

Hands-on Tutorial

Generating Knowledge Graphs with LLMS, prompt engineering, and RAG

TOTH 2025 Training

Dr Laure Berti-Equille, IRD (France)

Rafail Giannadakis, University of Crete, TALOS AI for SSH (Greece)

Rachel Milio, University of Crete, TALOS AI for SSH (Greece)

Meet The Team



Dr Laure Berti-Equille

Research Director
Computer Science, AI

<https://laureberti.github.io/website/>



Rafail Giannadakis

Research Assistant at
TALOS-AI4SSH (UoC)
Digital Humanities & Classics

<https://crete.academia.edu/RafailGiannadakis>



Rachel Milio

PhD Candidate at
TALOS-AI4SSH (UoC)

<https://orcid.org/0009-0002-0430-4711>

Agenda

Day 1 - June 3rd, 2025

Session 1

Introduction to
Knowledge Graphs &
LLMs

Session 2

LLMs & Prompt
Engineering

Day 2 - June 4th, 2025

Session 3

Knowledge Graph
Creation with GPT4

Session 4

Introduction to RAG

Schedule

<div>DAY</div> <div>TIME</div>	9:00 am - 10:00 am	10:00 am - 10:20 am	10:20 am - 12:30 pm	12:30 pm - 2:00 pm	2:00 pm - 3:00 pm	3:00 pm - 5:00 pm
Day 1	Knowledge Graphs	Coffee Break	Hands-on (1)	Lunch	LLMs	Hands-on (2)
Day 2	Prompting	Coffee Break	Hands-on (3)	Lunch	RAG	Hands-on (4)

Online Resources

Shared Google Drive with the corpus, slides, and exercises:

https://drive.google.com/drive/folders/12N9T1k4LbL4OL23HuFunwGG0fVdx2yy_?usp=sharing

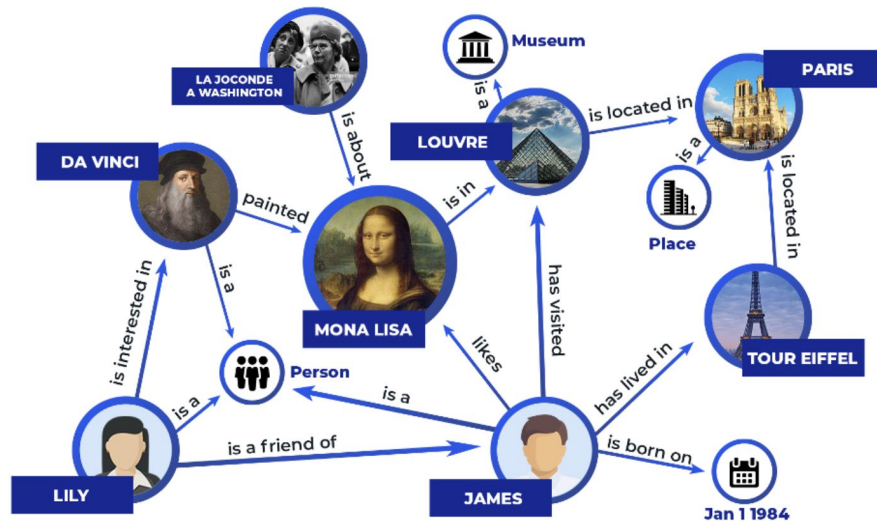
Introduction to KGs

Outline

- 1. Introduction to Knowledge Graphs (KGs)**
- 2. Two Popular Knowledge Graph Data Models:**
 - Resource Description Framework (RDF) (Query language: SPARQL)
 - Property Graphs (Query language: Cypher)
 - Comparison of RDF and Property Graphs
- 3. Combining KGs and LLMs**

Knowledge Graph Definition

- Represent facts about the world in a structured form
- Are directed multi-relational graphs composed of:
 - **entities** as the graph's **nodes**
 - **relationships** between entities as the graph's **edges**
 - An **organizing principle** that captures meta-information about core concepts
- Feature information regarding both real-world objects and abstract concepts structured as **triplets (head entity, relation, tail entity)** or **(h, r, t)**



Knowledge Graph Advantages

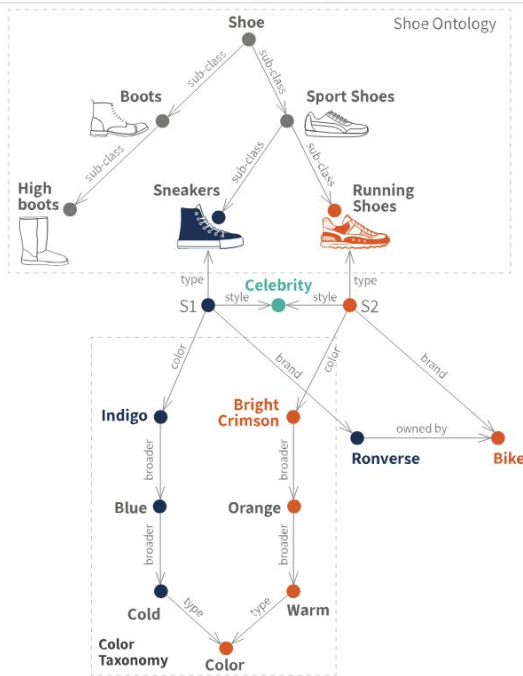
- **Expressivity:** The standards in the Semantic Web stack – RDF(S) and OWL – allow for a fluent representation of various types of data and content: data schema, taxonomies and vocabularies, all sorts of metadata, reference and master data
- **Performance:** All the specifications allow for efficient management of graphs of billions of facts and properties
- **Interoperability:** There is a range of specifications for data serialization, access (SPARQL Protocol for end-points), management (SPARQL Graph Store) and federation. The use of globally unique identifiers facilitates data integration and publishing.
- **Standardization** through the W3C community process.

Example of Knowledge Graph

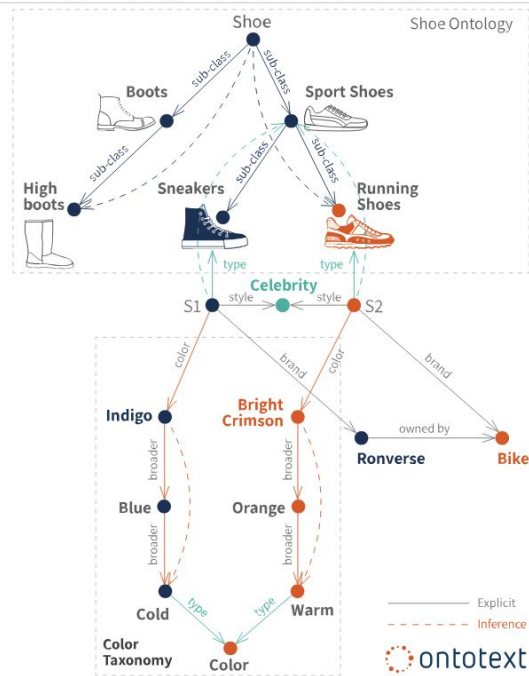
Plain Graph



Knowledge Graph



Knowledge Graph with Inference



Big Knowledge Graphs



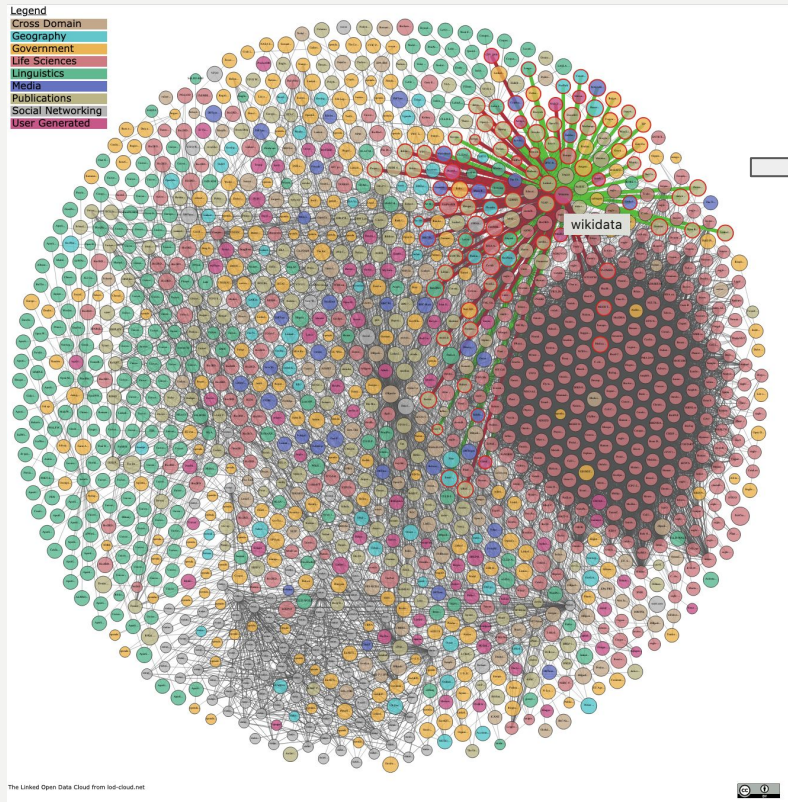
Google knowledge graph



NELL: Never-Ending Language Learning



DBPedia



lod-cloud.net/dataset/wikidata

Data Facts

Total size	12,500,000,000 triples
Namespace	http://www.wikidata.org/entity/
Links to doi	27,367,486 triples
Links to geonames-semantic-web	3,747,452 triples
Links to viaf	6,148,897 triples
Links to freebase	4,397,020 triples
Links to dnb-gemeinsame-normdatei	1,918,745 triples
Links to lcsb	1,338,101 triples
Links to data-bnf-fr	895,058 triples
Links to idref	528,040 triples
Links to oclc-fast	502,221 triples
Links to national-diet-library-authorities	124,052 triples
Links to chembl-rdf	100,223 triples
Links to libris	258,242 triples
Links to babelnet	65,942 triples
Links to getty-ign	29,226 triples
Links to getty-aat	25,080 triples

<https://lod-cloud.net/versions/latest/lod-cloud.svg>

(Knowledge) Graphs: What for?



Centrality

Outliers, Influencers, Vulnerabilities,...



Pathfinding

Shortest Path, Optimal path, Route Optimization,...



Community Detection

Recommendations, Homophily, Outliers,...



Similarity

Recommendations, What-if Analysis, Disambiguation,...



Embeddings

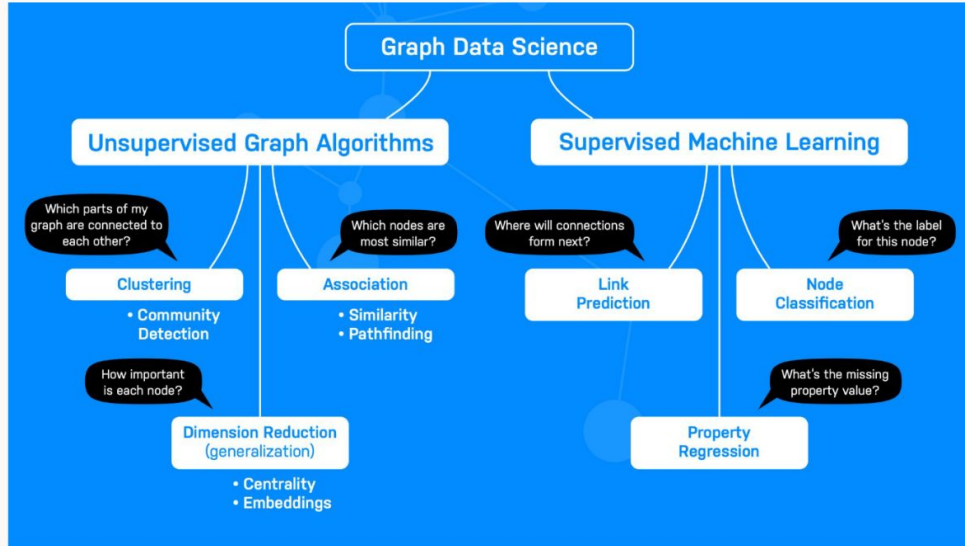
Dimensionality Reduction, Representation Learning, ..



Link Prediction

Link prediction, Recommendations, Next-Best Action,...

Leverage Graph Science



<https://neo4j.com/blog/genai/unifying-llm-knowledge-graph/>

Outline

1. Introduction to Knowledge Graphs (KGs)

2. Two Popular Knowledge Graph Data Models:

- Resource Description Framework (RDF) (Query language: SPARQL)
- Property Graphs (Query language: Cypher)
- Comparison of RDF and Property Graphs

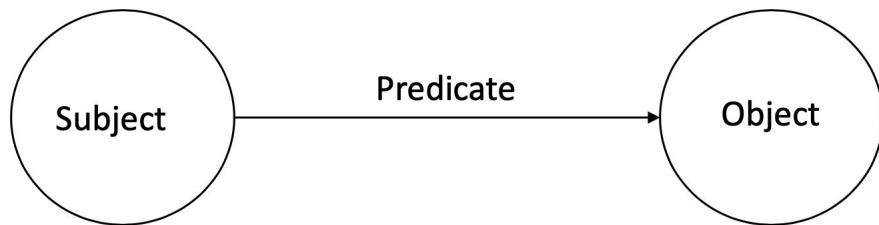
3. Combining KGs and LLMs

Resource Description Framework

- Designed to represent information on the web
- Standardized by World Wide Web (W3C) Consortium
- Triple is the basic unit of representation: Consists of subject, predicate, and object

Nodes can be of three types:

- Internationalized Resource Identifiers (IRI)
IRI: https://hi.wikipedia.org/हिन्दी_विकिपीडिया
- URL: <http://www.wikipedia.org>
- URI: www.wikipedia.org

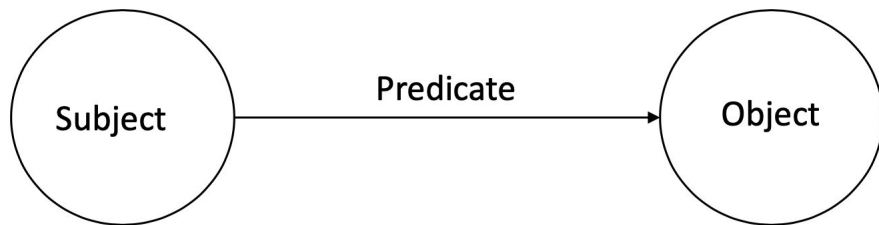


Resource Description Framework

- Designed to represent information on the web
- Standardized by World Wide Web (W3C) Consortium
- Triple is the basic unit of representation: Consists of subject, predicate, and object

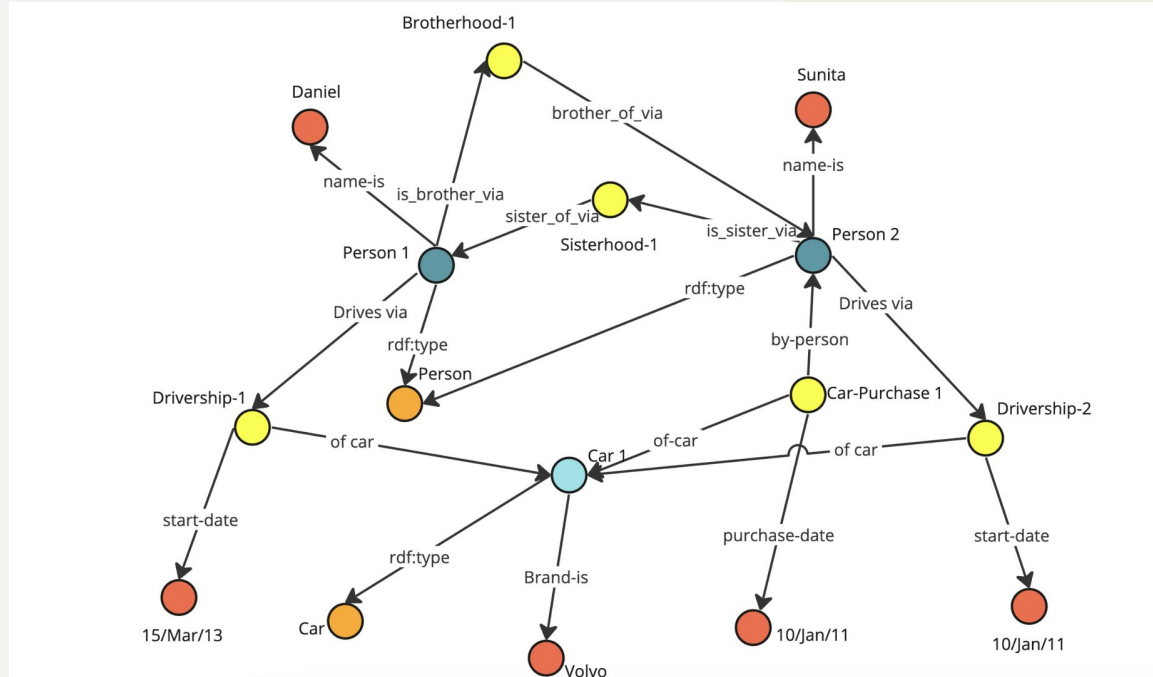
Nodes:

- Uniquely identifies resources on the web
- Literals
- A value of certain type (integer, string, etc.)
- Blank nodes
- A node with no identifier (anonymous)

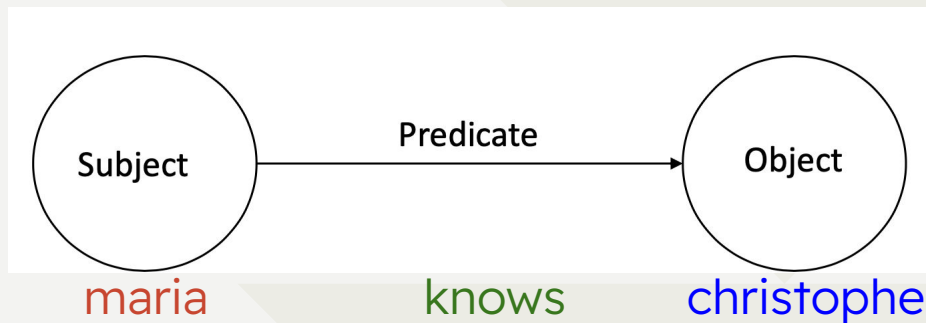


Example of RDF Model

A brother and sister, Daniel and Sunita. Sunita owns and drives a Volvo she purchased on January 10, 2011. Dan has also driven Sunita's car since March 15, 2013.



Simplified Example of RDF



`<http://example.org/maria>`
`<http://xmlns.com/foaf/0.1/knows>`
`<http://example.org/christophe>`

We can define prefixes

@prefix foaf: <http://xmlns.com/foaf/0.1/>

@prefix ex: <http://example.org/>

`ex:maria foaf:knows ex:christophe`

Querying a RDF Dataset



RDF Dataset

- A collection of RDF graphs with
 - Exactly one default graph
 - One or more named graphs
- Name can be a blank node or an IRI

Query Language: SPARQL

- Simple Protocol and Query Language (pronounced “sparkl”)
- Queries can go across multiple sources
- Full-featured query language
- Required/optional parameters
- Filtering the results
- Results can be graphs

Examples of SPARQL Query (1/2)

Who are the persons that maria knows?

```
SELECT ?person
```

```
WHERE
```

```
<http://example.org/maria> <http://xmlns.com/foaf/0.1/knows> ?person
```

Examples of SPARQL Query (2/2)

Who are the persons known by the persons that maria knows?

```
SELECT ?person ?person1
```

```
WHERE
```

```
<http://example.org/maria> <http://xmlns.com/foaf/0.1/knows> ?person  
?person <http://xmlns.com/foaf/0.1/knows> ?person1
```

or

```
PREFIX ex: <http://example.org/>
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT ?person ?person1
```

```
WHERE
```

```
ex:maria foaf:knows ?person
```

```
?person foaf:knows ?person1
```

Outline

1. Introduction to Knowledge Graphs (KGs)

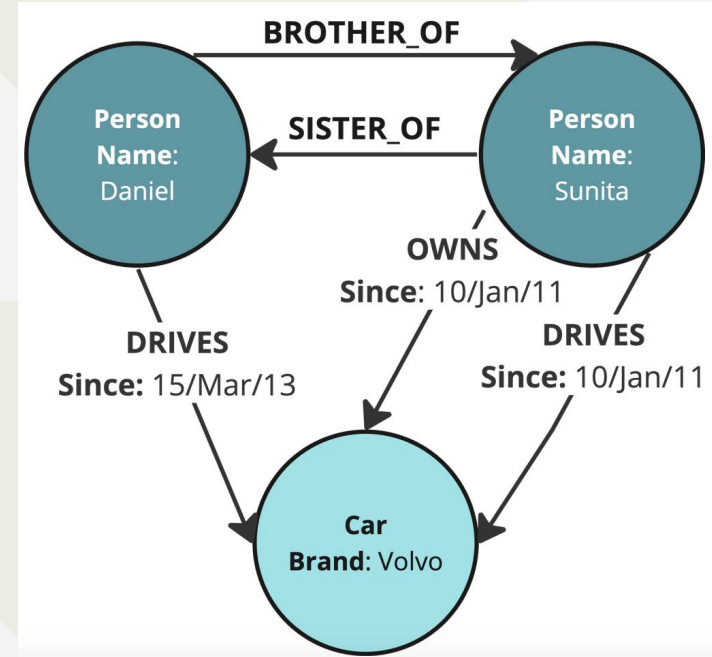
2. Two Popular Knowledge Graph Data Models:

- Resource Description Framework (RDF) (Query language: SPARQL)
- Property Graphs (Query language: Cypher)
- Comparison of RDF and Property Graphs

3. Combining KGs and LLMs

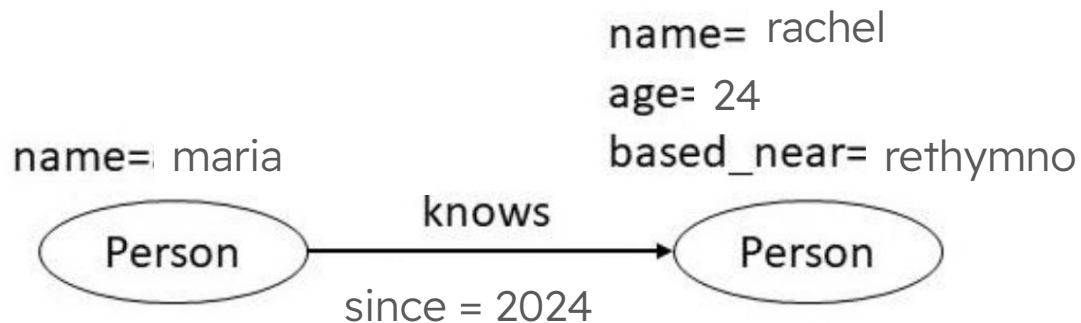
Property Graph Model

- Do not require a predefined schema
- Optimize graph traversals
- Used by many Graph DB
- Based on the ISO standard GQL
<https://www.iso.org/standard/76120.html>



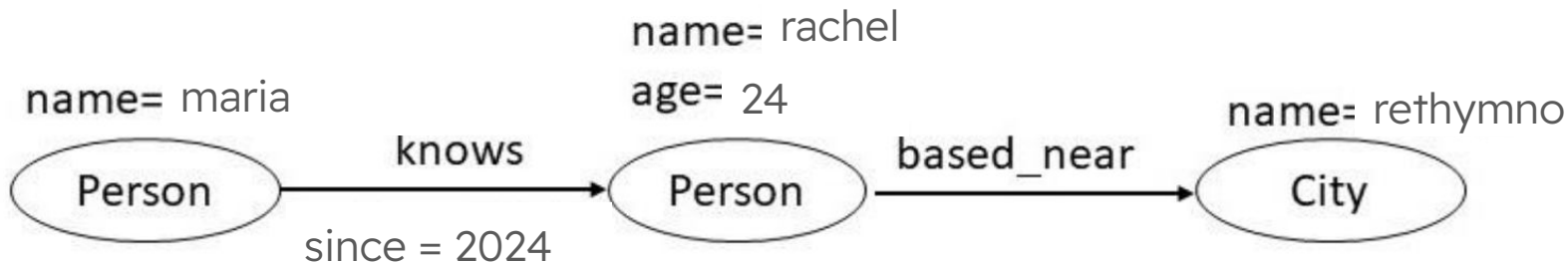
Property Graph Model

- Nodes, relationships and properties
- Each node and a relationship has a label and set of properties
- Properties are key value pairs
- Keys are strings, values can be any data types
- Each relationship has a direction



Property Graph Model

- Nodes, relationships and properties
- Each node and a relationship has a label and set of properties
- Properties are key value pairs
- Keys are strings, values can be any data types
- Each relationship has a direction



Cypher Queries on Property Graphs

- Query language for querying graph data
- Being considered for adoption as an ISO Standard
- Supports CRUD operations: Create, read, update, delete

Example:

Which people does maria know?

```
MATCH (p1:Person {name: maria}) -[:knows]-> (p2: Person)
RETURN p2
```

Cypher Query Examples

- Which people does maria know since 2010?

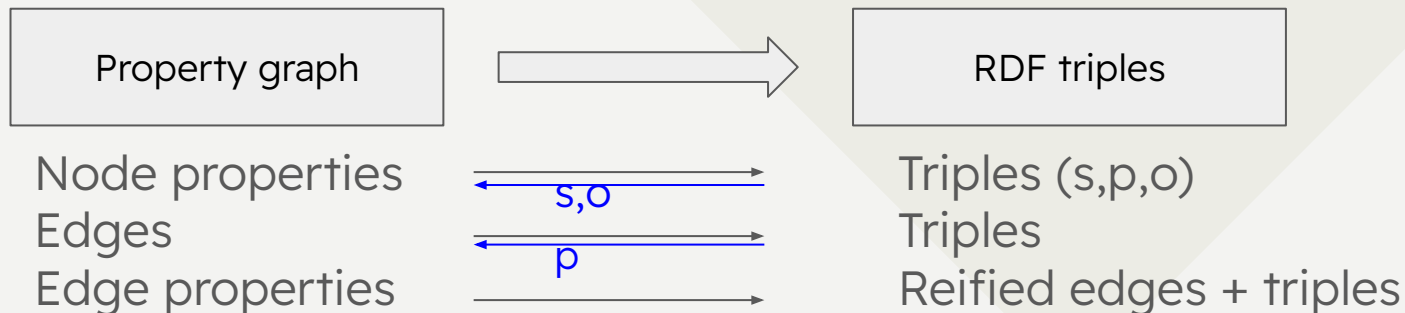
```
MATCH (p1:Person {name: maria}) -[:knows {since: 2010}]-> (p2: Person)
RETURN p1, p2
```

- Which people does maria know since 2010?

```
MATCH (p1:Person {name: maria}) -[:knows {since: Y}]-> (p2: Person)
WHERE Y <= 2010
RETURN p1, p2
```


RDF vs Property Graphs

- RDF supports several additional layers
- RDF Schema, Web Ontology, etc.
- Property graph model supports edge properties
- Property graph model does not require IRIs
- Property graph model does not support blank nodes



Cypher Query Examples

- Which people does maria know since 2010?

```
MATCH (p1:Person {name: maria}) -[:knows {since: 2010}]-> (p2: Person)
RETURN p1, p2
```

- Which people does maria know since 2010?

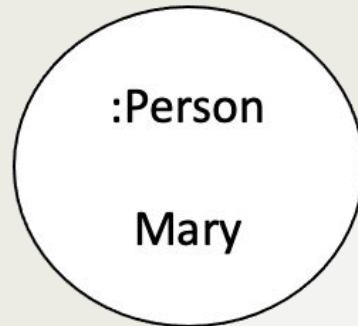
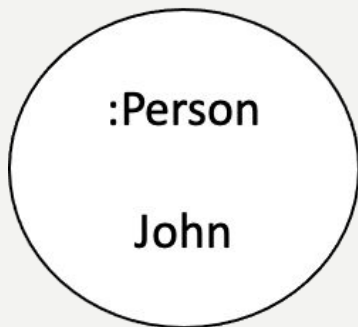
```
MATCH (p1:Person {name: maria}) -[:knows {since: Y}]-> (p2: Person)
WHERE Y <= 2010
RETURN p1, p2
```

Design a Property Graph

- Choosing nodes, labels and properties
- When to introduce relationships
- When to introduce relationship properties
- How to handle n-ary relationships

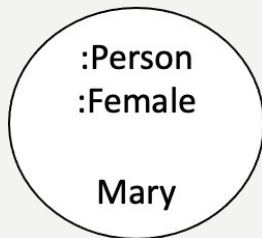
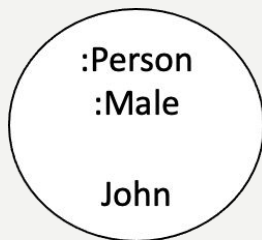
Choose Nodes, labels, and Properties

- Nodes usually represent entities in a domain
 - How to represent gender?



Choose Nodes, labels, and Properties

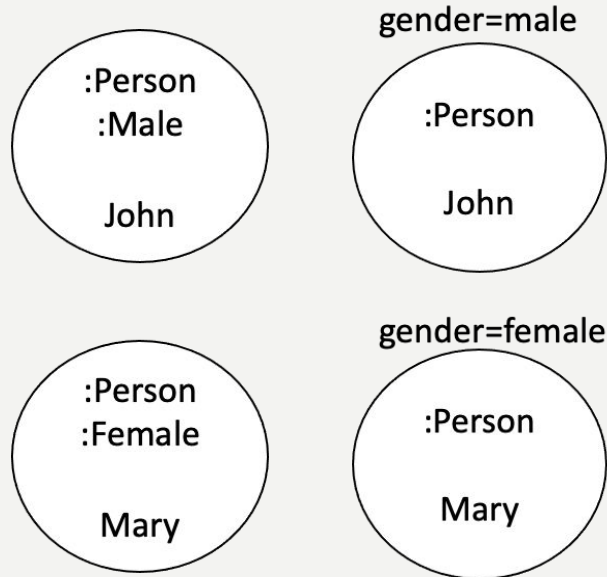
- Nodes usually represent entities in a domain
 - How to represent gender?



- Introduce a new Label
- Labels are like classes
- Labels should be natural
- Labels should not change with time
- Could benefit from indexing

Choose Nodes, labels, and Properties

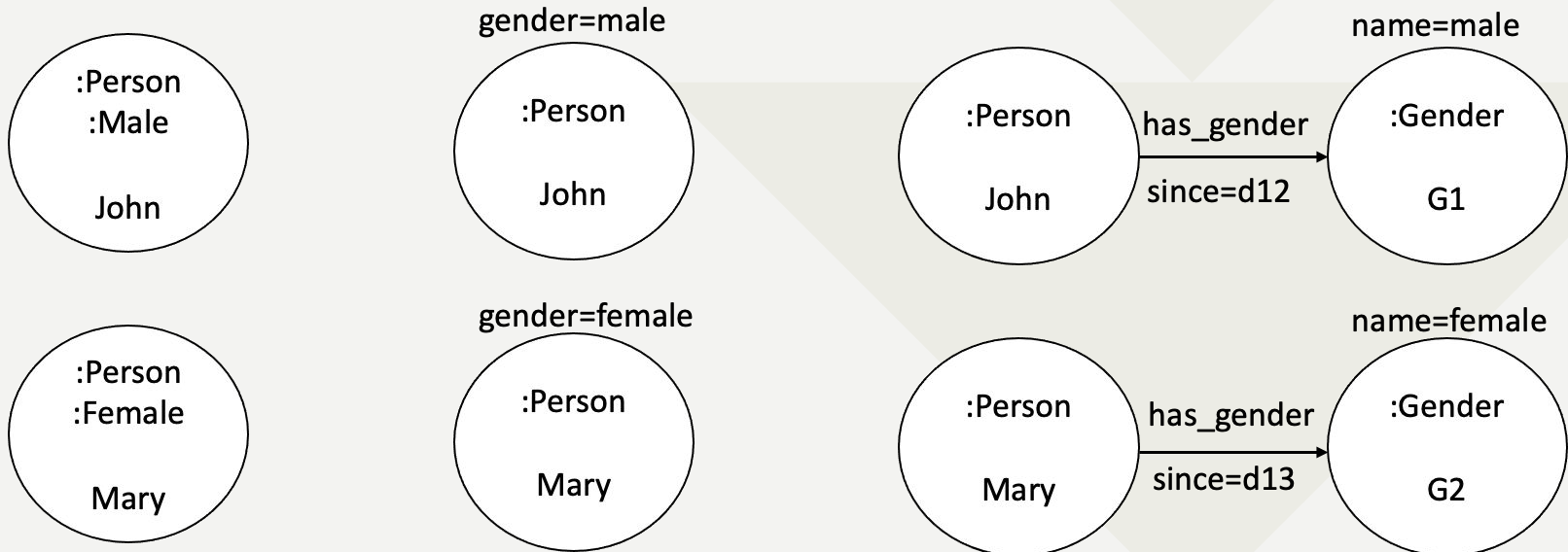
- Nodes usually represent entities in a domain
 - How to represent gender?



- Introduce a new node property
- Property should not change with time

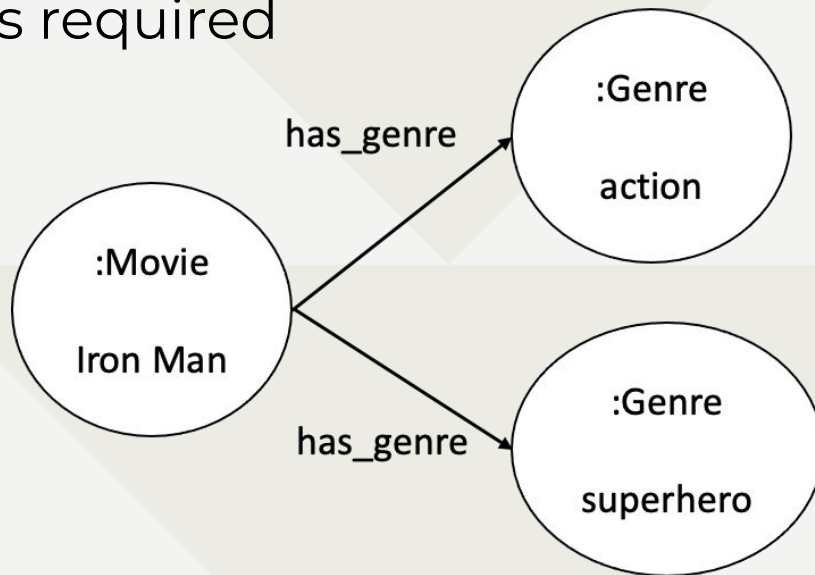
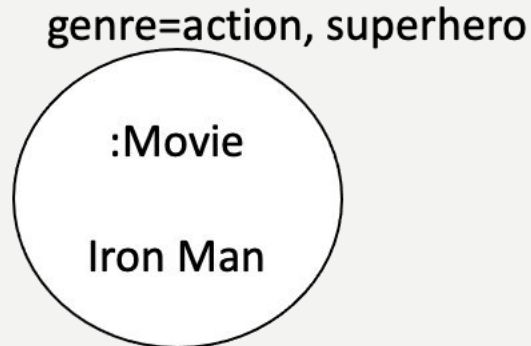
Choose Nodes, labels, and Properties

- Nodes usually represent entities in a domain
 - How to represent gender?



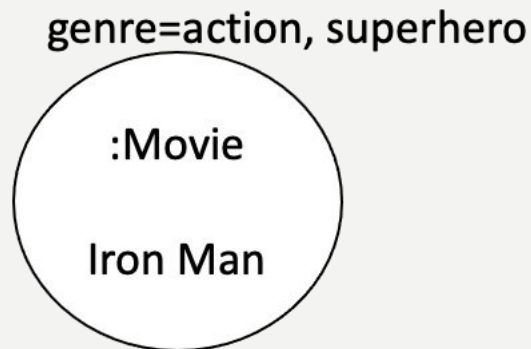
When to introduce a relationship?

- When efficient access is required



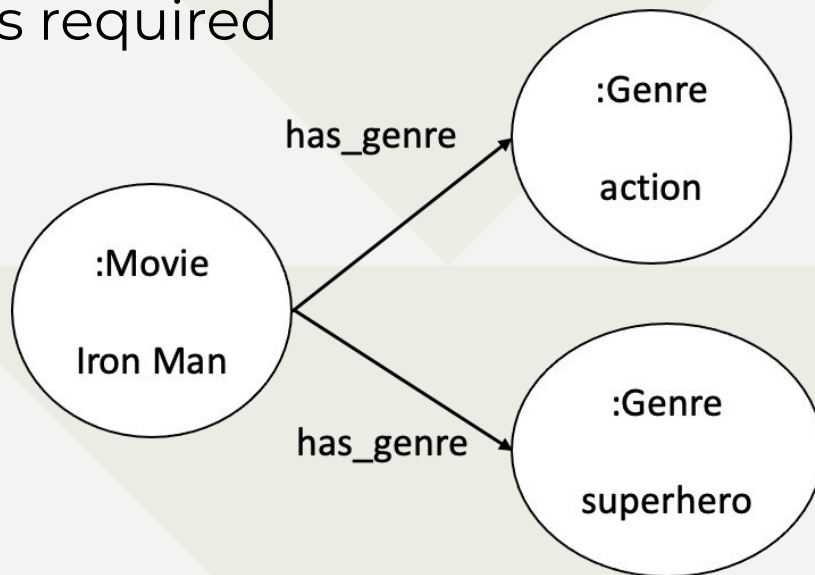
When to introduce a relationship?

- When efficient access is required



Find movies that have genres in common

```
MATCH (m1:Movie), (m2:Movie)
WHERE any(x IN m1.genre WHERE x IN m2.genre)
AND m1 <> m2
RETURN m1, m2
```



```
MATCH (m1:Movie)-[:has_genre]->(g:Genre),
      (m2:Movie)-[:has_genre]->(g)
WHERE m1 <> m2
RETURN m1, m2
```

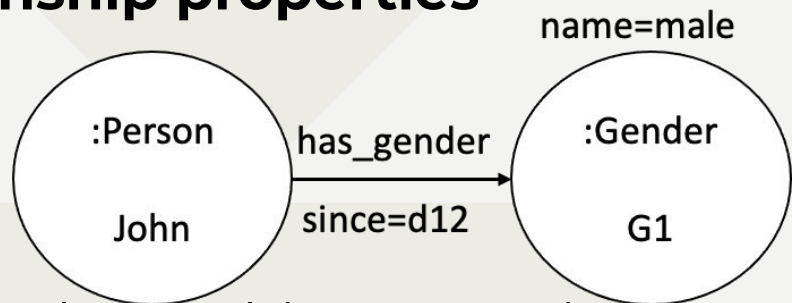
When to introduce relationship properties?

- **Common use cases for relationship properties**

- Time varying relationships
- Provenance
- Confidence
-

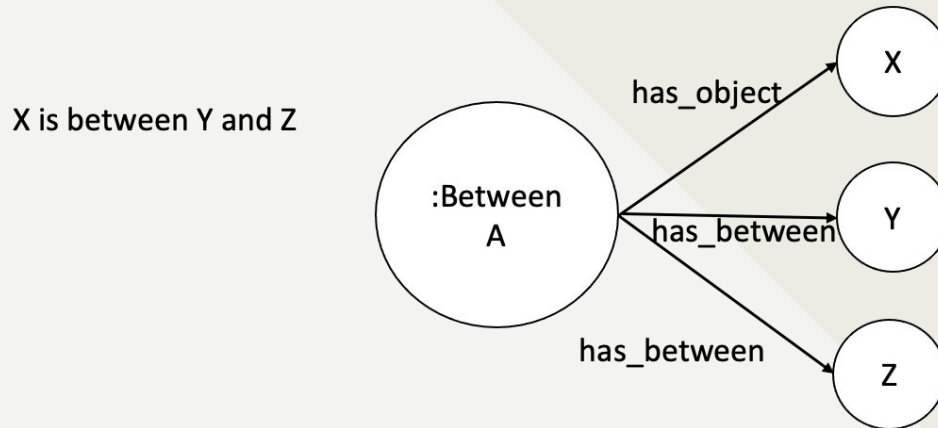
- **Disadvantages**

- Many systems do not index relationship properties
- This may not be a problem if relationship properties are used in the last stage of query processing
- For performance sensitive queries, it is better to reify the relationship



How to handle n-ary relationships?

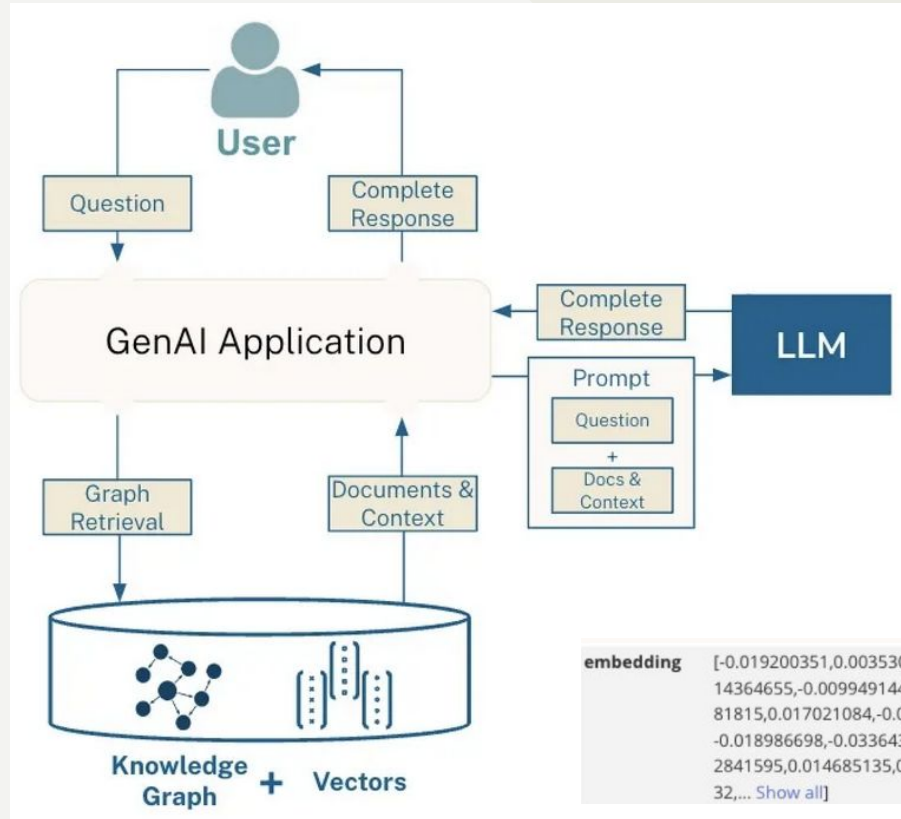
- Reification is a common technique to handle relationships with arity higher than 2
 - Create an object representing the relationship
 - Create objects for each argument of the relationship
 - Introduce relationships to connect them



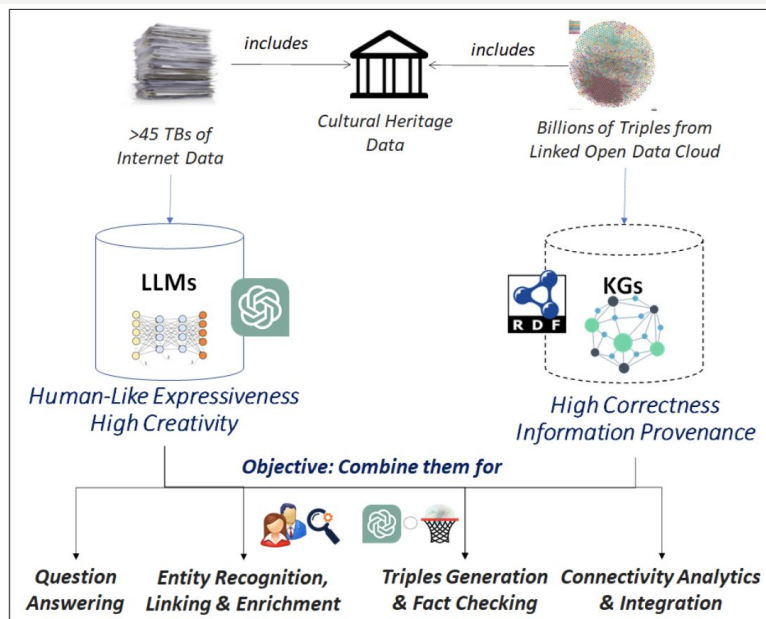
Outline

1. Introduction to Knowledge Graphs (KGs)
2. Two Popular Knowledge Graph Data Models:
 - Resource Description Framework (RDF) (Query language: SPARQL)
 - Property Graphs (Query language: Cypher)
 - Comparison of RDF and Property Graphs
3. Combining KGs and LLMs

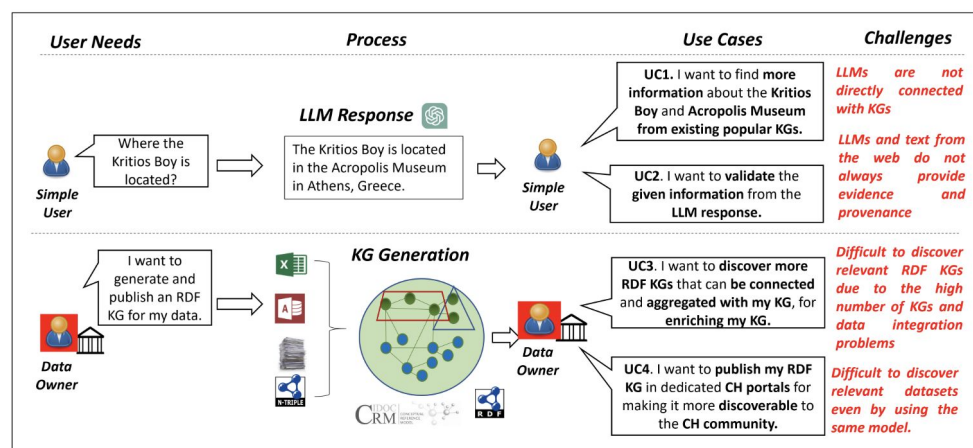
Generic Data Flow



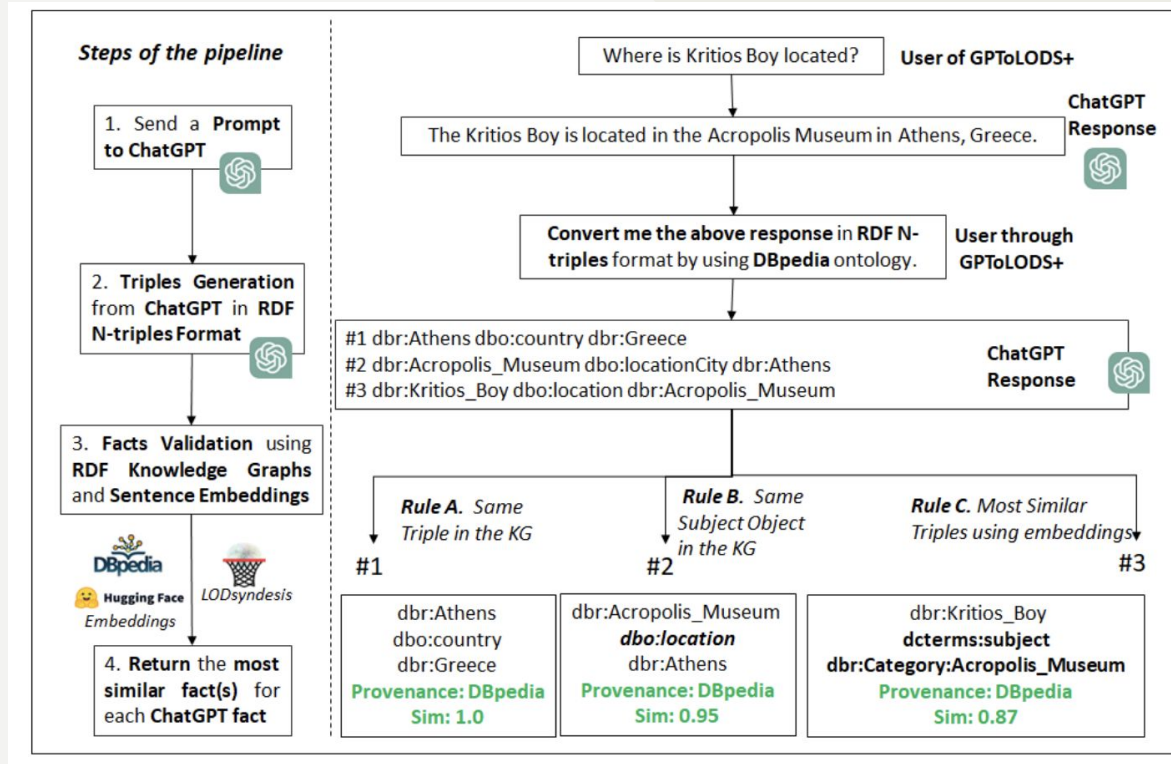
Use Cases in Cultural Heritage (1/3)



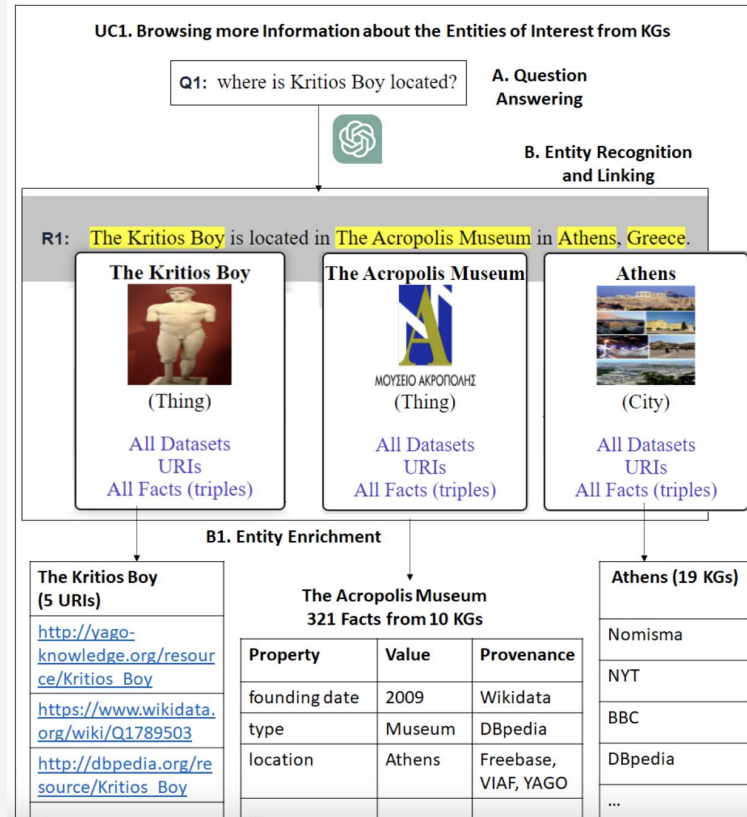
Vision: Combining LLMs and KGs over CH data

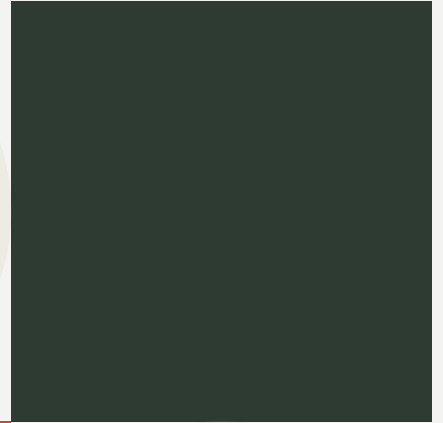


Use Cases in Cultural Heritage (2/3)



Use Cases in Cultural Heritage (2/3)





Further Reading

- A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, *ACM Computing Surveys (Csur)* 54 (2021) 1–37. <https://dl.acm.org/doi/abs/10.1145/3447772>
- Peng, C., Xia, F., Naseriparsa, M. et al. Knowledge Graphs: Opportunities and Challenges. *Artif Intell Rev* **56**, 13071–13102 (2023). <https://doi.org/10.1007/s10462-023-10465-9>
- S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang and X. Wu, Unifying Large Language Models and Knowledge Graphs: A Roadmap, in *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3580-3599, July 2024, doi: 10.1109/TKDE.2024.3352100.
- Ibrahim, N., Aboulela, S., Ibrahim, A. et al. A survey on augmenting knowledge graphs (KGs) with large language models (LLMs): models, evaluation metrics, benchmarks, and challenges. *Discov Artif Intell* **4**, 76 (2024). <https://doi.org/10.1007/s44163-024-00175-8>

Introduction to LLMs

Outline

1. Introduction to AI and LLMs

2. Key Elements of LLMs

- Parameters
- Embeddings
- Attention

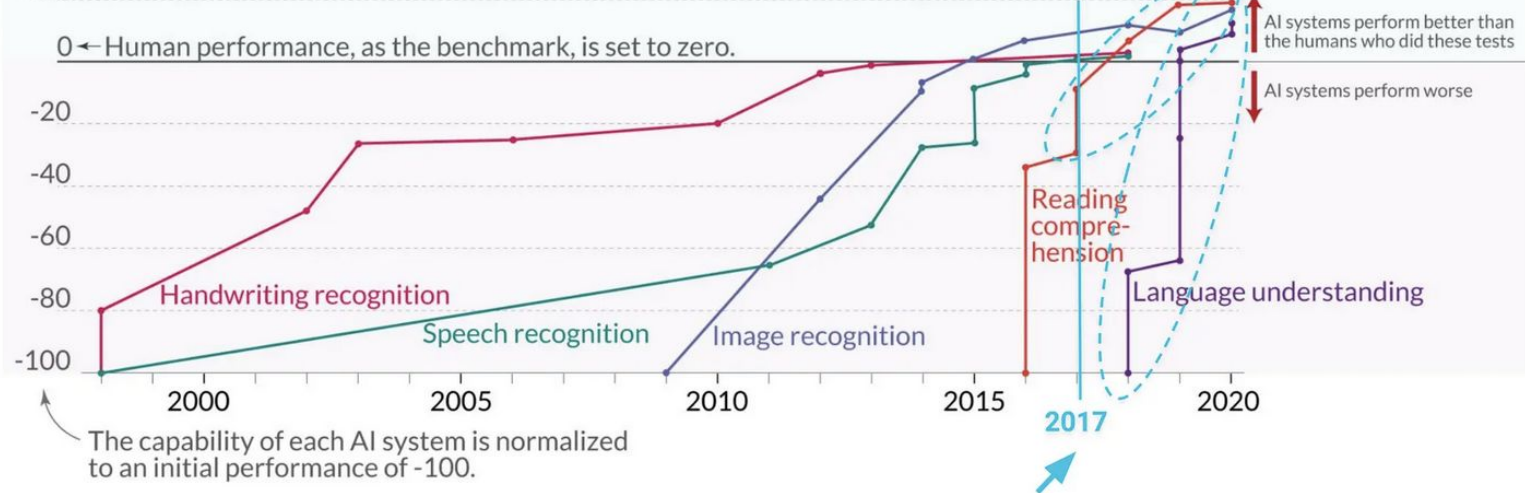
3. LLMs Architecture

- Types of architecture
- How LLM works
- How to set the LLM hyperparameters

Brief History of AI

Language and image recognition capabilities of AI systems have improved rapidly

Test scores of the AI relative to human performance
+20



Data source: Kiela et al. (2021) – Dynabench: Rethinking Benchmarking in NLP
OurWorldinData.org – Research and data to make progress against the world's largest problems.

Transformers

Licensed under CC-BY by the author Max Roser

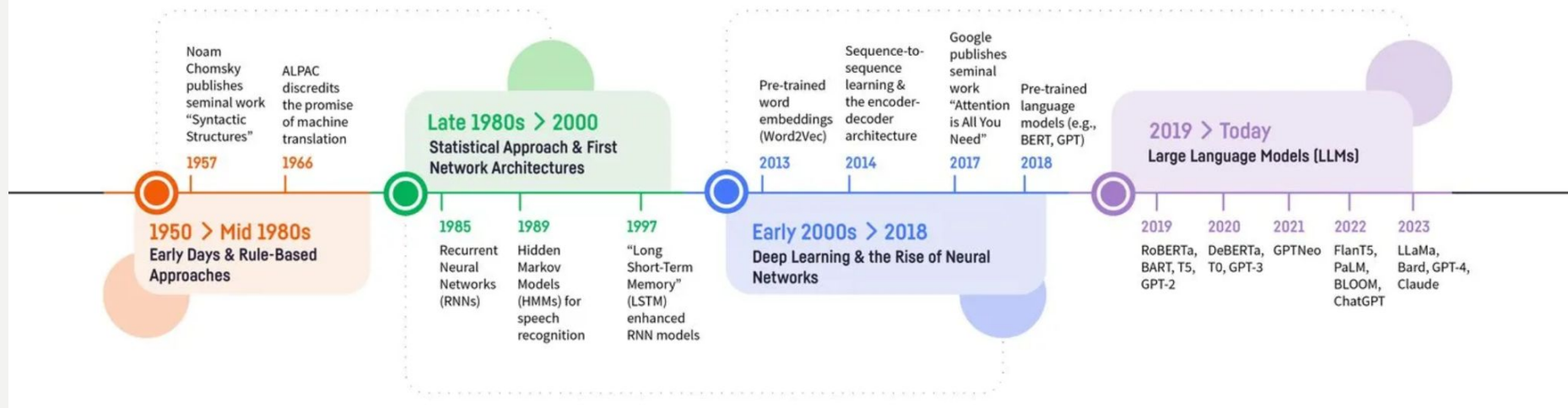
- Kiela et al. (2021), Dynabench: Rethinking Benchmarking in NLP: arxiv.org/abs/2104.14337
- Roser (2022), The brief history of artificial intelligence: The world has changed fast – what might be next?: ourworldindata.org/brief-history-of-ai

Text and shapes in blue have been added to the original work from Max Roser.

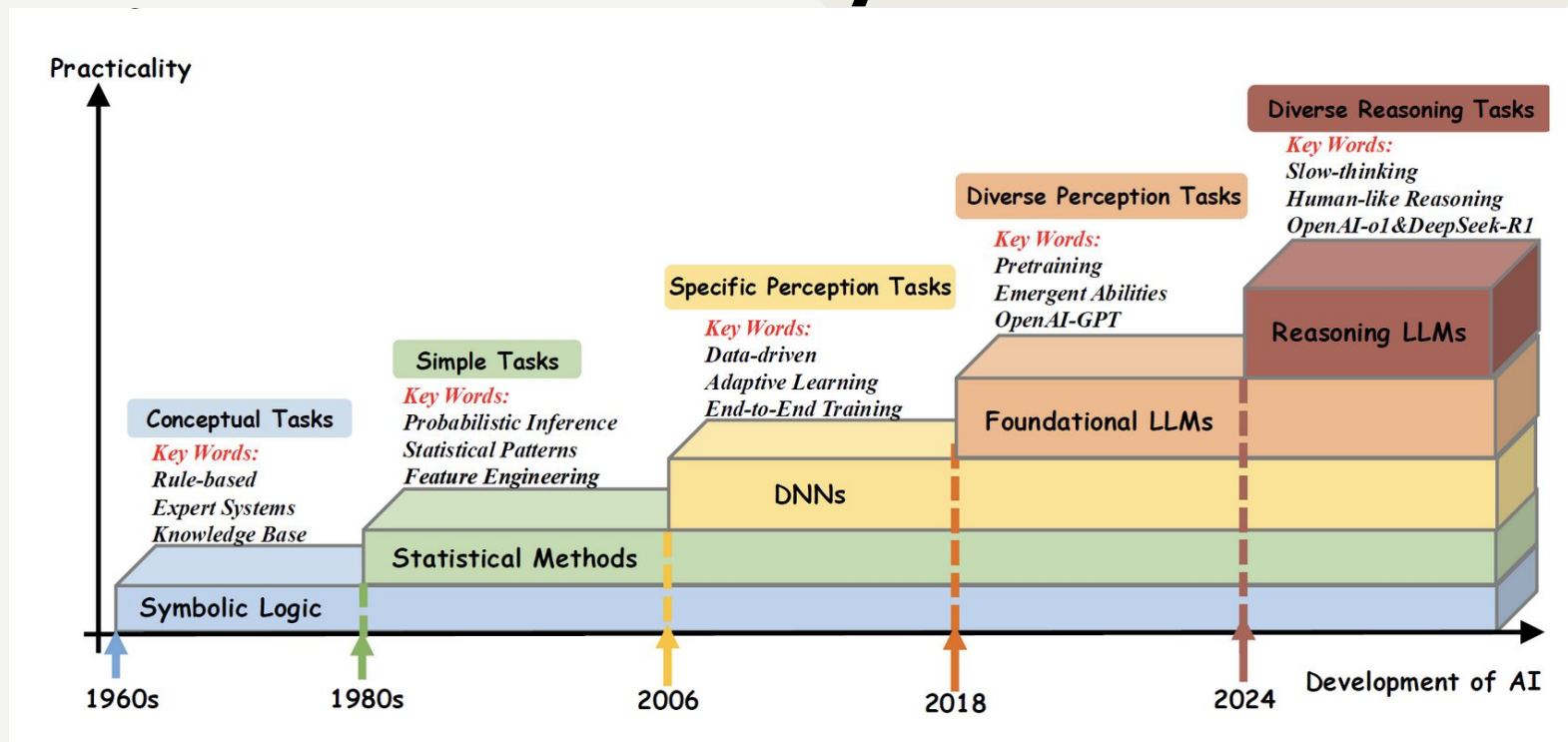
<https://ourworldindata.org/brief-history-of-ai>

History of NLP

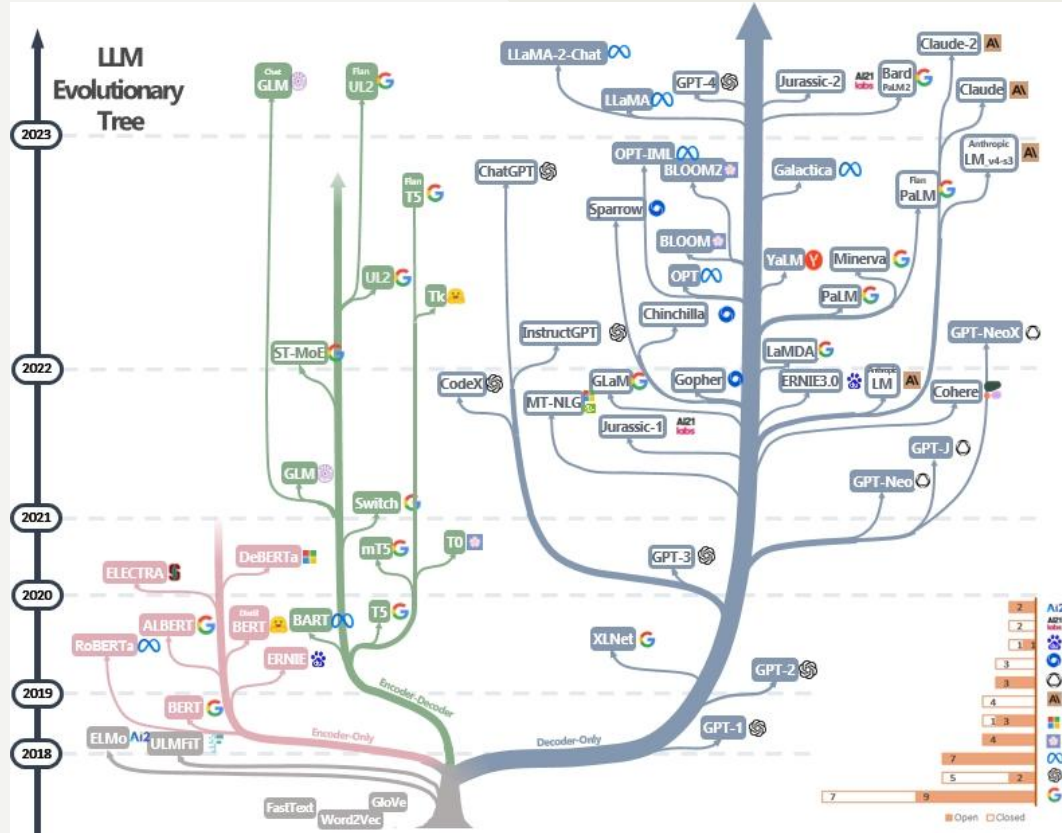
The History of NLP



Brief History of LLMs

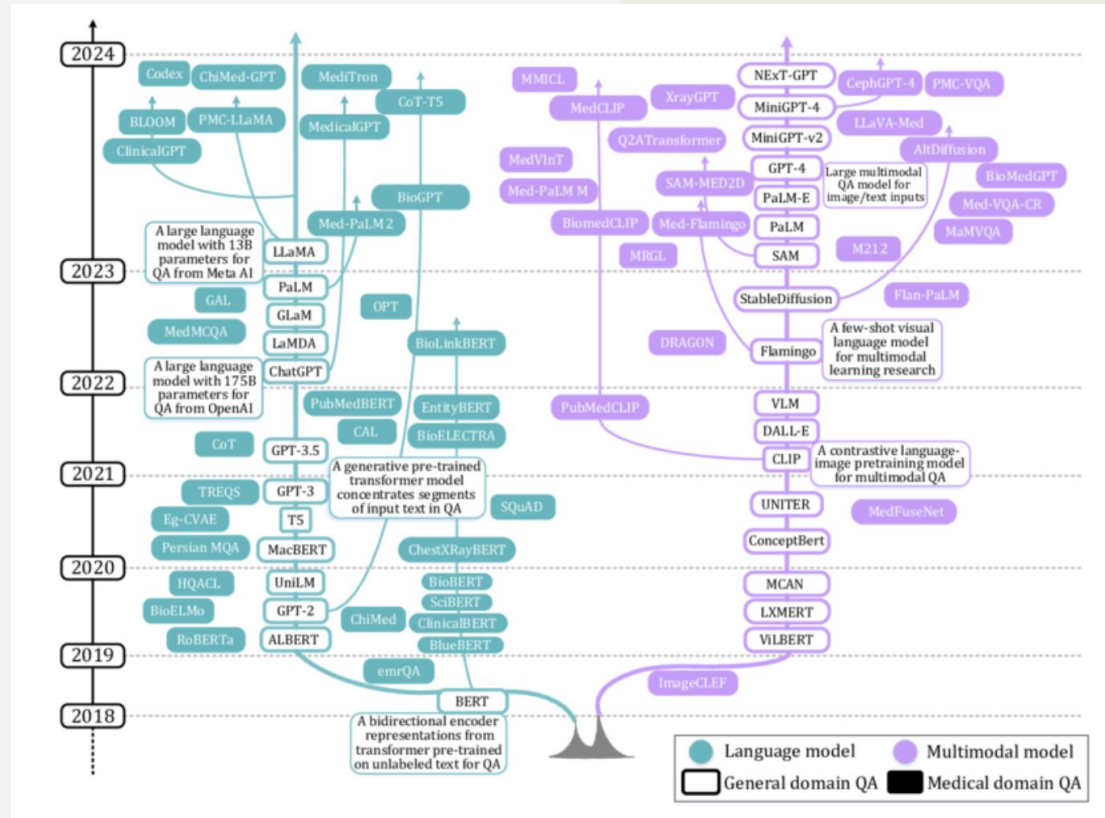


LLM Evolution

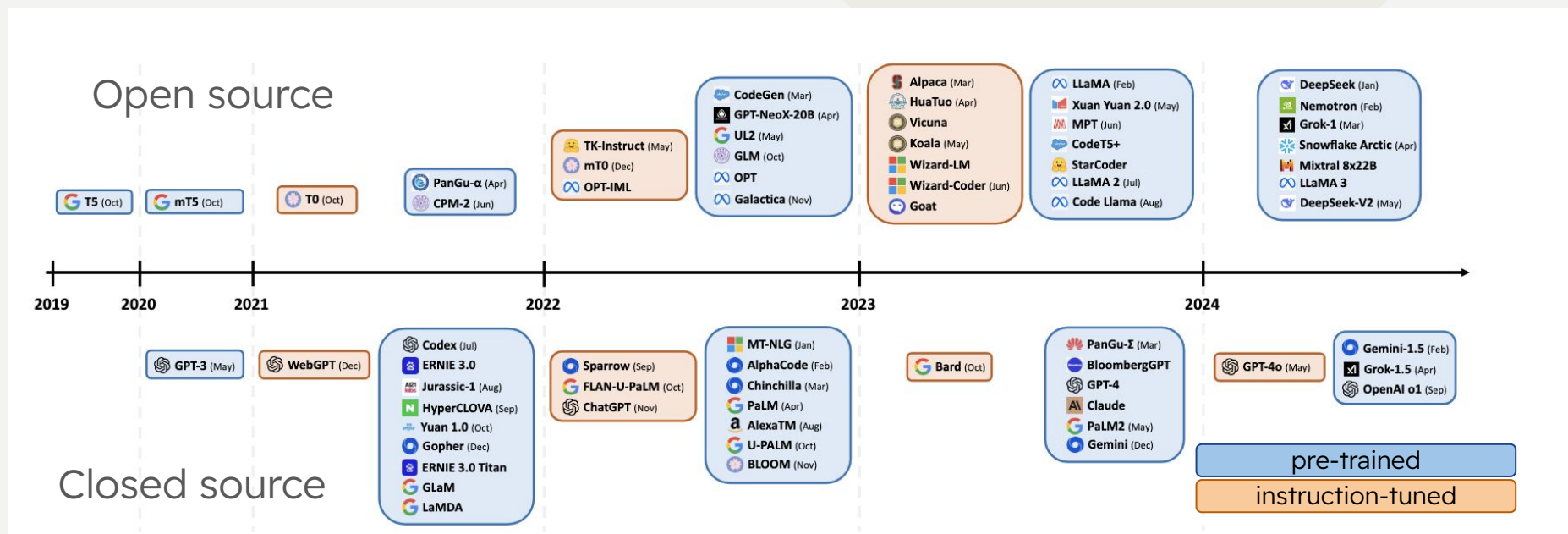


<https://github.com/Mooler0410/LLMsPracticalGuide?tab=readme-ov-file>

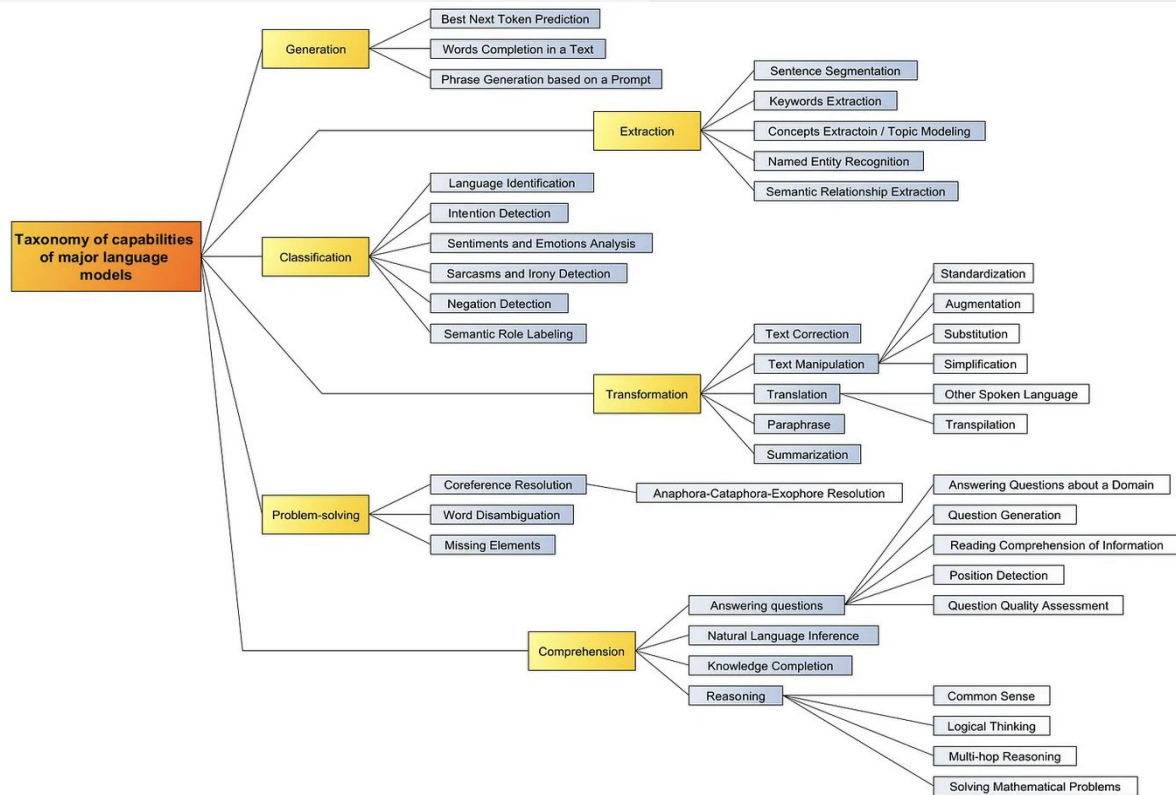
LLM Evolution



Open/Closed LLMs



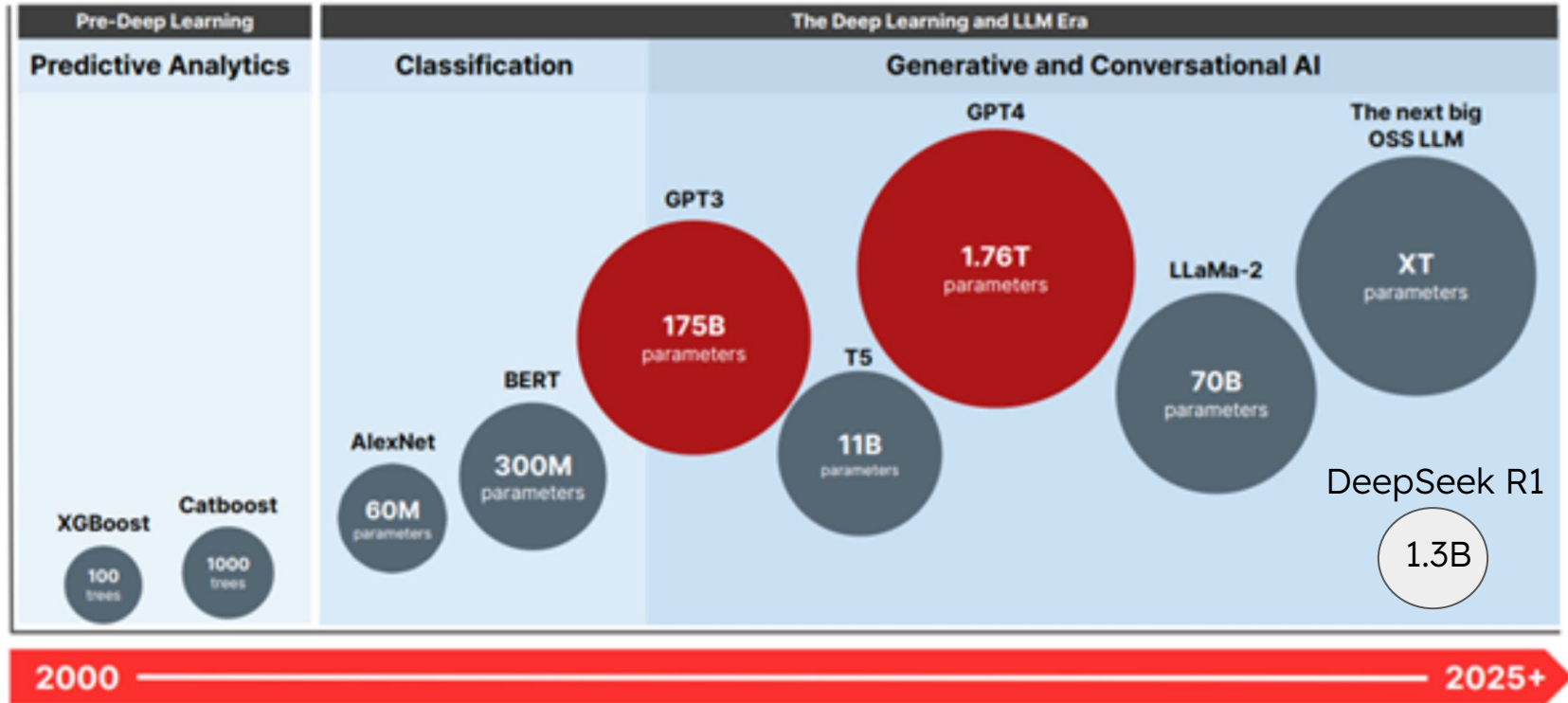
LLM Capabilities



Types of Models

- **Pretrained LLMs:** Model weights are learned from a vast repository of knowledge
- **Fine-tuned LLMs:** The pretrained model weights are used and updated from specialized knowledge tailored for specific tasks
- **Instruction-tuned LLMs:** Not only relies on data but also adapts the outputs based on provided instructions
- **RL-tuned LLMs:** Continuous learning and adaptation through feedback

LLM Size Evolution



Outline

1. Introduction to AI and LLMs

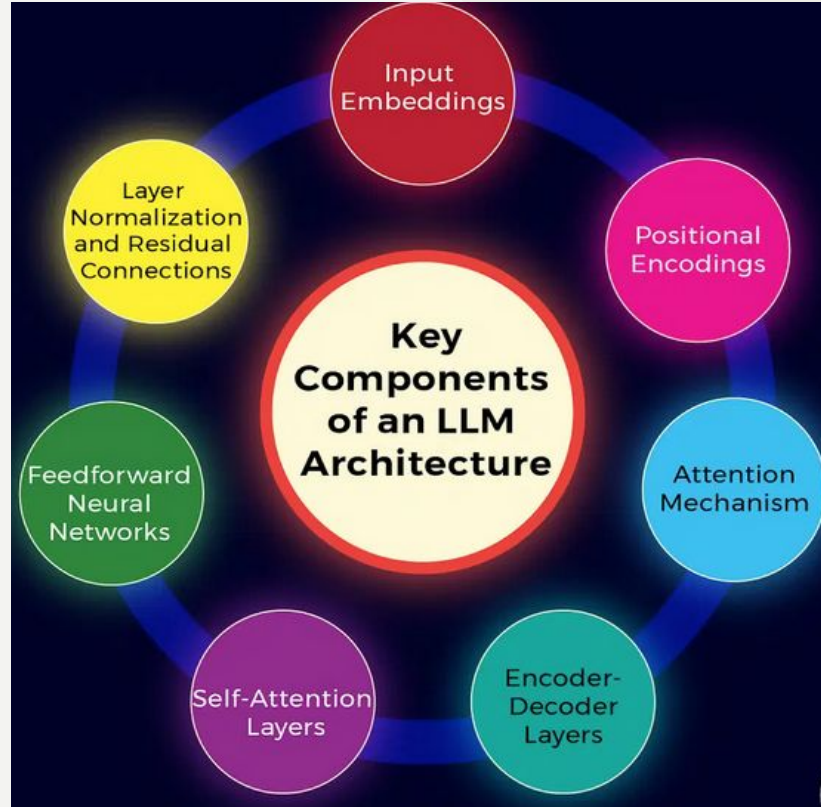
2. Key Elements of LLMs

- Parameters
- Embeddings
- Attention

3. LLMs Architecture

- Types of architecture
- How LLM works
- How to set the LLM hyperparameters

Key elements of LLMs



LLM Parameters

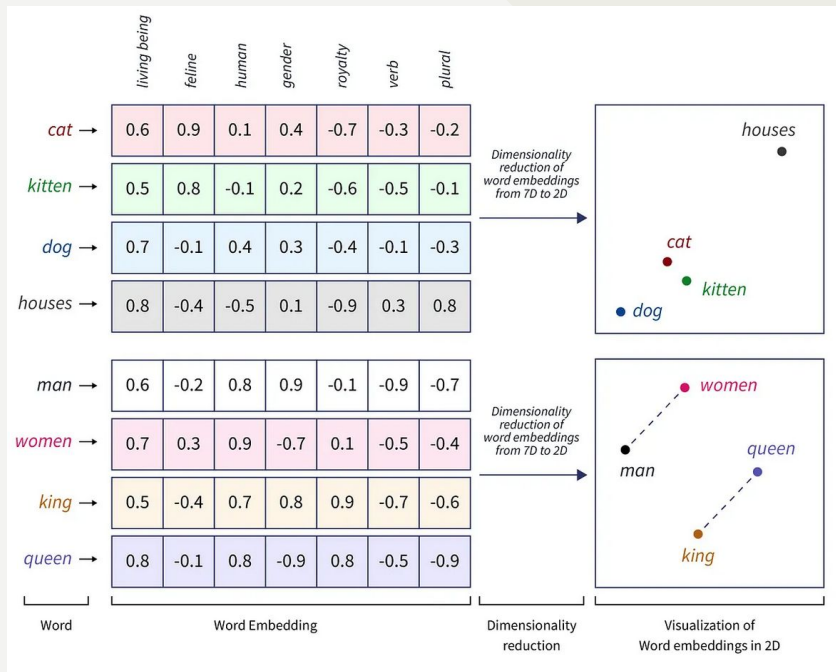
Attention weights and embedding vectors are the main LLM parameters.

- The attention mechanism empowers a model to selectively zoom into pivotal segments of input, sidelining the extraneous bits. The attention weights dictate this selective focus.
- Embedding vectors transmute textual tokens into arrays of numbers encapsulating semantics and context.

Embeddings

Word embeddings are dense, low-dimensional, and continuous vector representations of words that capture semantic and syntactic information.

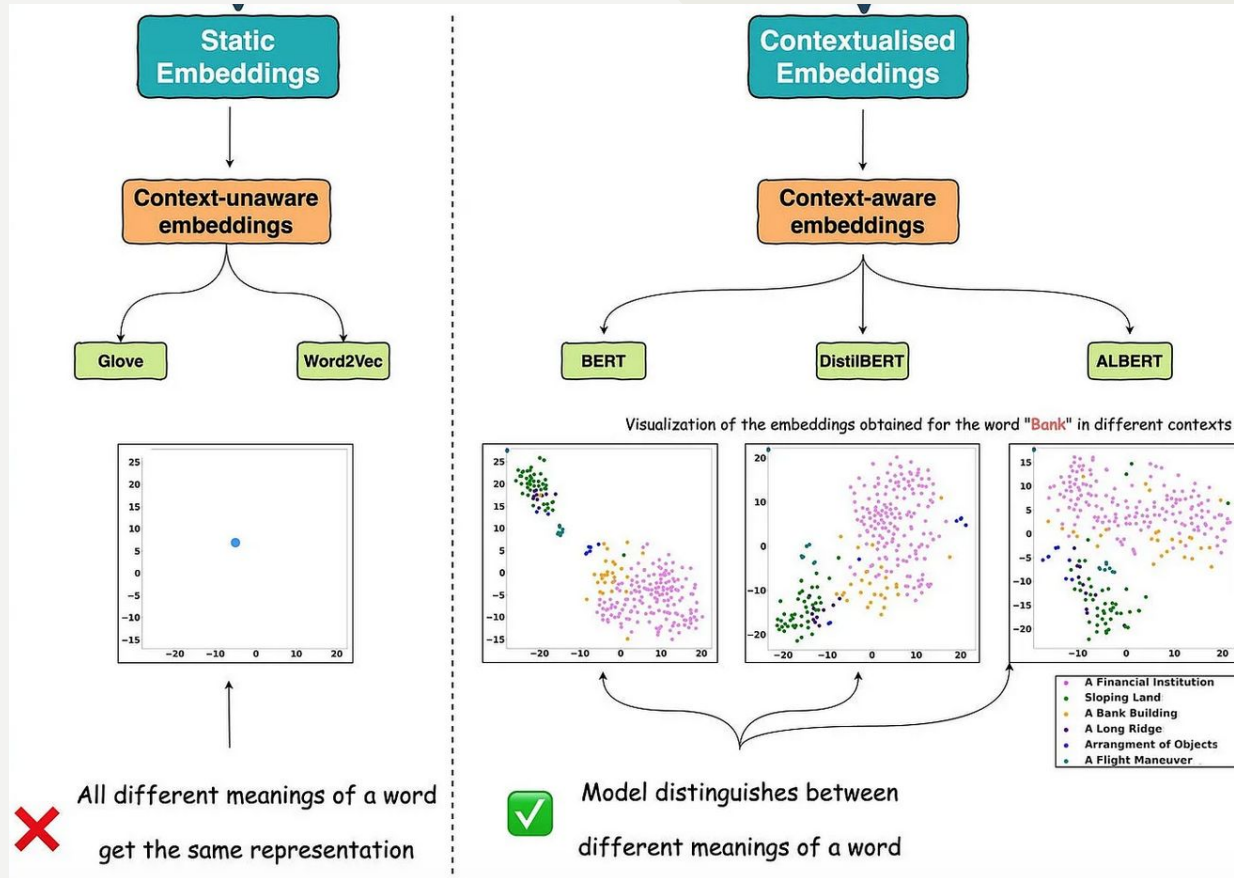
If “king” is represented by a vector v_{king} and “queen” by v_{queen} , the relationship between these vectors can capture the gender difference, as in $v_{\text{king}} - v_{\text{man}} + v_{\text{woman}} \approx v_{\text{queen}}$.



Many embeddings

- Word
- Sentence
- Document
- Image
- Audio
- Context
- User
- etc.

Types of Embeddings



Attention in LLM

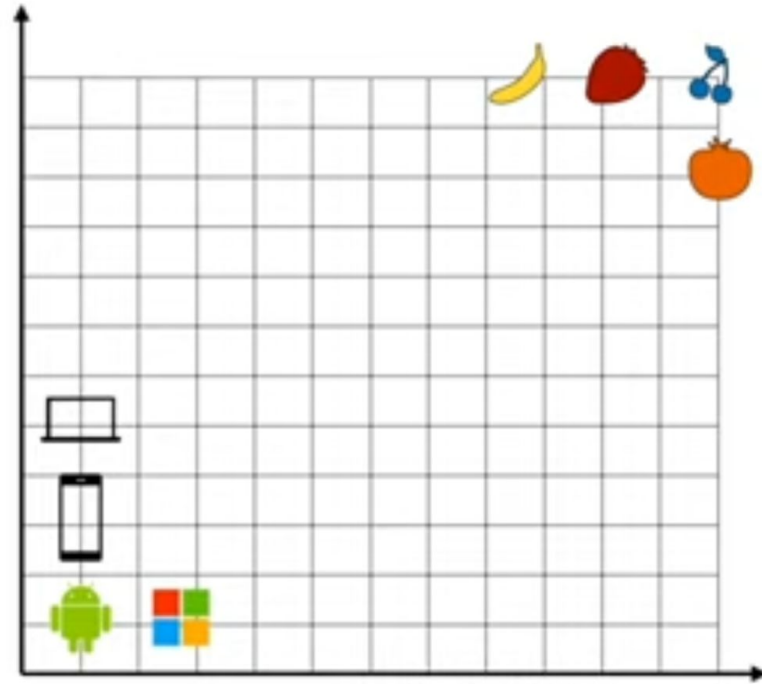
Attention mechanisms allows the model to focus on specific parts of the input text when generating an output.

Types of attention

- **Self-Attention** allows LLMs to understand the context of each word in relation to every other word in the sequence, capturing dependencies and relationships across long distances.
- **Scaled Dot-Product Attention** involves computing the dot product between a query vector and a set of key vectors, and then scaling the result by the square root of the dimensionality of the key vectors.
- **Multi-Head Attention** enhances the model's ability to process input sequences. Instead of relying on a single attention head, which computes weighted sums of input elements based on their relevance to a specific context, multi-head attention employs multiple attention heads simultaneously. Each head focuses on different aspects of the input, such as local dependencies, global context, or specific patterns.

Seminal paper “Attention is all you need” <https://arxiv.org/abs/1706.03762>


Simplified Example



Top right or bottom left?

Cherry 

Android 

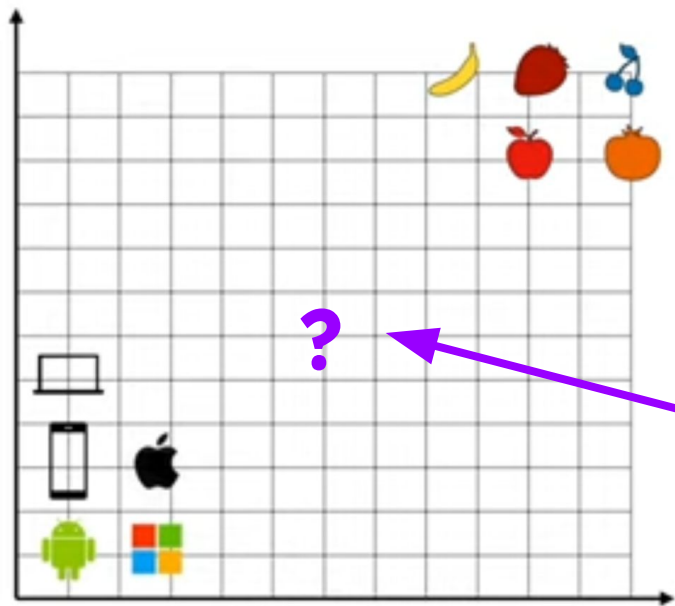
Laptop 

Banana 


Apple?

Simplified Example

Embeddings learn dependencies, proximity of words, and contextual similarity based on co-occurrence statistics



Top right or bottom left?

Cherry 

Android 

Laptop 

Banana 

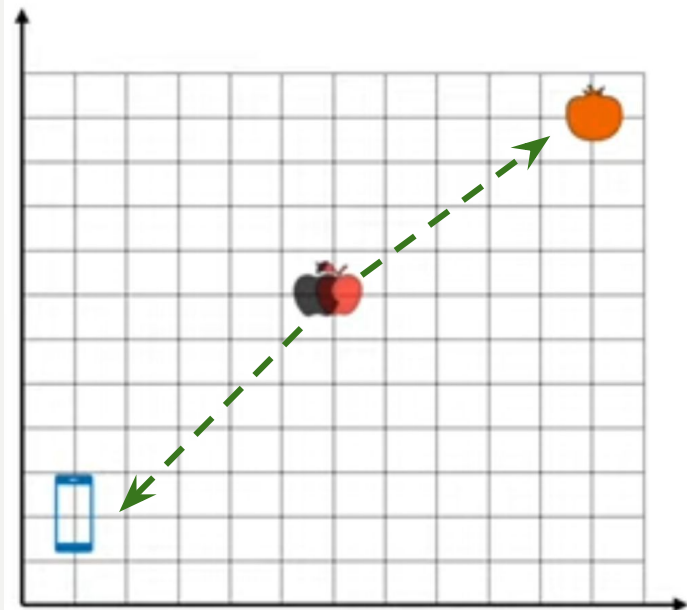
Apple?  

Simplified Example

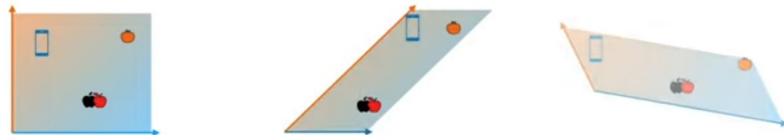
Embeddings are learned and updated from the different contexts

please buy an **apple** and an **orange**

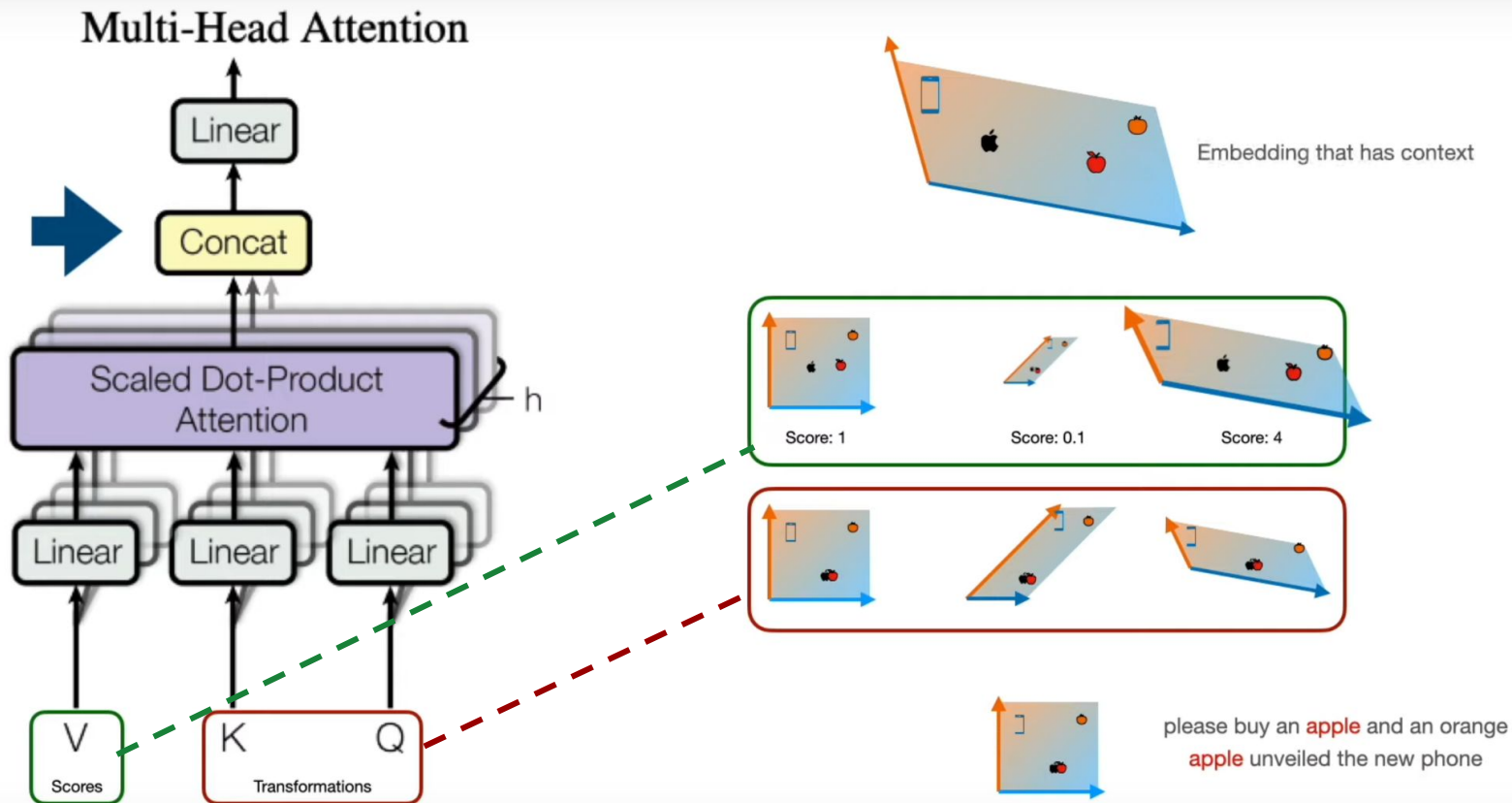
apple unveiled the new **phone**



There are many embeddings and linear transformations of existing embeddings can improve them



Simplified Example



Outline

1. Introduction to AI and LLMs

2. Key Elements of LLMs

- Parameters
- Embeddings
- Attention

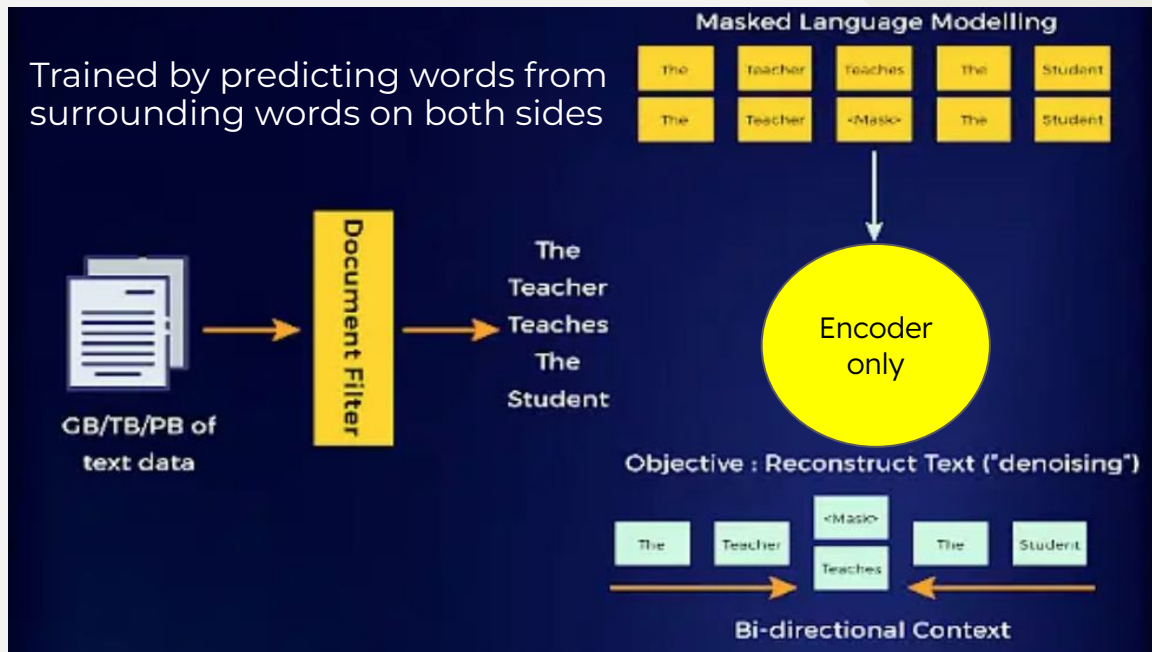
3. LLMs Architecture

- Types of architecture
- How LLM works
- How to set the LLM hyperparameters

Types of LLM Architecture: Encoder Only

Autoencoding Models: BERT family, ROBERTA

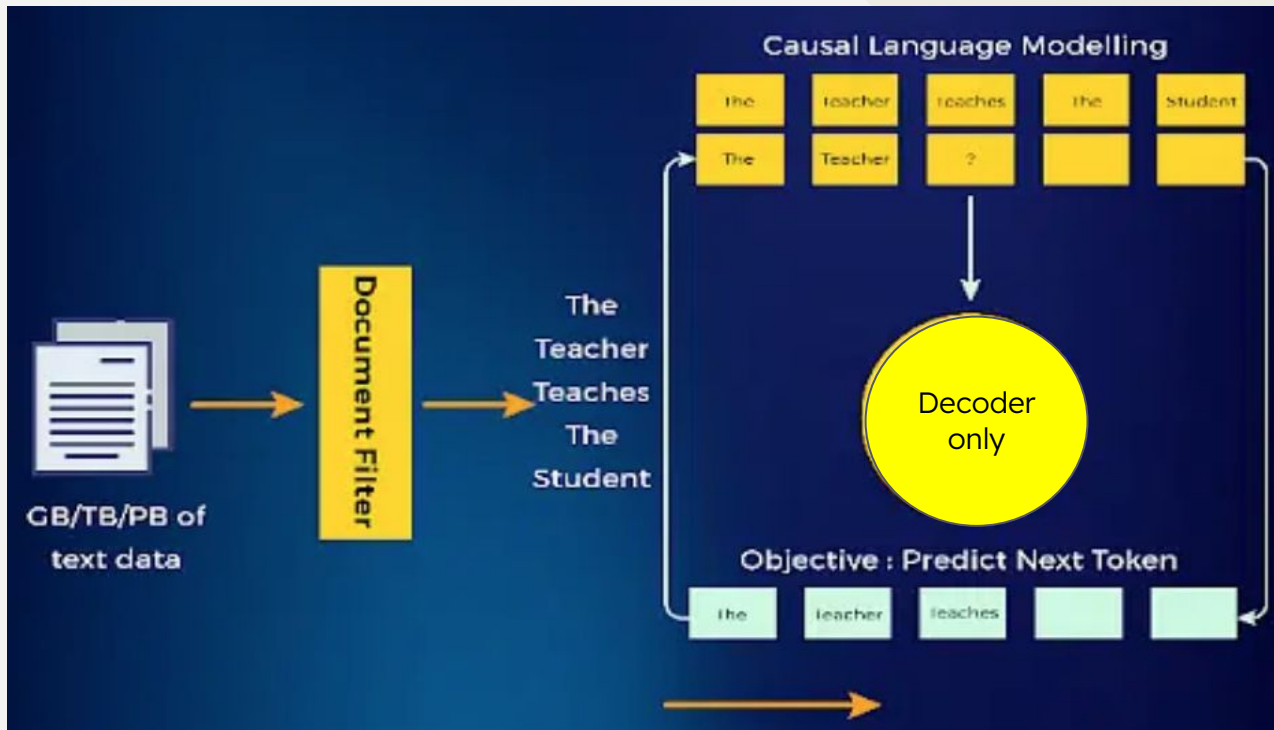
Use Cases: Sentiment Analysis, Named Entity Recognition, Topic Classification



Types of LLM Architecture: Decoder Only

Autoregressive Models: GPT, Claude, Llama, Mistral, Bloom

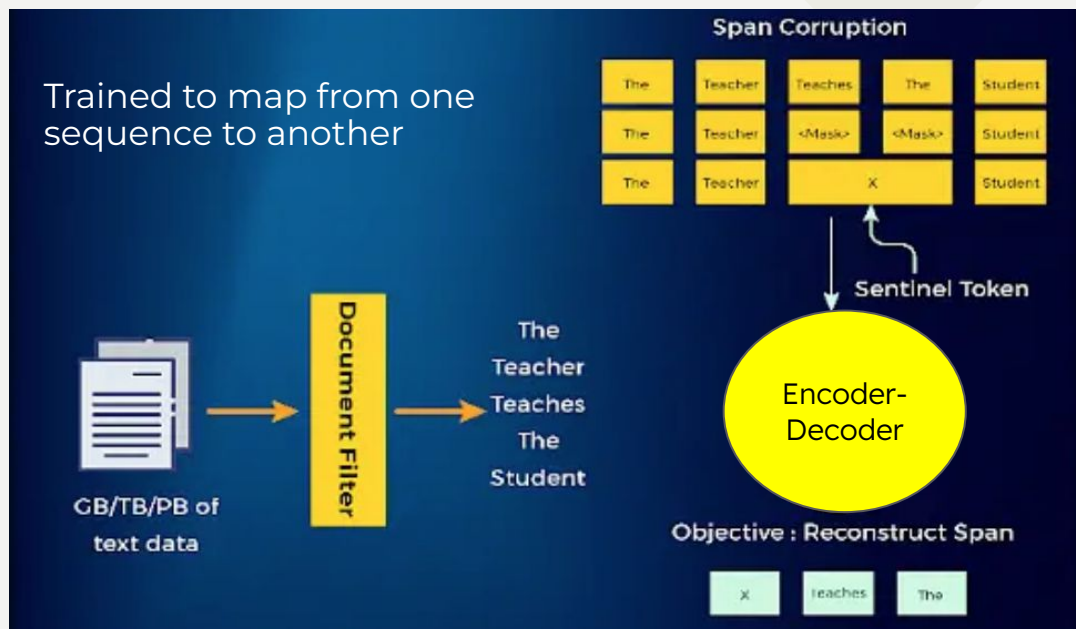
Generate text one token at a time based on the previously generated tokens



Types of LLM Architecture: Encoder-Decoder Models

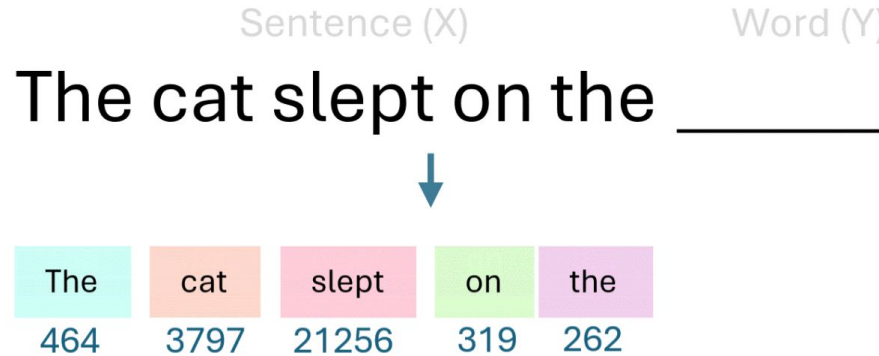
Sequence-to-Sequence Models: BART, Flan-T5, Whisper

Use Cases: Translation, Question-Answering, Text summarization



How LLMs work?

Tokenization



Tokenize, embed, encode input text

@NigelGebodh

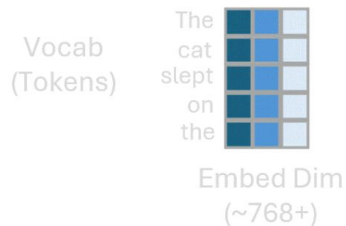
How LLMs work?

Embedding & Encoding

Sentence (X) Word (Y)

The cat slept on the _____

The	cat	slept	on	the
464	3797	21256	319	262
x_1	x_2	x_3	x_4	x_5



How LLMs work?

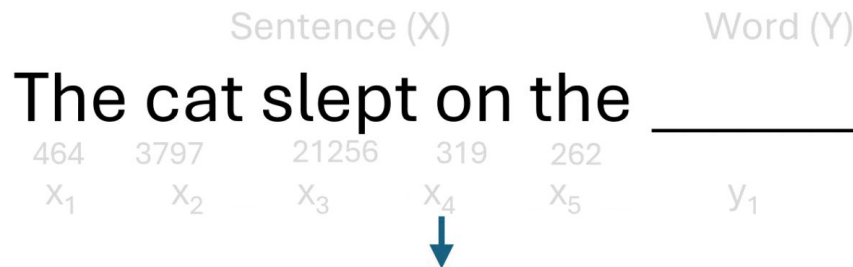
LLM Calculation

Sentence (X) Word (Y)

The cat slept on the _____

464 3797 21256 319 262

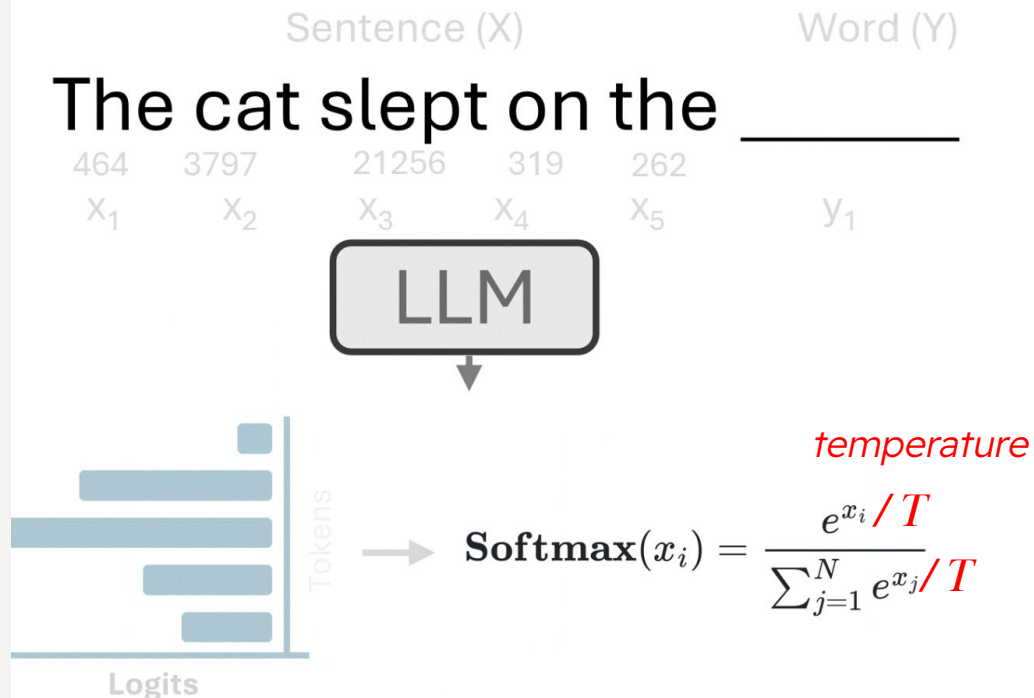
x_1 x_2 x_3 x_4 x_5 y_1



$$p(Y|X) = \prod_{t=1}^n p(y_t | y_{<t}, \mathbf{X})$$

How LLMs work?

LLM Calculation



How LLMs work?

Output

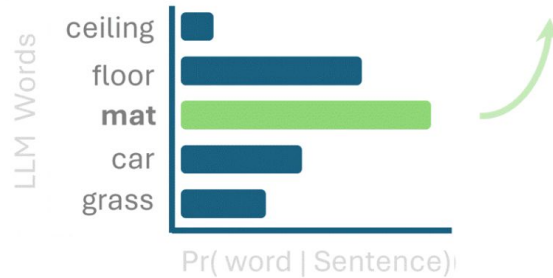
Sentence (X) Word (Y)

The cat slept on the mat

464 3797 21256 319 262 6759

X_1 X_2 X_3 X_4 X_5 y_1

LLM

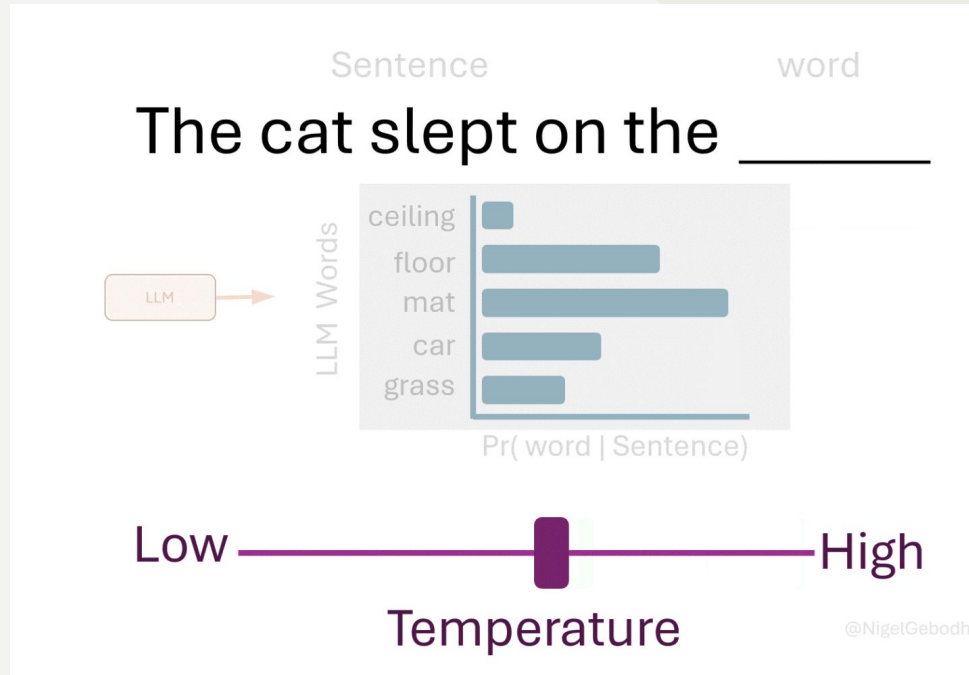


LLM Hyperparameters: Temperature

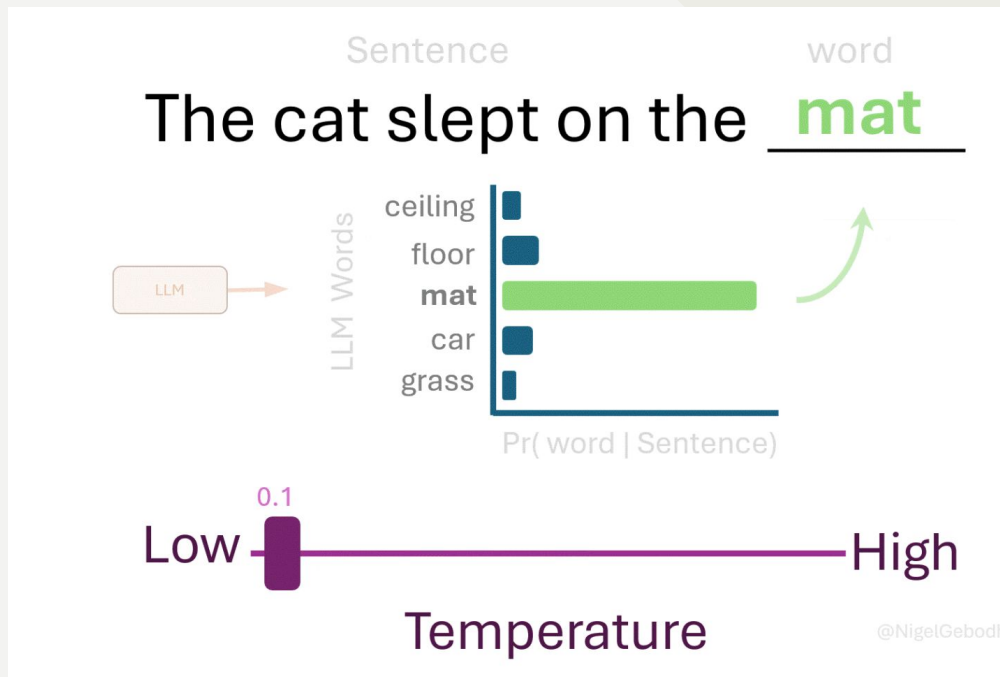
Controls how deterministic or creative the responses are by adjusting the probability distribution of the next word predicted.

- **High Temperature (e.g., 0.8–1.0):** Increases randomness and creativity. Useful for open-ended tasks like storytelling or brainstorming.
- **Low Temperature (e.g., 0.0–0.4):** Produces more deterministic and focused outputs. Suitable for factual or technical tasks.
- **Medium Temperature (e.g., 0.5–0.7):** Balances randomness and determinism, often a good starting point for general purposes.

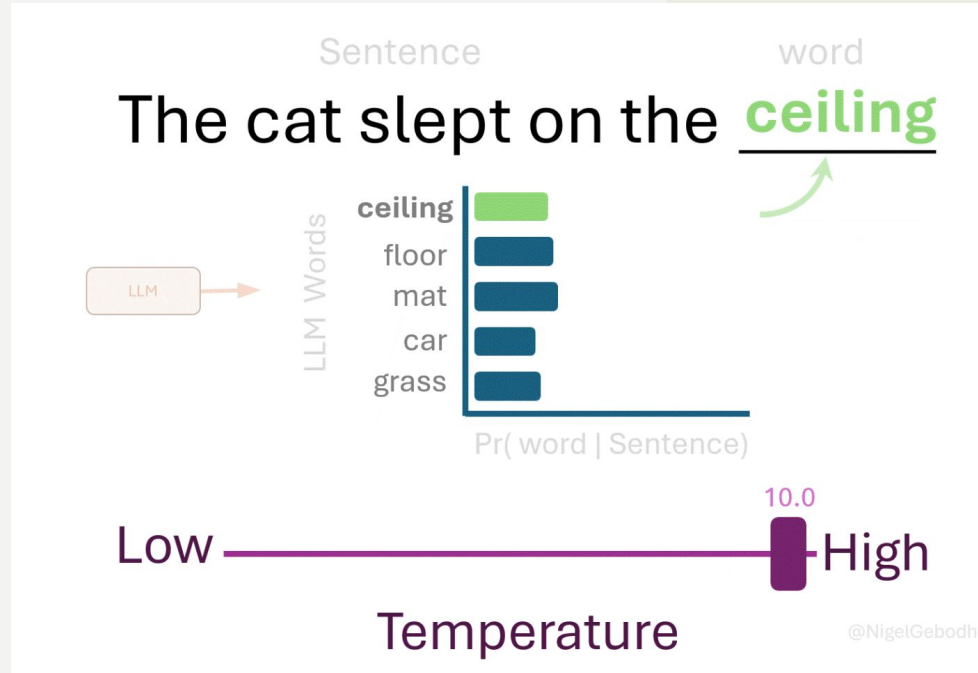
LLM Hyperparameters: Temperature



LLM Hyperparameters: Temperature



LLM Hyperparameters: Temperature



LLM Temperature: Examples (1/3)

Creative Writing (High Temperature)

Prompt: "Once upon a time in a galaxy far, far away,"

Temperature = 1.0: "Once upon a time in a galaxy far, far away, a talking nebula named Zorbo set out to collect cosmic dust to brew the universe's most powerful elixir. But little did Zorbo know, the dust was guarded by ancient star dragons."

Explanation: At a high temperature, the model generates imaginative and unpredictable ideas, ideal for creative storytelling.

LLM Temperature: Examples (2/3)

Technical Explanation (Low Temperature)

Prompt: "Explain Newton's second law of motion."

Temperature = 0.2: "Newton's second law of motion states that the force acting on an object is equal to the mass of the object multiplied by its acceleration. Mathematically, $F = ma$."

Explanation: At a low temperature, the model provides precise and reliable information with minimal deviation from established facts.

LLM Temperature: Examples (3/3)

Conversational Tone (Medium Temperature)

Prompt: "How do I bake a chocolate cake?"

Temperature = 0.6: "To bake a chocolate cake, start by preheating your oven to 350°F. Mix flour, sugar, cocoa powder, and baking powder in a bowl. In another bowl, whisk eggs, milk, and melted butter. Combine the wet and dry ingredients, pour into a greased pan, and bake for 30 minutes. Cool before serving."

Explanation: A medium temperature generates a coherent response that is both informative and conversational.

Best Practices for LLM Temperature

Recommendations

- Factual Answers 0.0–0.3 Deterministic, avoids creative deviations
- Code Generation 0.2–0.4 Predictable and accurate outputs
- Creative Writing 0.7–1.0 Diverse and imaginative results
- Brainstorming 0.8–1.0 Generates unique ideas
- Conversational Agents 0.5–0.7 Balanced and engaging dialogue

Best Practices for Experimentation

- **Start Simple:** Begin with a medium temperature (e.g., 0.7) and adjust based on the desired outcome.
- **Test Iteratively:** Evaluate outputs with different temperatures for the same prompt.
- **Combine with other hyperparameters:** Use temperature alongside parameters like top-k or top-p sampling for finer control over the output.

LLM Hyperparameters: Top-K

Sampling strategy used to generate text from a language model by selecting the next word in a sequence from the top K most probable words, rather than considering all possible words.

- Use **low K values (10–50)** when you want to restrict the model to only the most likely words, useful for focused, deterministic outputs
- Use **higher K values (100–1000)** to allow more diversity while still preventing the selection of highly improbable tokens
- Top-K is particularly effective for maintaining output quality in shorter sequences

Combining Temperature and Top-K

- Pairing these methods can offer fine-grained control over both the diversity and quality of generated text.
- Use a moderate K value (e.g., 50–100) to filter out unlikely tokens, then apply temperature to control randomness within this subset
- This combination is often more stable than using either method alone, especially for longer generation tasks

Guidelines for combined use

1. Start with a moderate K value (e.g., 50) and temperature (e.g., 0.7).
2. If outputs are too random or off-topic, decrease K or lower the temperature.
3. If outputs are too repetitive or predictable, increase K or raise the temperature.
4. Fine-tune based on your specific use case and desired output characteristics.

Other LLM Hyperparameters: Top-p

Top-p, aka nucleus sampling influences the randomness of LLM output. Top-p determines the threshold probability for including tokens in a candidate set used by the LLM to generate output.

Top-p

- Top-p parameter to 0.1 leads to deterministic and focused output
- Increasing Top-p to 0.8 allows for less constrained and more creative responses.

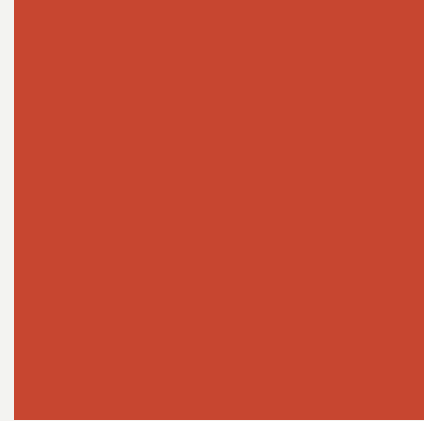
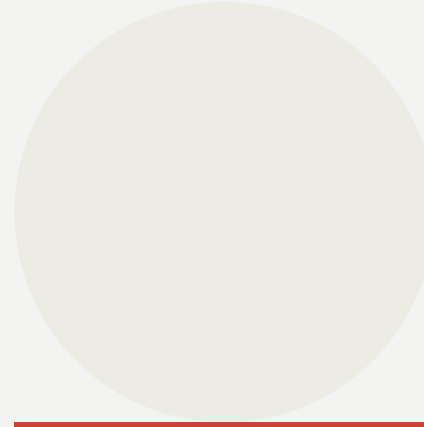
Other LLM Hyperparameters

Max Tokens & Context Window

- Context window and max tokens parameters add constraints on the size of the input accepted or the output generated by the model, directly affecting LLM performance and data processing capabilities.
- The context window, measured in tokens (which can be whole words, subwords, or characters), determines the number of words an LLM can process at once.
- Max tokens parameter sets the upper limit for the total number of tokens, encompassing both the input provided to the LLM as a prompt and the output tokens generated by the LLM in response to that prompt.

Context Window Effect

- **If the input exceeds the context window, the model starts to “forget” earlier information**, potentially resulting in less relevant and lower-quality output.
- **Context window places constraints on prompt engineering** techniques. Methods like Tree-of-Thoughts or Retrieval Augmented Generation (RAG) require large context windows to be effective
- **Context window limits**
 - GPT-4: 8,192 tokens,
 - GPT-4 Turbo: 128K tokens.
 - Claude from 9,000 tokens to 200K tokens in the Claude 2.1 version.



Further Reading

So many courses and resources on LLMs...

<https://gist.github.com/veekaybee/be375ab33085102f9027853128dc5f0e>

<https://github.com/Jason2Brownlee/awesome-llm-books>

<https://github.com/mlabonne/llm-course>

<https://github.com/Hannibal046/Awesome-LLM>

[The Llama 3 Herd of Models](#)

[Owen2.5 Technical Report](#)

[DeepSeek-V3 Technical Report](#)

[Mistral 7B](#)

[Llama 2: Open Foundation and Fine-Tuned Chat Models](#)

Prompting

Outline

1. Definition, Principles & Structure

2. Main Techniques

- Zero-shot Prompting
- Few-shot Prompting
- Chain-of-Thoughts Prompting
- Meta Prompting
- Self-Consistency Prompting
- Contrastive CoT Prompting
- Tree-of-Thoughts Prompting
- Multi-Chain-Reasoning

3. Prompting Frameworks

- CO-STAR
- AUTOMAT

What is Prompt Engineering?

Prompt engineering is the process of structuring or crafting an instruction in order to produce the best possible output from a generative artificial intelligence model. A prompt is natural language text describing the task that an AI should perform.

Prompt Principles

[Bsharat et al.](#) defined 26 ordered prompt principles, which can be organized into five distinct categories:

1. **Prompt Structure and Clarity:** Integrate the intended audience in the prompt.
2. **Specificity and Information:** Implement example-driven prompting (Use few-shot prompting)
3. **User Interaction and Engagement:** Allow the model to ask precise details and requirements until it has enough information to provide the needed response
4. **Content and Language Style:** Instruct the tone and style of response
5. **Complex Tasks and Coding Prompts:** Break down complex tasks into a sequence of simpler steps as prompts

Basic Prompt Structure

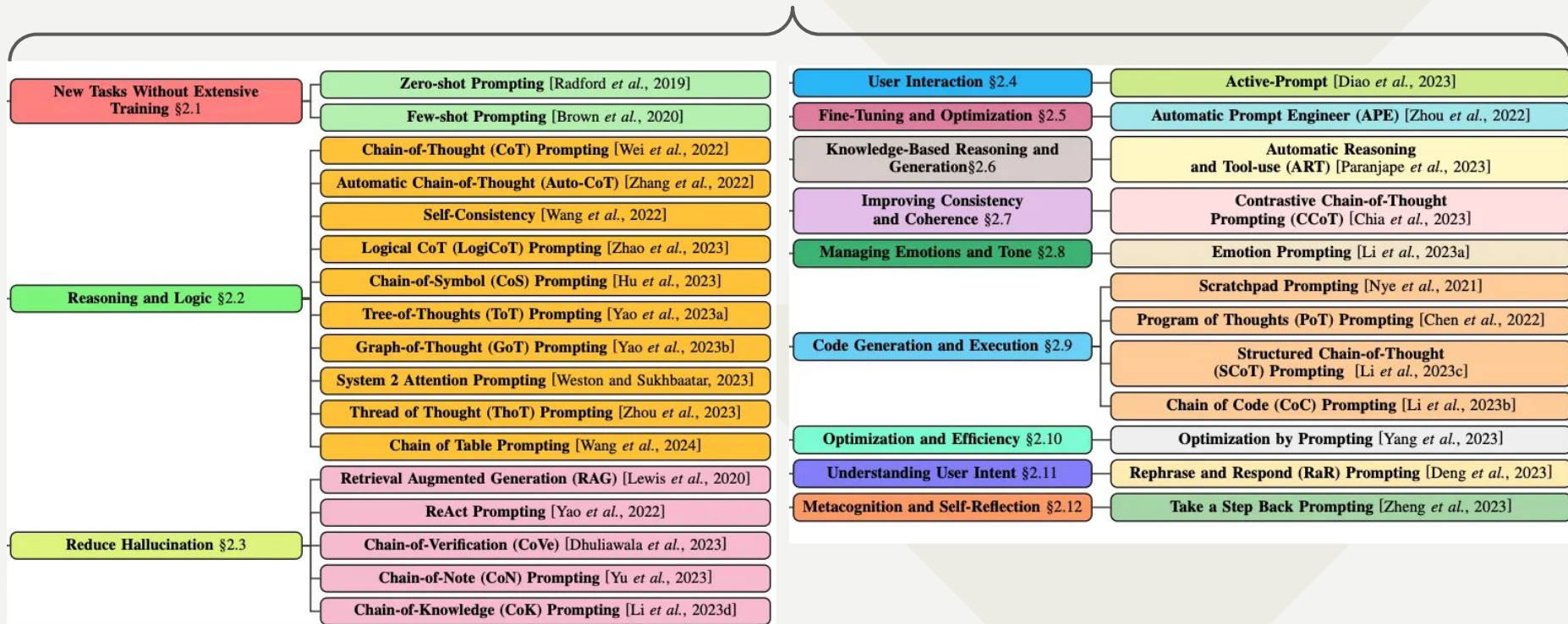
A well-crafted prompt typically consists of:

- **Examples:** Sample inputs/outputs to guide the expected response.
- **Role (Persona):** The perspective or tone the AI should adopt.
- **The Directive:** The main instruction or task.
- **Output Formatting:** Specifications for how the response should be structured.
- **Additional Information:** Background or context that informs the task.

Role-based prompting assigns a specific persona to the AI, such as a doctor or historian, to tailor the response's tone, style, and content. This helps ensure the output matches the intended context or professional setting.

One-shot prompting involves giving the AI a single example before the task. This clarifies the expected output and can improve accuracy for tasks that benefit from a sample demonstration.

Techniques of Prompt Engineering



Sahoo et al. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications <https://arxiv.org/pdf/2402.07927>

Outline

1. Definition & Structure

2. Main Techniques

- Zero-shot Prompting
- Few-shot Prompting
- Chain-of-Thoughts Prompting
- Meta Prompting
- Self-Consistency Prompting
- Contrastive CoT Prompting
- Tree-of-Thoughts Prompting
- Multi-Chain-Reasoning

3. Prompting Frameworks

- CO-STAR
- AUTOMAT

Zero-shot Prompting

Technique that instructs an LLM to perform a task without providing any examples within the prompt. A zero-shot prompt relies on the LLM's ability to understand the task based on the instructions alone, leveraging the vast amount of data it has been trained on.

Example for a sentiment analysis task:

```
> Classify the following text as neutral, negative, or positive.  
> Text: I think the vacation was okay. Sentiment:
```

Tasks: Translation, summarization, or content moderation, where pre-defined examples are not always available or even necessary. Massive training and perhaps fine-tuning, combined with an easy-to-understand zero-shot prompt, enable the LLM to perform these tasks accurately.

Best practice: If zero-shot prompting proves insufficient, switching to few-shot prompting might help.

Few-shot Prompting

Technique where examples are included in the prompt to facilitate learning.

Example of task: using a new word correctly in a sentence

> An example of a sentence using the word flamingo is: We saw many red flamingos on our trip to Florida.

> Write a short story about a flamingo that found itself on a ship bound for California.

Tasks: Text generation, summarization

Best practice: For more complex tasks, few-shot prompting may be insufficient, requiring more advanced techniques like chain-of-thought prompting.

Chain-of-Thoughts (CoT) Prompting

Technique that enhances the reasoning abilities of LLMs by breaking down complex tasks into simpler sub-steps. It instructs LLMs to solve a given problem step-by-step, enabling them to field more intricate questions.

Example of reasoning task

```
> I started out with 8 marbles. I gave 3 to a friend, and then  
found 4 more. How many marbles do I have now? Think step by step .
```

Tasks: Reasoning, explanation

Best practice: CoT prompting includes providing clear logical steps in the prompt as well as a few examples to guide the model. Combining CoT with few-shot prompting can be particularly effective for complex tasks.

Few-shot vs CoT Prompting

Few-shot 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. ✅

Meta-Prompting

Focuses on structuring and guiding LLM responses in a more organized and efficient manner. Unlike few-shot prompting, which relies on detailed examples to steer the model, meta prompting is a more abstract approach that emphasizes the format and logic of queries.

Example of task for solving a math problem

```
> Step 1: Define the variables.  
> Step 2: Apply the relevant formula.  
> Step 3: Simplify and solve.
```

Tasks: Code generation

Best practices: focusing on logical structures, keeping prompts abstract, and ensuring the task's format is clearly defined. The meta prompt engineering technique is especially useful for token efficiency and for tasks where traditional few-shot examples can lead to biases or inconsistencies.

Self-consistency Prompting

Improves the accuracy of CoT reasoning. Instead of relying on a single, potentially flawed flow of logic, self-consistency generates multiple reasoning paths and then selects the most consistent answer from them.

Example of reasoning task

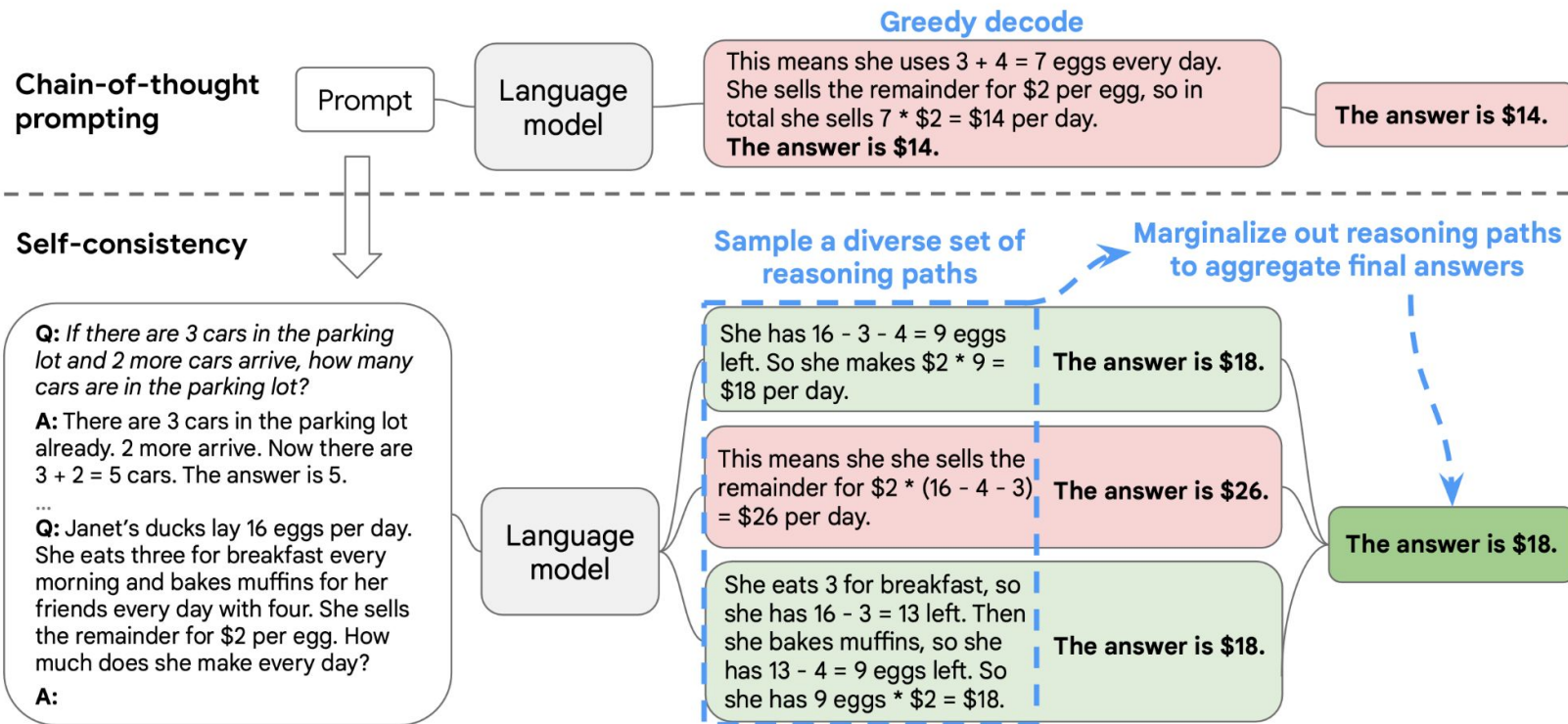
> When I was 6, my sister was half my age.
> Now I'm 70. How old is my sister?

A LLM might answer 35 (half one's age) but, with self-consistency prompting, the model generates additional reasoning paths, such as:

> When you were 6, your sister was 3.
> The difference in your ages is 3 years and that doesn't vary.
> Now that you're 70, she must be 67.

Best practice: Useful for improving model performance on complex reasoning tasks and can be applied to a variety of domains, from arithmetic problems to real-world decision-making

Self-consistency vs CoT Prompting




Contrastive CoT Prompting

CCoT employs negative examples along with positive ones to enhance the reasoning capabilities of language models.


Standard Prompting

Model Input

 Question: James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?



Answer: 624

 Question: James has 30 teeth. His dentist drills 4 of them and caps 7 more teeth than he drills. What percentage of James' teeth does the dentist fix?


Model Output





Answer: 37.5%

Chain-of-Thought (CoT)

Model Input

 Question : James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?

 Explanation: He writes each friend $3 \times 2 = 6$ pages a week So he writes $6 \times 2 = 12$ pages every week. That means he writes $12 \times 52 = 624$ pages a year.

 Question: James has 30 teeth. His dentist drills 4 of them and caps 7 more teeth than he drills. What percentage of James' teeth does the dentist fix?


Model Output





Explanation: The dentist fixes a total of $4 + 7 = 11$ teeth. To find the percentage, we divide the number of teeth fixed by the total number of teeth and multiply by 100: $11/30 \times 100 = 36.67\%$


Contrastive Chain-of-Thought

Model Input

 Question : James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?

 Explanation: He writes each friend $3 \times 2 = 6$ pages a week. So he writes $6 \times 2 = 12$ pages every week. That means he writes $12 \times 52 = 624$ pages a year.

 Wrong Explanation: He writes each friend $12 \times 52 = 624$ pages a week. So he writes $3 \times 2 = 6$ pages every week. That means he writes $6 \times 2 = 12$ pages a year.

 Question: James has 30 teeth. His dentist drills 4 of them and caps 7 more teeth than he drills. What percentage of James' teeth does the dentist fix?

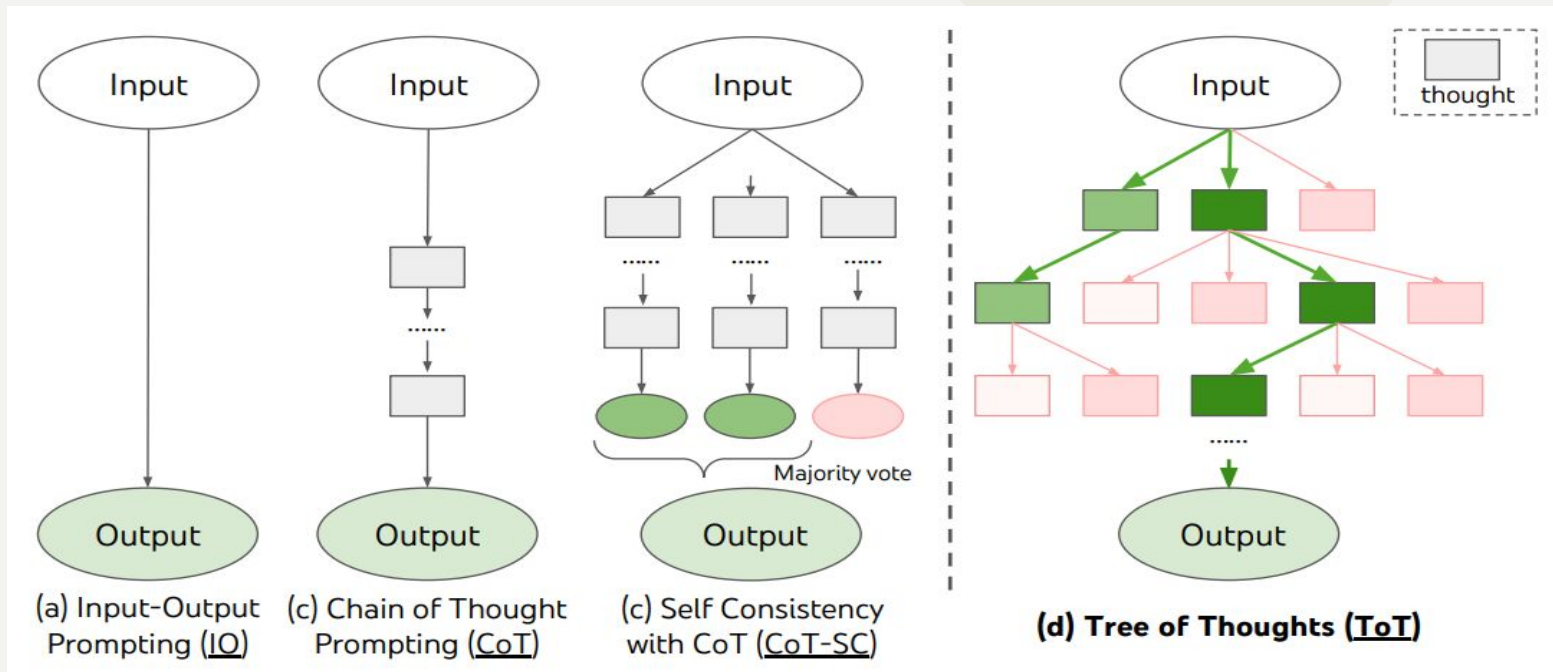
Model Output



Explanation: The dentist drills 4 teeth, so there are $30 - 4 = 26$ teeth left. The dentist caps 7 more teeth than he drills, so he caps $4 + 7 = 11$ teeth. Therefore, the dentist fixes a total of $4 + 11 = 15$ teeth. To find the percentage of teeth the dentist fixes, we divide the number of teeth fixed by the total number of teeth and multiply by 100: $15/30 \times 100 = 50\%$

Tree-of-Thought Prompting (ToT)

ToT enables problem solving with the generation and evaluation of thoughts, then combined with search algorithms (e.g., breadth-first search and depth-first search) to enable systematic exploration of thoughts with lookahead and backtracking



Multi-Chain Reasoning (MCR)

How many ants would fit into The Shard?

1. Decomposition

Iteratively generate the reasoning chain by interleaving retrieved evidence

2. Retrieval

3. Meta-Reason

Generate the final answer based on the content of sampled reasoning chains

Reasoning chain #1

Q1: What is the height of The Shard?

E1: The Shard comprises a 26-floor office complex, occupied by...

E1: The Shard comprises a 26-floor...

E2: Ants are eusocial insects of the...

Q1: What is the height of The Shard?

A1: The height of The Shard is 26-floors, standing 309.6 metres tall.

Q2: What is the volume of an ant?

A2: The volume of an ant is $2e-23$ cubic meters.

So the answer is: $4e+5/(2e-23)=2e+28$.

Reasoning chain #2

Q1: How high is the shard?

E1: 1,016 feet: Standing 309.6 metres (1,016 feet) high, The Shard is ...

E1: Standing 309.6 metres (1,016 feet) high, The Shard ...

E2: Ants range in size from 0.75 ...

Q1: How high is the shard?

A1: The shard is 1016 feet in height.

Q2: How long is the average ant?

A2: The length of an average ant is 0.75mm.

So the answer is: $1e+6*1016/0.75=1.354e+9$

Reasoning chain #3

Q1: How many ants are in the world?

E1: Ant: This study also puts a conservative estimate of the ants at ...

E1: Ant: This study also puts ...

E2: The Shard comprises ...

Q1: How many ants are in the world?

A1: There are around 20 quadrillion ants in the world.

Q2: What is the volume of The Shard?

A2: The volume of The Shard is 90000 cubic meters.

So the answer is: $20e15/90000=2.222222e+12$.

What is the height of The Shard? The height of The Shard is 26-floors, standing 309.6 metres tall.

What is the volume of an ant? The volume of an ant is $2e-23$ cubic meters.

How high is the shard? The shard is 1016 feet in height.

How long is the average ant? The length of an average ant is 0.75mm.

How many ants are there in the world? There are around 20 quadrillion ants in the world.

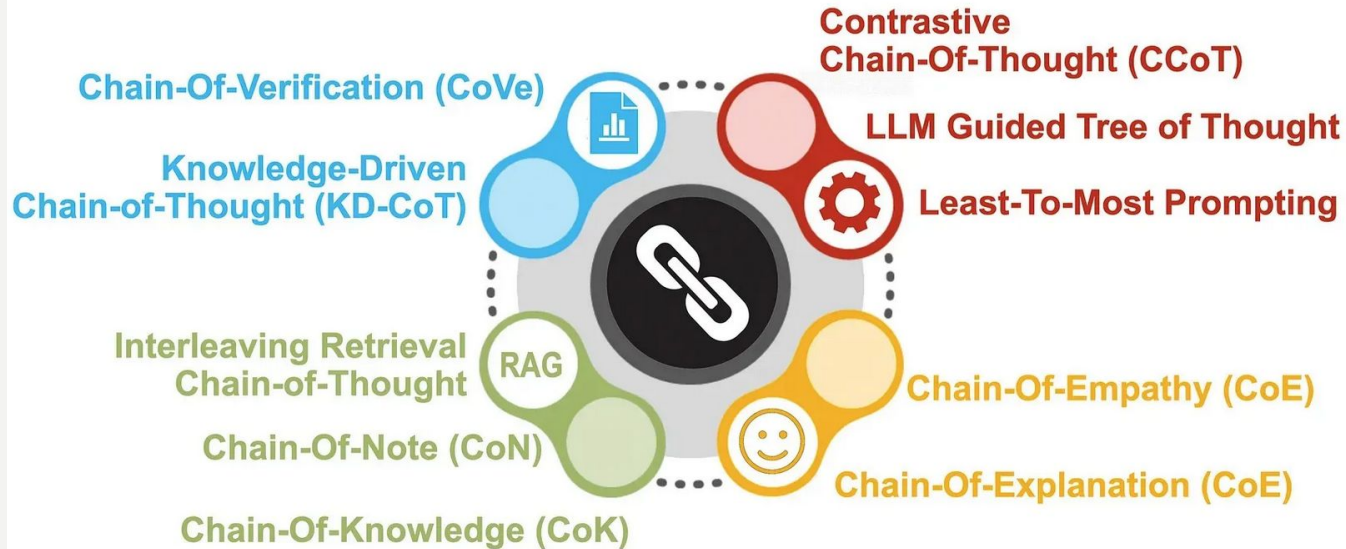
What is the volume of The Shard? The volume of The Shard is 90000 cubic meters

Answer: The volume of The Shard is 90000 cubic meters. The volume of an ant is $2e-23$ cubic meters.

So the answer is: $4.5e+22$.



Other Chain-of-* Prompting



ReAct Prompting

Generates reasoning traces allowing the model to induce, track, and update action plans, and handle exceptions.

(2) AlfWorld

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.
Your task is to: Put some pepper shaker on a drawer.

(2a) Act-Only

Act 1: Go to drawer 1
Obs 1: The drawer 1 is closed.

Act 2: Open drawer 1
Obs 2: You open the drawer 1. The drawer 1 is open. In it, you see a dish sponge 2, and a spoon 1.

Act 3: Go to sinkbasin 1
Obs 3: On the sinkbasin 1, you see a dish sponge 3, a spatula 1, and a spoon 2.

Act 4: Take peppershaker 1 from sinkbasin 1
Obs 4: Nothing happens.

Act 5: Take peppershaker 1 from sinkbasin 1
Obs 5: Nothing happens.



(2b) ReAct (Reason + Act)

Act 1: **Think**[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]
Act 2: Go to cabinet 1
Obs 2: On the cabinet 1, you see a vase 2.

(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

Act 6: Go to countertop 3
Obs 6: On the countertop 3, you see a apple 1, a bread 1, a pepper shaker 1, and a vase 3.

Act 7: Take pepper shaker 1 from countertop 3
Obs 7: You pick up the pepper shaker 1 from the countertop 3.

Act 8: **Think**[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]
Act 9: Go to drawer 1
Obs 9: Drawer 1 is closed.

Act 10: Open drawer 1
Obs 10: You open Drawer 1 ...

Act 11: Put pepper shaker 1 in/on drawer 1
Obs 11: You put pepper shaker 1 in/on the drawer 1.



Outline

1. Definition & Structure

2. Main Techniques

- Zero-shot Prompting
- Few-shot Prompting
- Chain-of-Thoughts Prompting
- Meta Prompting
- Self-Consistency Prompting
- Contrastive CoT Prompting
- Tree-of-Thoughts Prompting
- Multi-Chain-Reasoning

3. Prompting Frameworks

- CO-STAR
- AUTOMAT

CO-STAR Framework

- **Context:** Set the scene! Provide background details for the LLM to understand the situation.
- **Objective:** What do you want to achieve? Clearly define the task for focused results.
- **Style & Tone:** Dress it up! Specify the desired writing style and emotional tone for your LLM's response.
- **Audience:** Know your reader. Identify who you're targeting to tailor the LLM's output.
- **Response:** Pick your format. Define the output format (text, code, etc.) for the LLM's response.

<https://towardsdatascience.com/how-i-won-singapores-gpt-4-prompt-engineering-competition-34c195a93d41/#10b2>

Example of CO-STAR

In [8]:

```
user_prompt = """
# CONTEXT #
I want to share our company's new product feature for
serving open source large language models at the lowest cost and lowest
latency. The product feature is Anyscale Endpoints, which serves all Llama series
models and the Mistral series too.

#####

# OBJECTIVE #
Create a LinkedIn post for me, which aims at Gen AI application developers
to click the blog link at the end of the post that explains the features,
a handful of how-to-start guides and tutorials, and how to register to use it,
at no cost.

#####

# STYLE #

Follow the simple writing style common in communications aimed at developers
such as one practised and advocated by Stripe.

Be persuasive yet maintain a neutral tone. Avoid sounding too much like sales or marketing
pitch.

#####

# AUDIENCE #
Tailor the post toward developers seeking to look at an alternative
to closed and expensive LLM models for inference, where transparency,
security, control, and cost are all imperatives for their use cases.

#####

# RESPONSE #
Be concise and succinct in your response yet impactful. Where appropriate, use
appropriate emojis.
"""
```

AUTOMAT Framework

1. Act as a Particular persona (Who is the bot impersonating?)
2. User Persona & Audience (Who is the bot talking to?)
3. Targeted Action (What do you want the bot to do?)
4. Output Definition (How should the bot's response be structured?)
5. Mode / Tonality / Style (How should its response be communicated?)
6. Atypical Cases (Are there edge cases where the bot should react differently?)
7. Topic Whitelisting (What relevant topics can the bot talk about?)

<https://medium.com/the-generator/the-perfect-prompt-prompt-engineering-cheat-sheet-d0b9c62a2bba>

AUTOMAT Framework

A

Act as a ..., Bot Persona

Define the **bot persona** of the AI assistant just in a few words.

✓ Be very specific in your description.

Act as a sensitive elderly psychotherapist ...
Act as a patient support staff ...
Act as a professional journalist ...
Act as a pebble, a car in love with its driver ...
Act as a 4th grader math tutor ...
Act as a csh-terminal on the mac ...

✗ Don't describe a behaviour that the AI exhibits anyway.

Act as a helpful AI ...

U

User Persona, Audience

Describe the audience, their background, the expected level of knowledge of the **recipients** in a few words

✓ Describe the audience.

Explain it like to someone with an MSc in software engineering ...
... like to a 5-year-old child
... to the owner of the Tesla model S ...

✗ Don't be **unspecific** about the audience.

... tell me ...
... to the user ...

T

Targeted action

Use a **meaningful verb and objects** describing the transformation from input to output or the way the model should produce or create the output

✓ Describe the task.

... summarize ...
... list ...
... translate ...
... classify ...
... explain ...
... extract ...
... format ...
... comment ...
... document the code ...

✗ Avoid using verbs like "answer".

... answer the question ...
... write a ...
... give me ...

AUTOMAT Framework

0

Output Definition

The output can be described in a separate section in great detail, see below.

✓ Describe the output.

... a list of steps ...
... a formula ...
... a table ...
... python code ...
... a JSON ...
... a floating-point number between 0.0 and 1.0 ...
... a recipe with a list of ingredients for 4 persons ...
... a list of two-letter ISO country codes ...
... a iambic pentameter ...

✗ Don't be too general.

... an answer ...
... a text ...
... a few ...

M

Mode/ Tonality/ Style

Define **the way the model should convey the message**.

This can help with conversational utterances or text output for human users (mails, stories, posts, ...)

✓ Describe the mode/ tone/ style.

... empathetic ...
... confident ...
... aggressive ...
... moaning ...
... sarcastic ...
... witty ...
... stuttering ...
... Hemingway style ...
... like in a legal text ...

✗ Don't describe a behaviour that the AI tries to **exhibit anyway** (without specific prompting).

... friendly ...
... neutral ...
... smart ...
... intelligent ...

A

Atypical cases

This mainly makes sense for a model integrated into an application or for a prompt which is used for several requests.

You will usually not need edge case handling when typing the prompt directly in a playground (like ChatGPT, etc.)

✓ Describe atypical, edge cases.

... and list these movies in a table with the columns "title", "director", "release date". If "director" or "release date" is missing, put a "-" to the cell. If the title is not known, don't put the movie into the table.

... if the answer on the question is not in the provided context, tell the user, you can't answer the question on basis of your material ...

✗ Don't forget to say what should be done if an assumption is not correct.

... answer only on the basis of your knowledge ...
> and if you don't know, what then?

... translate the English input text to French ...
> and if someone gives French input? Leave it in French or translate it to English?

AUTOMAT Framework

T

Topic whitelisting

When building a **conversational** system, you may not want the model to talk about anything and everything, because it may hallucinate, touch critical topics, etc.

✓ List permitted conversation topics.

... answer only questions regarding the CRB2004, it's features and operations. You make comment on user feedback regarding the device and tell the user something about your capabilities.

✗ Don't tell the model what not to talk about.
The list will **typically not be exhaustive**.

... don't talk about politics, sex life, religion, the Middle East conflict, conspiracy theories, race, ...
> but talking about how to best commit suicide, hack into government servers, making your girlfriend submissive is fine?



Further Reading

- <https://learnprompting.org/docs/introduction>
- <https://medium.com/the-generator/the-perfect-prompt-prompt-engineering-cheat-sheet-d0b9c62a2bba>
- [OpenAI Cookbook](#) has many in-depth examples for how to utilize LLM efficiently.
- [Prompt Engineering Guide](#) repo contains a pretty comprehensive collection of education materials on prompt engineering.
- <https://github.com/dmatrix/genai-cookbook/blob/main/llm-prompts/README.md>
- learnprompting.org
- [PromptPerfect](#)
- [Semantic Kernel](#)

Introduction to RAG

Outline

1. Definition & Evolution

2. Main Components of a RAG Pipeline

- Ingestion
- Retrieval

3. Comparisons

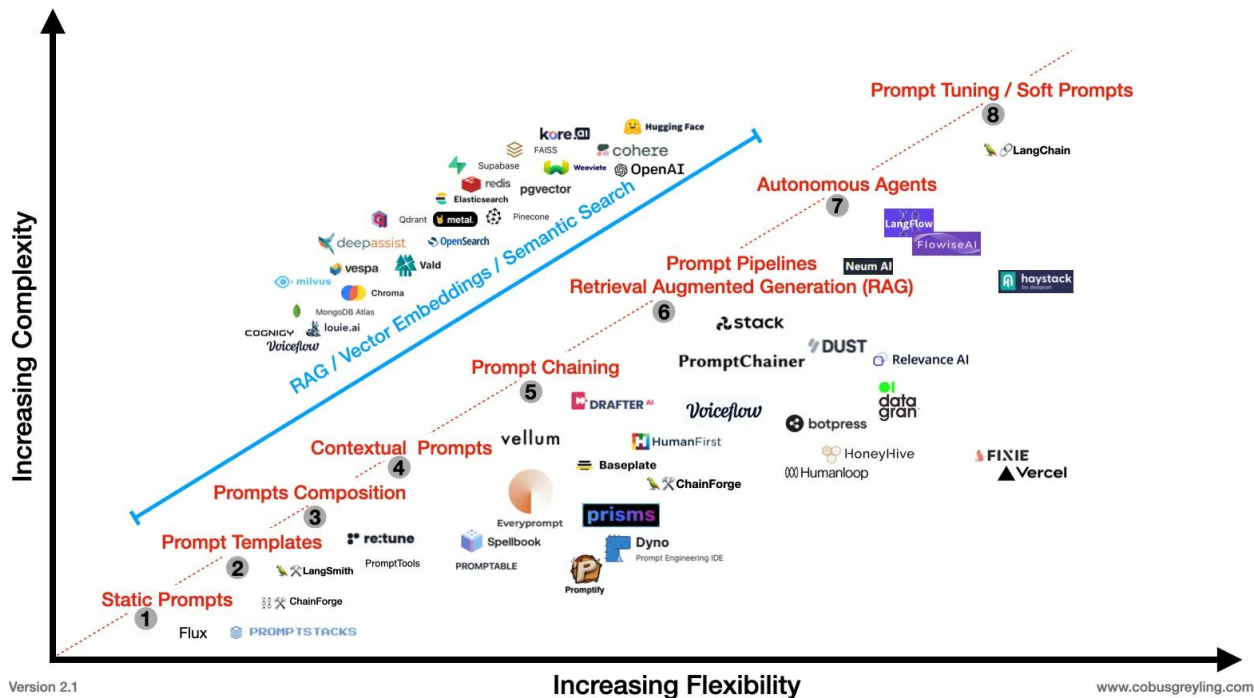
- RAG vs LLM
- RAG vs Fine-tuning

What is RAG?

Retrieved Augmented Generation is a technique that allows generative AI models to search and incorporate new additional external data and knowledge that are relevant to create more accurate domain-specific content without needing further training.

RAG & Prompting Evolution

Emerging RAG & Prompt Engineering Architectures for LLMs



When to Choose RAG

Use Prompt Engineering:

- You need a quick, low-resource solution.
- The task doesn't require access to external or updated information.

Use Fine-Tuning:

- You have a large, domain-specific dataset.
- High accuracy is critical, and you have the computational resources.

Use RAG:

- Your application requires up-to-date or specialized knowledge.
- You want to minimize hallucinations by grounding responses in external data.
- You need a balance between performance and resource efficiency.

Best Practices for Implementing RAG

- **Maintain a High-Quality Knowledge Base**

Ensure that the documents used for retrieval are accurate, relevant, and up-to-date.

- **Optimize Retrieval Mechanisms**

Use effective embedding models and similarity measures to improve the relevance of retrieved documents.

- **Monitor and Evaluate Performance**

Regularly assess the system's responses for accuracy and relevance, and adjust components as needed.

- **Ensure Data Privacy and Security**

Implement appropriate measures to protect sensitive information in your knowledge base.

Outline

1. Definition & Evolution

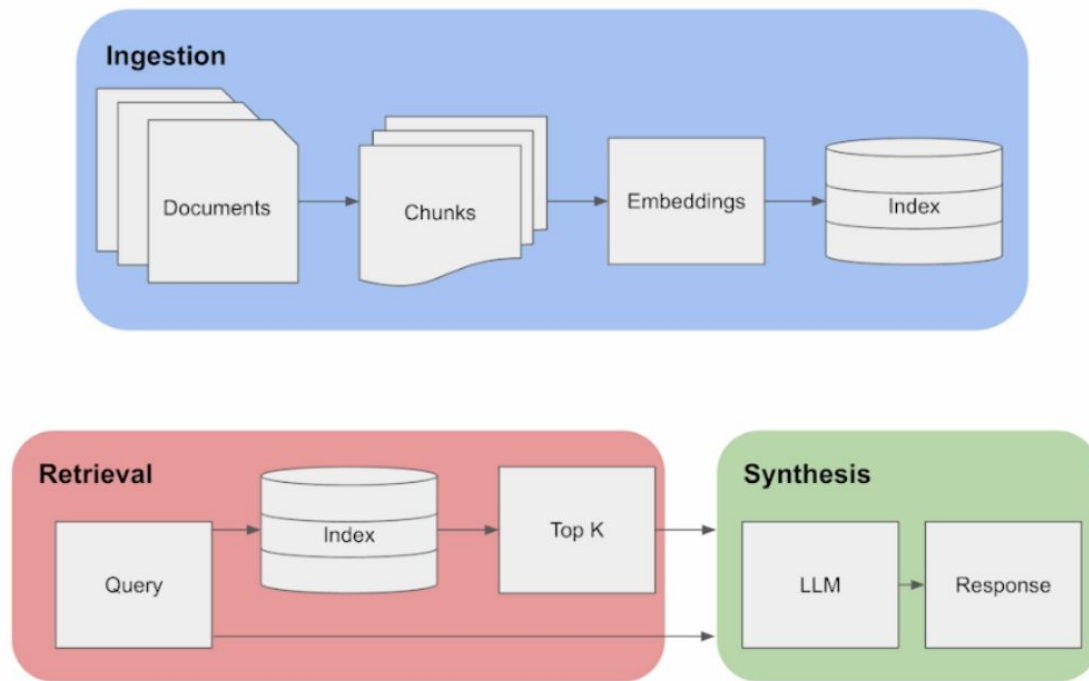
2. Main Components of a RAG Pipeline

- Ingestion
- Retrieval

3. Comparisons

- RAG vs LLM
- RAG vs Fine-tuning

Basic RAG pipeline



Main Chunking Strategies (1/2)

Fixed-size chunking

- The most common and straightforward approach
- decide the number of tokens in our chunk and, optionally, whether there should be any overlap between them.
- overlap between chunks to make sure that the semantic context doesn't get lost between chunks.
- computationally cheap and simple to use since it doesn't require the use of any NLP libraries.

Recursive Chunking

- It divides the input text into smaller chunks in a hierarchical and iterative manner using a set of separators.
- If the initial attempt at splitting the text doesn't produce chunks of the desired size or structure, the method recursively calls itself on the resulting chunks with a different separator or criterion until the desired chunk size or structure is achieved.

Specialized chunking

Main Chunking Strategies (2/2)

Semantic Chunking

1. **Split** the documents into sentences based on separators(.,?!,)
2. **Index** each sentence based on position.
3. **Group**: Choose how many sentences to be on either side. Add a buffer of sentences on either side of our selected sentence.
4. **Calculate** distance between group of sentences.
5. **Merge** groups based on similarity i.e. keep similar sentences together.
6. **Split** the sentences that are not similar.

The water cycle is a continuous process by which water moves through the Earth and atmosphere. It involves processes such as evaporation, condensation, precipitation, and collection. Evaporation occurs when the sun heats up water in rivers, lakes, or oceans, turning it into vapor or steam. This vapor rises into the air and cools down, forming clouds. Eventually, the clouds become heavy and water falls back to the earth as precipitation, which can be rain, snow, sleet, or hail. This water then collects in bodies of water, continuing the cycle.

- 1 Split text into sentences or paragraphs
- 2 Vectorize windows of sentences
- 3 Calculate cosine distance between all pairs
- 4 Merge until breakpoint is reached

Regular vs Semantic Chunking

Regular Chunking

Chunk 1: Abstract, Intro, part of Methods

Chunk 2: Rest of Methods, part of Results

Chunk 3: Rest of Results, part of Discussion

Chunk 4: Rest of Discussion, Conclusion, References

Semantic Chunking

Chunk 1: Abstract and Introduction

Chunk 2: Methods

Chunk 3: Results

Chunk 4: Discussion and Conclusion

Chunk 5: References

Query Example

"What were the methods used to measure blood pressure in studies that found a significant reduction in hypertension?"

Semantic Search Results

Regular Chunking

- Retrieves parts of Chunks 1 and 2
- Combines relevant info from multiple chunks
- May include some irrelevant information
- Requires more complex combination of info

Semantic Chunking

- Retrieves Chunk 2 (entire Methods section)
- All relevant information in one coherent chunk
- Minimal irrelevant information included
- Preserves full context of Methods

Potential Advantages of Semantic Chunking

1. Better coherence and context preservation
2. Reduced noise and irrelevant information
3. Potentially more efficient retrieval (fewer chunks needed)
4. Improved handling of long-range dependencies
5. Possible better ranking of most relevant information
6. Easier for model to understand complete ideas

Chunk Size Matters

Choosing the right **chunk_size** can influence the efficiency and accuracy of a RAG system in several ways:

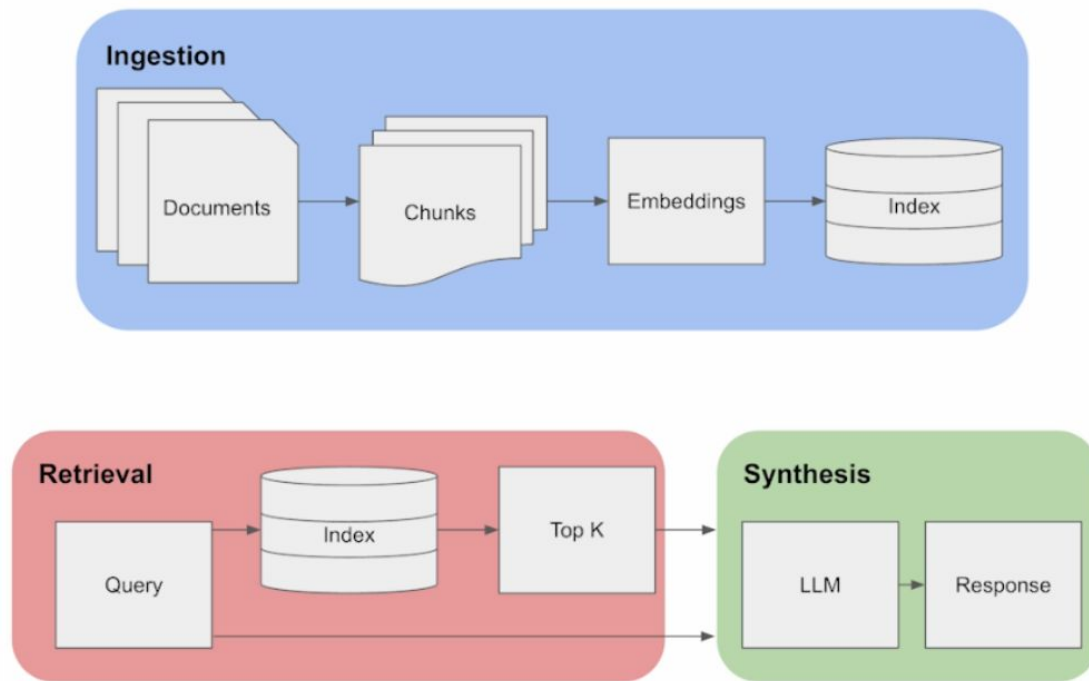
1. Relevance and Granularity:

- A small **chunk_size**, like 128, yields more granular chunks. This granularity, however, presents a risk: vital information might not be among the top retrieved chunks, especially if the **similarity_top_k** setting is as restrictive as 2.
- Conversely, a chunk size of 512 is likely to encompass all necessary information within the top chunks, ensuring that answers to queries are readily available. To navigate this, we employ the **Faithfulness** and **Relevancy** metrics. These measure the **absence of 'hallucinations'** and the 'relevancy' of responses based on the query and the retrieved contexts respectively.

2. Response Generation Time

- As the **chunk_size** increases, so does the volume of information directed into the LLM to generate an answer. This might also slow down the system. Ensuring that the added depth doesn't compromise the system's responsiveness is crucial.

Basic RAG pipeline



Lexical Retrieval

Traditional approach to information retrieval based on exact word matches and term frequency.

Main Approaches

- **TF-IDF (Term Frequency-Inverse Document Frequency)** evaluates the importance of a word in a document relative to a corpus. It increases proportionally with the number of times a word appears in the document but is offset by how frequently the word appears across all documents.
- **BM25 (Best Matching 25)**. A more refined version of TF-IDF. It introduces term frequency saturation and document length normalization, improving relevance scoring.

Advantages of Lexical Retrieval

- Fast and computationally efficient.
- Easy to interpret and implement.
- Works well when exact keyword matching is important.

Limitations

- Cannot handle synonyms or paraphrased queries effectively.
- Limited ability to capture semantic meaning.

Semantic Retrieval

Use neural networks to embed both queries and documents into a shared vector space, where semantic similarity can be calculated using metrics like cosine similarity.

- **How it works**

- **Vector Encoding:** Both queries and documents are transformed into dense vectors using pre-trained or fine-tuned encoders. These encoders are typically trained on large datasets, enabling them to capture semantic nuances beyond surface-level keyword overlap.
- **Semantic Matching:** Vectors are compared to identify the most semantically relevant documents, even if they don't share explicit terms with the query.

- **Advantages of Semantic Retrieval**

- Handles paraphrasing, synonyms, and conceptual similarity effectively.
- Supports more natural and conversational queries.
- Multilingual capabilities are often built-in.

- **Challenges and Considerations**

- Requires significant computational resources.
- Retrieval quality is sensitive to training data and may reflect biases.
- Updating document embeddings for dynamic content can be complex.

Hybrid Retrieval: Lexical + Semantic

Reciprocal Rank Fusion (RRF) merges the rankings from different retrieval models (e.g., BM25 and a neural retriever) by assigning higher scores to documents that consistently rank well across systems.

How RRF works: Each document receives a score based on its position in the ranked lists from multiple retrieval methods.

$$\text{RRF Score}(d) = \sum_{i=1}^n \frac{1}{k + \text{rank}_i(d)}$$

where:

- d is the document,
- $\text{rank}_i(d)$ is the rank position of document d in the i^{th} ranked list,
- k is a constant (typically set to 60),
- n is the number of retrieval systems.

Benefits of Hybrid Retrieval:

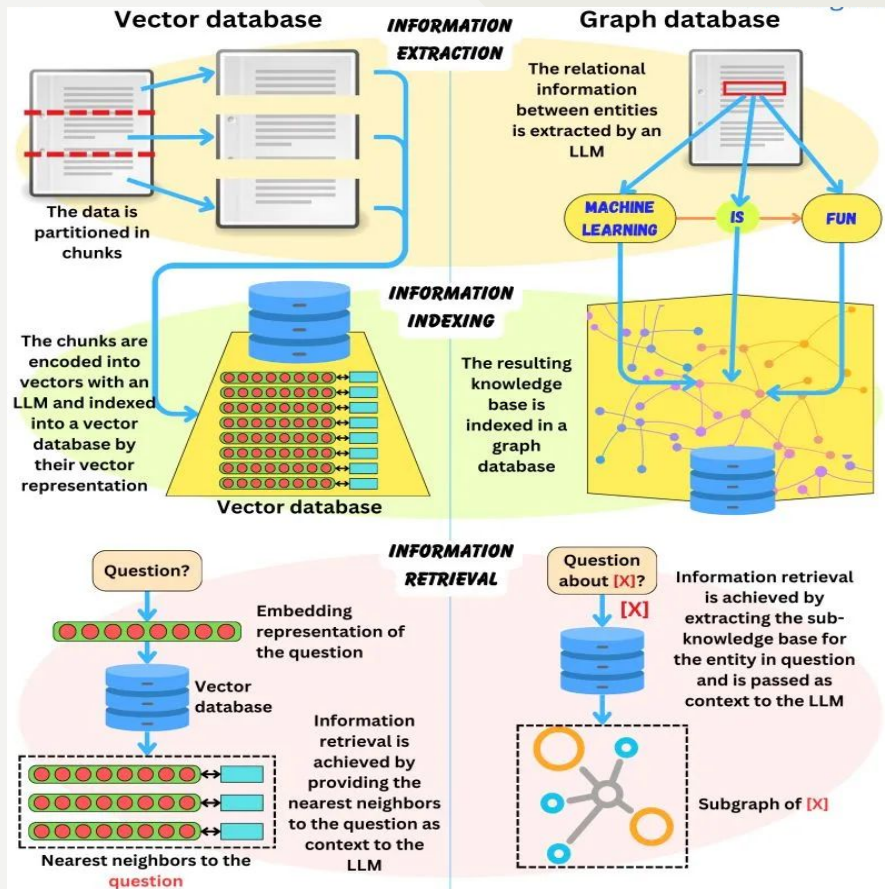
- Increases recall by retrieving relevant documents that either lexical or semantic methods might miss individually.
- Balances precision and coverage.
- Makes the retrieval system more resilient to query variations and noise.

RAG Retriever^r

Retriever here could be any of the following depending on the need for semantic retrieval or not:

- **Vector database:** Typically, queries are embedded using models like BERT for generating dense vector embeddings. Alternatively, traditional methods like TF-IDF can be used for sparse embeddings. The search is then conducted based on term frequency or semantic similarity.
- **Graph database:** Constructs a knowledge base from extracted entity relationships within the text. This approach is precise but may require exact query matching, which could be restrictive in some applications.
- **Regular SQL database:** Offers structured data storage and retrieval but might lack the semantic flexibility of vector databases.

Vector DB vs Graph DB



https://www.linkedin.com/posts/damienbenveniste_machinelearning-datascience-artificialintelligence-activity-7119708674868051969-5HA1/?utm_source=share&utm_medium=member_desktop

Choice of Vector Database

DB Attributes	Open-source & free to self-host	Managed Cloud Offering	Disk-based Index	Multi-tenancy Support	In-built Text Embeddings creation (Bring-your-own-model)	In-built Image Embedding creation	Metadata Filtering	Embeddable	Multiple vectors per point	Langchain integration	Llama index integration	Hybrid Search	BM25 support	Sparse Vectors	Full-text
1 Pinecone	✗	✓		✓ via	✗		✓	✗	✗	✓	✓	✗	✗	✓	✗
2 Qdrant	✓	✓		✓ via	✓ via FastEm	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗
3 Weaviate	✓	✓		✓	✓	✓	✗	✓	✓	✓	✓ RRF	✓	✗	✗	✗
4 pgvector	✓	✓ (supabas	✓	✗		✓	✗	✓	✓	✓	✓ https	✓	✗	✓ htt	
5 Vespa	✓	✓	✓ https://doc	✓	✗	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓
6 Milvus	✓	✓	✓	✓ http	✗		✓	✗		✓	✓		✗	✗	✗
7 MongoDB Atlas	✗	✓		✓ via	✗		✓	✗	✓	✓	✓ https	✓	✗	✓	
8 Marqo	✓	✓			✓	✓ st	✗	✓	✓	✗	✓ via w	✓	✗	✗	✗
9 Vectara	✗	✓		✓ via	✗ [Note Vecta	✓	✗		✓	✓	✓ Only	✗	✗	✗	✗
10 Elasticsearch	✗	✓	✓	✓	✓		✗	✓	✓	✓	✓	✓	✓	✓	✓
11 OpenSearch	✓	✓	✓	✓	✓	✓ ht	✗	✓	✓	✓	✓ Only	✓	✗	✓	
12 Chroma	✓	✗	✗	✗	✓		✓	✓	✗	✓	✗	✗	✗	✗	✗

Re-Ranking

Re-ranking in RAG refers to the process of evaluating and sorting the retrieved documents or info snippets based on their relevance to the given query or task.

- **Lexical Re-Ranking:** Based on lexical similarity between the query and the retrieved documents. Common Methods: BM25 or cosine similarity with TF-IDF
- **Semantic Re-Ranking:** Uses semantic understanding to judge the relevance of documents. It often involves neural models like BERT or other transformer-based models to understand the context and meaning beyond mere word overlap.
- **Learning-to-Rank (LTR) Methods:** Involve training a model specifically for the task of ranking documents (point-wise, pair-wise, and list-wise) based on features extracted from both the query and the documents. This can include a mix of lexical, semantic, and other features.
- **Hybrid Methods:** These combine lexical and semantic approaches, possibly with other signals like user feedback or domain-specific features, to improve re-ranking.

Outline

1. Definition & Evolution

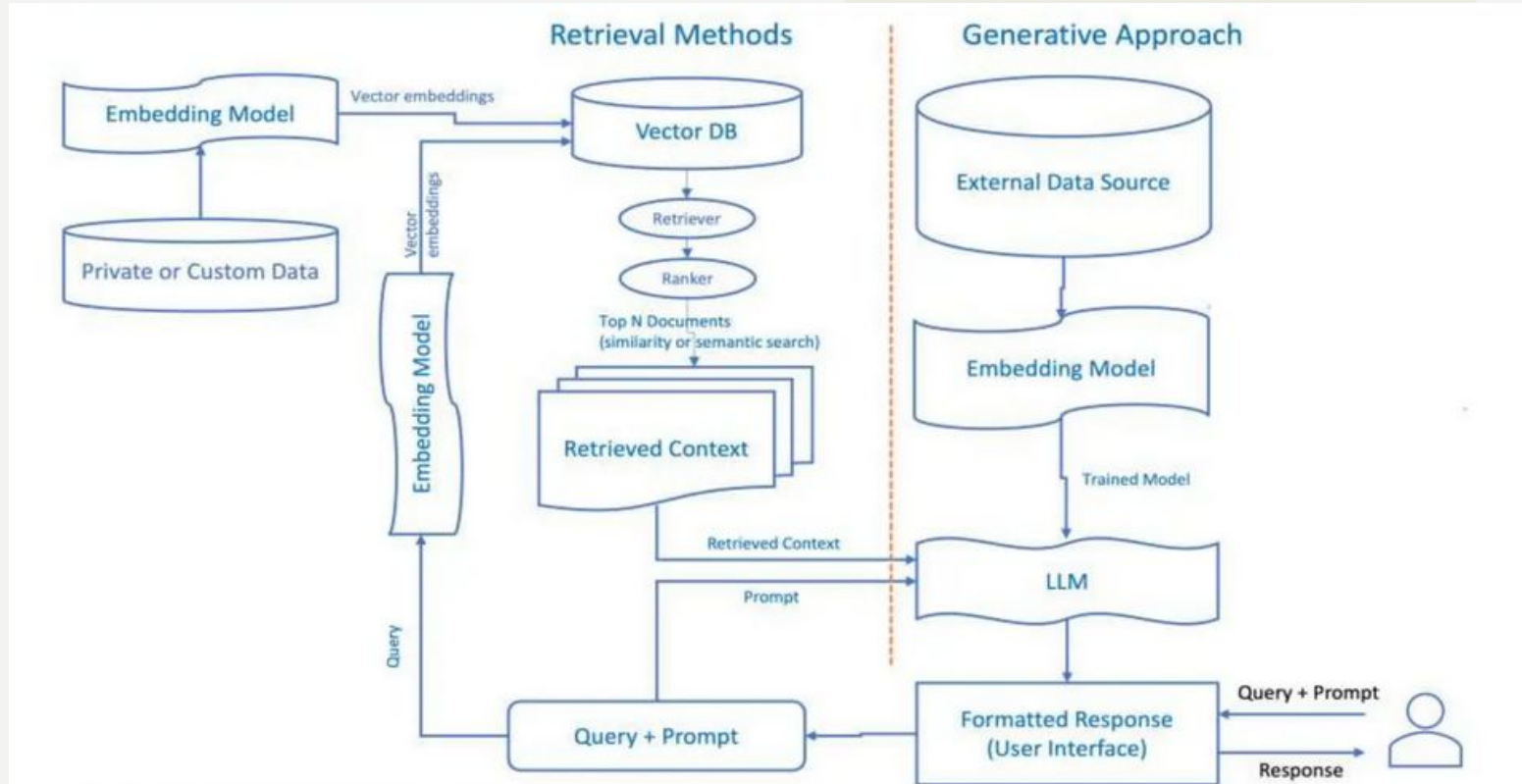
2. Main Components of a RAG Pipeline

- Ingestion
- Retrieval

3. Comparisons

- RAG vs LLM
- RAG vs Fine-tuning

RAG vs LLM

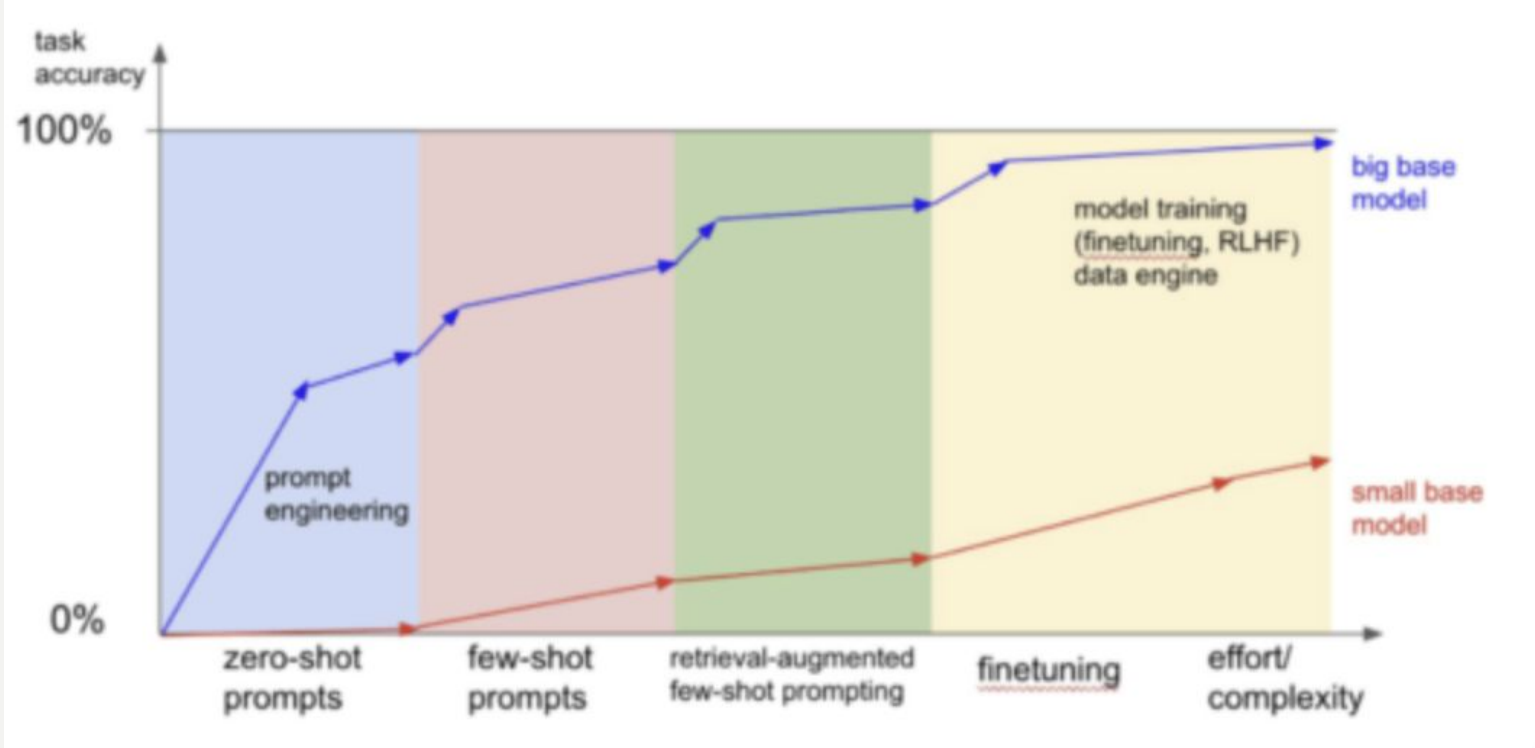


RAG vs Fine-tuning

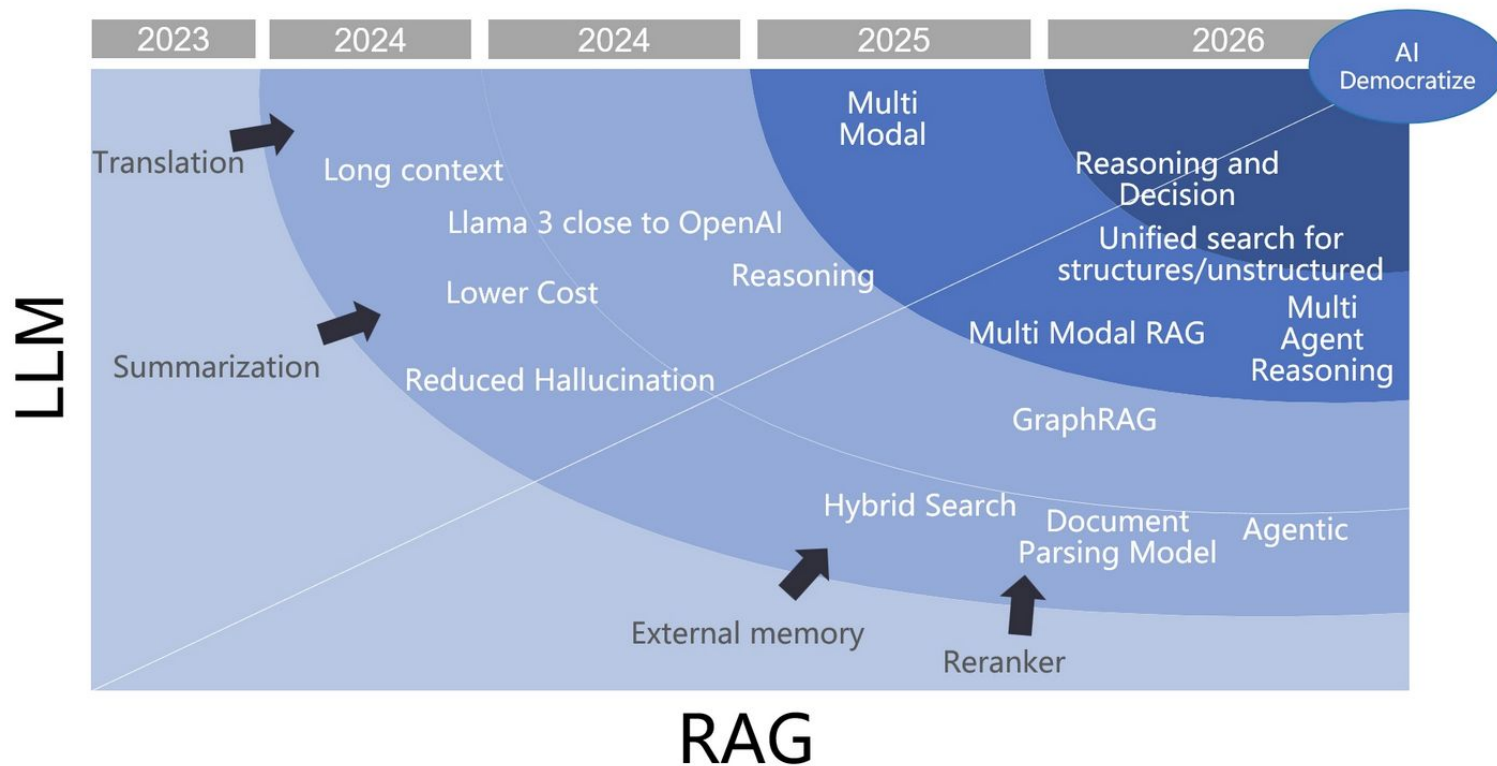
Feature Comparison	RAG	Fine-tuning
Knowledge Updates	Directly updates the retrieval knowledge base, ensuring information remains current without the need for frequent retraining, suitable for dynamic data environments.	Stores static data, requiring retraining for knowledge and data updates.
External Knowledge	Proficient in utilizing external resources, particularly suitable for documents or other structured/unstructured databases .	Can be applied to align the externally learned knowledge from pretraining with large language models, but may be less practical for frequently changing data sources.
Data Processing	Requires minimal data processing and handling .	Relies on constructing high-quality datasets , and limited datasets may not yield significant performance improvements.
Model Customization	Focuses on information retrieval and integrating external knowledge but may not fully customize model behavior or writing style .	Allows adjustments of LLM behavior , writing style, or specific domain knowledge based on specific tones or terms.
Interpretability	Answers can be traced back to specific data sources , providing higher interpretability and traceability.	Like a black box , not always clear why the model reacts a certain way, with relatively lower interpretability.
Computational Resources	Requires computational resources to support retrieval strategies and technologies related to databases . External data source integration and updates need to be maintained.	Preparation and curation of high-quality training datasets , definition of fine-tuning objectives, and provision of corresponding computational resources are necessary.
Latency Requirements	Involves data retrieval, potentially leading to higher latency .	LLM after fine-tuning can respond without retrieval, resulting in lower latency .
Reducing Hallucinations	Inherently less prone to hallucinations as each answer is grounded in retrieved evidence.	Can help reduce hallucinations by training the model based on specific domain data but may still exhibit hallucinations when faced with unfamiliar input.
Ethical and Privacy Issues	Ethical and privacy concerns arise from storing and retrieving text from external databases .	Ethical and privacy concerns may arise due to sensitive content in the training data .

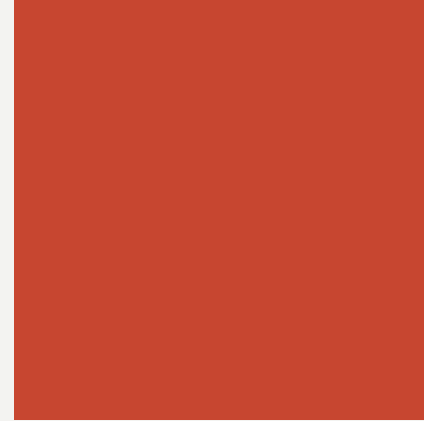
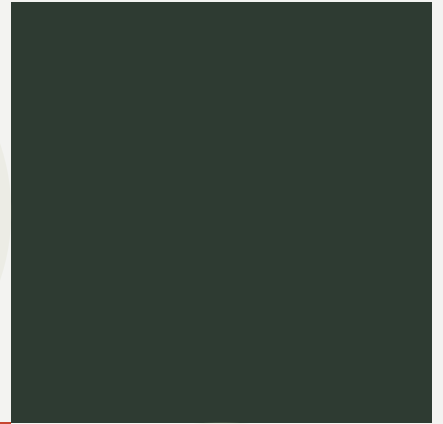
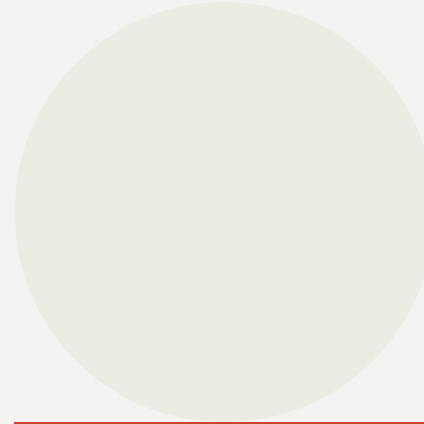
<https://arxiv.org/pdf/2312.10997v1>

Efficiency Comparison



RAG





Further Reading

- Fan et al., 2024. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. KDD 2024 <https://arxiv.org/pdf/2405.06211>
- Gao et al. 2024 Retrieval-Augmented Generation for Large Language Models: A Survey <https://arxiv.org/abs/2312.10997>
- Retrieval Augmented Generation. <https://aman.ai/primers/ai/RAG/>
- Everything you need to know about Vector Databases — A Deep Dive. <https://generativeai.pub/everything-you-need-to-know-about-vector-databases-a-deep-dive-4903a40e67a9>
- <https://github.com/Danielskry/Awesome-RAG?tab=readme-ov-file>
- <https://medium.com/@OpenRAG/modular-rag-and-rag-flow-part-%E2%85%B0-e69b32dc13a3>



Thank You

For Attending!