineering – what is it?

- A **prompt** serves as the input or instruction provided to a LLM, guiding it to generate relevant and contextually appropriate outputs to accomplish a given task.
- Prompt engineering - field that involves optimizing prompts for efficient use of LLMs for a set of applications
- Why?
  - Better results when using LLMs
  - Helps to test LLMs and their capabilities and limitations
  - Allows you to build innovative applications on top of LLMs' APIs

Resources: https://www.promptingguide.ai/
Video: https://www.youtube.com/watch?v=dOxUroR57xs

Prompt engineering – elements of the prompt



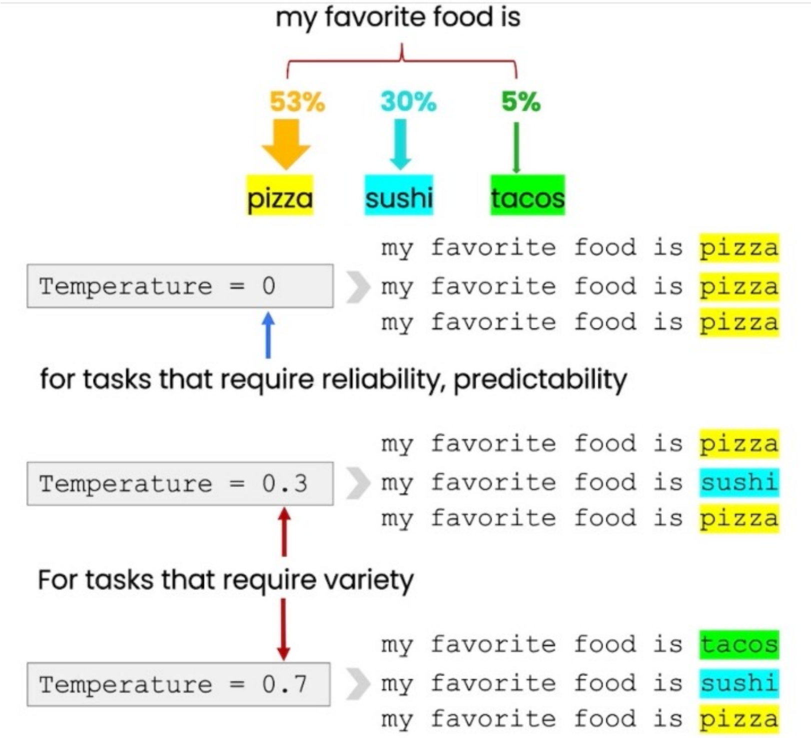
- Instructions
- Context
- Input data
- Output indicator

```
Classify the text into neutral, negative or positive

Text: I think the food was okay.

Sentiment:
```

https://www.youtube.com/watch?v=dOxUroR57xs

Prompt engineering – parameters/ settings

- The most relevant parameter in the APIs of LLMs – ***temperature*** – allows to control stochasticity of the responses
- It is a value that determines the probability of choosing the most likely token at each step of generation
- Higher values favor greater diversity in responses
- Lower values should be used for greater predictability



```
Temperature = 0
```

my favorite food is pizza
my favorite food is pizza
my favorite food is pizza

for tasks that require reliability, predictability

```
Temperature = 0.3
```

my favorite food is pizza
my favorite food is sushi
my favorite food is pizza

For tasks that require variety

```
Temperature = 0.7
```

my favorite food is tacos
my favorite food is sushi
my favorite food is pizza

https://learn.deeplearning.ai/chatgpt-prompt-eng

Other settings: https://www.promptingguide.ai/introduction/settings

Prompt Engineering – principles

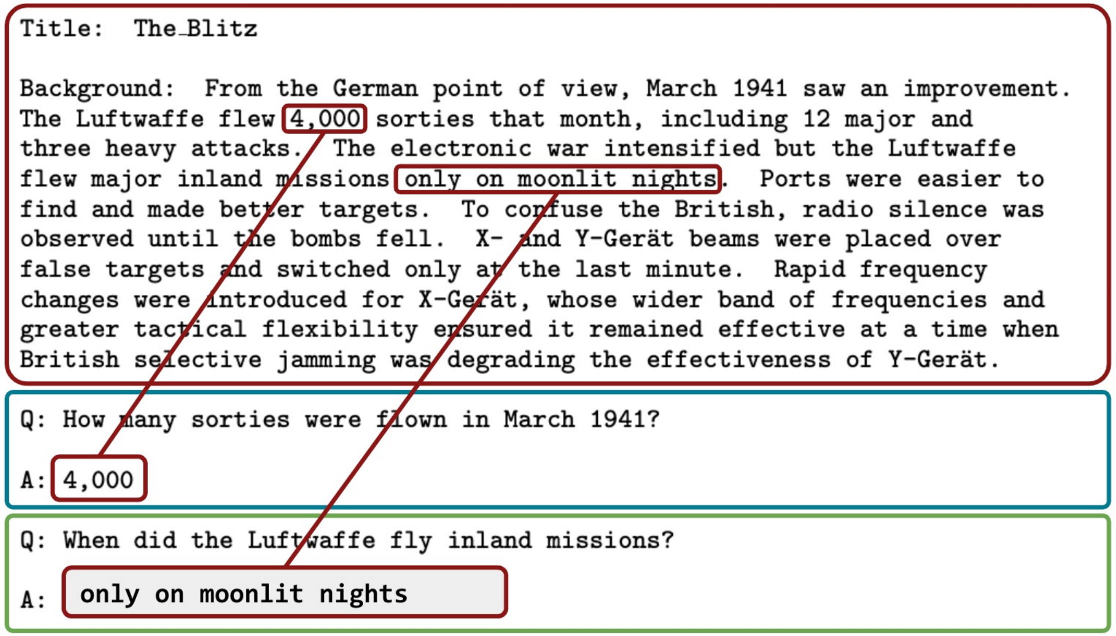
- Write clear sentences that are as unambiguous as possible
  - Various types of separators can be used to give structure to requests when they become more complex
  - Structure can be requested in the output(e.g. JSON ou HTML)
- Iterative process
  - Try what you want, analyze the result and clarify the instructions by improving the context
  - Use examples in context to clarify what you want

Course: https://learn.deeplearning.ai/chatgpt-prompt-eng

Using LLMs: context

- The ability to do "in-context" learning can be used in LLMs to improve the outcomes of prompts, feeding the model with texts that can create an appropriate context for the task or questions that follow
- Can be used to perform one- or few-shot learning, giving examples to the LLM that allow "generalization"
- It can be used to give "instructions" on how to perform some tasks, how to organize answers, etc
- Can be used to change the text generation style, for example in “role playing”

In-context learning in LLMs



Title: The_Blitz

Background: From the German point of view, March 1941 saw an improvement. The Luftwaffe flew 4,000 sorties that month, including 12 major and three heavy attacks. The electronic war intensified but the Luftwaffe flew major inland missions only on moonlit nights. Ports were easier to find and made better targets. To confuse the British, radio silence was observed until the bombs fell. X- and Y-Gerät beams were placed over false targets and switched only at the last minute. Rapid frequency changes were introduced for X-Gerät, whose wider band of frequencies and greater tactical flexibility ensured it remained effective at a time when British selective jamming was degrading the effectiveness of Y-Gerät.

Q: How many sorties were flown in March 1941?

A: 4,000

Q: When did the Luftwaffe fly inland missions?

A: only on moonlit nights

LLMs allow you to create a context that can then be used for subsequent dialogs and tasks to ask the LLM

This can be used as a form of one- or few-shot learning

Hey model, here is an example of nervous anticipation: “My palms started to sweat as the lotto numbers were read off.”

Hey model, here’s an example without nervous anticipation: “...”

Adapted from Christopher Potts – Stanford

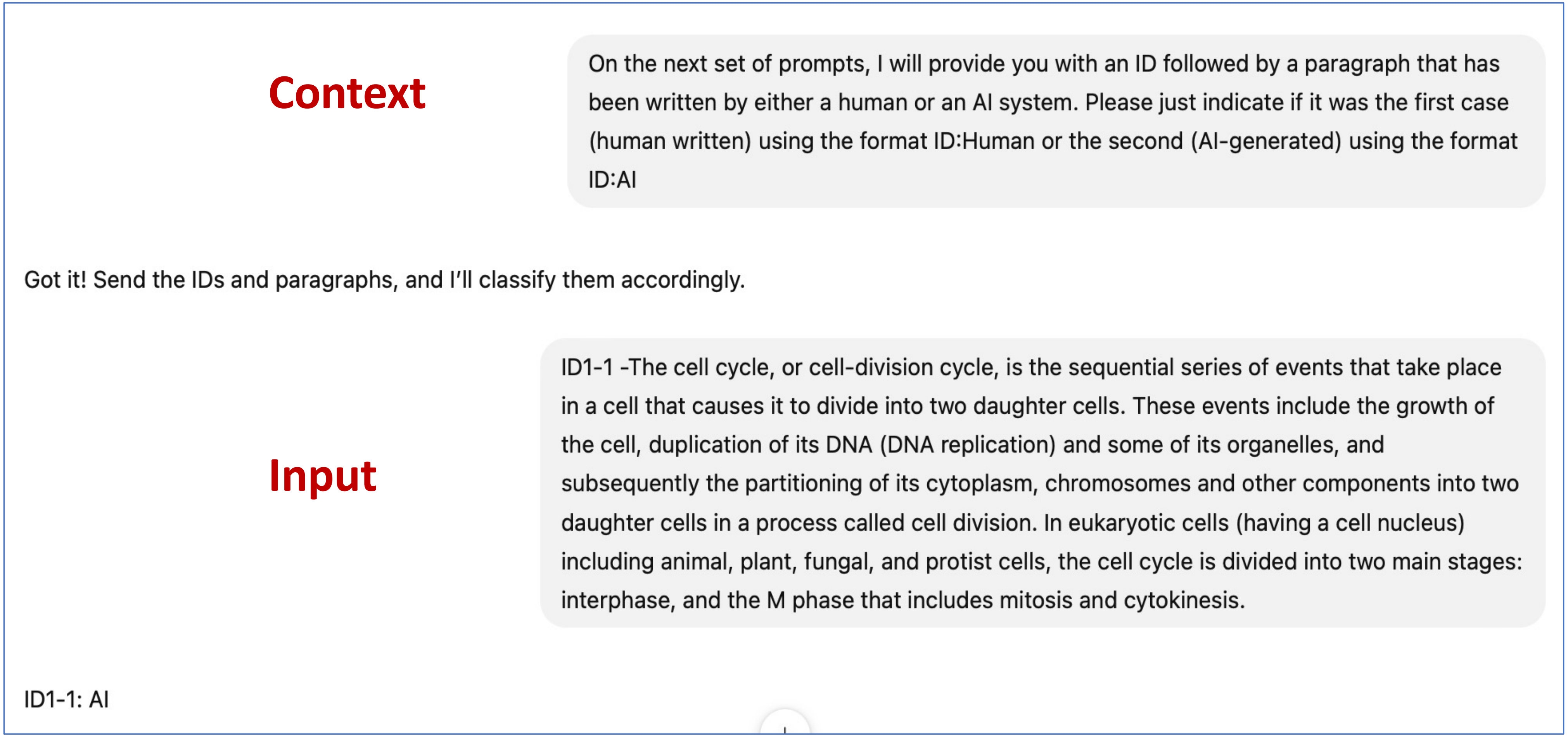
Zero-shot learning

- Large LLMs are tuned to follow instructions and are trained on large amounts of data; so they are capable of performing some tasks "zero-shot”
- This means **no further learning** is needed to solve some problems/ tasks
- Example:

Prompt: Classify the text into neutral, negative or positive.
Text: I think the vacation is okay.
Sentiment: Neutral

- Note that in the prompt above we didn't provide the model with any examples of text alongside their classifications, the LLM already understands "sentiment” from previous training.
- Instruction tuning has shown to improve zero-shot learning - Wei et al. (2022) - the concept of finetuning models on datasets described via instructions.

Zero-shot learning example with context



Context

On the next set of prompts, I will provide you with an ID followed by a paragraph that has been written by either a human or an AI system. Please just indicate if it was the first case (human written) using the format ID:Human or the second (AI-generated) using the format ID:AI

Got it! Send the IDs and paragraphs, and I'll classify them accordingly.

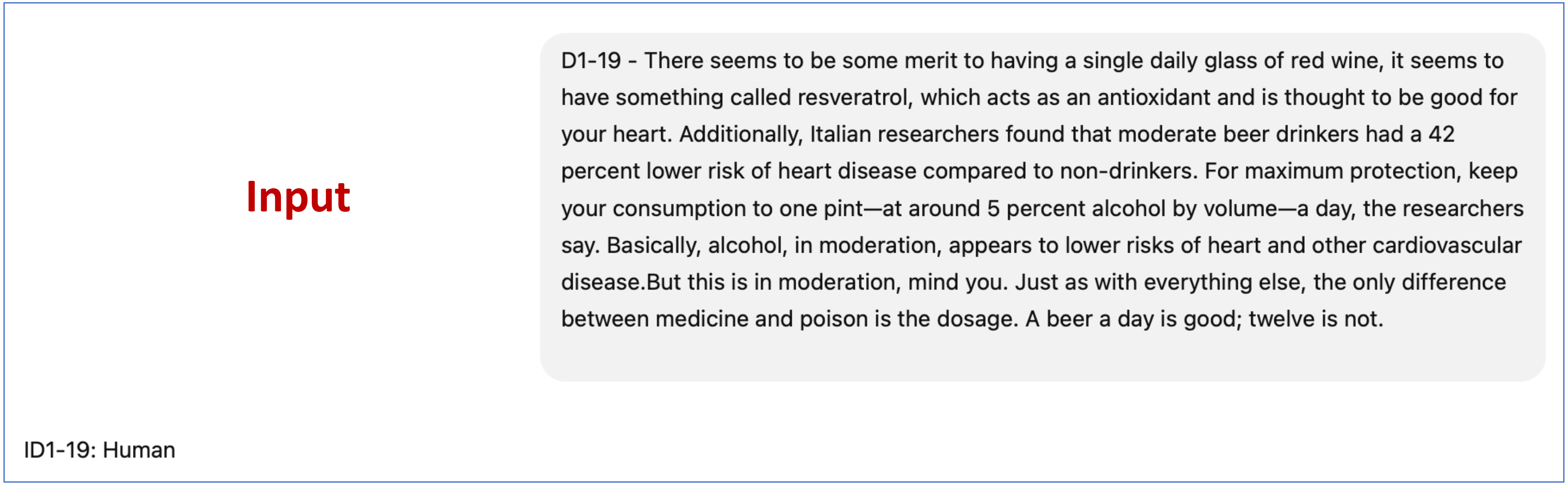
Input

ID1-1 -The cell cycle, or cell-division cycle, is the sequential series of events that take place in a cell that causes it to divide into two daughter cells. These events include the growth of the cell, duplication of its DNA (DNA replication) and some of its organelles, and subsequently the partitioning of its cytoplasm, chromosomes and other components into two daughter cells in a process called cell division. In eukaryotic cells (having a cell nucleus) including animal, plant, fungal, and protist cells, the cell cycle is divided into two main stages: interphase, and the M phase that includes mitosis and cytokinesis.

ID1-1: AI

Generated in Mar 19 2025 by Chat GPT

Zero-shot learning example



Input

D1-19 - There seems to be some merit to having a single daily glass of red wine, it seems to have something called resveratrol, which acts as an antioxidant and is thought to be good for your heart. Additionally, Italian researchers found that moderate beer drinkers had a 42 percent lower risk of heart disease compared to non-drinkers. For maximum protection, keep your consumption to one pint—at around 5 percent alcohol by volume—a day, the researchers say. Basically, alcohol, in moderation, appears to lower risks of heart and other cardiovascular disease.But this is in moderation, mind you. Just as with everything else, the only difference between medicine and poison is the dosage. A beer a day is good; twelve is not.

ID1-19: Human

Generated in Mar 19 2025 by Chat GPT

An example of one-shot learning

- **Prompt**:

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:

We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:

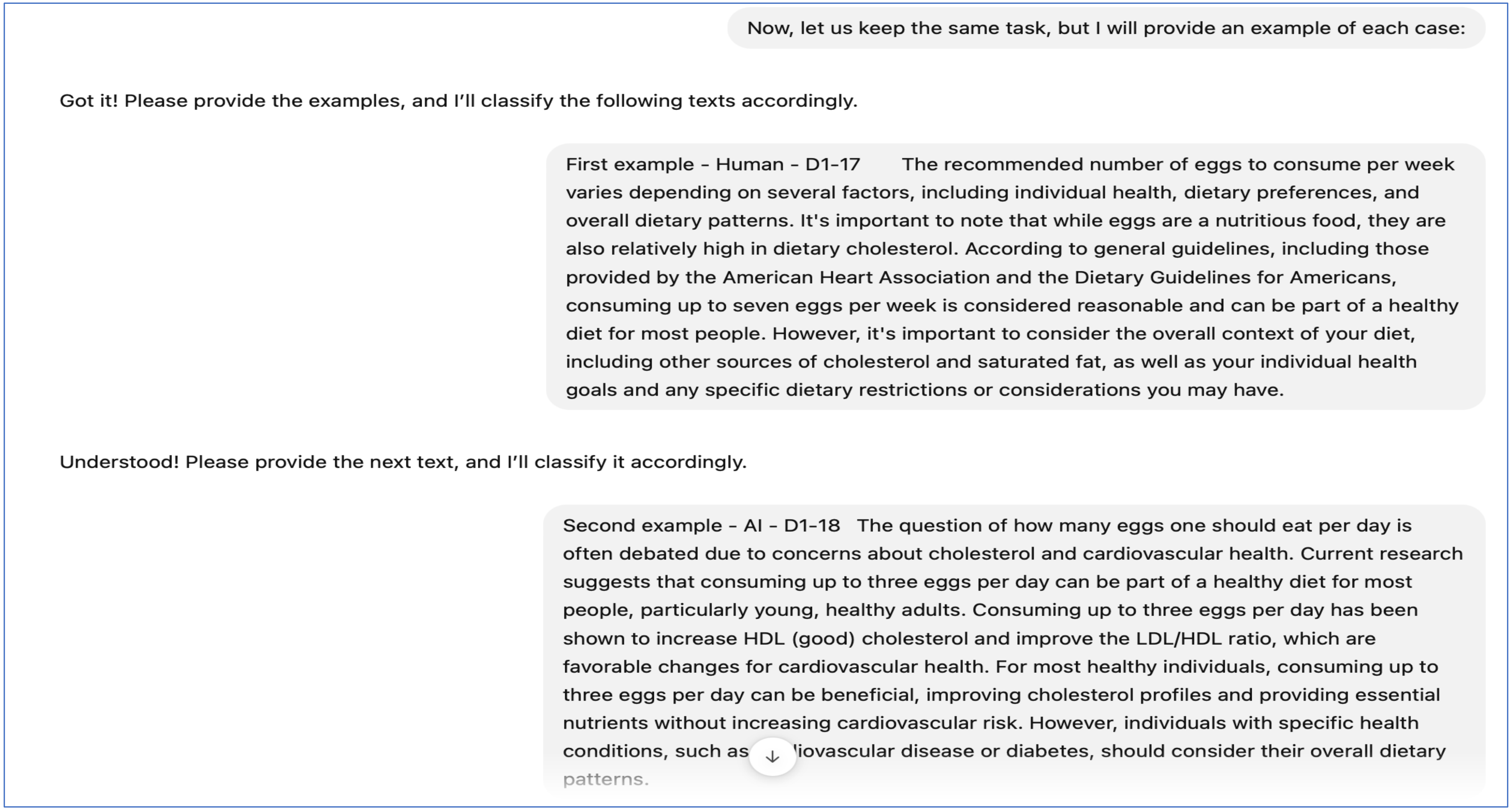
- **Output**:

When we won the game, we all started to farduddle in celebration.

We can observe that the model has somehow learned how to perform the task by providing it with just one example (i.e., 1-shot). For more difficult tasks, we can experiment with increasing the demonstrations (e.g., 3-shot, 5-shot, …)

One-shot learning example

Generated in Mar 19 2025 by Chat GPT



Now, let us keep the same task, but I will provide an example of each case:

Got it! Please provide the examples, and I'll classify the following texts accordingly.

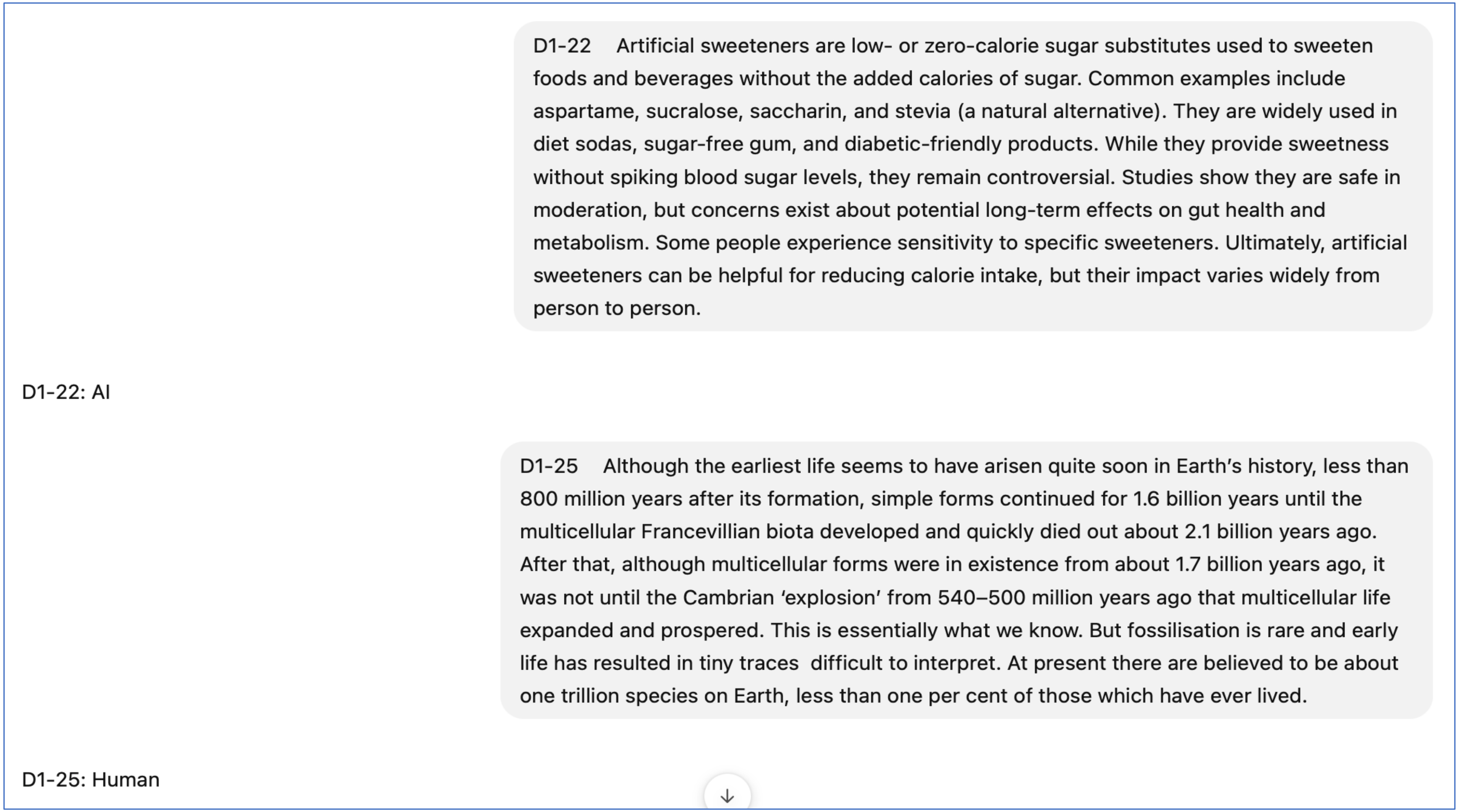
First example - Human - D1-17 The recommended number of eggs to consume per week varies depending on several factors, including individual health, dietary preferences, and overall dietary patterns. It's important to note that while eggs are a nutritious food, they are also relatively high in dietary cholesterol. According to general guidelines, including those provided by the American Heart Association and the Dietary Guidelines for Americans, consuming up to seven eggs per week is considered reasonable and can be part of a healthy diet for most people. However, it's important to consider the overall context of your diet, including other sources of cholesterol and saturated fat, as well as your individual health goals and any specific dietary restrictions or considerations you may have.

Understood! Please provide the next text, and I'll classify it accordingly.

Second example - AI - D1-18 The question of how many eggs one should eat per day is often debated due to concerns about cholesterol and cardiovascular health. Current research suggests that consuming up to three eggs per day can be part of a healthy diet for most people, particularly young, healthy adults. Consuming up to three eggs per day has been shown to increase HDL (good) cholesterol and improve the LDL/HDL ratio, which are favorable changes for cardiovascular health. For most healthy individuals, consuming up to three eggs per day can be beneficial, improving cholesterol profiles and providing essential nutrients without increasing cardiovascular risk. However, individuals with specific health conditions, such as [illegible]iovascular disease or diabetes, should consider their overall dietary patterns.

One-shot learning example

Generated in Mar 19 2025 by Chat GPT



D1-22 Artificial sweeteners are low- or zero-calorie sugar substitutes used to sweeten foods and beverages without the added calories of sugar. Common examples include aspartame, sucralose, saccharin, and stevia (a natural alternative). They are widely used in diet sodas, sugar-free gum, and diabetic-friendly products. While they provide sweetness without spiking blood sugar levels, they remain controversial. Studies show they are safe in moderation, but concerns exist about potential long-term effects on gut health and metabolism. Some people experience sensitivity to specific sweeteners. Ultimately, artificial sweeteners can be helpful for reducing calorie intake, but their impact varies widely from person to person.

D1-22: AI

D1-25 Although the earliest life seems to have arisen quite soon in Earth’s history, less than 800 million years after its formation, simple forms continued for 1.6 billion years until the multicellular Francevillian biota developed and quickly died out about 2.1 billion years ago. After that, although multicellular forms were in existence from about 1.7 billion years ago, it was not until the Cambrian ‘explosion’ from 540–500 million years ago that multicellular life expanded and prospered. This is essentially what we know. But fossilisation is rare and early life has resulted in tiny traces difficult to interpret. At present there are believed to be about one trillion species on Earth, less than one per cent of those which have ever lived.

D1-25: Human

Few-shot learning

- ***Few-shot prompting*** can be used as a technique to enable in-context learning where we provide demonstrations in the prompt to steer the model to better performance.
- A few more tips:
  - the label space and the distribution of the input text specified by the demonstrations are both important
  - the format you use also plays a key role in performance
  - selecting random labels from a true distribution of labels (instead of a uniform distribution) helps.

- **Prompt**:

This is awesome! // Negative

This is bad! // Positive

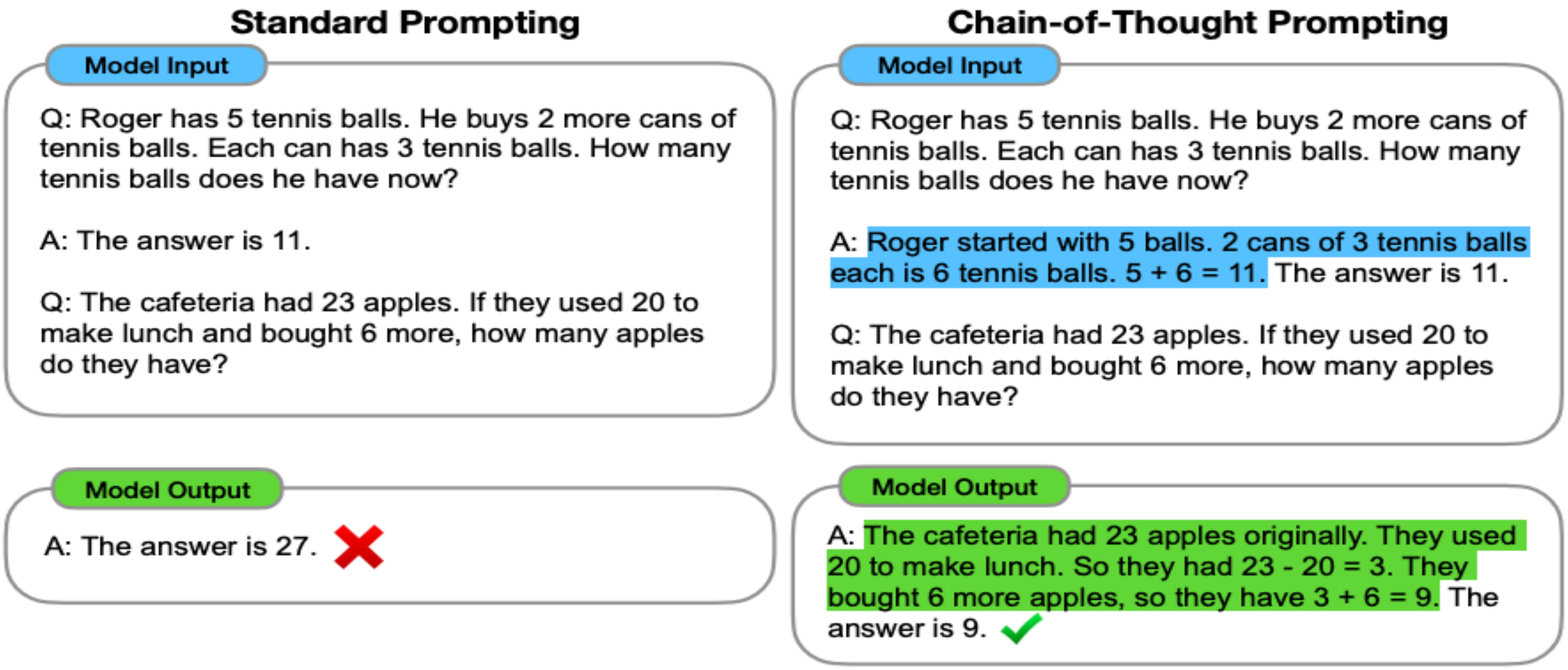
Wow that movie was rad! // Positive

What a horrible show!

- **Output**:

Negative

Chain of thought



Standard Prompting

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. ✗

Chain-of-Thought Prompting

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?

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. ✓

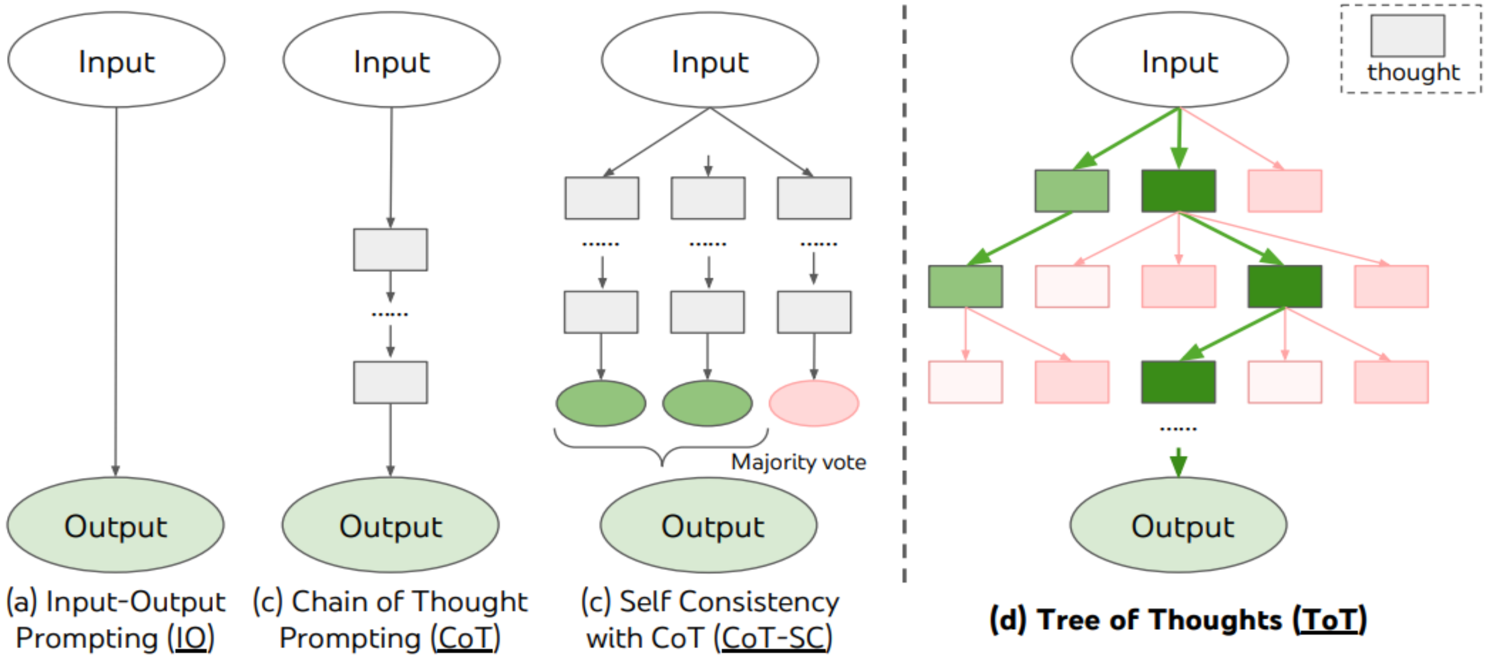
- **Chain-of-thought (CoT)** prompting enables complex reasoning capabilities through intermediate reasoning steps.

Prompt chaining

- To improve the reliability and performance of LLMs, one of the important prompting engineering techniques is to break tasks into its **subtasks**.
- Once subtasks have been identified, the LLM is prompted with a subtask and then its response is used as input to another prompt- **prompt chaining**
- Useful to accomplish complex tasks, where prompts perform transformations or additional processes on the responses before reaching a final desired state.
- Helps to boost transparency of your LLM application, increases controllability, and reliability.
- Particularly useful when building LLM-powered conversational assistants and improving the personalization and user experience of your applications.

Tree of thoughts

- For complex tasks, traditional or simple prompting techniques fall short. Alternatives have been proposed including Tree of Thoughts (ToT), a framework that generalizes chain-of-thought prompting and encourages exploration over thoughts that serve as intermediate steps.


Input
Output
(a) Input-Output Prompting (IO)
Input
Output
(c) Chain of Thought Prompting (CoT)
Input
Majority vote
Output
(c) Self Consistency with CoT (CoT-SC)
Input
thought
Output
(d) Tree of Thoughts (ToT)

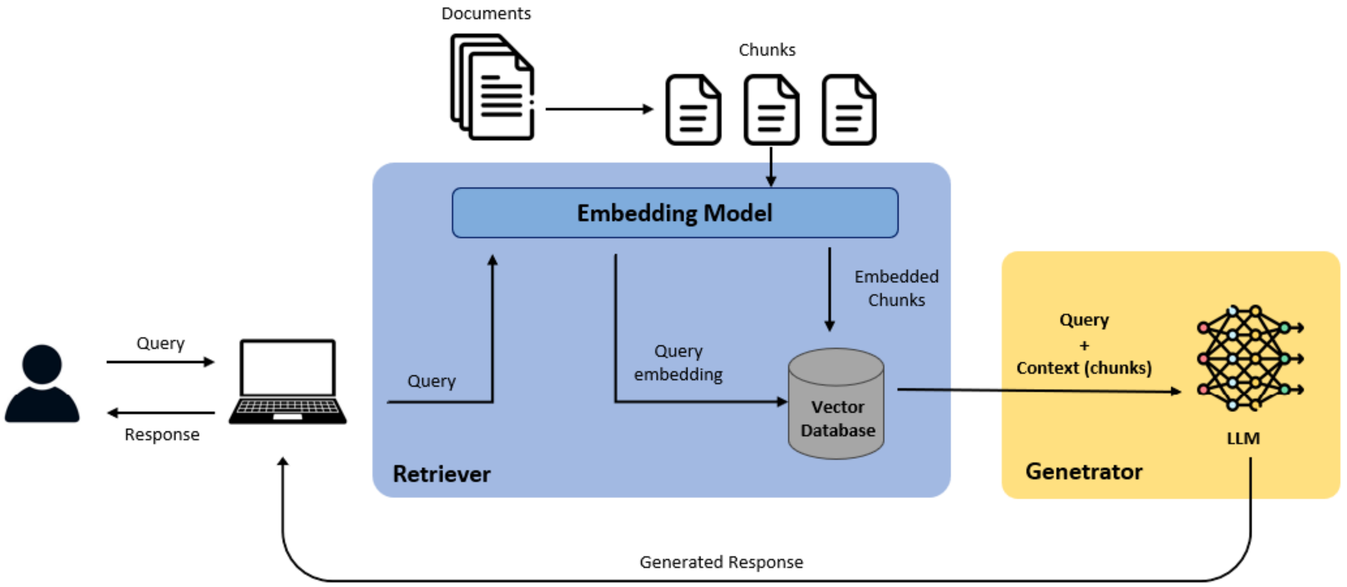
https://www.promptingguide.ai/techniques/tot

Retrieval Augmented Generation (RAG)

- For complex and knowledge-intensive tasks, it's possible to build an LLM-based system that accesses external knowledge sources to complete tasks.
- This enables more factual consistency, improves reliability of the generated responses, and helps to mitigate the problem of "hallucination".
- RAG can be finetuned and its internal knowledge can be modified in an efficient manner **without needing retraining of the entire model**.
- This makes RAG adaptive for situations where facts could evolve over time, enabling access to the latest information.

Retrieval Augmented Generation (RAG)

- RAG combines an **information retrieval component** with a **text generator model**.
- RAG takes an input and retrieves a set of relevant/supporting documents given a source.
- The documents are provided as context with the original input prompt and fed to the text generator which produces the final output.



Retrieval Augmented Generation (RAG)

- Step 1 – **Database construction:**

Creation of a vectorized database to store documents. Due to LLM’s context window limits and to improve the similarity search, documents are segmented into manageable sections, transformed into numerical vectors and efficiently stored for rapid retrieval

- Step 2 – **Querying**

Similarly to the documents in the external knowledge base, user queries undergo vectorization through the same embedding model.

- Step 3 – **Information retrieval**

The retriever component conducts a search of the database using the vectorized query, identifying closely related sections within the documents. This process relies on calculating similarity

- Step 4 – **Response generation**

Following the retrieval of the most relevant sections, the prompt is assembled by combining the initial query with these retrieved sections as context. This prompt is then input into the LLM, seeking a coherent and contextually appropriate response to the user’s query

Vector databases for RAG

- Vector databases, or vector stores, are specialized databases designed to store and retrieve data efficiently as high-dimensional numerical vectors.
- These allow accurate semantic similarity search based on vector similarity
- Allows to handle complex/ unstructured data
- Can be implemented to allow improved performance - Creating a robust vector database is crucial for the performance of a **RAG** system
- Steps in the construction:
  - Chunking - split the documents into smaller units
  - Converting textual data into numerical vectors – embedding
  - Indexing the numerical vectors for good performance

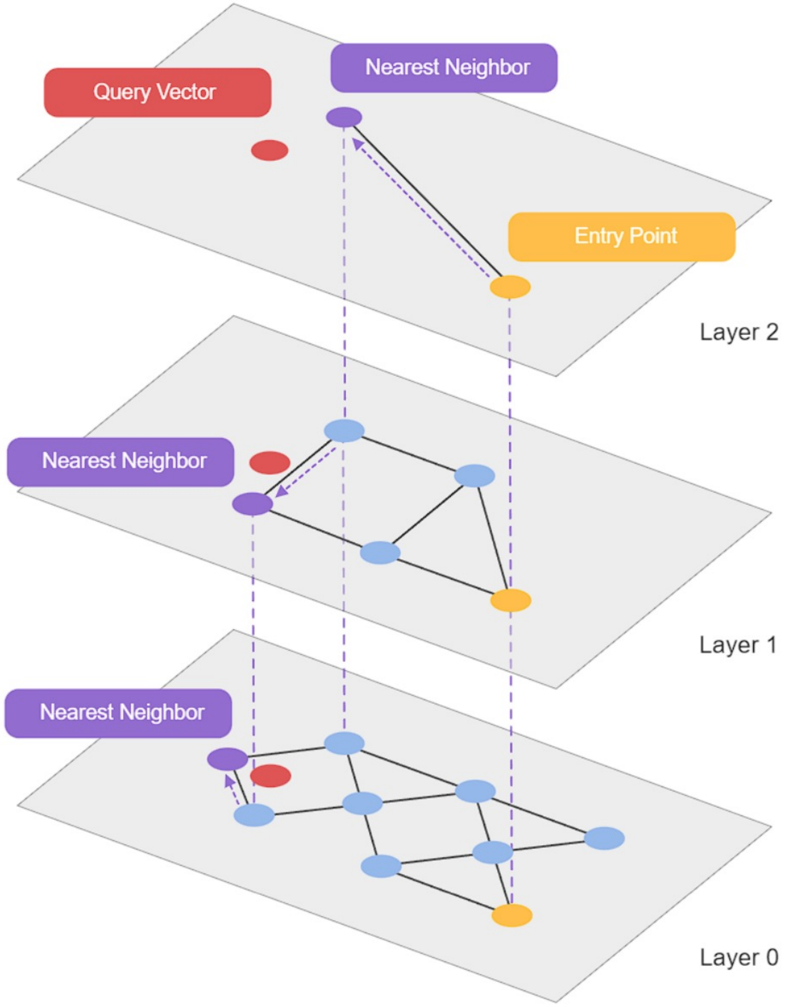
Similarity search/ retrieval algorithms for RAG

- **Nearest Neighbor Search - NNS**
  - seeks to find the vectors in a database that are closest to a specified query vector
  - operates based on distance metrics, such as Euclidean distance or cosine similarity, measuring the similarity between vectors.
  - common search algorithms include brute-force approaches and tree-based approaches
  - can have high computational complexity
- **Approximate NNS**
  - provide an approximate solution, relaxing the requirement for an exact match and reducing computational costs
  - One popular alternative is the use of Navigable Small World (NSW) graphs that introduce a trade-off between accuracy and efficiency
  - **Hierarchical NSW** further refines the structure. It introduces a hierarchy of layers, each containing a subset of nodes. Nodes within a layer are connected to nodes in the same and neighboring layers, creating a navigable structure

To learn more about this: https://www.deeplearning.ai/short-courses/building-applications-vector-databases/

Similarity search/ retrieval algorithms for RAG

HNSW

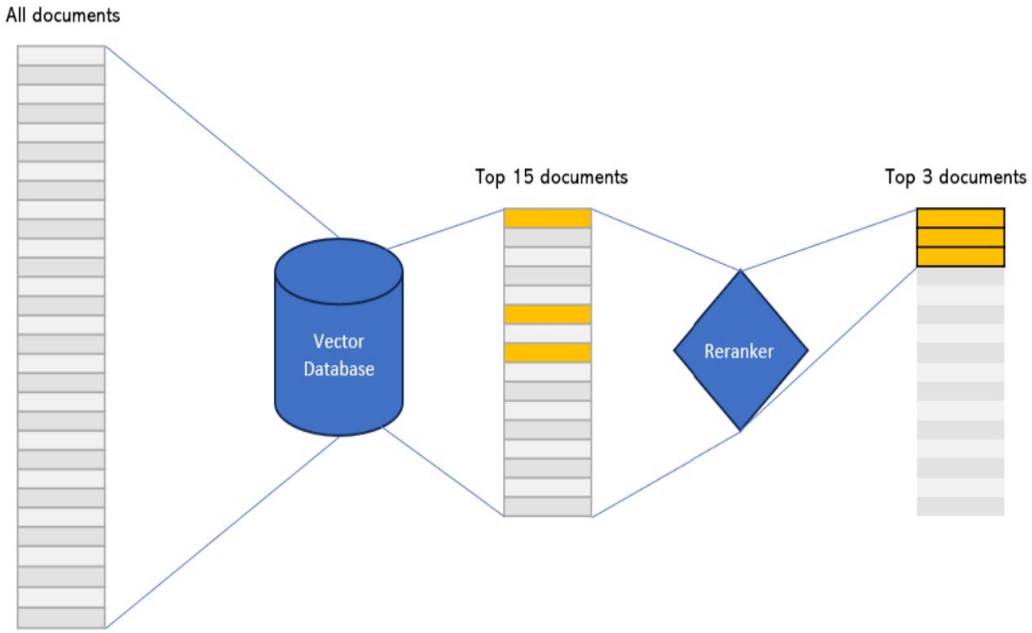


To learn more about this: https://www.deeplearning.ai/short-courses/building-applications-vector-databases/

RAG - rerankers

Some information loss occurs as the data is compressed into a singular vector - not uncommon that the top-ranked documents in search (e.g., the top three documents) may overlook pertinent information.

Addressing this issue may involve expanding the set of retrieved documents in the search and subsequently implementing a **reranking process**.



This is typically done by **cross-encoders**, a model that, given a query and document pair, will output a similarity score.

RAG - assessment

- Need to assess both the retriever and the generator
- Evaluating the retriever involves checking the ability to find relevant information from external sources- recall, precision, relevancy
- Generator can be evaluated by the capacity to create consistent and relevant responses, given the context
- Important to have quantitative metrics with defined cases - many times involves manual curation of prompts
- Some relevant metrics:
  - faithfulness (accuracy of the generated answer against the retrieved context)
  - answer relevance (if the generated answer addresses the question that was provided)
  - context relevancy and recall (recovers all relevant context minimizing redundancy)
  - answer semantic similarity (to the ground truth)

Relevant frameworks for RAG evaluation: RAGAS and ARES:

- Ragas: Automated evaluation of retrieval augmented generation. ArXiv, abs/2309.15217, 2023.
- Ares: An automated evaluation framework for retrieval-augmented generation systems. ArXiv, abs/2311.09476, 2023.

Agents

- LLM agents seek to give the ability to integrate LLMs with external tools and services.
- When building LLM-based agents, the LLM serves as the central controller, or ”brain”, orchestrating a sequence of operations necessary to fulfil a task or user request.
- This central role involves interpreting the user’s input and determining the appropriate actions/tools to be executed
- When integrated with domain- specific tools, these agents can perform complex operations beyond their native capabilities, providing deeper insights, more accurate predictions, and customized solutions.

Agents - architecture

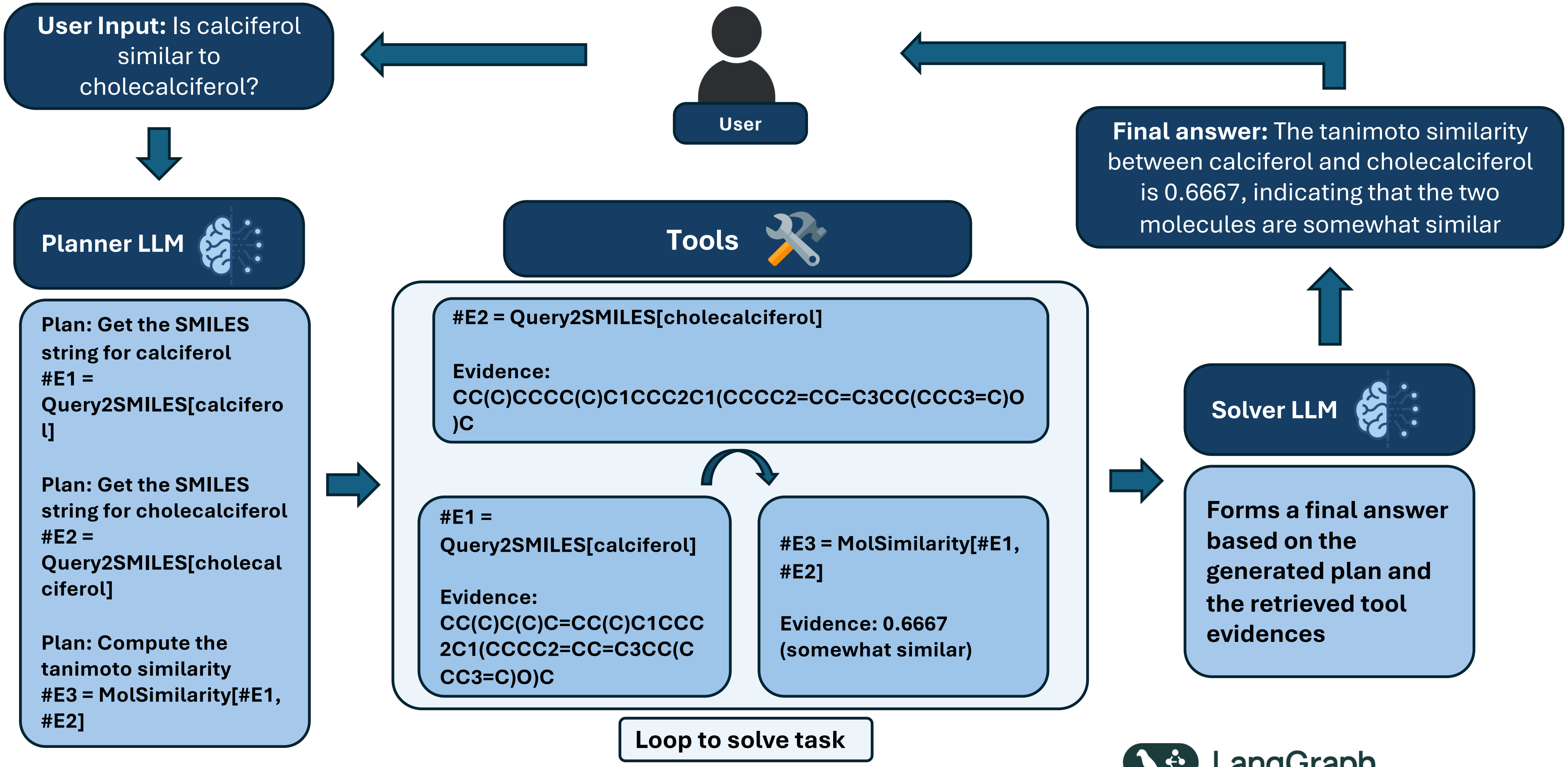
- To solve complex tasks, humans typically deconstruct them into subtasks and tackle them individually.
- Inspired by this human capability, most agent architecture designs aim to empower agents with a similar ability to break down and systematically address complex tasks.
- This enables the agent to create a sequence of actions or decisions, ensuring that each step contributes to the completion of the overall task.
- Existing studies on agent architecture design can be summarized on the basis of whether the agent can receive **feedback** during the planning process.

Agents - architecture

Planning without feedback:

- LLM operates without receiving input from external tools after executing them. Consequently, the outputs of these tools cannot shape future decision-making processes, potentially leading to less adaptive or suboptimal strategies.
- **Reasoning without Observation (REWOO)** - separates plans from external observations, where agents first generate plans and obtain observations independently, and then combine them to derive the final results
- Planning without feedback may present some obstacles. In many real-world scenarios, agents must make long-term planning to solve complex tasks.

Agents – an example with REWOO



User Input: Is calciferol similar to cholecalciferol?

User

Planner LLM

Plan: Get the SMILES string for calciferol
#E1 = Query2SMILES[calciferol]

Plan: Get the SMILES string for cholecalciferol
#E2 = Query2SMILES[cholecalciferol]

Plan: Compute the tanimoto similarity
#E3 = MolSimilarity[#E1, #E2]

Tools

#E2 = Query2SMILES[cholecalciferol]

Evidence: CC(C)CCCC(C)C1CCC2C1(CCCC2=CC=C3CC(CCC3=C)O)C

#E1 = Query2SMILES[calciferol]

Evidence: CC(C)C(C)C=CC(C)C1CCC2C1(CCCC2=CC=C3CC(CCC3=C)O)C

#E3 = MolSimilarity[#E1, #E2]

Evidence: 0.6667 (somewhat similar)

Loop to solve task

Solver LLM

Forms a final answer based on the generated plan and the retrieved tool evidences

Final answer: The tanimoto similarity between calciferol and cholecalciferol is 0.6667, indicating that the two molecules are somewhat similar

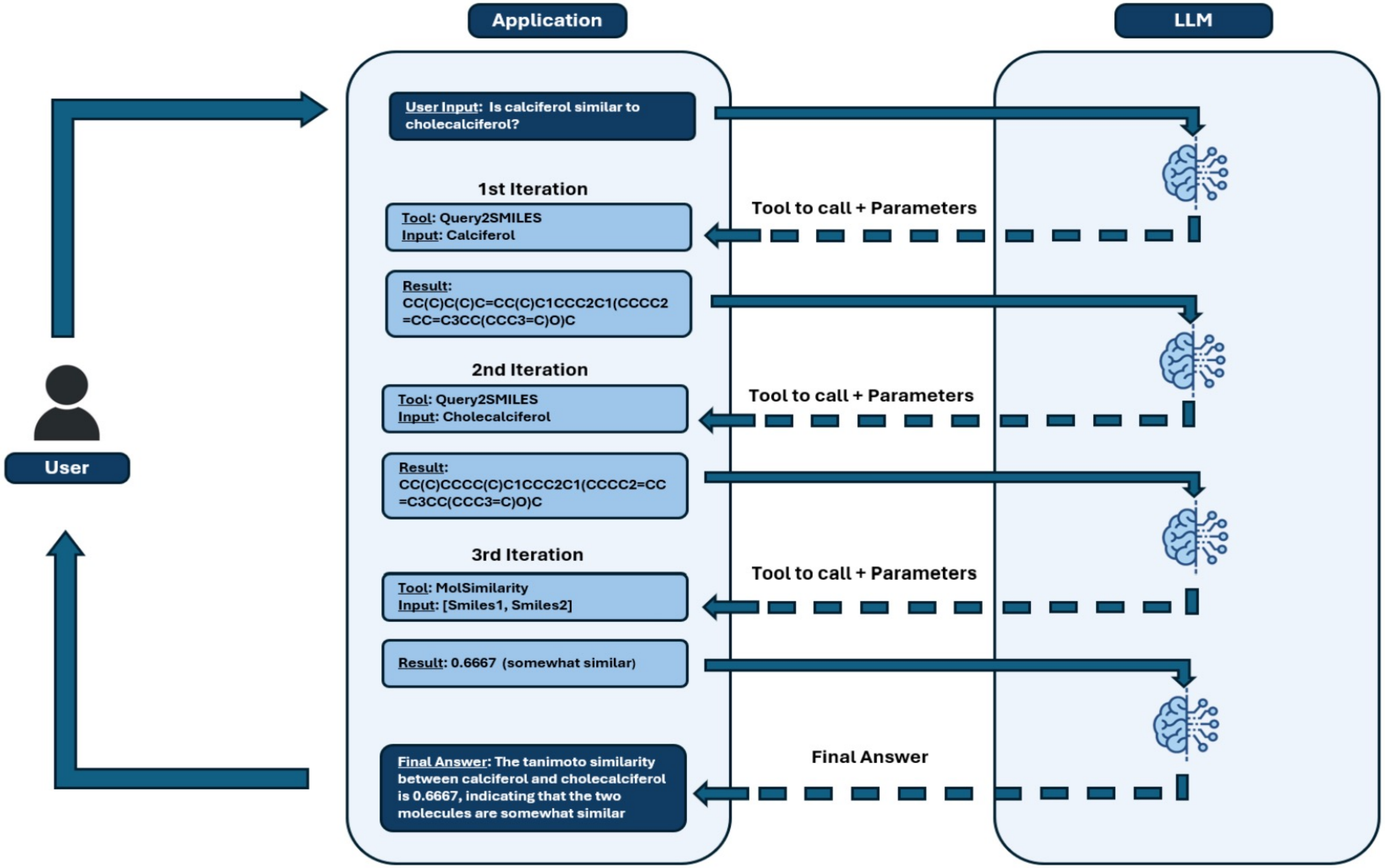
LangGraph

Agents - architecture

Planning with feedback:

- Enables LLM-based agents to adapt and improve their actions based on external inputs and signals. This can be an advantage, especially when examining how humans tackle complex tasks, iteratively making and changing their plans based on external feedback
- **Reasoning and acting (ReAct)** - operates on a cycle of action generation, and the respective observation/result for that action. Initially, the LLM chooses an action to take based on its understanding of the task and context. These actions are then executed, and feedback is collected from the action result itself
- The feedback signals are used to refine the agent’s understanding and decision-making process. The agent analyzes the feedback, identifies areas for improvement, and adjusts its action generation strategy accordingly. This iterative cycle continues, allowing the agent to continuously learn and optimize its actions over time

Agents – an example with ReAct



User Input: Is calciferol similar to cholecalciferol?

1st Iteration

Tool: Query2SMILES
Input: Calciferol

Result: CC(C)C(C)C=CC(C)C1CCC2C1(CCCC2=CC=C3CC(CCC3=C)O)C

2nd Iteration

Tool: Query2SMILES
Input: Cholecalciferol

Result: CC(C)CCCC(C)C1CCC2C1(CCCC2=CC=C3CC(CCC3=C)O)C

3rd Iteration

Tool: MolSimilarity
Input: [Smiles1, Smiles2]

Result: 0.6667 (somewhat similar)

Final Answer: The tanimoto similarity between calciferol and cholecalciferol is 0.6667, indicating that the two molecules are somewhat similar

Tool to call + Parameters
Tool to call + Parameters
Tool to call + Parameters
Final Answer

Paulo Seixal – MSc thesis - 2024

Some useful libraries for LLMs

- **AI suite** (https://github.com/andrewyng/aisuite) - use multiple LLM through a standardized interface
- **Langchain** (https://github.com/langchain-ai; https://github.com/langchain-ai/langchain; https://github.com/langchain-ai/langgraph) - framework for building LLM-powered applications (RAG, agents, etc.)
- **PromptBench** (https://github.com/microsoft/promptbench) – evaluation of LLMs (pytorch)
- **GraphRAG** (https://github.com/microsoft/graphrag) – extraction of information from texts
- **Argilla** (https://github.com/argilla-io/argilla) – building datasets to evaluate
- **HuggingFace** (https://huggingface.co/docs/hub/index) – model repository and many other resources