LangChain for Generative AI

Bruno Gonçalves

graphs4sci.substack.com

https://github.com/DataForScience/LangChain

Events

data4sci.substack.com

Generative AI with the OpenAI API for Developers
Oct 9, 2024 - 10am-2pm (PST)

ChatGPT and Competing LLMs
Nov 13, 2024 - 10am-2pm (PST)

Bruno Gonçalves

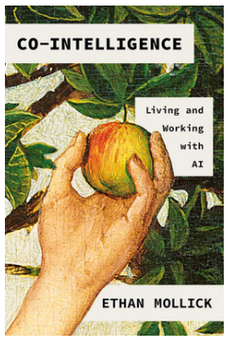
https://data4sci.com

info@data4sci.com

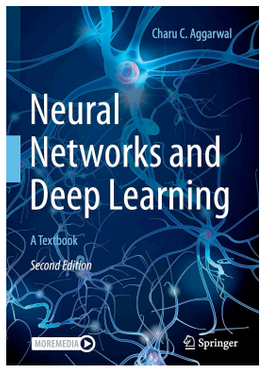
https://data4sci.com/call

References

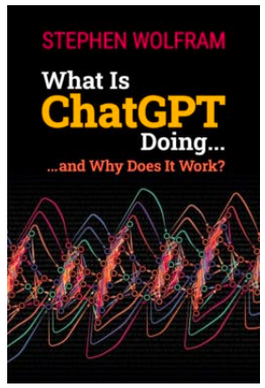
https://github.com/DataForScience/LangChain



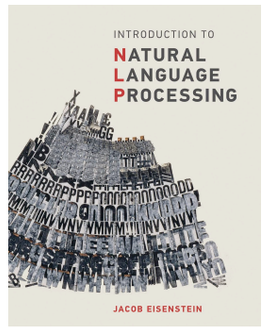
https://amzn.to/3KcLlsf



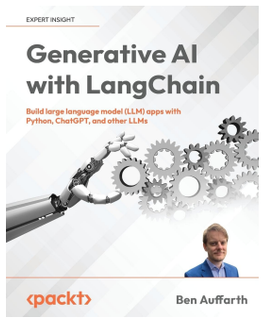
https://amzn.to/48rZn9X



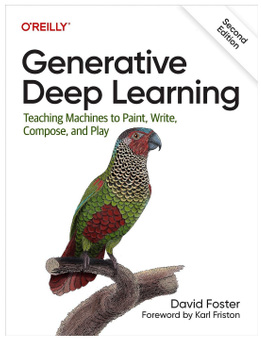
https://amzn.to/3LBRdBY



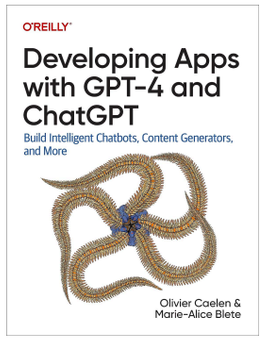
https://amzn.to/3ZMnTih



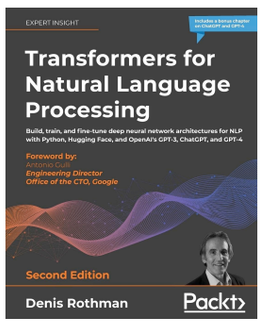
https://amzn.to/3VquiPj



https://amzn.to/3t8PuxM



https://amzn.to/3RHRkQa

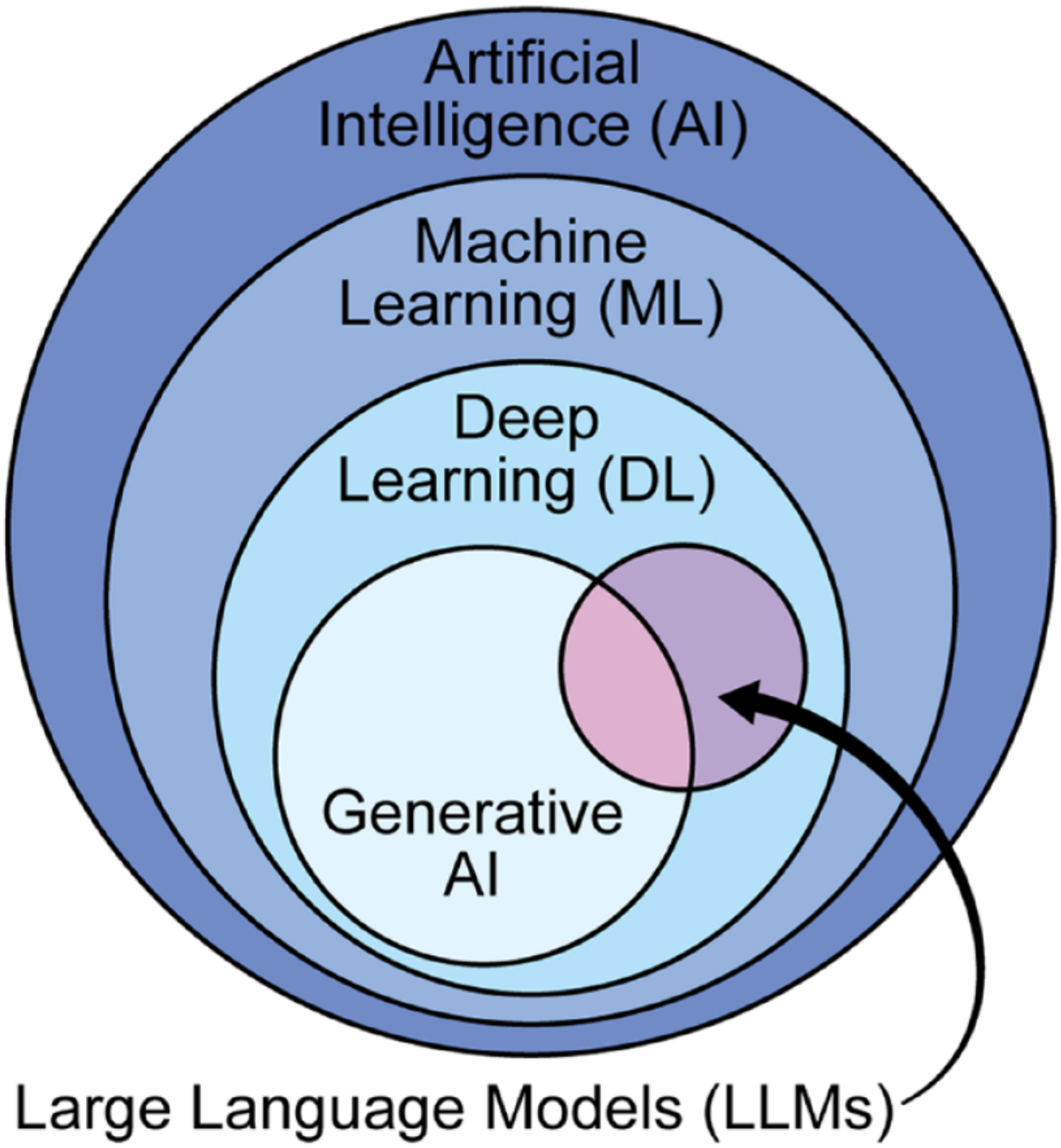


https://amzn.to/46kOo02

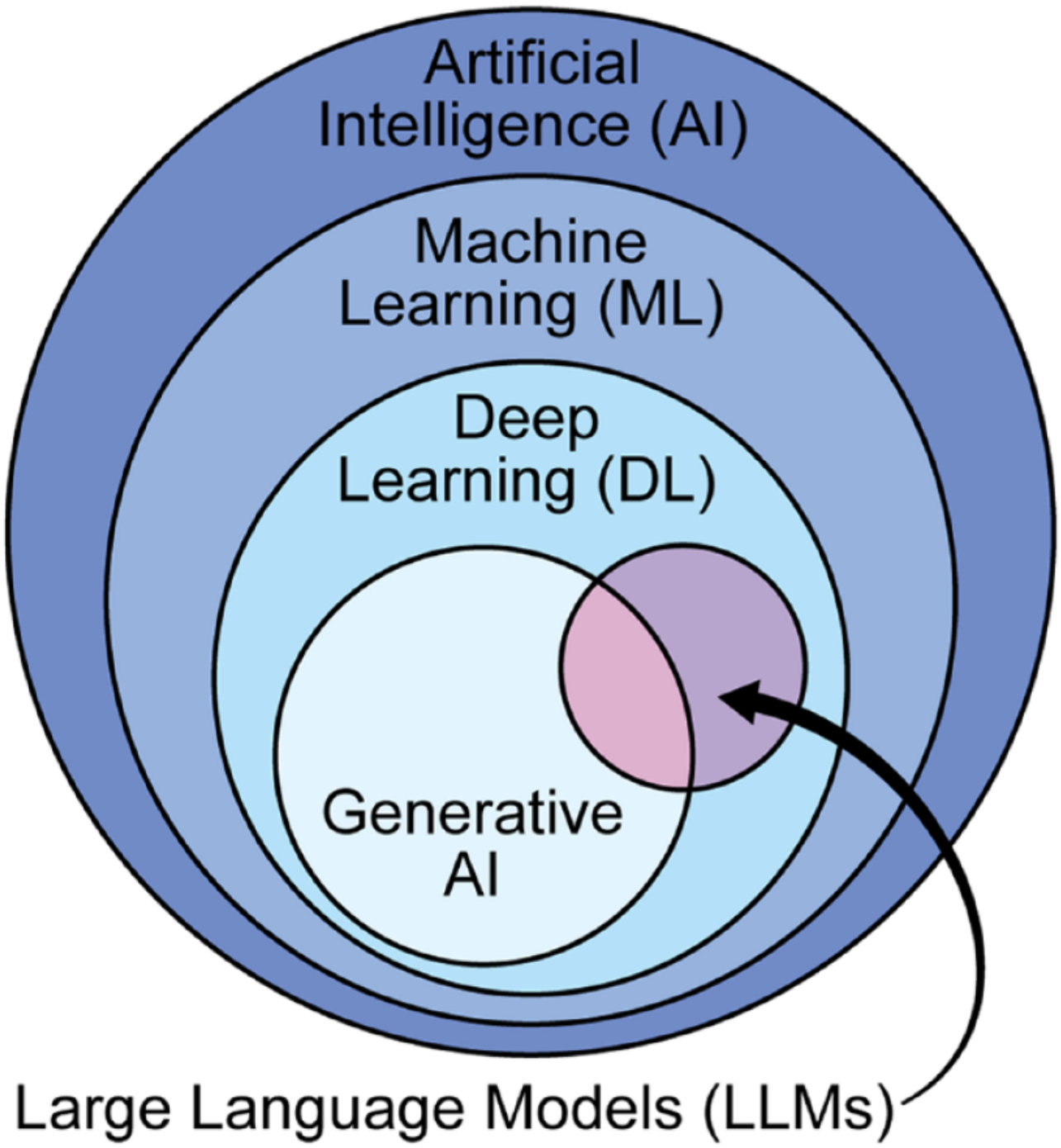
Table of Contents

1. Generative AI

Generative Models

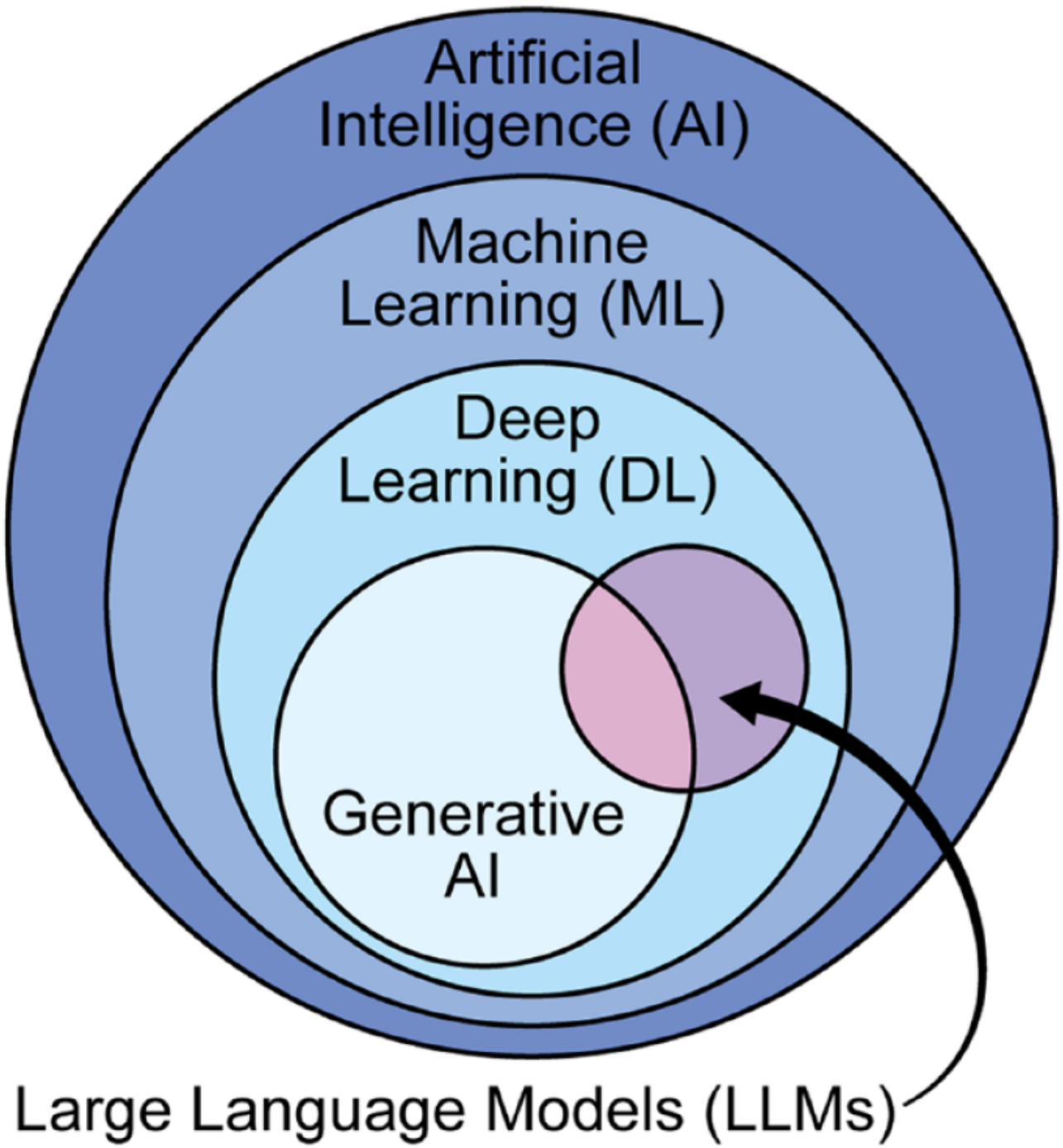


Generative Models



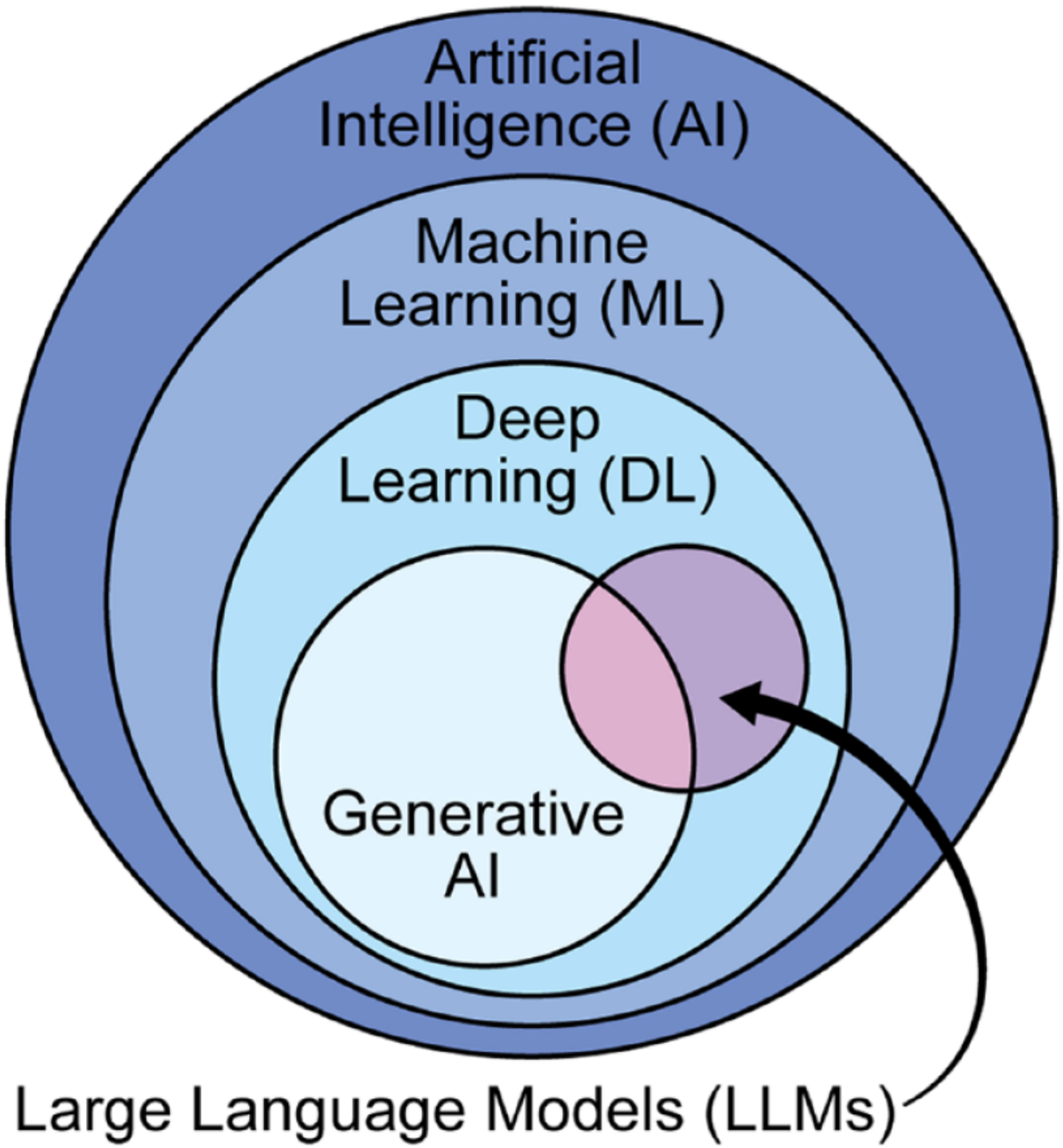
- Artificial Intelligence (AI): A field of CS focused on creating intelligent agents.

Generative Models



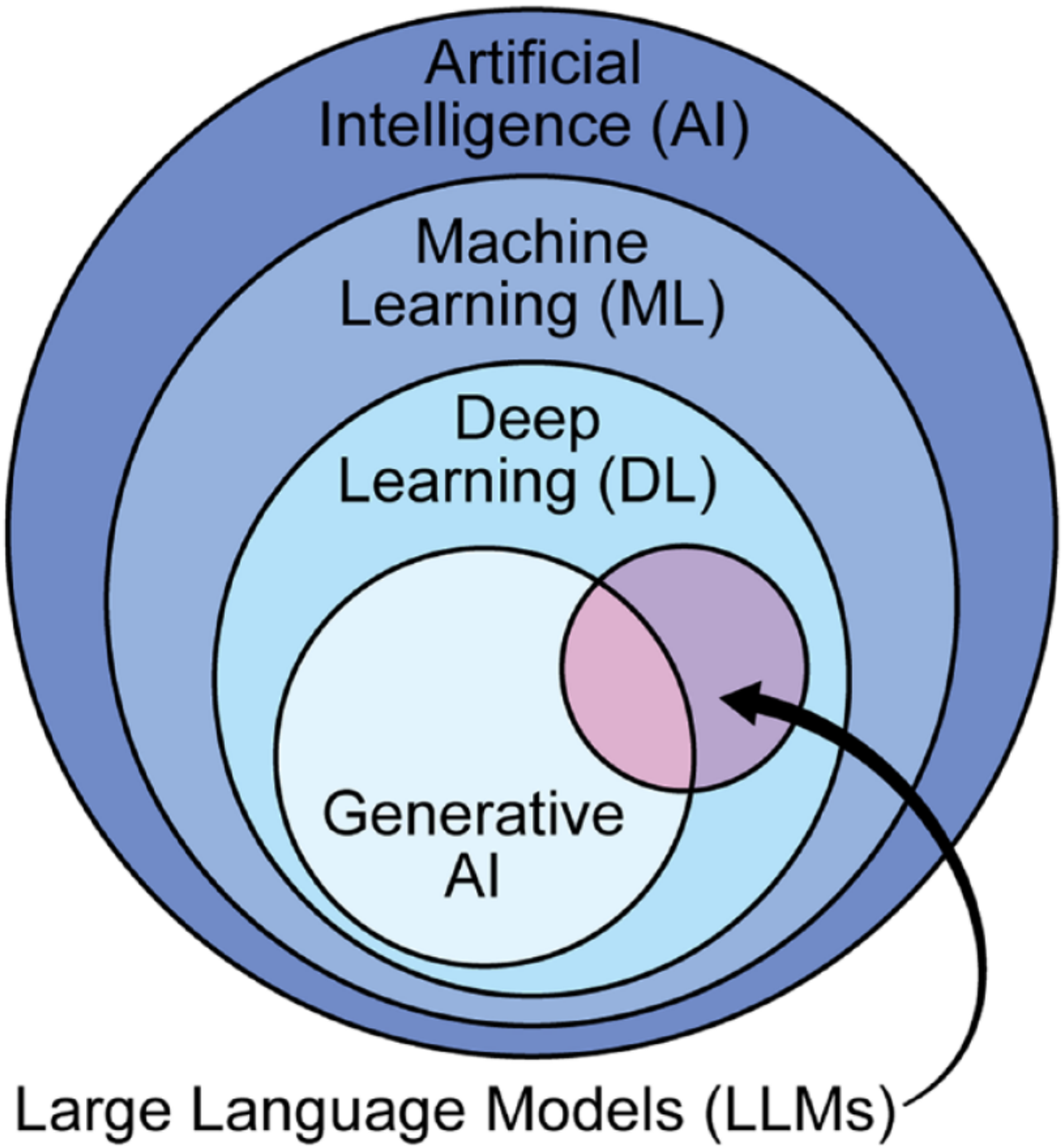
- Artificial Intelligence (AI): A field of CS focused on creating intelligent agents.
- Machine Learning (ML): Subset of AI focused on developing algorithms that can learn from data.

Generative Models



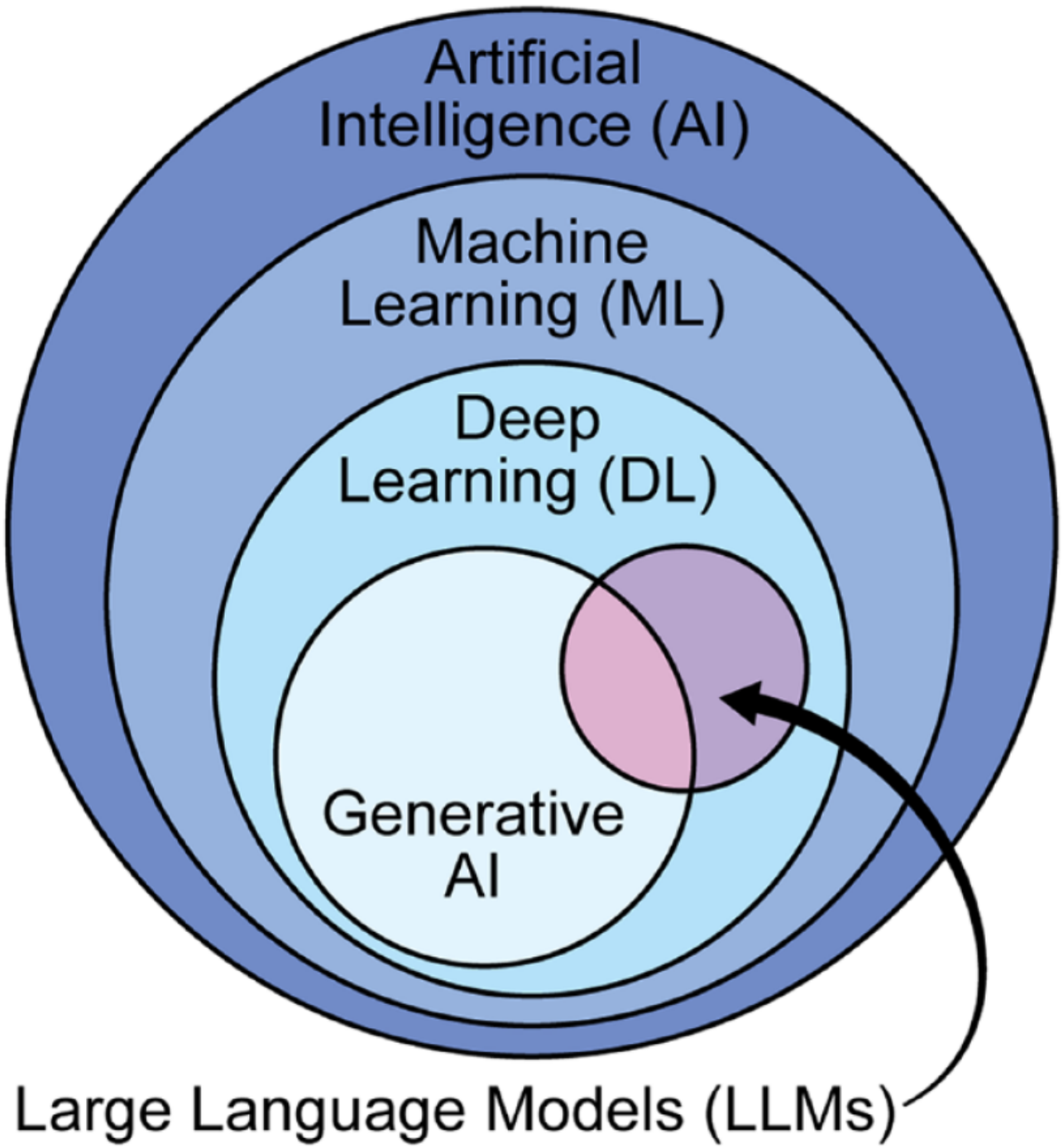
- Artificial Intelligence (AI): A field of CS focused on creating intelligent agents.
- Machine Learning (ML): Subset of AI focused on developing algorithms that can learn from data.
- Deep Learning (DL): Neural networks with many layers

Generative Models



- Artificial Intelligence (AI): A field of CS focused on creating intelligent agents.
- Machine Learning (ML): Subset of AI focused on developing algorithms that can learn from data.
- Deep Learning (DL): Neural networks with many layers
- Generative Models: Models that can generate new data based on learned patterns

Generative Models



- Artificial Intelligence (AI): A field of CS focused on creating intelligent agents.
- Machine Learning (ML): Subset of AI focused on developing algorithms that can learn from data.
- Deep Learning (DL): Neural networks with many layers
- Generative Models: Models that can generate new data based on learned patterns
- Large Language Models (LLMs): Statistical models used to predict words in natural language

Generative Models

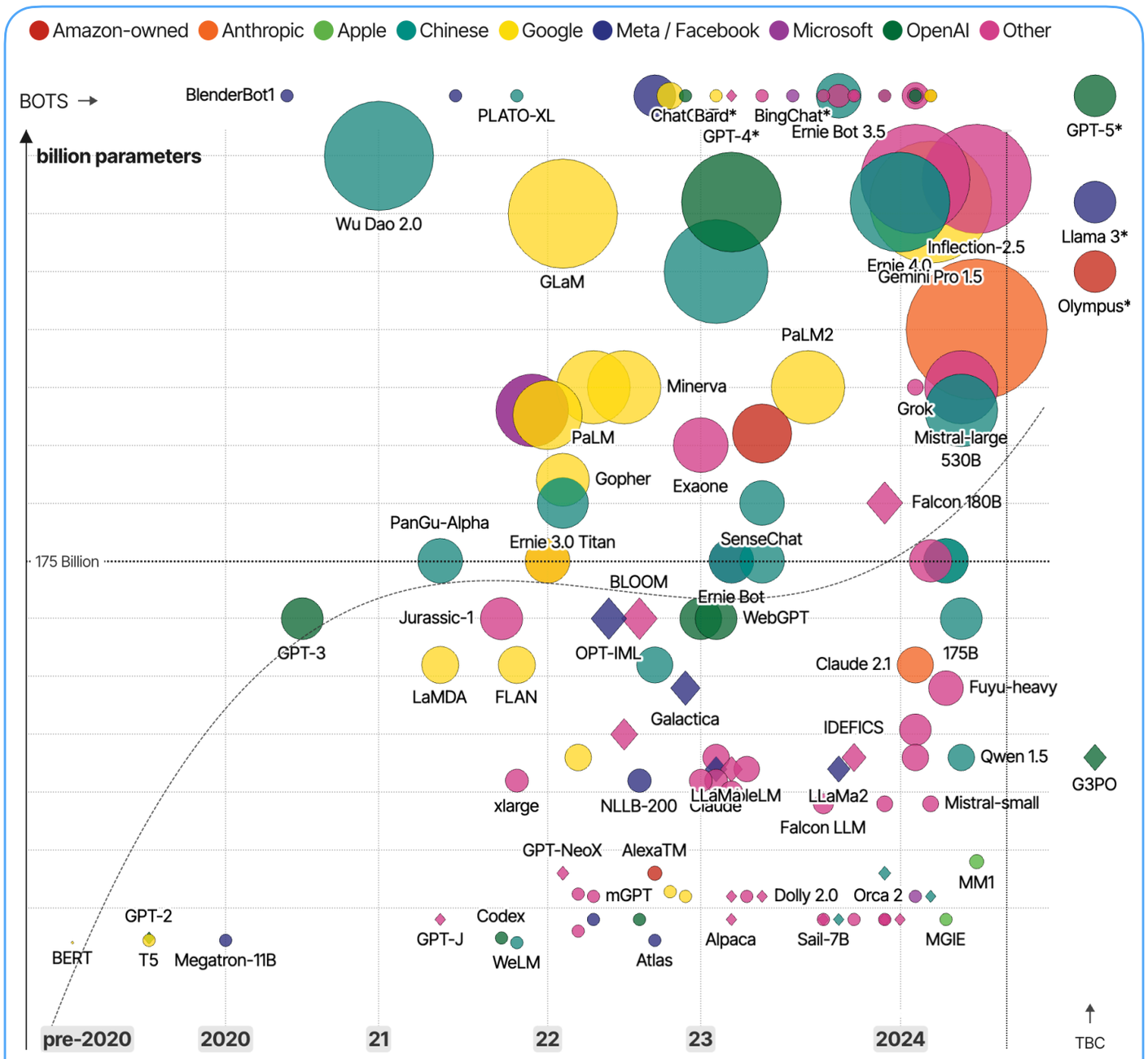
- Come in many flavors, depending on what the input and outputs are
    - Text-to-Text - Dialog based systems, like ChatGPT
    - Text-to-Image - Capable of generating images based on prompts, like DALL-E
    - Image-to-Text - Capable of generating text that describes an image
    - Text-to-Speech - Generate audio from written text
    - Speech-to-Text - Transcribe audio into text
    - Text-to-Code - Generate programming code based on a problem descriptions, like GitHub CoPilot
    - Image-to-Image - Automatically image editing and manipulation

Language Models

- A foundational component of Natural Language Processing and AI.
- Software that predicts and generates human language based on patterns observed in training data.
- Applications to:
    - Machine translation
    - Summarization
    - Sentiment analysis
    - Chatbots and Virtual Assistants
    - Content Generation
    - etc…

LLM Comparison

https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/



size = no. of parameters

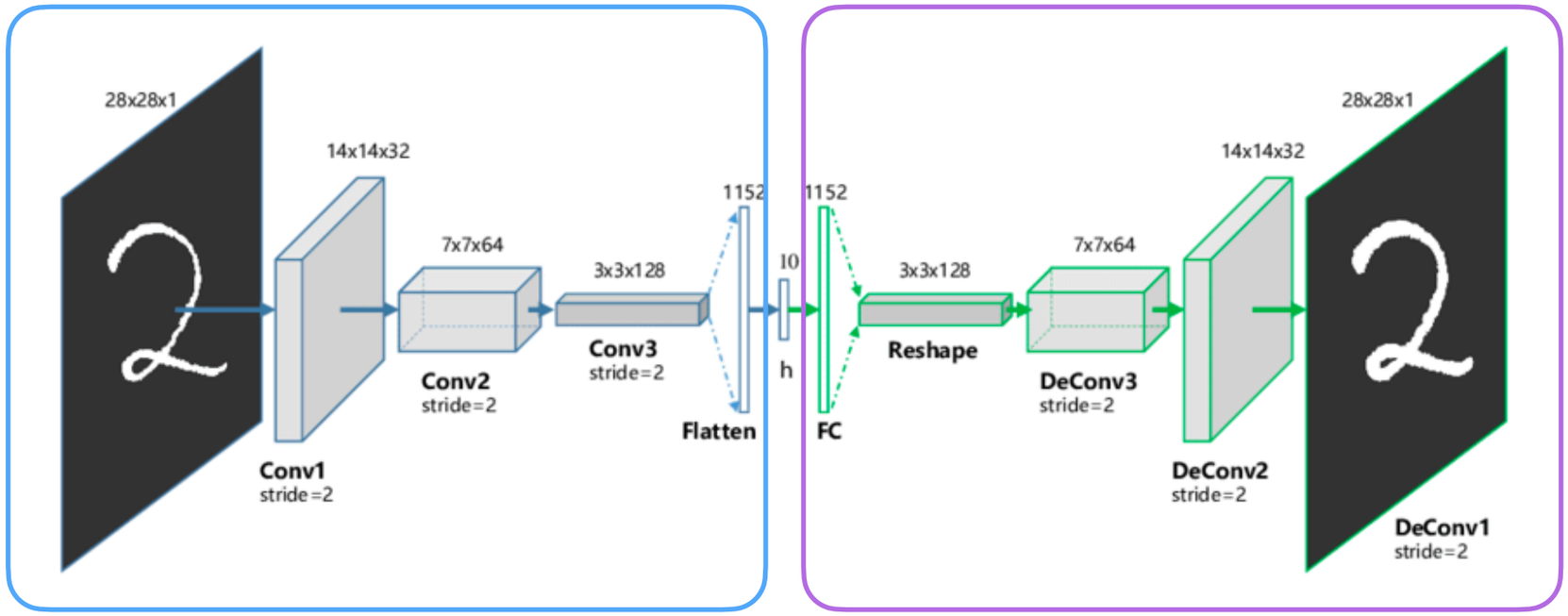
open-access

David McCandless, Tom Evans, Paul Barton
Information is Beautiful // UPDATED 20th Mar 24

source: news reports, LifeArchitect.ai
* = parameters undisclosed // see the data

MADE WITH VIZsweet

Encoder/Decoder Architecture



Encoder

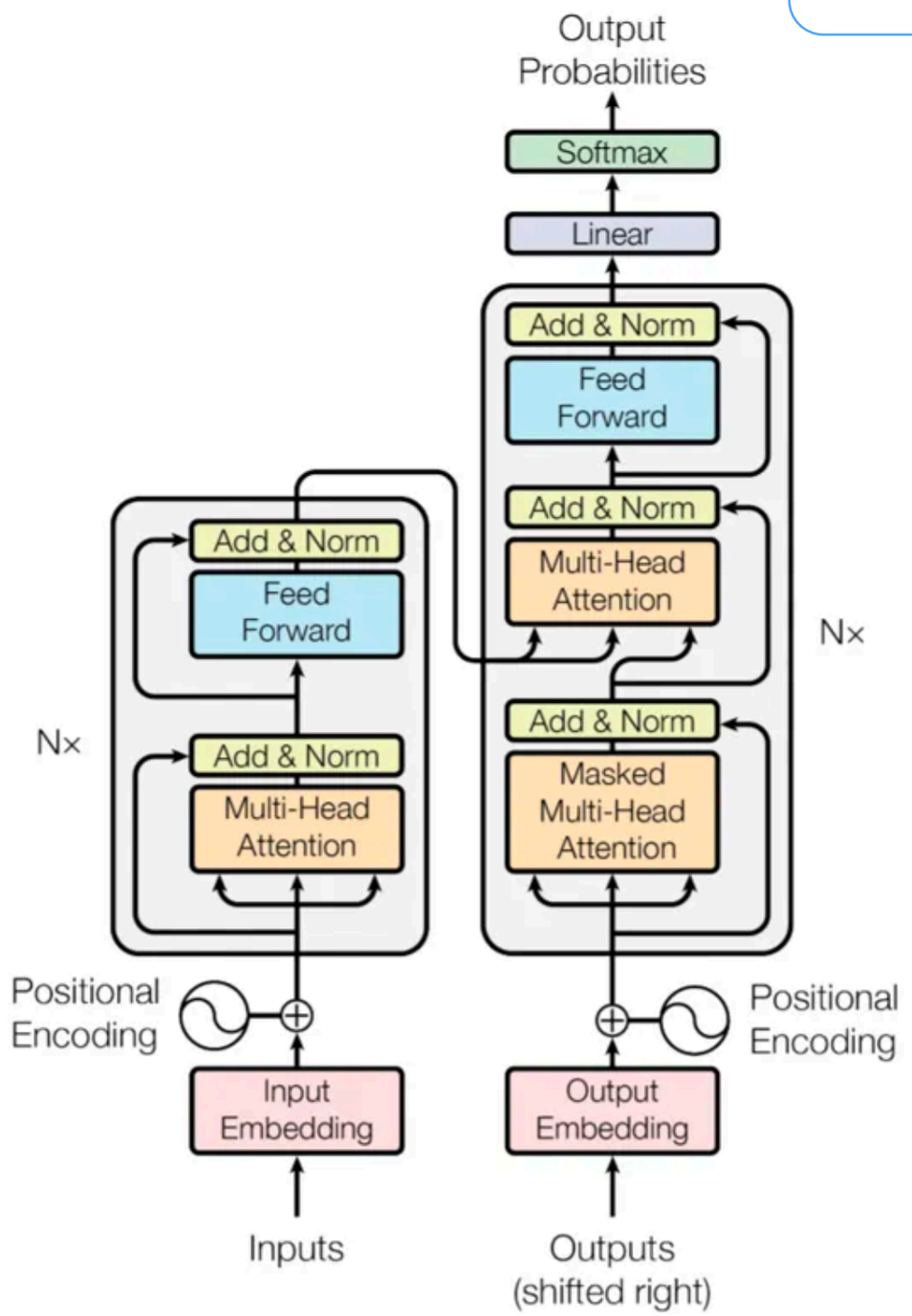
Input -> Internal Representation

Decoder

Internal Representation -> Output

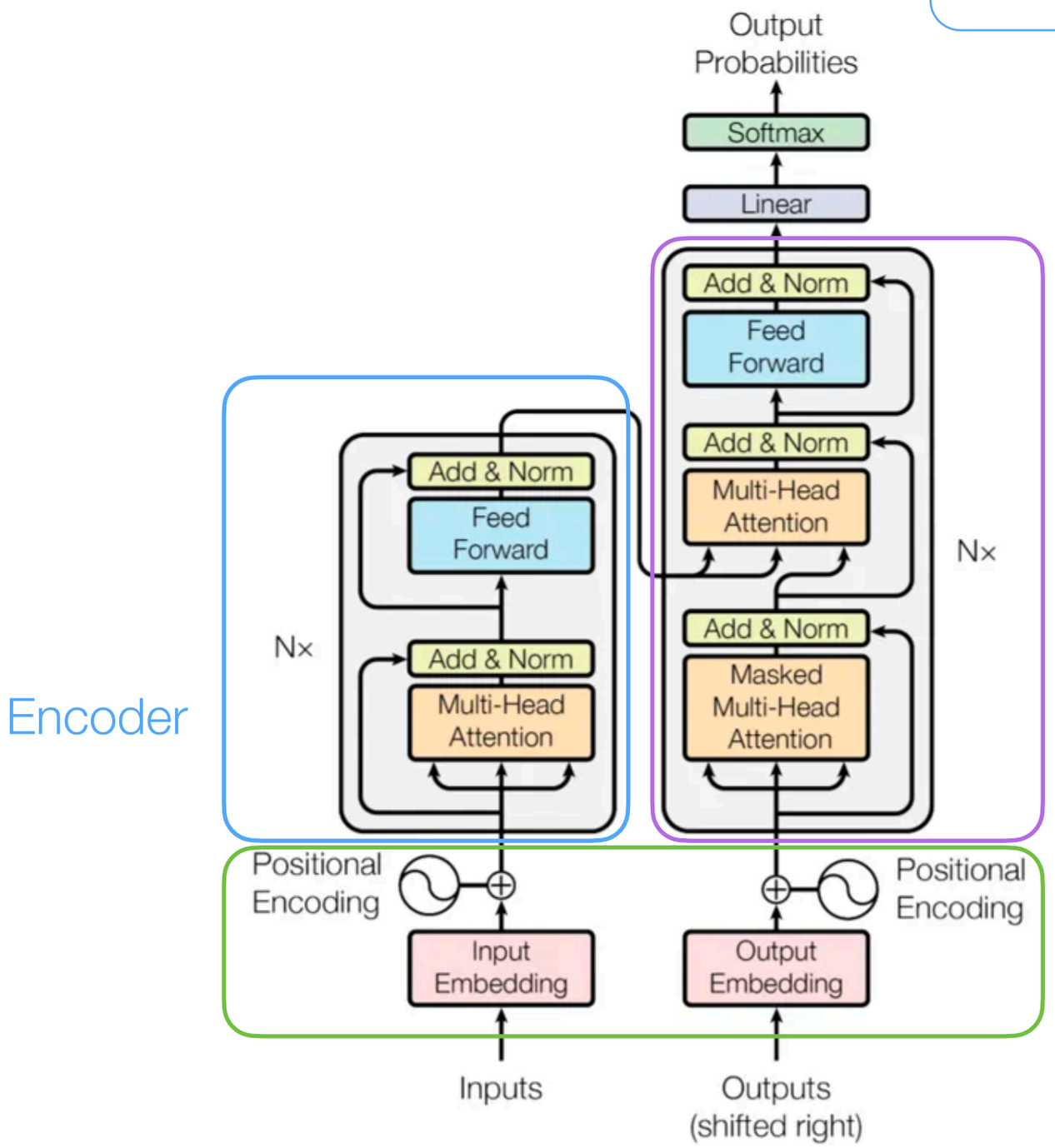
Transformers

arXiv:1706.03762



Transformers

arXiv:1706.03762



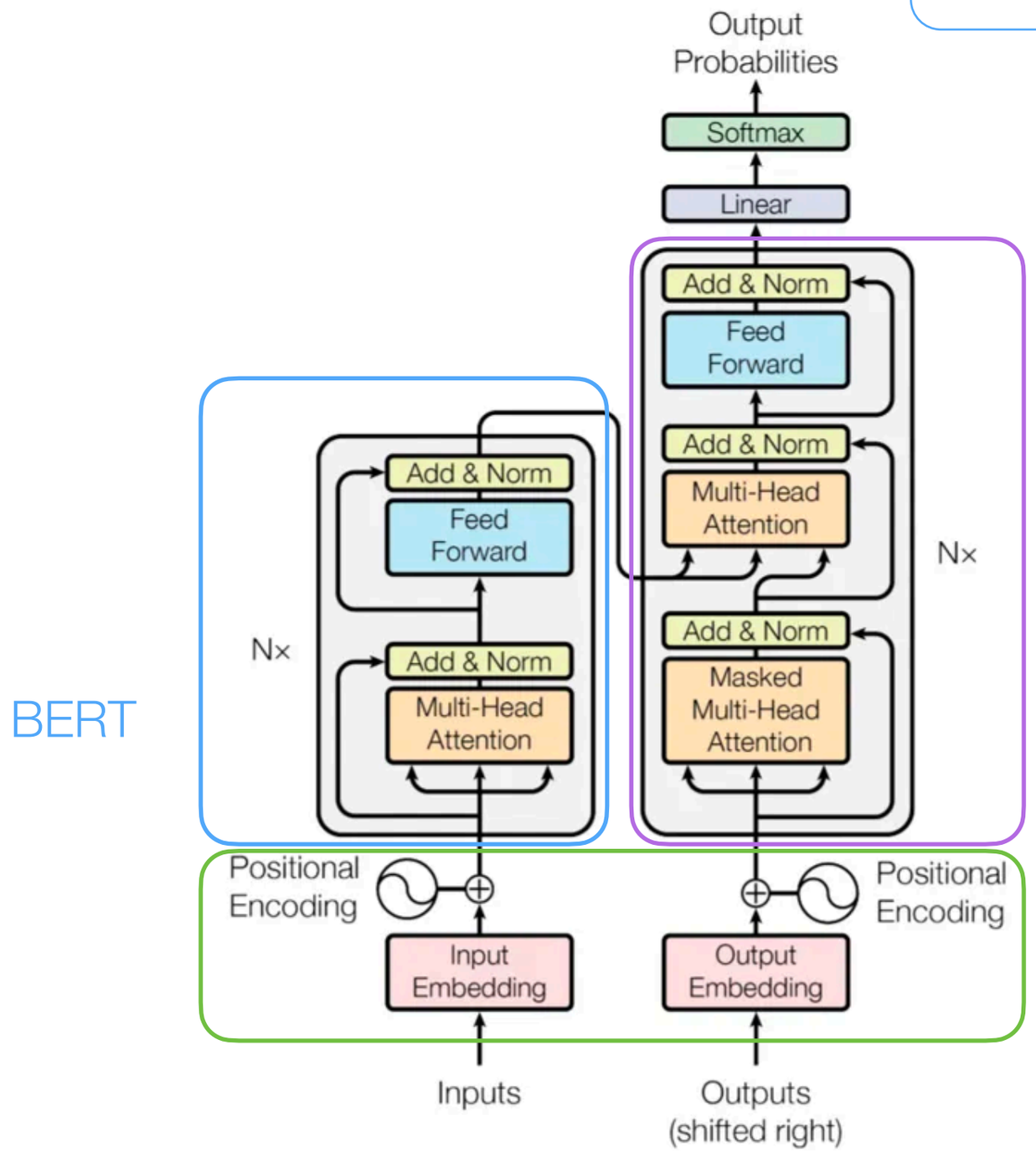
Encoder

Decoder

Preprocessing

Transformers

arXiv:1706.03762



BERT

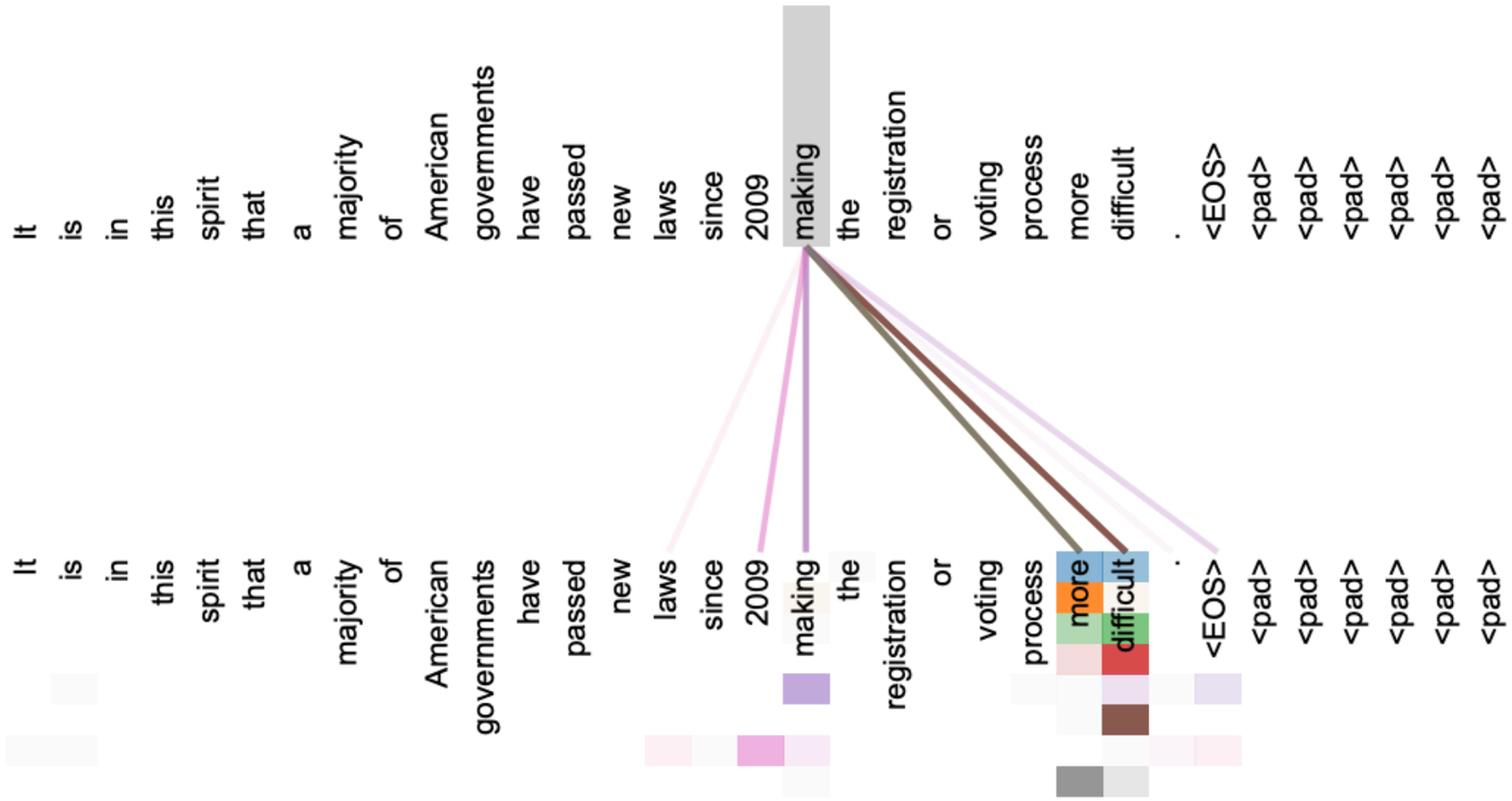
GPT

Preprocessing

Attention

arXiv:1706.03762

- "Simple" mechanism to allow each token to take the context it appears in into account
- Requires exponentially more weights to be computed (from each token to every other token)

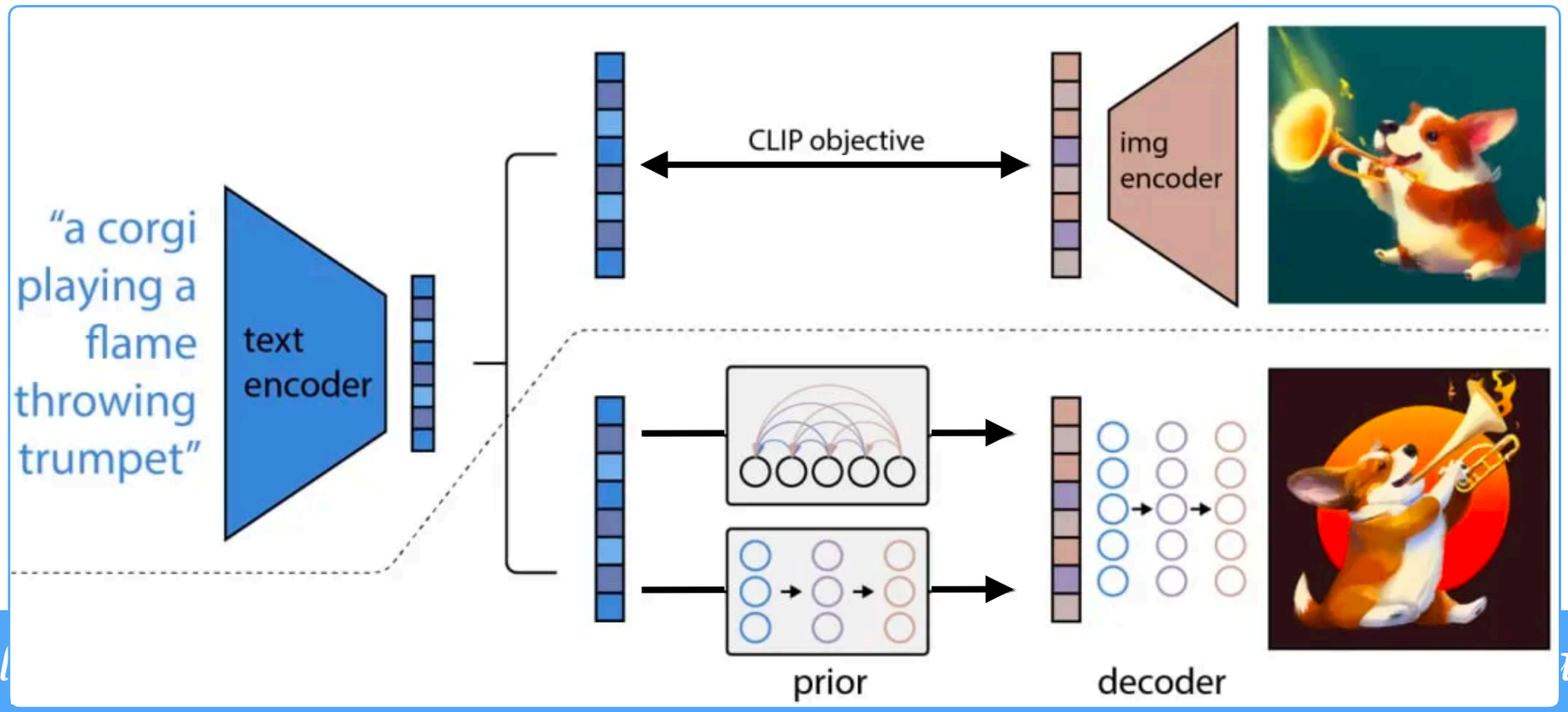


- Weights indicate the importance of each word relative to the current one

Text to Image Models

arXiv:2204.06125

- Image models like DALL-E build on top of the LLM approach outlined above
- Essentially, DALL-E replaces the LLM decoder with an image decoder
- The image decoder is trained using Contrastive Training:
  - A large list of images and textual descriptions is used to train the image decoder to generate an internal representation that matches the encoded description
  - Inverting the process produces a model that converts textual descriptions into images
- As a final improvement, we add a prior in between the output of the text encoder and the input of the image decoder.



Code - Generative AI

https://github.com/DataForScience/LangChain

2. LangChain

LangChain

- LangChain is an open-source Python framework designed to simplify the development of applications using large language models (LLMs).

- Main features:
  - Model Abstraction: LangChain manages inputs and outputs seamlessly by abstracting away model details

  - Modular Structure: Allows developers to combine different prompts and even multiple LLMs within a single application.

  - Chains: Multiple components can be chained together to create complex applications

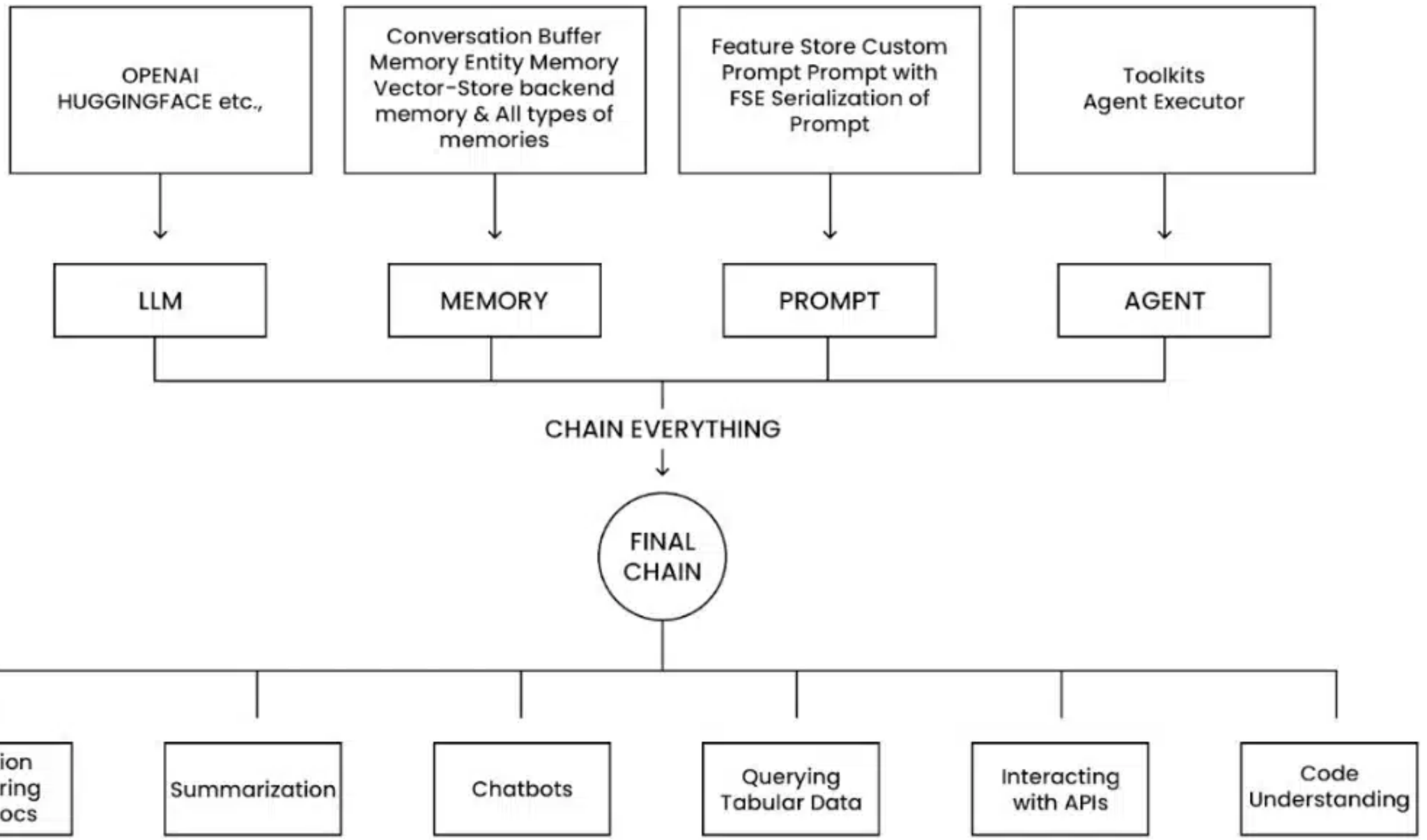
  - Data Integration: Allows LLMs to transform, store, and retrieve data from databases and online sources.

  - Memory Management: Provides utilities for adding memory to LLM systems, retaining conversation history or summaries for context.

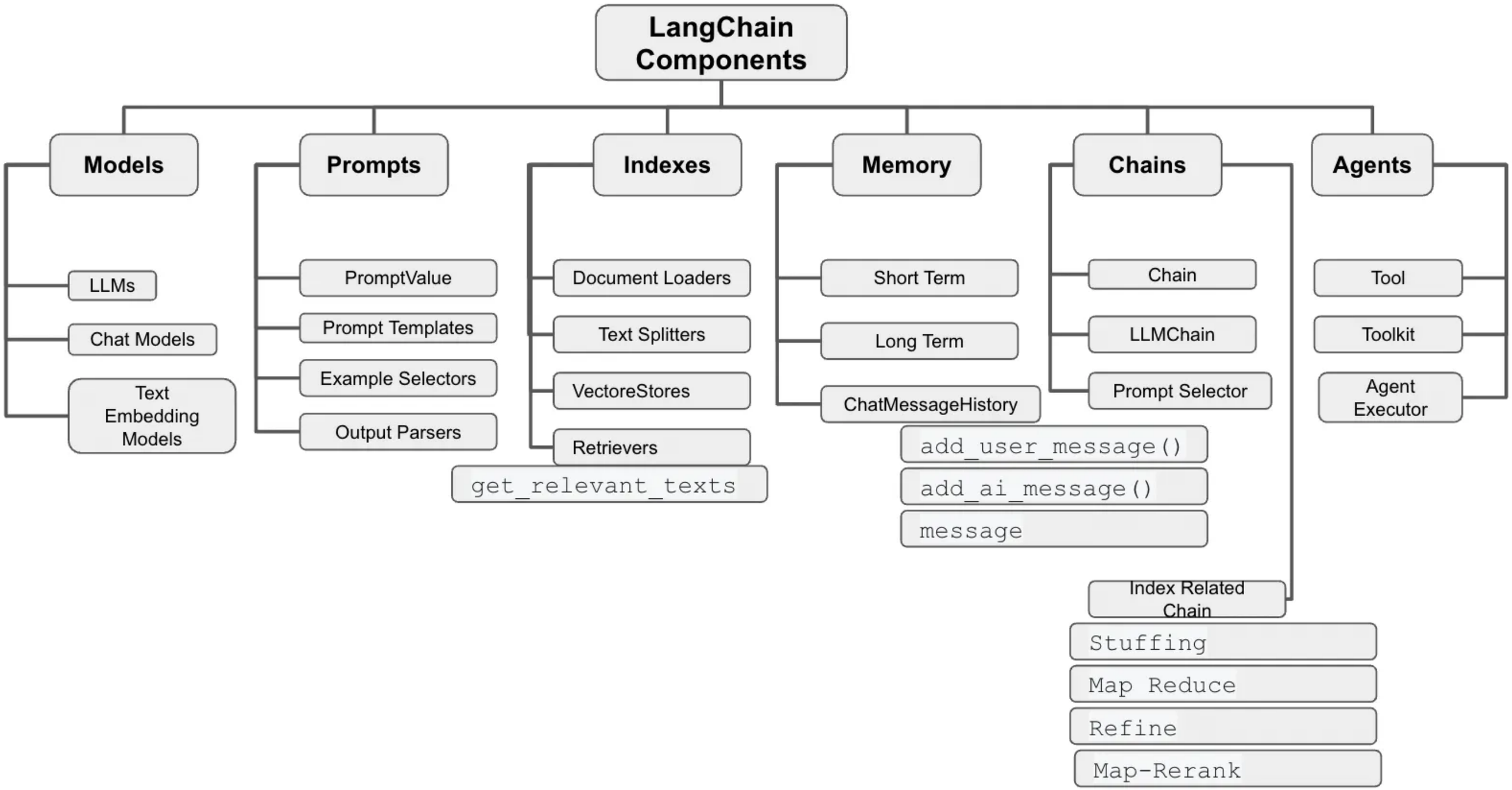
LangChain Applications

- LangChain can be used to build a wide range of LLM-powered applications:
    - Chatbots
    - Coding assistants and code security analysis
    - Recommendation systems
    - Document analysis and summarization
    - Question-answering systems
    - Data augmentation
    - Text classification and summarization
    - Sentiment analysis
    - Machine translation

LangChain Structure



LangChain Structure



Prompts

- Prompts are the basic way to interact with an LLM model. LangChain provides different classes to manage and generate prompts

    - PromptValue - Represents the text input to a model

    - PromptTemplate - Jinja-like templating engine for a prompt. It takes input variables and returns a PromptValue. Used for HumanMessages, SystemMessages, and AIMessages.

    - StrOutputParser - Helps structure language model responses

Chains

- A Chain is a sequence of calls to components, including other chains.
- Conceptually similar to sklearn Pipelines
- Component Examples:
    - LLMMath - For math related queries
    - SQLDatabaseChain - To query databases
    - OpenAIModerationChain - Check content against moderation rules
    - ConstitutionalChain - Legal applications
    - LLMCheckerChain - Prevent hallucinations

Advantages of Chains

- Modularity - Each component is an isolated module
- Composability - Modules can be easily connected together
- Readability - Each step in the chain is clear
- Maintainability - The sequence of steps can be easily expanded or changed
- Reusability - Common functionality becomes chains
- Tool Integration - Integrate LLMs, databases, APIs, etc
- Productivity - Facilitates prototyping and development

Agents

- Agents are autonomous software that are capable of taking actions to accomplish specific goals and tasks
- Agents leverage chains to take goal-driven actions and can combine and orchestrate chains
- Advantages of agents:
    - Goal-oriented execution: Plan chains of logic targeting specific goals.
    - Dynamic responses: React and adapt to environment changes.
    - Statefulness: Maintain memory and context.
    - Robustness: Error handling.
    - Composition: Combine reusable component chains.

Memory

- A key limitation of agents and chains is their statelessness
- Memory refers to the persisting state between executions of a Chain or Agent.
- Several versions:
    - ConversationBufferMemory - store all messages
    - ConversationBufferWindowMemory - retain only recent messages
    - ConversationKGMemory - Summarizes exchanges as a knowledge graph
    - EntityMemory - persists agent state and facts through a database
    - SQLDatabase - Database integration

Using tools

https://python.langchain.com/v0.2/docs/integrations/tools/

- Tools are interfaces that Agents can use to interact with the world.
- Examples:
  - ShellTool - To run commands on the command line
  - DuckDuckGoSearchRun - To search on DDG
  - AlphaVantageAPIWrapper - Financial data
  - RequestsGetTool - Access a specific URL
  - WikipediaQueryRun - Query Wikipedia
  - YouTubeSearchTool - Query YouTube
  - And many more…

Code - LangChain

https://github.com/DataForScience/LangChain

3. Information Processing

Information processing

- Information processing is perhaps the killer application for LLMs

Information processing

- Information processing is perhaps the killer application for LLMs
- Large models ability to understand context and nuance lets them quickly and effectively extracting relevant information from various types of documents:
    - Unstructured text
    - Complex document formats (PDF, Word, JSON, etc)
    - Images with text (OCR)
    - mixed content formats.

Information processing

- Information processing is perhaps the killer application for LLMs
- Large models ability to understand context and nuance lets them quickly and effectively extracting relevant information from various types of documents:
    - Unstructured text
    - Complex document formats (PDF, Word, JSON, etc)
    - Images with text (OCR)
    - mixed content formats.
- LLMs can sift through large volumes of data to identify and isolate critical details, making them invaluable for tasks like data categorization, entity extraction, sentiment analysis, etc

Information processing

- Information processing is perhaps the killer application for LLMs
- Large models ability to understand context and nuance lets them quickly and effectively extracting relevant information from various types of documents:
    - Unstructured text
    - Complex document formats (PDF, Word, JSON, etc)
    - Images with text (OCR)
    - mixed content formats.
- LLMs can sift through large volumes of data to identify and isolate critical details, making them invaluable for tasks like data categorization, entity extraction, sentiment analysis, etc
- LLMs are able to work across multiple languages

Information processing

- Information processing is perhaps the killer application for LLMs
- Large models ability to understand context and nuance lets them quickly and effectively extracting relevant information from various types of documents:
    - Unstructured text
    - Complex document formats (PDF, Word, JSON, etc)
    - Images with text (OCR)
    - mixed content formats.
- LLMs can sift through large volumes of data to identify and isolate critical details, making them invaluable for tasks like data categorization, entity extraction, sentiment analysis, etc
- LLMs are able to work across multiple languages
- And more importantly, LLMs can be easily automated

Text Summarization

- Text summarization is a classical NLP problem and a common application for LLMs

Text Summarization

- Text summarization is a classical NLP problem and a common application for LLMs
- Two main types of summarization approaches:
    - Extractive Summarization: Identify and extract the most important sentences from a text and combine them to form a summary.
    - Abstractive Summarization: Generate new sentences that capture the essence of the original text.

Text Summarization

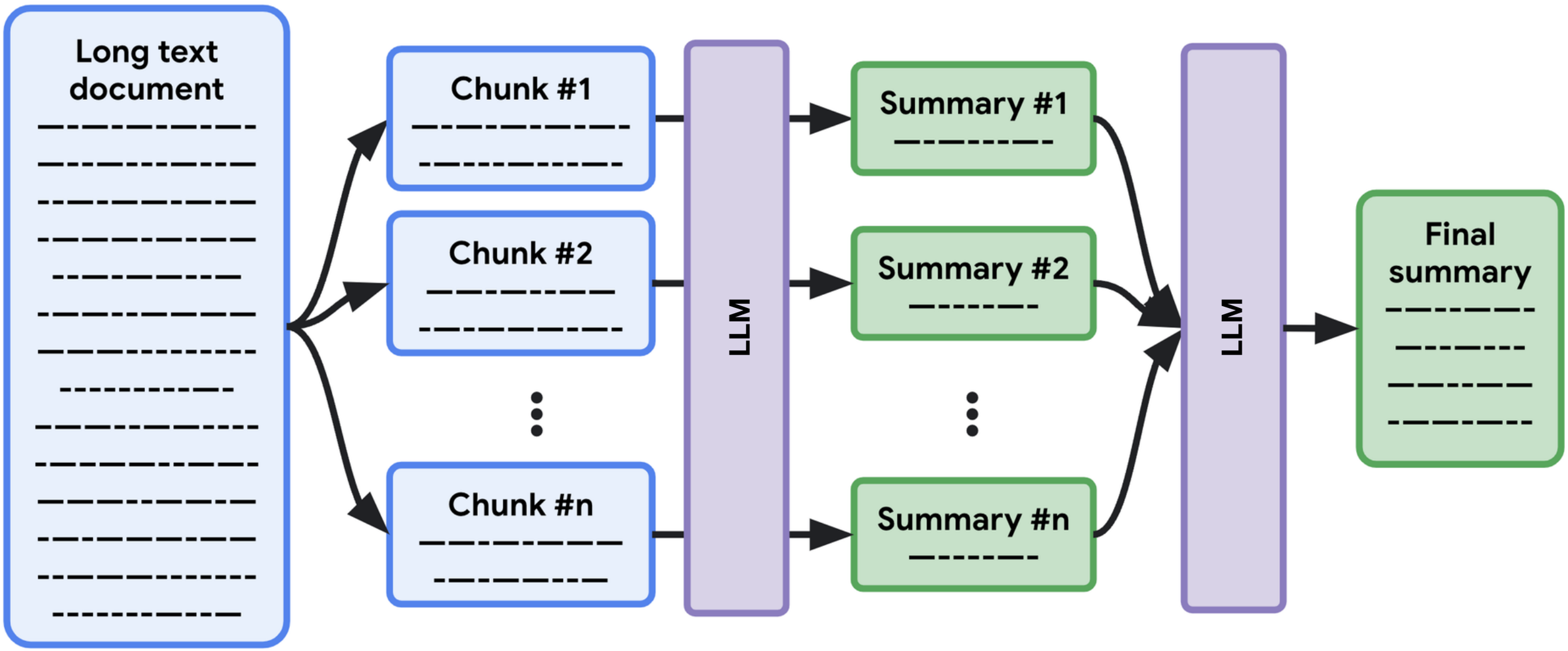
- Text summarization is a classical NLP problem and a common application for LLMs
- Two main types of summarization approaches:
  - Extractive Summarization: Identify and extract the most important sentences from a text and combine them to form a summary.
  - Abstractive Summarization: Generate new sentences that capture the essence of the original text.

- Several approaches are possible based on text size:
  - Document fits in context window: A basic Prompt is enough - "Write a summary of the following:"
  - Multiple documents: StuffDocumentsChain combines all documents into a single object that can be processed by the LLM
  - Multiple Large Documents: MapReduceDocumentsChain to iteratively summarize each document (Map) and then combine all summaries into a single result (Reduce)

MapReduceDocumentsChain

https://cloud.google.com/blog/products/ai-machine-learning/long-document-summarization-with-workflows-and-gemini-models

- MapReduceDocumentsChain - uses multiple chains and prompts to obtain the final result
- Different prompting techniques can be used to improve the results



Information Extraction

- Information extraction is the process of automatically extracting structured data from unstructured or semi-structured sources such as webpages, PDFs, etc
- LangChain requires us to define the schema of the information we are trying to extract
- Once defined, the schema is passed to the .with_structured_output() function of the LLM

```
extraction_chain = prompt | llm.with_structured_output(schema=Data)
```

Code - Information Processing

https://github.com/DataForScience/LangChain

4. Chatbots

Retrieval Augmented Generation

- Retrieval Augmented Generation (RAG) is a technique to allow LLMs to retrieve and use relevant information from external sources

- Advantages of RAG

    - Improved Accuracy: LLMs generate more accurate and relevant responses with up-to-date information not included in their pre-training data.

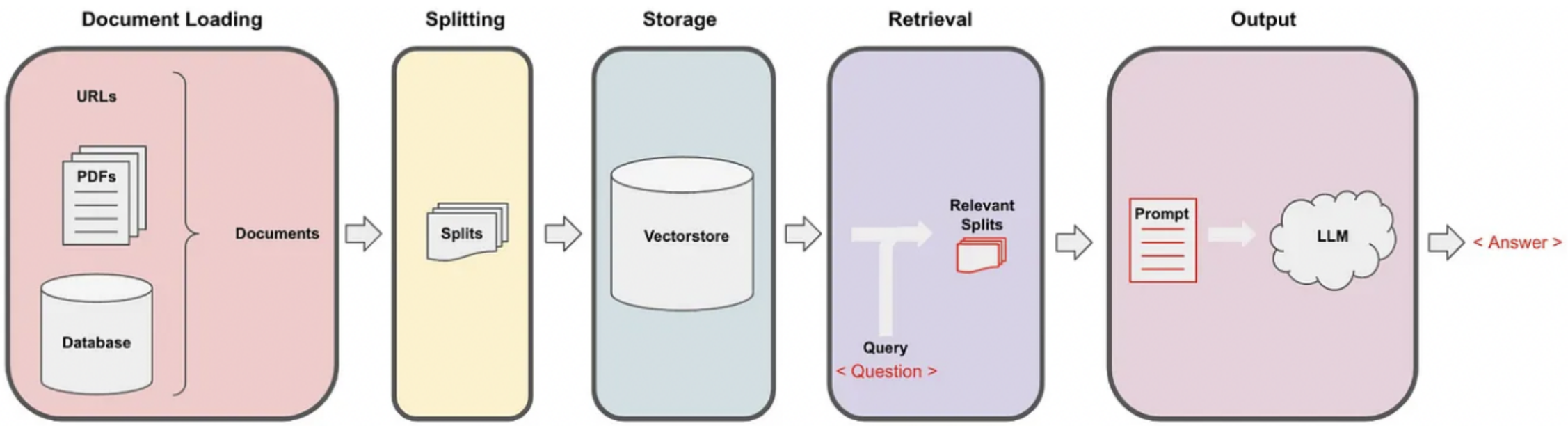
    - Verifiable Outputs: LLMs can report the sources they used to generate their output.

    - Reduced Hallucinations: Grounds LLMs generation on factual information

    - Adaptability: LLMs can be adapted to new domains

Retrieval Augmented Generation

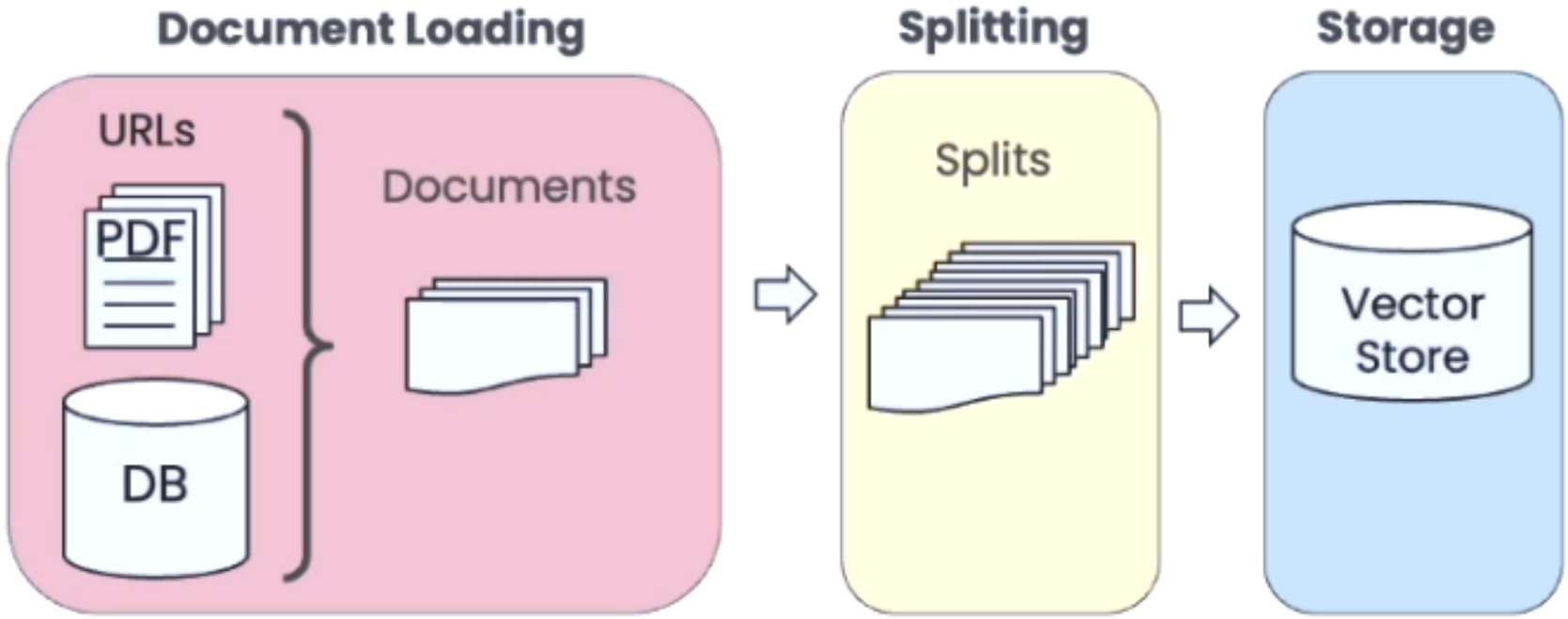
- RAG is a multistep process



Vector Storage and Retrieval

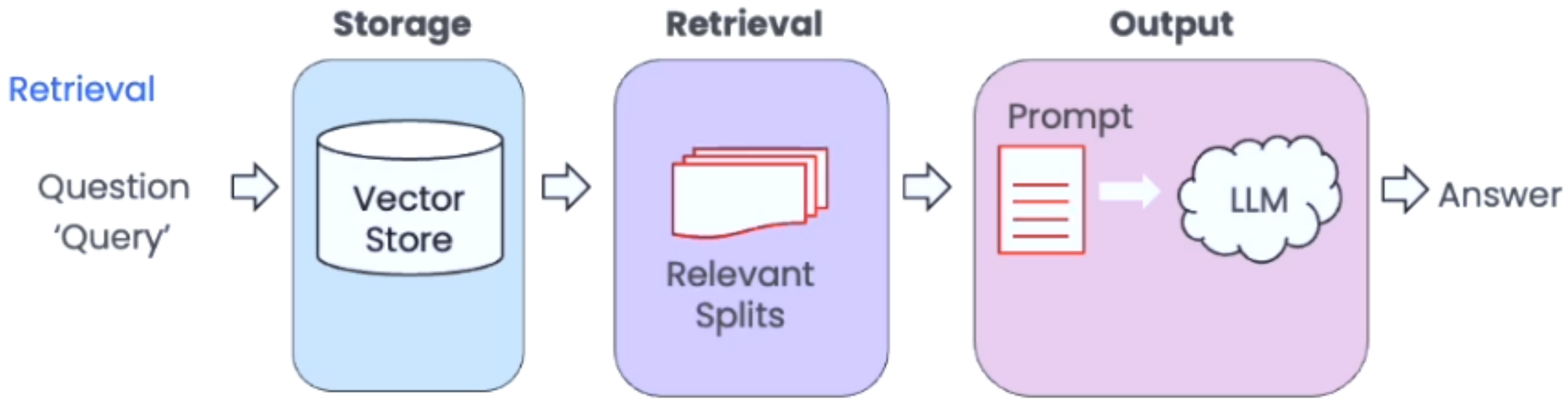
- Vector embeddings are used to represent the semantic meaning of each document, chunk or query

- A vector database allows us to quickly find related documents to provide to the LLM



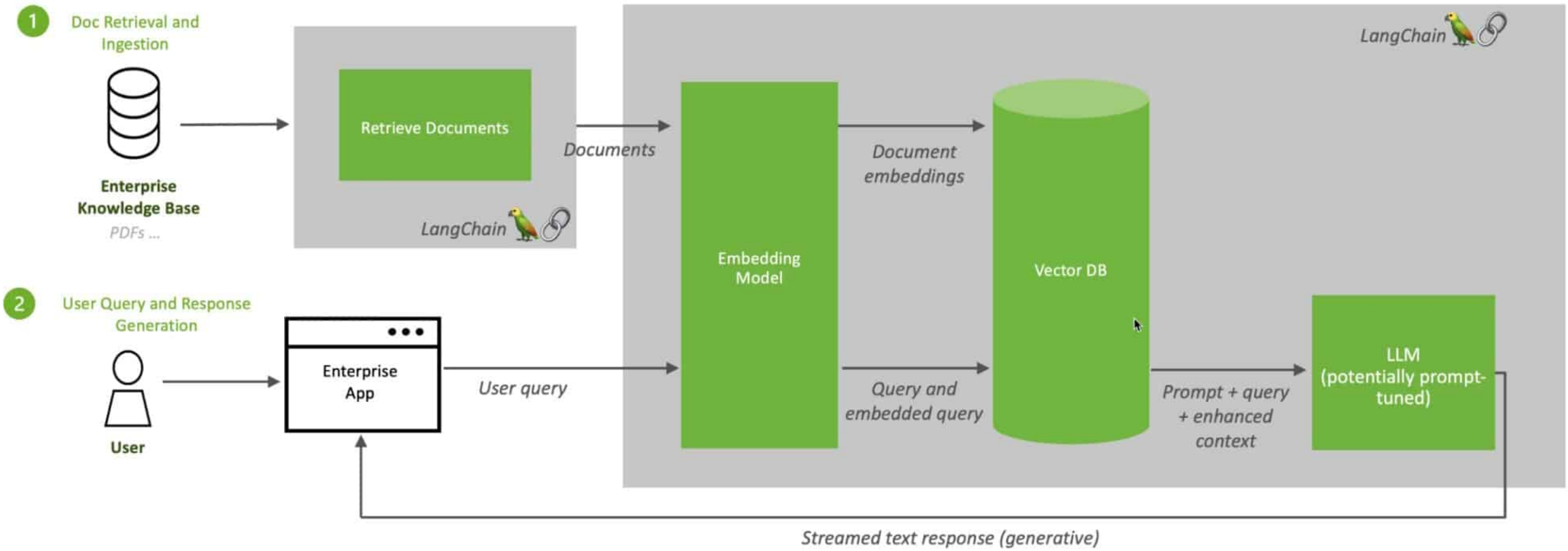
Vector Storage and Retrieval

- Vector embeddings are used to represent the semantic meaning of each document, chunk or query
- A vector database allows us to quickly find related documents
- Retrieved documents can be added to the prompt context window to help the LLM provide better answers



Simple RAG ChatBot

https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/



Code - Chatbots

https://github.com/DataForScience/LangChain

5. Prompt Engineering

Prompts

https://www.pinecone.io/learn/series/langchain/langchain-prompt-templates/

- Prompts can have multiple parts, but now all prompts use all pieces

INSTRUCTIONS Tell the model what to do

```
"""Answer the question based on the context below. If the
question cannot be answered using the information provided answer
with "I don't know".

Context: Large Language Models (LLMs) are the latest models used in NLP.
Their superior performance over smaller models has made them incredibly
useful for developers building NLP enabled applications. These models
can be accessed via Hugging Face's `transformers` library, via OpenAI
using the `openai` library, and via Cohere using the `cohere` library.

Question: Which libraries and model providers offer LLMs?

Answer: """
```

CONTEXTS (EXTERNAL INFO)

Additional source of knowledge

PROMPTER INPUT

OUTPUT INDICATOR

- PromptTemplates allow us to quickly generate customized prompts for each use

PromptTemplate

https://www.pinecone.io/learn/series/langchain/langchain-prompt-templates/

- Template parameters are identified by curly braces {query}

```python
from langchain import PromptTemplate

template = """Answer the question based on the context below. If the
question cannot be answered using the information provided answer
with "I don't know".

Context: Large Language Models (LLMs) are the latest models used in NLP.
Their superior performance over smaller models has made them incredibly
useful for developers building NLP enabled applications. These models
can be accessed via Hugging Face's `transformers` library, via OpenAI
using the `openai` library, and via Cohere using the `cohere` library.

Question: {query}

Answer: """

prompt_template = PromptTemplate(
    input_variables=["query"],
    template=template
)
```

PromptTemplate

https://www.pinecone.io/learn/series/langchain/langchain-prompt-templates/

- Template parameters are identified by curly braces {query}
- To generate a new prompt, we must simply provide the value of all the parameters to the format() method

```python
print(
    prompt_template.format(
        query="Which libraries and model providers offer LLMs?"
    )
)
```

```python
from langchain import PromptTemplate

template = """Answer the question based on the context below. If the
question cannot be answered using the information provided answer
with "I don't know".

Context: Large Language Models (LLMs) are the latest models used in NLP.
Their superior performance over smaller models has made them incredibly
useful for developers building NLP enabled applications. These models
can be accessed via Hugging Face's `transformers` library, via OpenAI
using the `openai` library, and via Cohere using the `cohere` library.

Question: {query}

Answer: """

prompt_template = PromptTemplate(
    input_variables=["query"],
    template=template
)
```

Simple Prompting strategies

https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/

- Prompting is how we interact with LLMs
- Choosing the correct type of prompt can have a major effect on the results we obtain
- Common prompting approaches:
    - Zero-Shot - Explain the task and ask for the result (most examples we've seen so far)
    - Few-Shot - Provide a few examples of both inputs and outputs that the model can generalize from

Simple Prompting strategies

https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/

- Prompting is how we interact with LLMs
- Choosing the correct type of prompt can have a major effect on the results we obtain
- Common prompting approaches:
    - Zero-Shot - Explain the task and ask for the result (most examples we've seen so far)
    - Few-Shot - Provide a few examples of both inputs and outputs that the model can generalize from
- The FewShotPromptTemplate makes it easy to generate custom prompts with a few examples, to test the effect of different number of examples, etc

```
few_shot_prompt_template = FewShotPromptTemplate(
    examples=examples,
    example_prompt=example_prompt,
    prefix=prefix,
    suffix=suffix,
    input_variables=["query"],
    example_separator="\n\n"
)
```

A list of examples to be used
The template of each example
Instructions proceeding the examples
Instructions following the examples
variables for the few-shot prompt template
Characters to use to separate examples

Prompting Tips

https://www.promptingguide.ai/introduction/tips

- Start simple and integrate by gradually adding the details necessary to improve results
- Break down complex tasks into smaller, more manageable, subtasks
- Use action commands ("Write", "Classify", "Summarize", "Translate", "Order", etc.) to indicate what the goal is, such as
- More descriptive and detailed the prompts produce better results.

Chain-of-Thought prompts

arXiv: 2201.11903 (2022)

- Chain of Thought (CoT) prompts encourages LLMs to generate intermediate steps before providing the final solution to a problem.

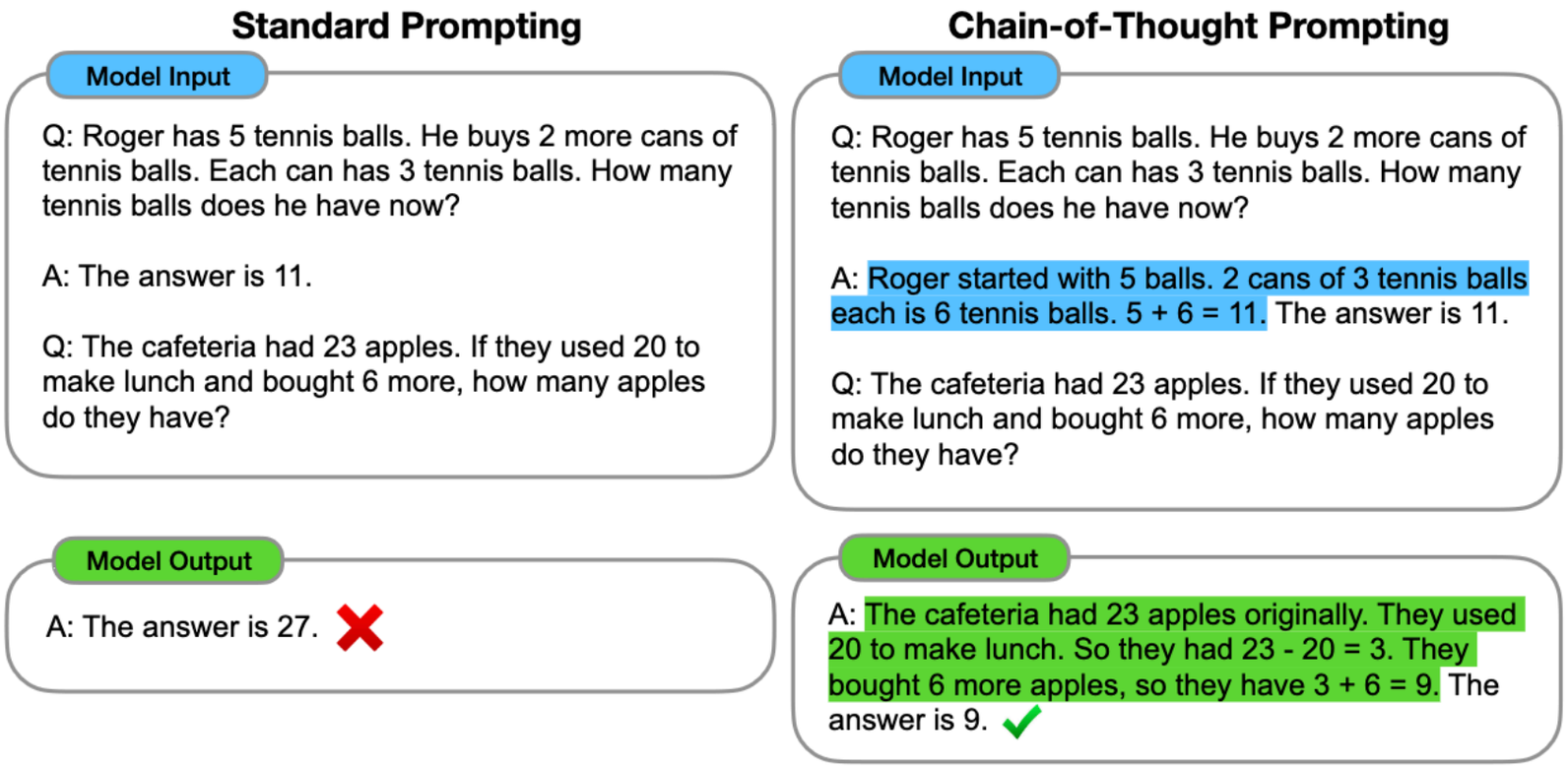
- CoT breaks down complex problems into manageable, intermediate steps and mimic an human thought process when working through multi-step problems.

- Relatively small change to the prompt can have a profound effect

Chain-of-Thought prompts

arXiv: 2201.11903 (2022)

- Chain of Thought (CoT) prompts encourages LLMs to generate intermediate steps before providing the final solution to a problem.
- CoT breaks down complex problems into manageable, intermediate steps and mimic an human thought process when working through multi-step problems.
- Relatively small change to the prompt can have a profound effect



Chain-of-Thought prompts

arXiv: 2201.11903 (2022)

- Chain of Thought (CoT) prompts encourages LLMs to generate intermediate steps before providing the final solution to a problem.
- CoT breaks down complex problems into manageable, intermediate steps and mimic an human thought process when working through multi-step problems.
- Relatively small change to the prompt can have a profound effect

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

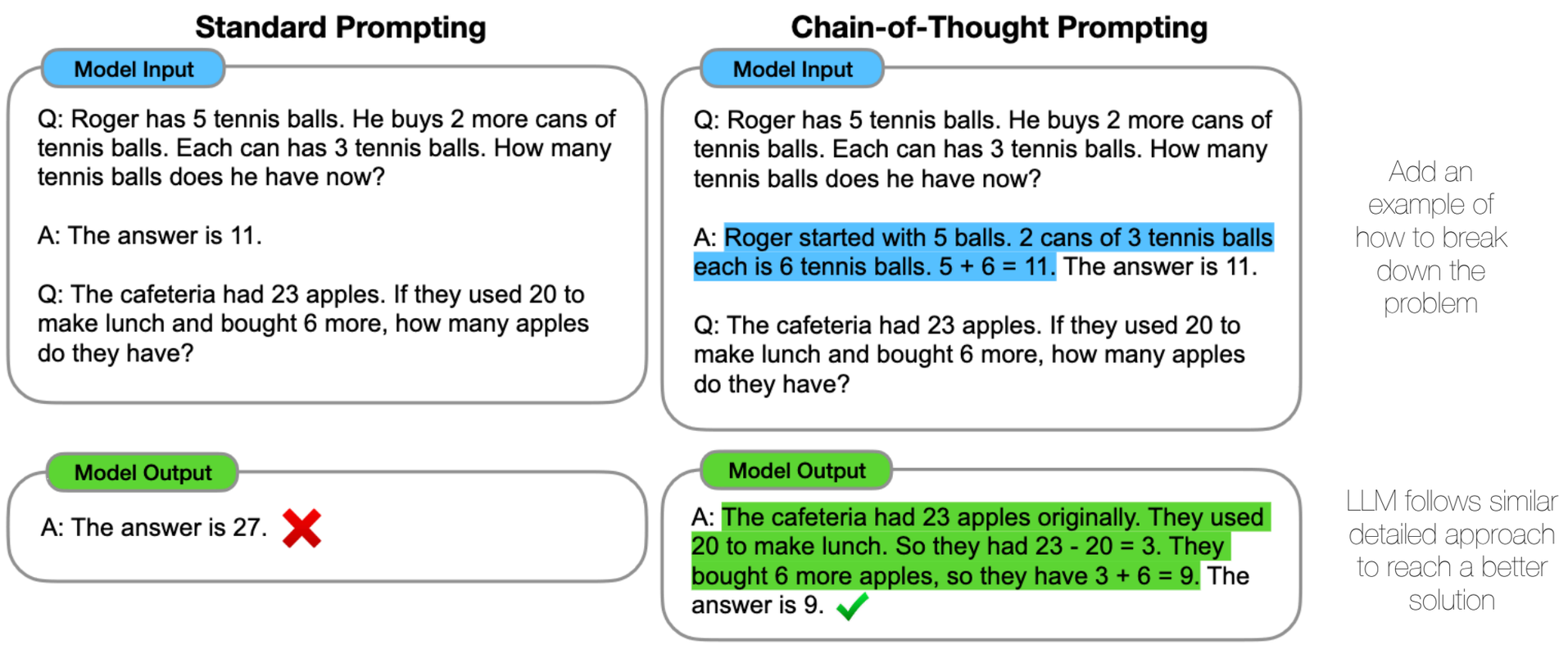
A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Add an example of how to break down the problem

Chain-of-Thought prompts

arXiv: 2201.11903 (2022)

- Chain of Thought (CoT) prompts encourages LLMs to generate intermediate steps before providing the final solution to a problem.
- CoT breaks down complex problems into manageable, intermediate steps and mimic an human thought process when working through multi-step problems.
- Relatively small change to the prompt can have a profound effect



Chain-of-Thought prompts

arXiv: 2201.11903 (2022)

Math Word Problems (free response)

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.

Math Word Problems (multiple choice)

Q: How many keystrokes are needed to type the numbers from 1 to 500?
Answer Choices: (a) 1156 (b) 1392 (c) 1480 (d) 1562 (e) 1788

A: There are 9 one-digit numbers from 1 to 9. There are 90 two-digit numbers from 10 to 99. There are 401 three-digit numbers from 100 to 500. 9 + 90(2) + 401(3) = 1392. The answer is (b).

CSQA (commonsense)

Q: Sammy wanted to go to where the people were. Where might he go?
Options: (a) race track (b) populated areas (c) desert (d) apartment (e) roadblock

A: The answer must be a place with a lot of people. Race tracks, desert, apartments, and roadblocks don't have a lot of people, but populated areas do. So the answer is (b).

StrategyQA

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

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

Date Understanding

Q: The concert was scheduled to be on 06/01/1943, but was delayed by one day to today. What is the date 10 days ago in MM/DD/YYYY?

A: One day after 06/01/1943 is 06/02/1943, so today is 06/02/1943. 10 days before today is 05/23/1943. So the answer is 05/23/1943.

Sports Understanding

Q: Is the following sentence plausible? "Joao Moutinho caught the screen pass in the NFC championship."

A: Joao Moutinho is a soccer player. The NFC championship is part of American football, not soccer. So the answer is no.

SayCan (Instructing a robot)

Human: How would you bring me something that isn't a fruit?

Explanation: the user wants something to eat that isn't a fruit. An energy bar is not a fruit, so I will bring the user an energy bar.
Plan: 1. find(energy bar) 2. pick(energy bar) 3. find(user) 4. put(energy bar) 5. done().

Last Letter Concatenation

Q: Take the last letters of the words in "Lady Gaga" and concatenate them.

A: The last letter of "Lady" is "y". The last letter of "Gaga" is "a". Concatenating them is "ya". So the answer is ya.

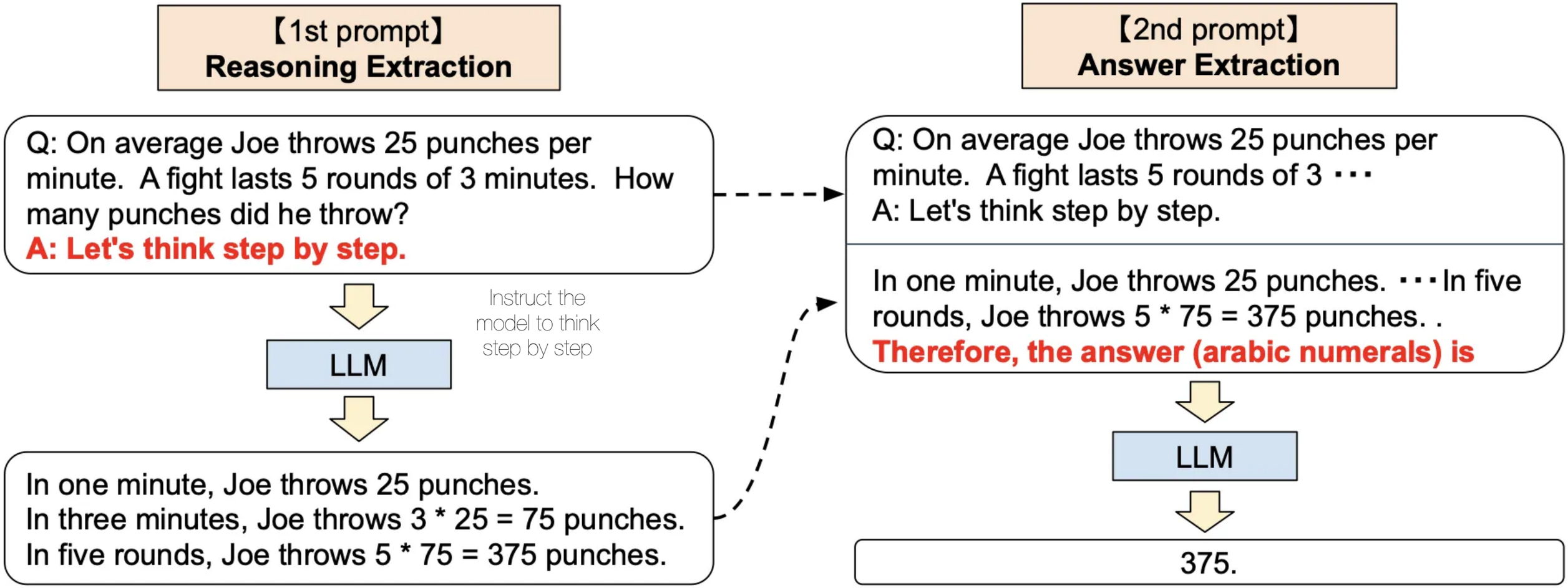
Coin Flip (state tracking)

Q: A coin is heads up. Maybelle flips the coin. Shalonda does not flip the coin. Is the coin still heads up?

A: The coin was flipped by Maybelle. So the coin was flipped 1 time, which is an odd number. The coin started heads up, so after an odd number of flips, it will be tails up. So the answer is no.

Chain-of-Thought - Zero Shot

arXiv: 2201.11903 (2022)



Code - Prompt Engineering

https://github.com/DataForScience/LangChain

Events

data4sci.substack.com

Generative AI with the OpenAI API for Developers
Oct 9, 2024 - 10am-2pm (PST)

ChatGPT and Competing LLMs
Nov 13, 2024 - 10am-2pm (PST)

Bruno Gonçalves

https://data4sci.com

info@data4sci.com

https://data4sci.com/call

VIDEO

Times Series Analysis for Everyone

★★★★★ 1 review

By Bruno Goncalves

https://bit.ly/Timeseries_LL

TIME TO COMPLETE:
6h

TOPICS:
Time Series

PUBLISHED BY:
Pearson

PUBLICATION DATE:
November 2021

Start

6 Hours of Video Instruction

The perfect introduction to time-based analytics

Overview

Times Series Analysis for Everyone LiveLessons covers the fundamental tools and techniques for the analysis of time series data. These lessons introduce you to the basic concepts, ideas, and algorithms necessary to develop your own time series applications in a step-by-step, intuitive fashion. The lessons follow a gradual progression, from the more specific to the more abstract, taking you from the very basics to some of the most recent and sophisticated algorithms by leveraging the statsmodels, arch, and Keras state-of-the-art models.

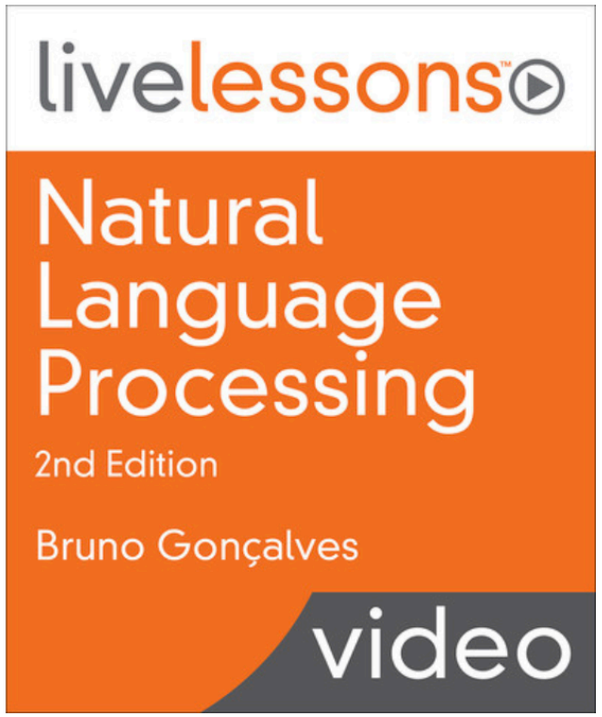
VIDEO

Natural Language Processing, 2nd Edition

Write the first review

By Bruno Goncalves

https://bit.ly/NLP_LL



TIME TO COMPLETE:
5h 23m

TOPICS:
Natural Language Processing

PUBLISHED BY:
Addison-Wesley Professional

PUBLICATION DATE:
October 2021

Start

5 Hours of Video Instruction

Overview

Natural Language Processing LiveLessons covers thefundamentals of Natural Language Processing in a simple and intuitive way,empowering you to add NLP to your toolkit. Using the powerful NLTK package, itgradually moves from the basics of text representation, cleaning, topicdetection, regular expressions, and sentiment analysis before moving on to theKeras deep learning framework to explore more advanced topics such as textclassification and sequence-to-sequence models. After successfully completingthese lessons you'll be equipped with a fundamental and practical understandingof state-of-the-art Natural Language Processing tools and algorithms.

END