Slot 1: Introduction to RAG systems with LLMs and KGs

ISWC 2024

ISWC Baltimore, USA

Fraunhofer FIT

RWTH AACHEN UNIVERSITY

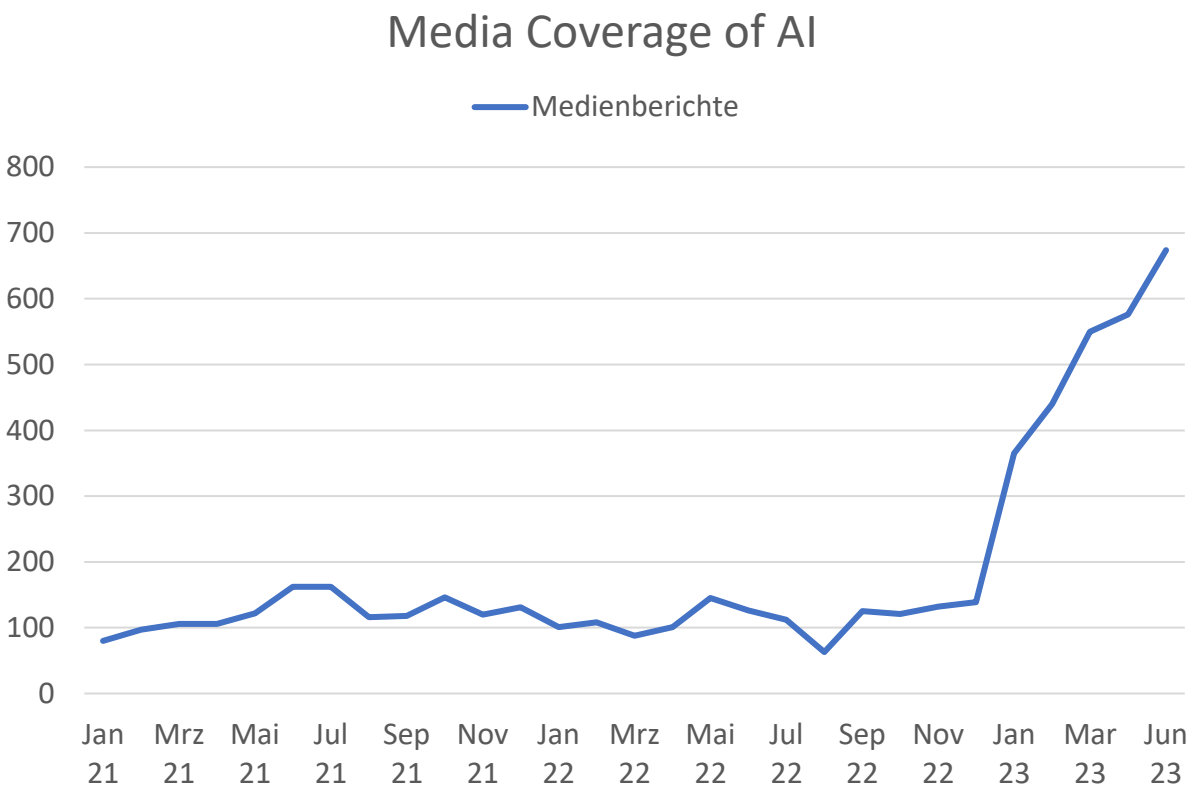
TIB LEIBNIZ-INFORMATIONSZENTRUM TECHNIK UND NATURWISSENSCHAFTEN UNIVERSITÄTSBIBLIOTHEK

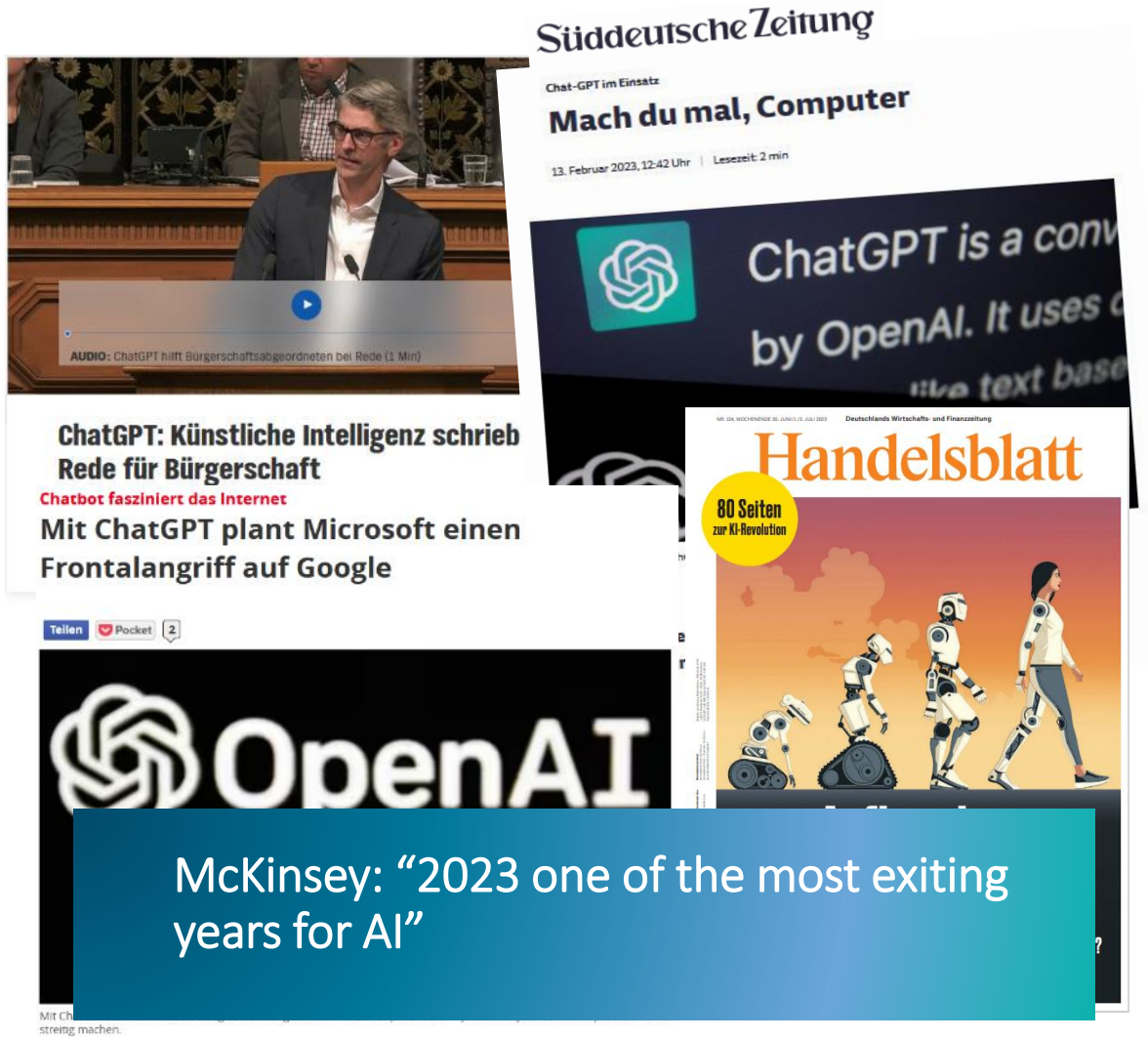
UPB

Part 1: Introduction

Large Language Models, Retrieval Augmented Generation (RAG), and Graph RAG

Artificial intelligence is the number one topic of conversation





McKinsey: “2023 one of the most exiting years for AI”

Large language models (LLMs) are taking the world by storm

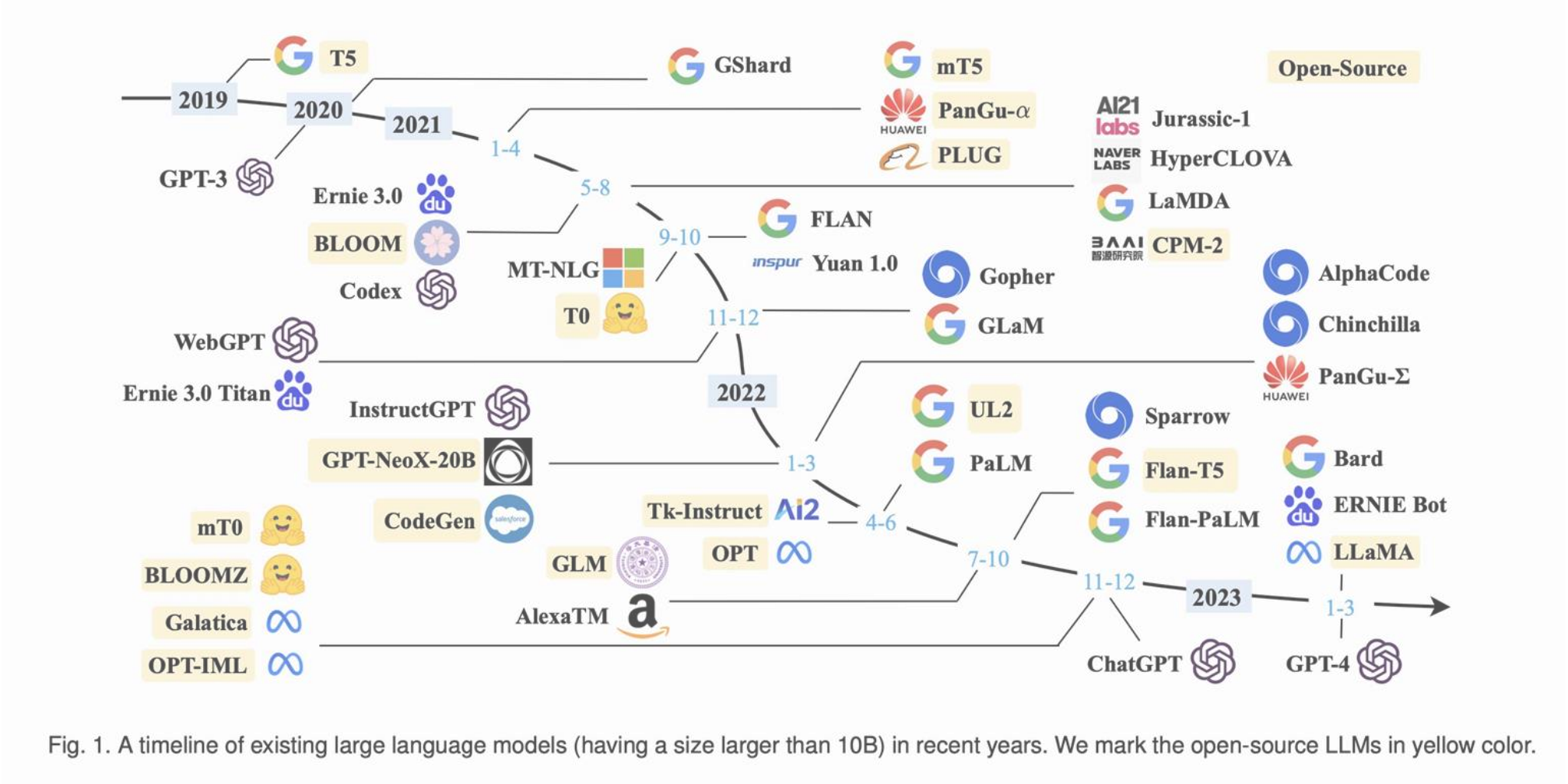


Fig. 1. A timeline of existing large language models (having a size larger than 10B) in recent years. We mark the open-source LLMs in yellow color.

Image source: A Survey of Large Language Models https://arxiv.org/abs/2303.18223

The bigger the LLM, the more capabilities arise

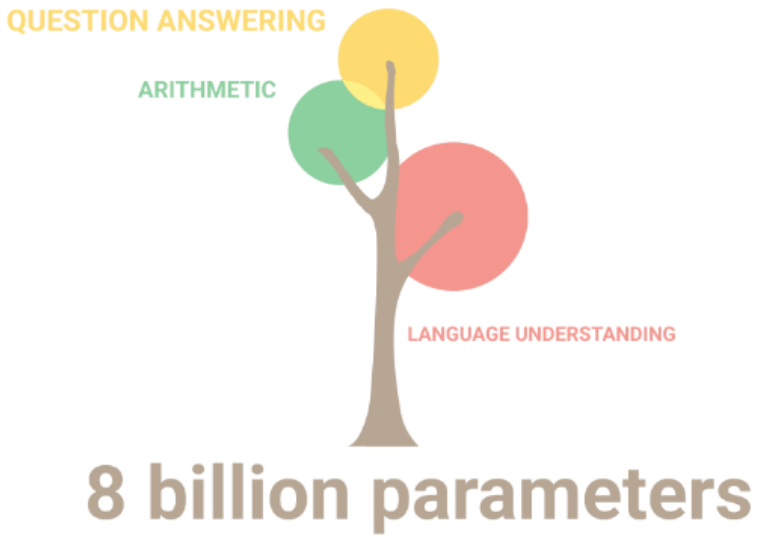


Image source: https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

The bigger the LLM, the more capabilities arise

Question answering

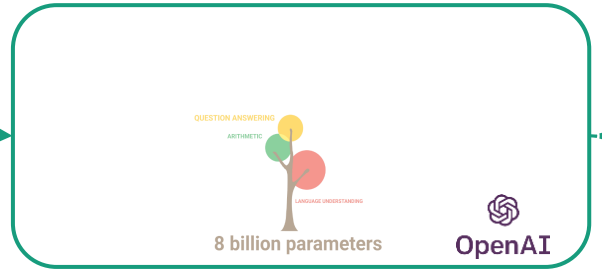
What is the capital of Germany?

Translation

Translate “Welcome to the workshop” into Spanish

Code generation

Write a Python function to calculate the Pythagoras theorem



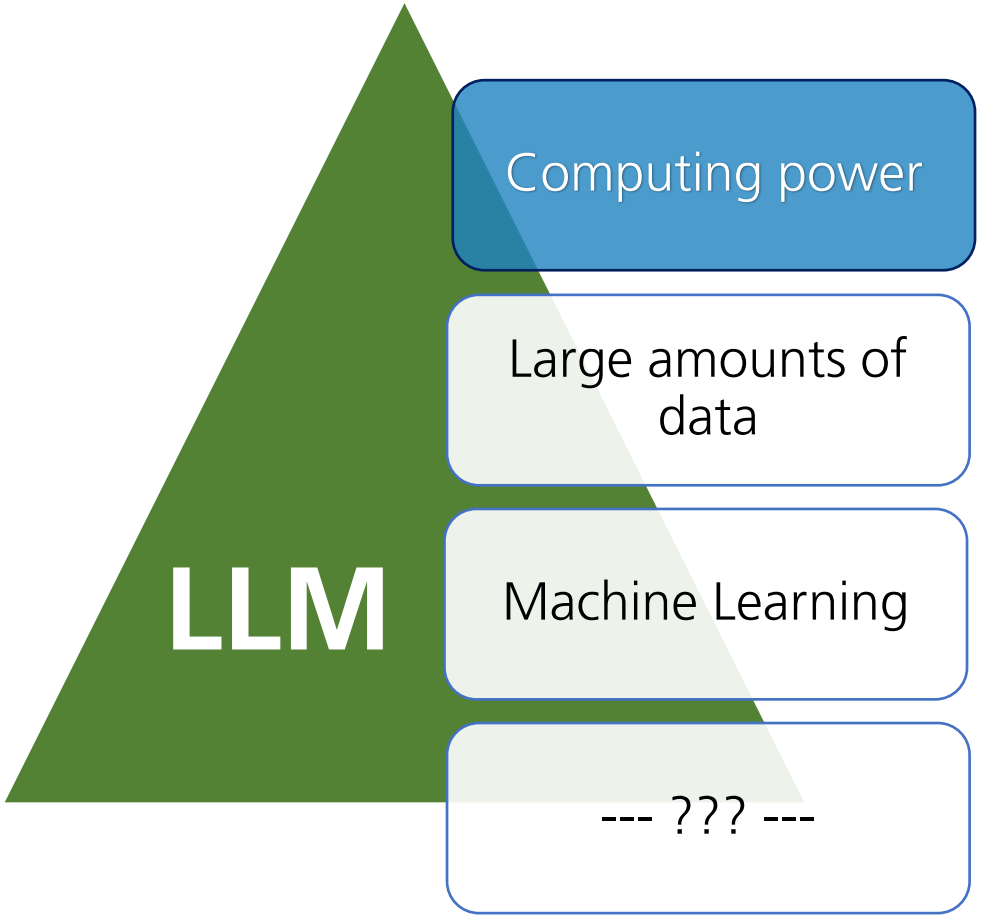
The capital of Germany is Berlin.

"Welcome to the workshop" translates to "Bienvenidos al taller" in Spanish.

```
def pythagoras_theorem( a, b):
    c_squared = a**2 + b**2
    c = math.sqrt(c_squared)
    return c
```

Answer generated by GPT-3.5 Turbo 16K

The core elements of an LLM



GPT-3 175B model **requires O(1,000–10,000) V100 GPUs**

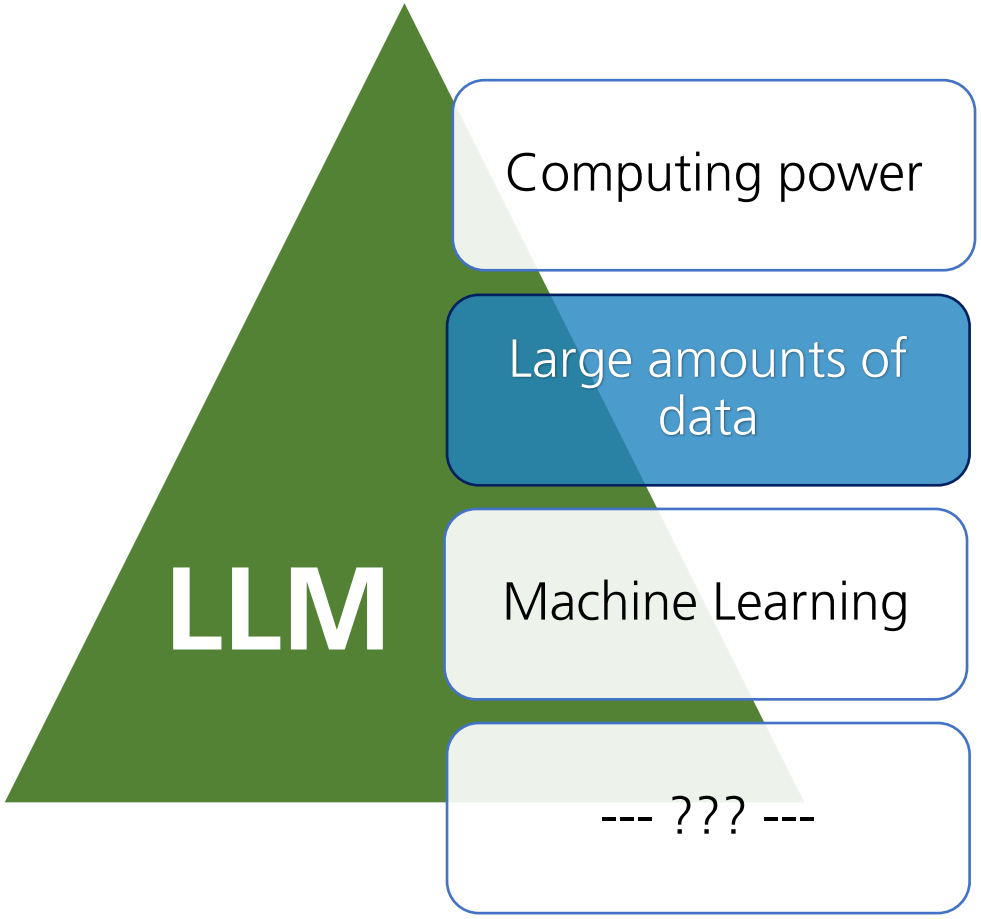
Costs O(1–10) $M for a single training run

O(1) month of training

"'The supercomputer built for OpenAI is a single system with more than 285,000 CPU cores, 10,000 GPUs, and 400 Gigabits per second of network connectivity for each GPU server,' the companies explained in a blog post."[2]

Sources:
1. https://news.developer.nvidia.com/openai-presents-gpt-3-a-175-billion-parameters-language-model/
2. https://news.microsoft.com/source/features/innovation/openai-azure-supercomputer/

The core elements of an LLM



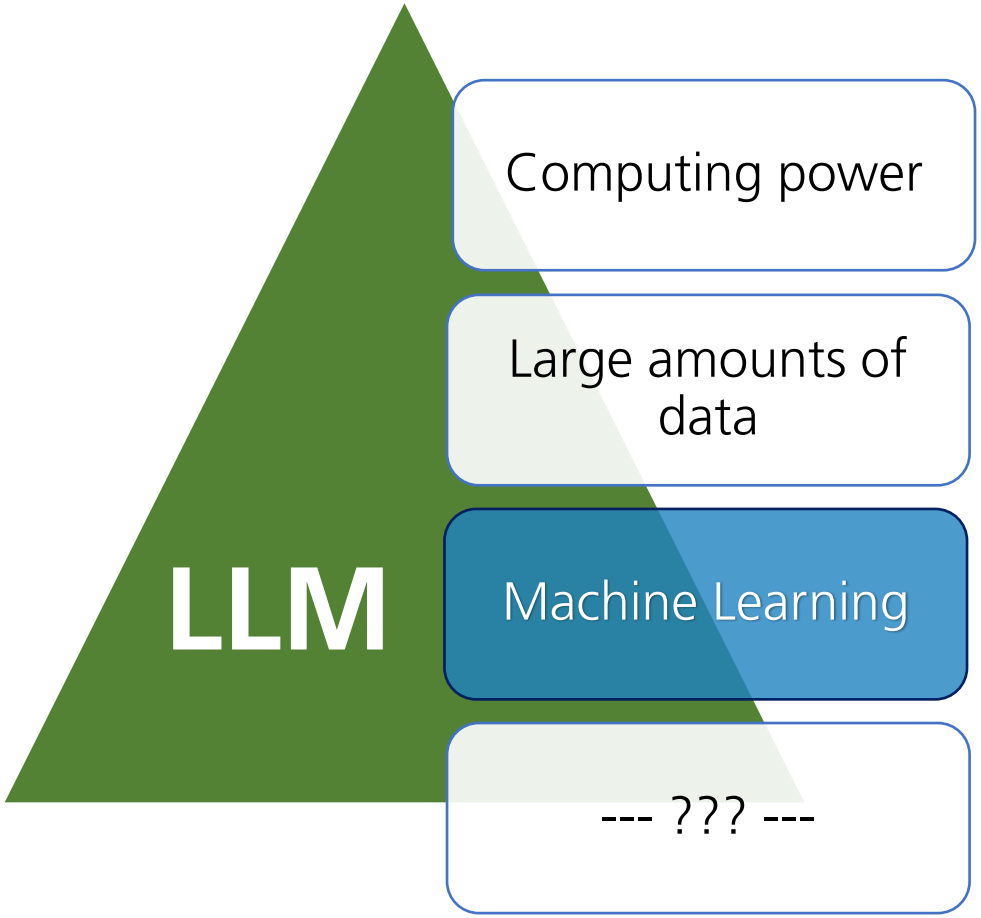
GPT-3 was trained on **570 GB of text**

- Number of documents per language:

English	235.987.420
German	3.014.597
French	2.568.341

- Paper shows that the quality of the model depends on the size of the model, the amount of data and the number of training runs
- The texts contain a total of approx. **500 billion tokens** (note: 1 token != 1 word)

The core elements of an LLM



BERT models: Prediction of masked tokens

- Powerful Embeddings

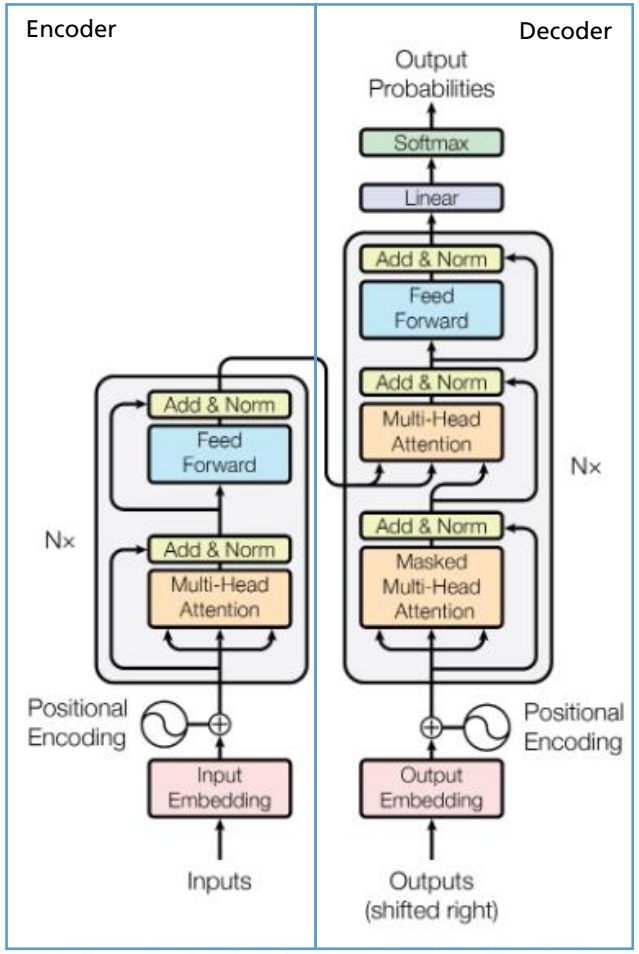


Figure 1: The Transformer - model architecture.

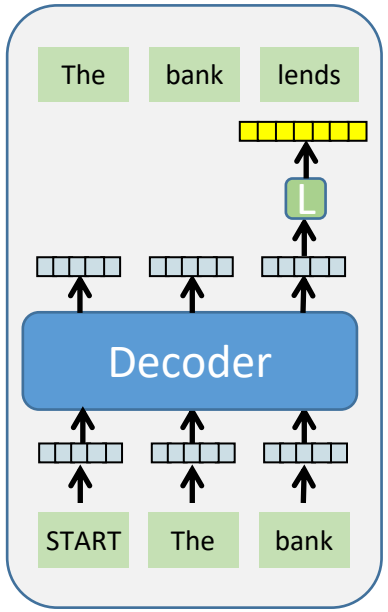
Quelle: Vaswani et al. – Attention is all you need (2017)

GPT models: Prediction of the next token

- In-Context Learning via Prompting

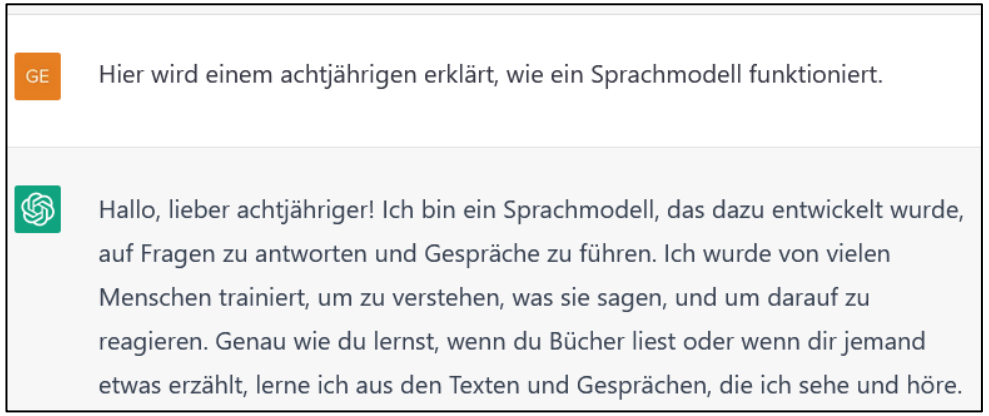
Adaption via Prompting

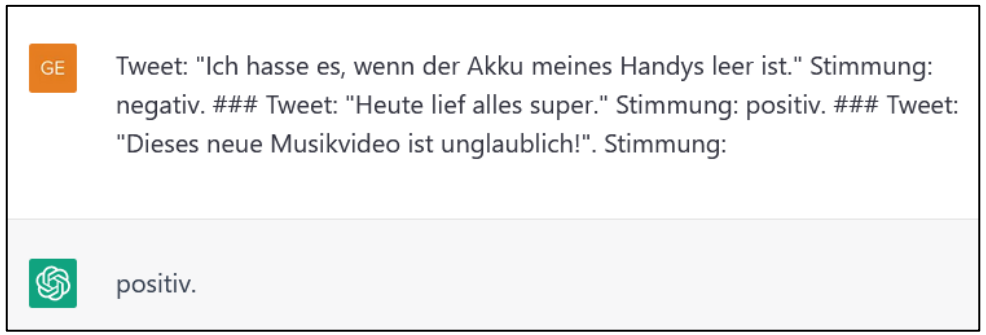
GPT language models respond to instructions



- A GPT language model is given a **starting text**
  - It generates a continuation that is syntactically correct and plausible in terms of content
  - If the starting text contains a **direct instruction**, the corresponding response is generated
  - Self-attention only on **previous** tokens

- The GPT language model can be taught how to solve a task using one or more **examples**.





- Through **fine-tuning** (e.g., RLHF) the LLM can be trained to
  - avoid **abusive** statements
  - fulfil any request **correctly** and accurately
  - respond to the dialogue partner and make the dialogue **interesting**

Limitations of Large Language Models

Dresden is a city in

The president of the USA is

Is the BMW E4 car fully electric?

Knowledge is **somehow stored** in the billions of parameters of the LLM

Hallucination

Dresden is a city in the **state of Bavaria**

Updateability problem

The president of the USA is **Donald Trump**

Provenance problem

…As of now, there is no official BMW E4 released. However, **there are reports and speculations about** an upcoming E4 SUV, which is expected to be a fully electric…

Answer generated by GPT-3.5 Turbo 16K

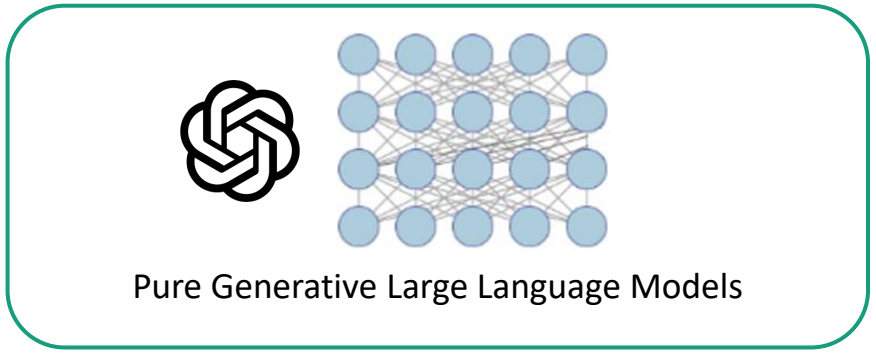
Retrieval Augmented Generation

To the rescue

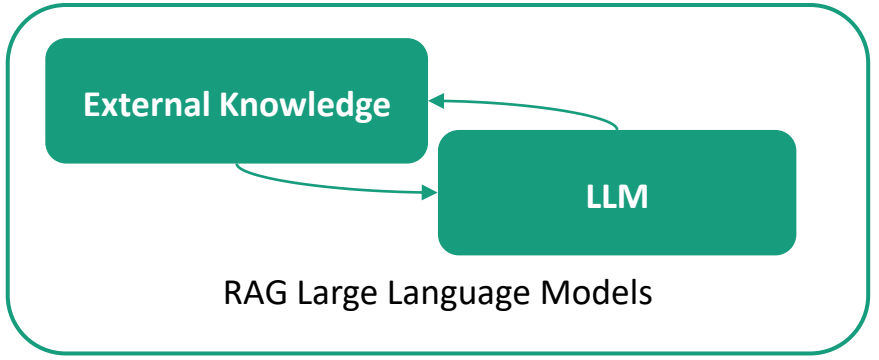
Retrieval Augmented Generation

Separate Generative AI into two components, i.e., Knowledge Store and Linguistic Capabilities

From



To

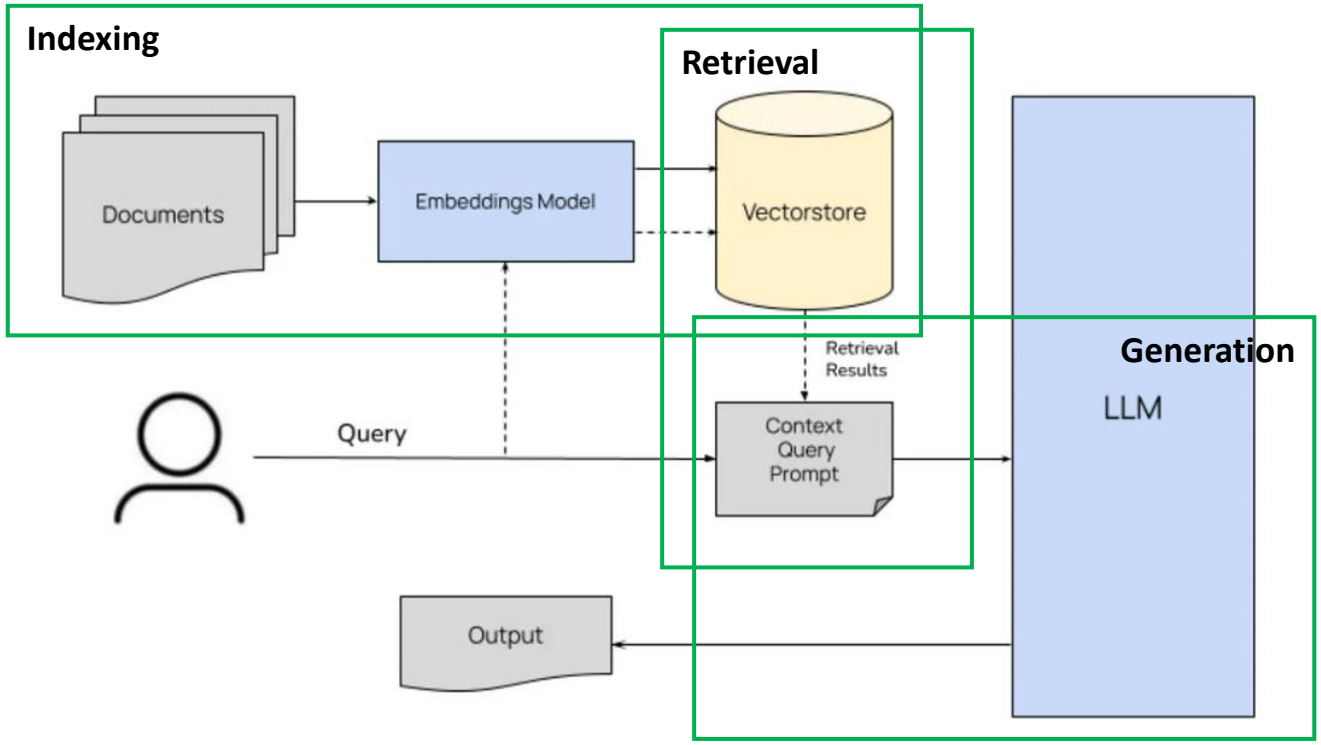


- Efficiency **Yellow**
- Factual correctness **Red**
- Updatability **Red**
- Provenance **Red**
- Effectiveness **Green**
- Synthesis **Green**

- Efficiency **Green**
- Factual correctness **Green**
- Updatability **Green**
- Provenance **Green**
- Effectiveness **Green**
- Synthesis **Green**

Retrieval Augmented Generation (Vector RAG)

- Factual grounding
- Mitigating hallucinations
- Dependent on quality and accuracy of dataset
- Latency concerns with vast datasets



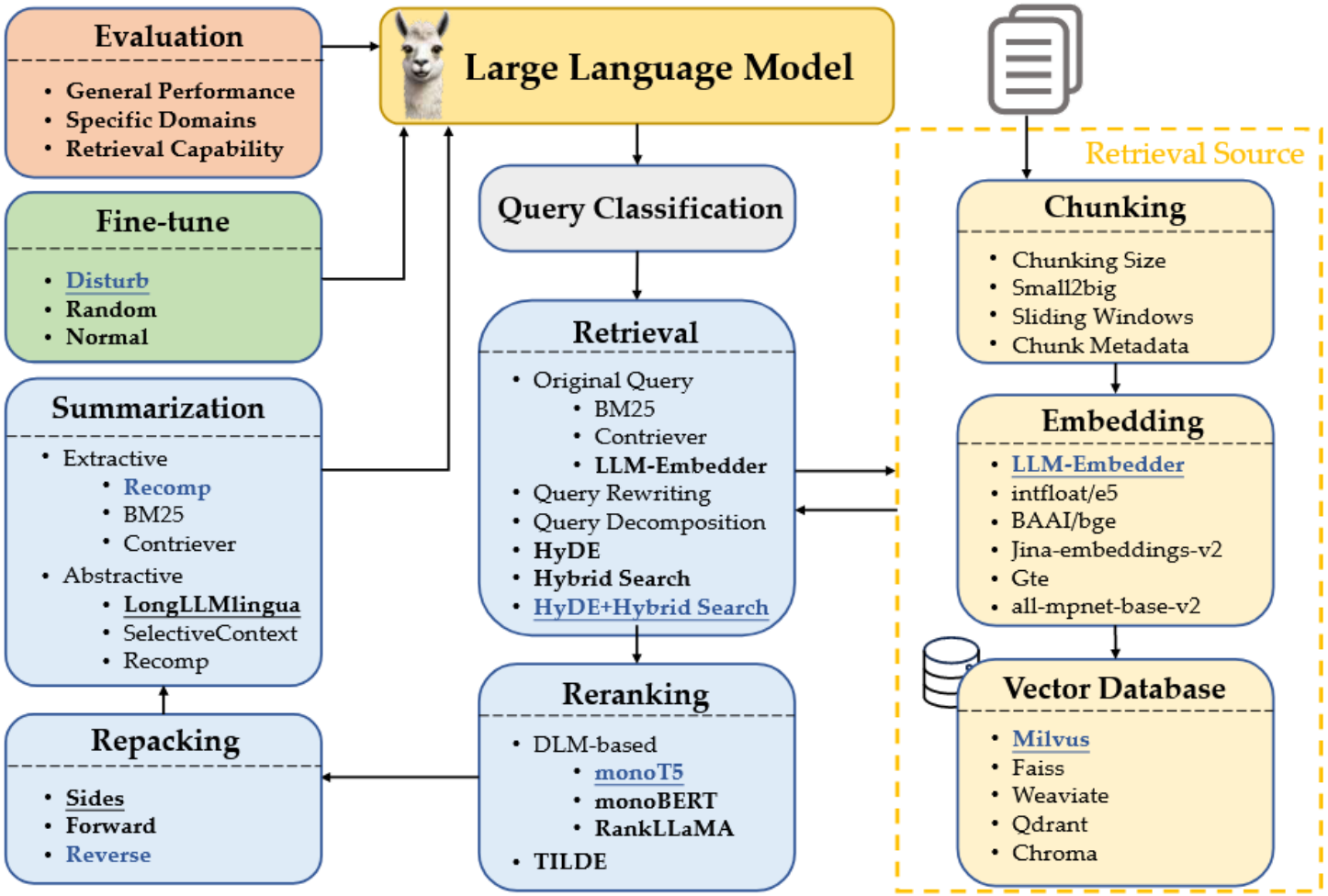
A typical RAG system [6]

Hands-On 1.1

Simple Vector RAG

Retrieval Augmented Generation (Vector RAG)

In practices RAG gets more complicated



Source: https://arxiv.org/abs/2407.01219

Limitations of Vector RAG

- **Limited Contextual Understanding:** Struggling to understand the context of queries and the relationships between different pieces of information
- **Difficulty in Handling Global Queries:** Hard to answer global queries that require understanding information spread across an entire dataset
- **Inefficient for Complex Search and Content Management:** Handling multiple language variants, version control, and fine-grained content filtering can be challenging using traditional vector-based RAG
- **Chunking and Context Loss:** Breaking documents into documents can lead to the loss of important contextual information
- **Limited Reasoning Capabilities**
  - Multi-hop reasoning: Navigating multiple relationships to arrive at an answer
  - Abstractive reasoning: Answering with knowledge not explicitly present in the text requires deeper reasoning and understanding

What are the side effects of medications used to treat diseases like asthma?

Answering this accurately requires identifying medications for asthma, finding similar diseases, and then identifying side effects of treatments for those diseases—a multi-step process difficult for conventional RAG models

Graph RAG

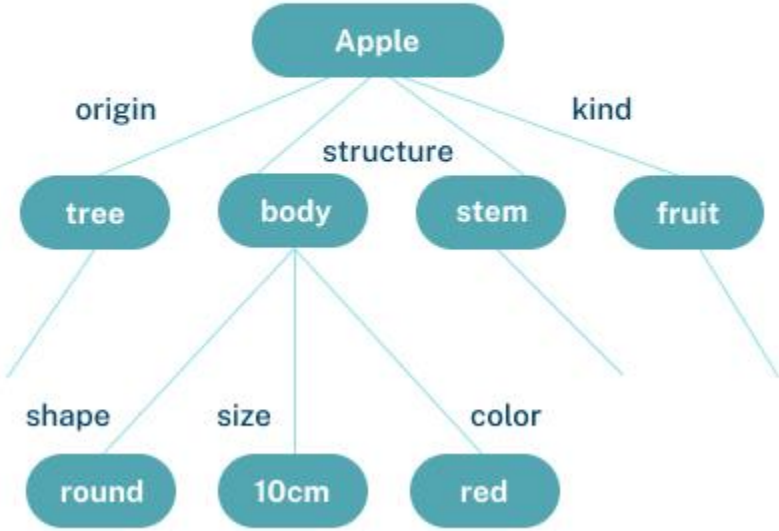
To the rescue

Graph RAG - Knowledge Representation, Vectors & Graphs

Human View of an Apple

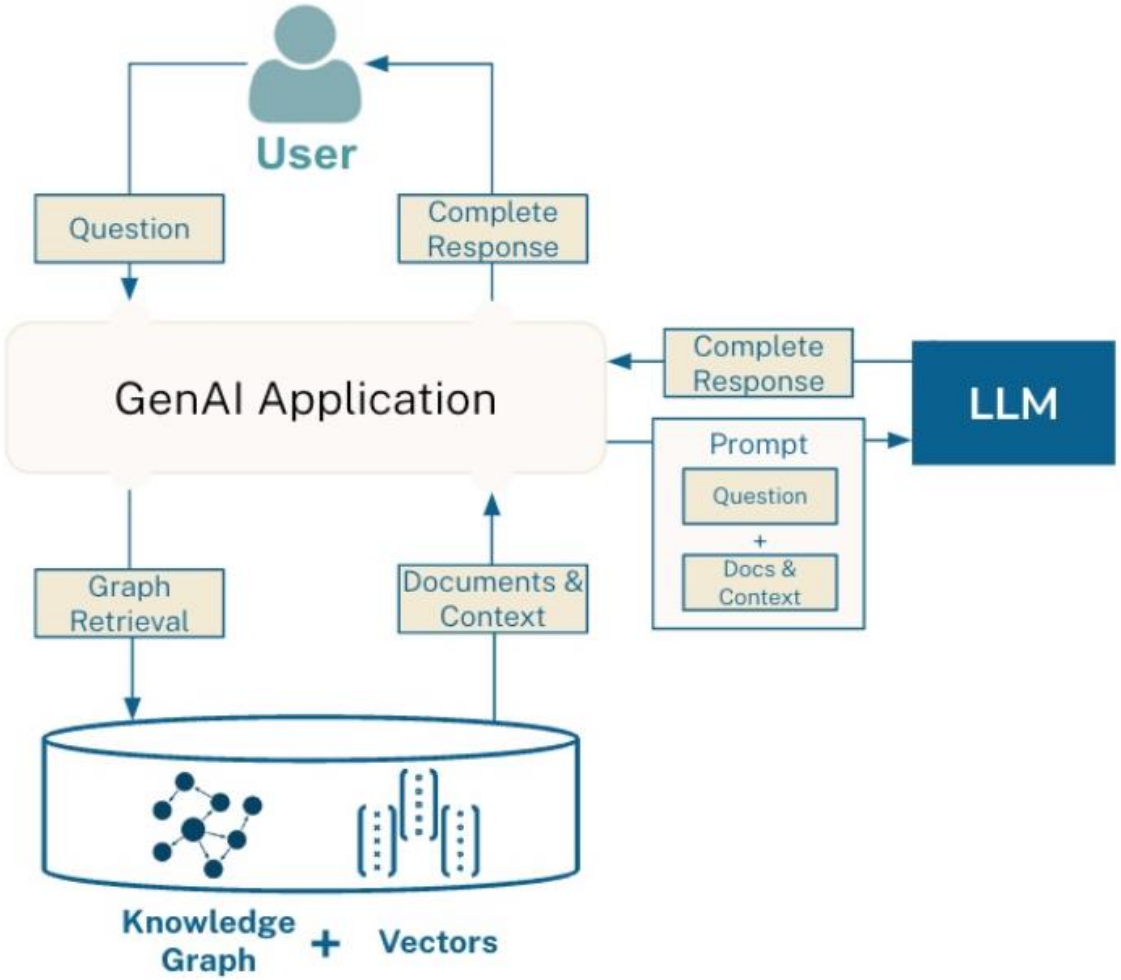
Vector View of an Apple

Knowledge Graph View of an Apple



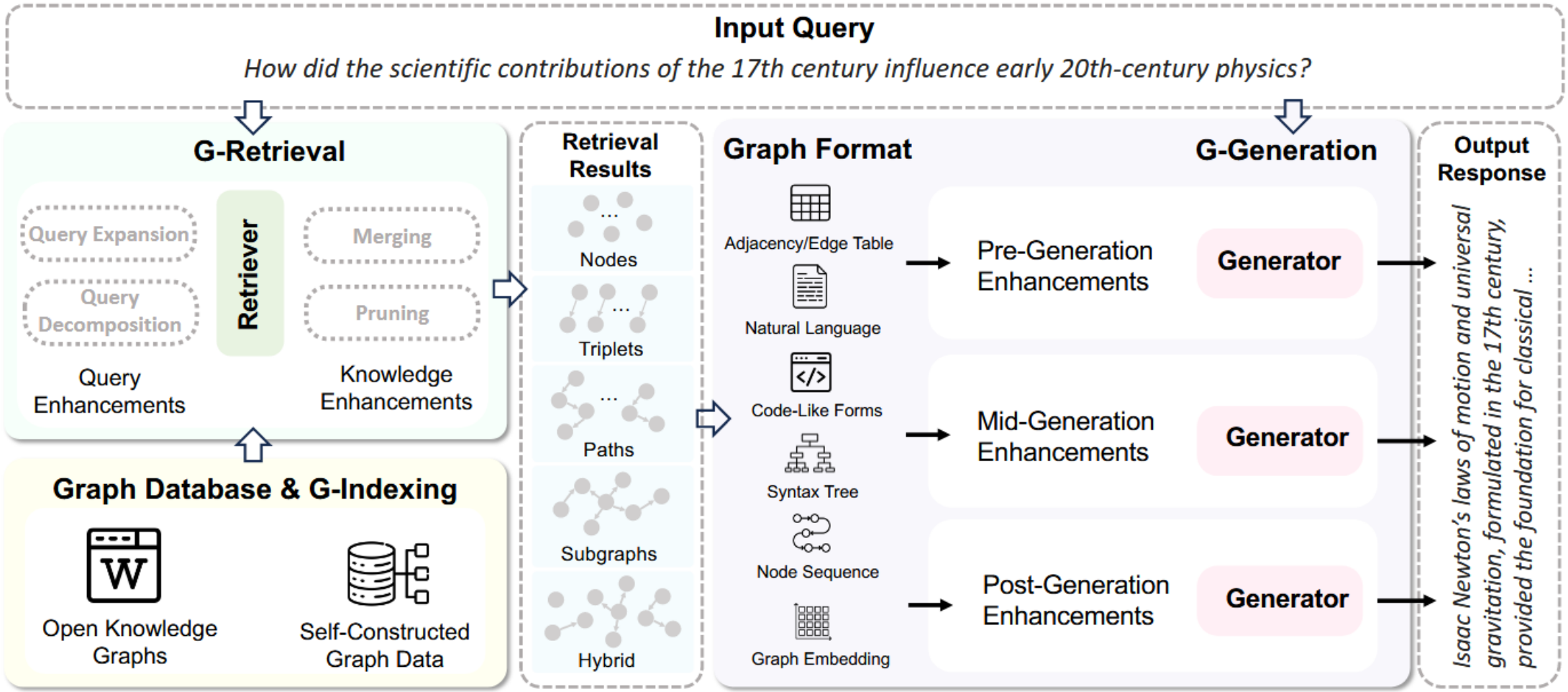
Source of the image -- The GraphRAG Manifesto: Adding Knowledge to GenAI: https://neo4j.com/blog/graphrag-manifesto/

Graph RAG - A typical Graph RAG system



Source of the image -- The GraphRAG Manifesto: Adding Knowledge to GenAI: https://neo4j.com/blog/graphrag-manifesto/

Graph RAG



Source of the image -- Graph Retrieval-Augmented Generation: A Survey https://arxiv.org/abs/2408.08921

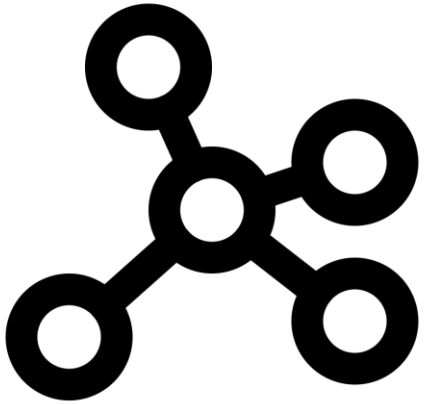
Part 1: G-Indexing

Creating a Knowledge Graph with LLMs for RAG systems

Graph Indexing

- Entity Linking
- Constructing the KG

- Text Chunks
- Documents Metadata, e.g., versions



Typical Pipeline

LlamaIndex approach - Knowledge Graph Index

- A specialized index designed to handle KG construction from unstructured text, enabling entity-based querying.
- **Builds a KG in several steps:**
  - Load Your Data: Using LlamaIndex's data connectors
  - Instantiate KnowledgeGraphIndex object. You can pass your loaded documents, specify the maximum number of triplets per chunk
  - Configure Storage Context: LlamaIndex provides simple in-memory storage or integration with external graph databases
  - Run the Indexer: This process will automatically extract relationships, create nodes, and build the knowledge graph.
- It allows **manual addition of triplets** to enhance or refine the automatically constructed KG

Building the Knowledge Graph

```
from llama_index.core import SimpleDirectoryReader, KnowledgeGraphIndex
from llama_index.core.graph_stores import SimpleGraphStore

from llama_index.llms.openai import OpenAI
from llama_index.core import Settings
from IPython.display import Markdown, display
```

```
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
```

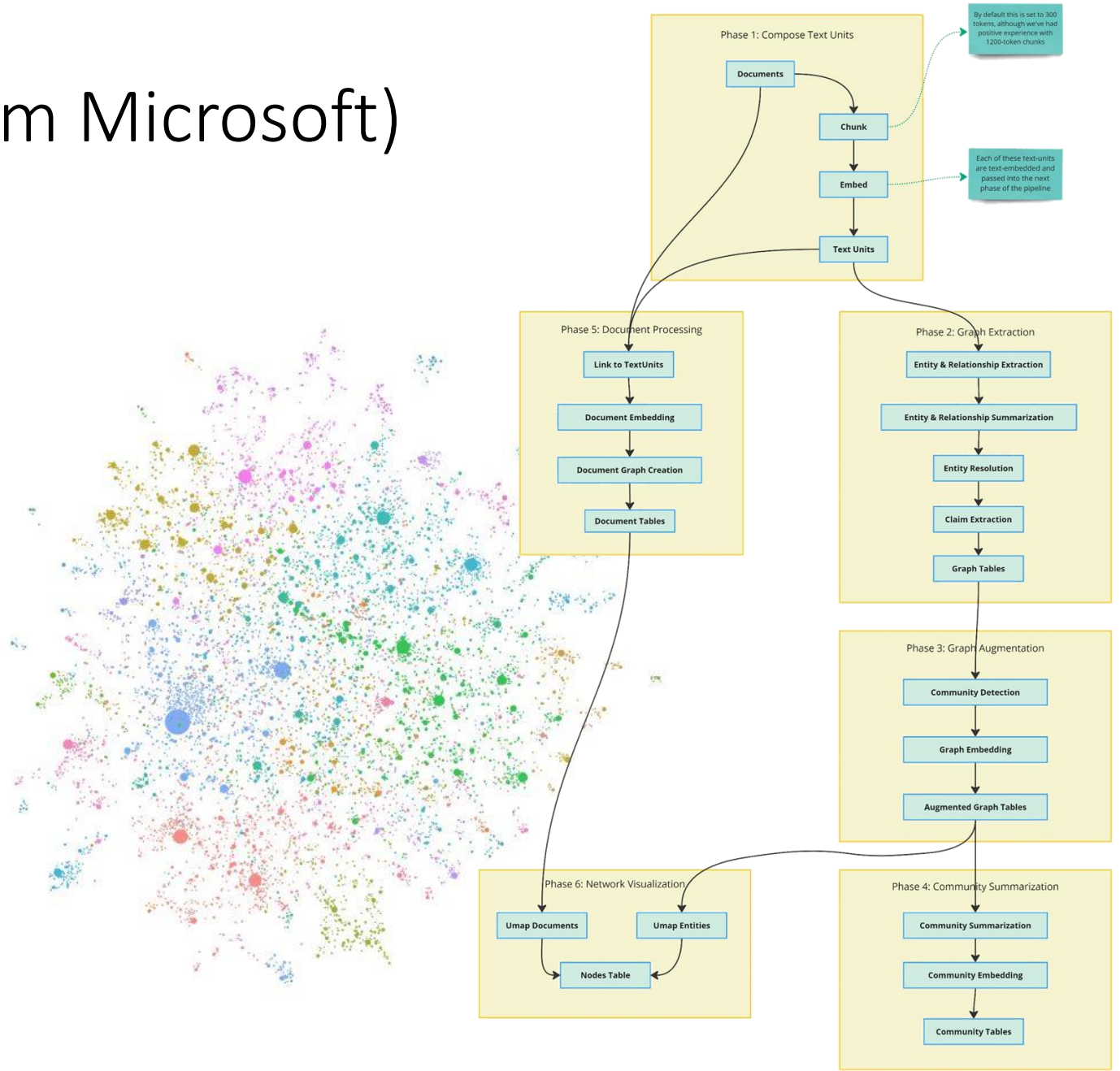
```
documents = SimpleDirectoryReader(
    "../../../../examples/paul_graham_essay/data"
).load_data()
```

```
# define LLM
# NOTE: at the time of demo, text-davinci-002 did not have rate-limit errors

llm = OpenAI(temperature=0, model="text-davinci-002")
```

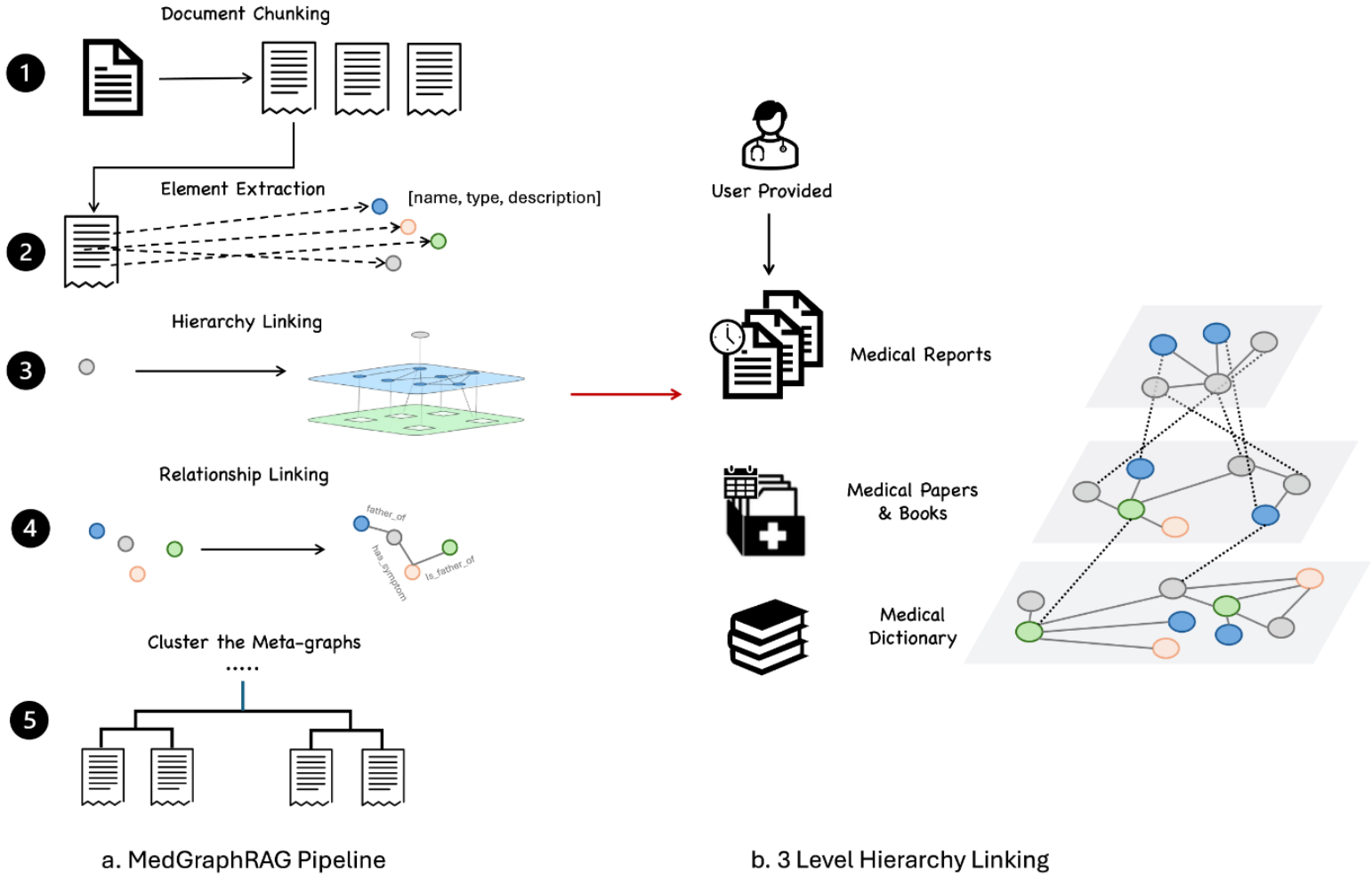
```
from llama_index.indices.property_graph import SchemaLLMPathExtractor

entities = Literal["PERSON", "PLACE", "THING"]
relations = Literal["PART_OF", "HAS", "IS_A"]
schema = {
    "PERSON": ["PART_OF", "HAS", "IS_A"],
    "PLACE": ["PART_OF", "HAS"],
    "THING": ["IS_A"],
}

kg_extractor = SchemaLLMPathExtractor(
  llm=llm,
  possible_entities=entities,
  possible_relations=relations,
  kg_validation_schema=schema,
  strict=True,  # if false, allows values outside of spec
)
```

GraphRAG approach (from Microsoft)

More coming in session 3



Layered Knowledge Graph



a. MedGraphRAG Pipeline

b. 3 Level Hierarchy Linking

Source of the image -- [Medical Graph RAG: Towards Safe Medical Large Language Model via Graph Retrieval-Augmented Generation]

Hands-On 1.2

Building a Knowledge Graph out of my repository of scientific papers

End of session 1