

Engineering Information Systems with LLMs and AI-based techniques

Massimo Mecella

Marco Calamo, Flavia Monti

Francesca De Luzi, Jerin Mathew, Mattia Macrì,
Jacopo Rossi, Matteo Marinacci, Filippo Bianchini,
Giovanni Della Pelle



SAPIENZA
UNIVERSITÀ DI ROMA

Who we are ? (1/2)

- The **Processes, Services and Software Engineering** group of Sapienza Università di Roma (currently one of the largest *IS Engineering group* in Italy)
- Massimo Mecella
 - Professor (Assist – '06 / Assoc – '15 / Full '21)
 - One of the inventor of the «Roman model» for automatic service composition (so far the most impactful in SOC)
 - Recently GC of CAiSE 2019, BPM 2021 (the 1st «back in presence» conference of our community after pandemic), ICSOC 2023
 - Sits in the SCs of CAiSE, ICSOC, SummerSOC, Intelligent Environments, AVI
 - Research interests in IS Engineering, BPM, SOC, smart environments (recently Industry 4.0), advanced user interfaces (ubiquitous, immersive), software engineering (also in aerospace)

Who we are ? (2/2)

- Flavia Monti – PhD student in Eng CS on Industry 4.0 (ends in Jan 2025)
- Marco Calamo – PhD student in EngCS on AI & Analytics (will end in Jan 2027)
- Francesca De Luzi – PostDoc on innovative technologies and management
- Jerin Mathew – PhD student in EngCS on data management (ends in Jan 2025)
- Mattia Macrì – PhD student in Mgmt Eng on knowledge management via LLMs (will end in Jan 2026)
- Jacopo Rossi – PhD student in aerospace software eng (will end in Oct 2026)
- Matteo Marinacci – PhD student in immersive environments for aerospace (will end in Oct 2026)
- Filippo Bianchini – PhD student in knowledge management (will end in Jan 2027)
- Giovanni Della Pelle – MSc student in Eng CS (will end in Oct 2024, then hopefully a PhD student)



Outline of the tutorial

- Introducing AI and GenAI and LLMs
- Once upon a time ...
- Surveying LLMs in IS engineering phases
- A vision for the future: **low design** and **design assistants**
- Three current researches of our towards this vision
- Concluding remarks

Introducing AI, GenAI, LLMs

A quick&dirty introduction with the eyes of an IS engineer

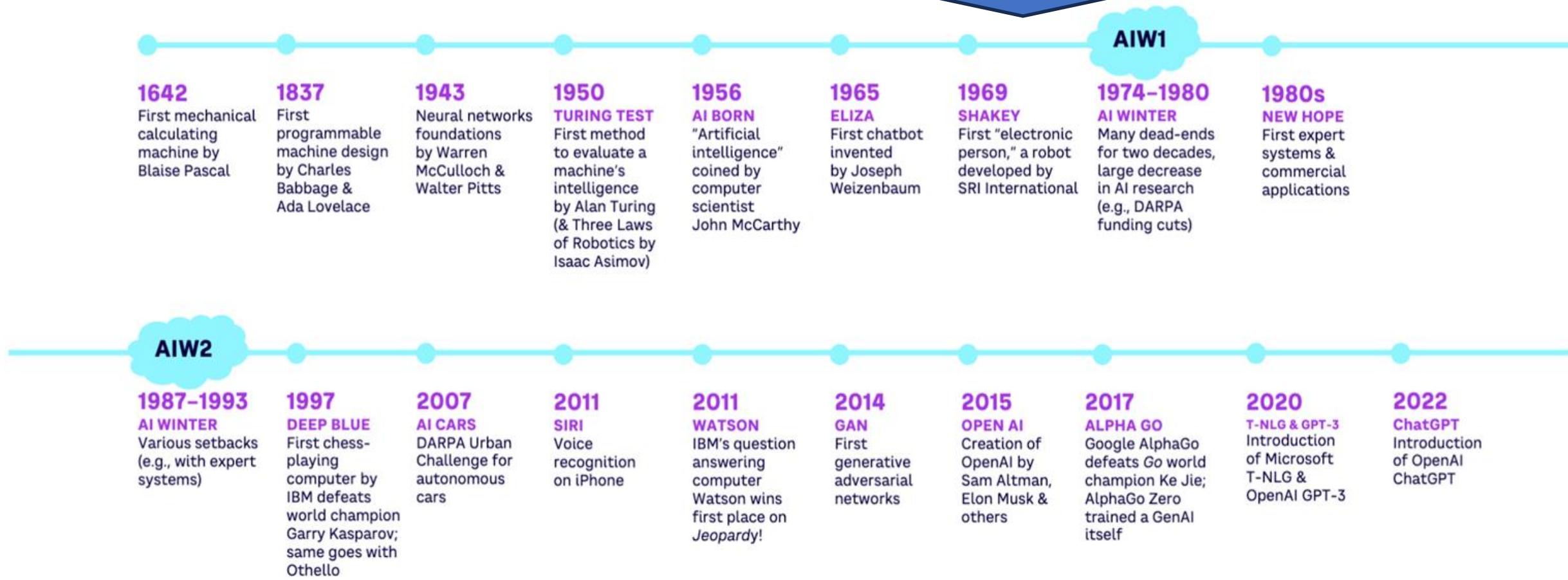


SAPIENZA
UNIVERSITÀ DI ROMA

AI Timeline

Artificial intelligence (AI) is technology that enables computers and machines to simulate human intelligence and problem-solving capabilities

Source: <https://www.ibm.com/topics/artificial-intelligence>



Source: <https://www.adlittle.com/en/insights/report/generative-artificial-intelligence-toward-new-civilization>



SAPIENZA
UNIVERSITÀ DI ROMA

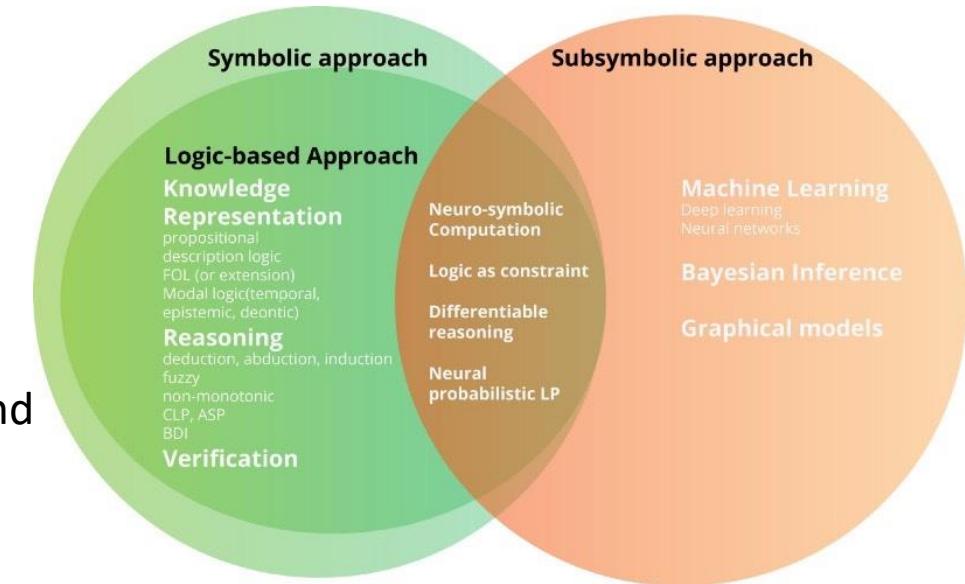
Symbolic and Subsymbolic AI

Symbolic AI

also called rule-based or traditional AI, it refers to an artificial intelligence approach that relies on a base of **knowledge represented in the form of symbols or rules**

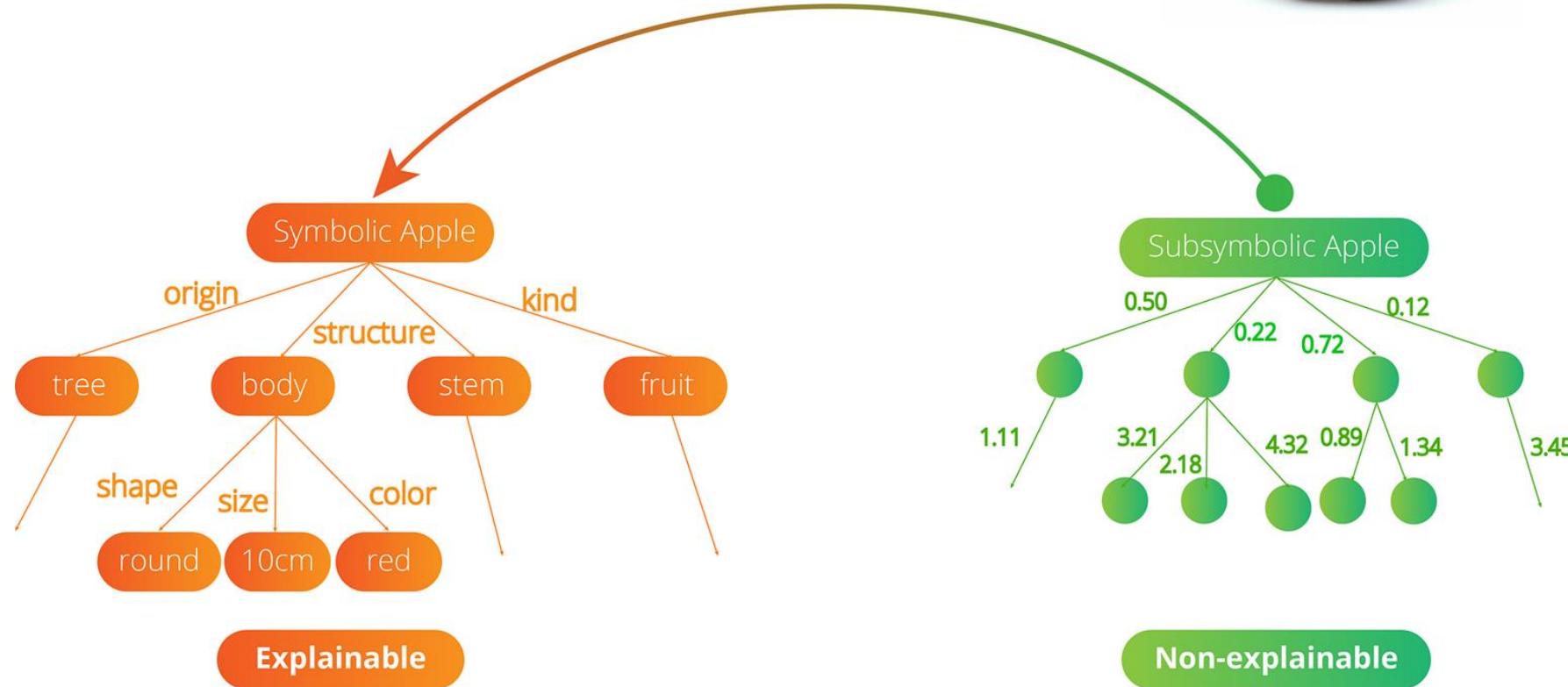
Subsymbolic AI

it refers to the approach that focuses on creating intelligent behavior by simulating the behavior in biological systems. It deals with the processing and analysis of data using **numerical methods and statistical models** to solve problems without the need for explicit rules or algorithms



Source: <https://www.vizio.ai/blog/black-box-and-white-box-models-towards-explainable-ai>

Symbolic and Subsymbolic AI



Source: <https://towardsdatascience.com/symbolic-vs-subsymbolic-ai-paradigms-for-ai-explainability-6e3982c6948a>



SAPIENZA
UNIVERSITÀ DI ROMA

History of Language Models

Source: [Evolution of Language Models: From Rules-Based Models to LLMs](#)

What is a language model?

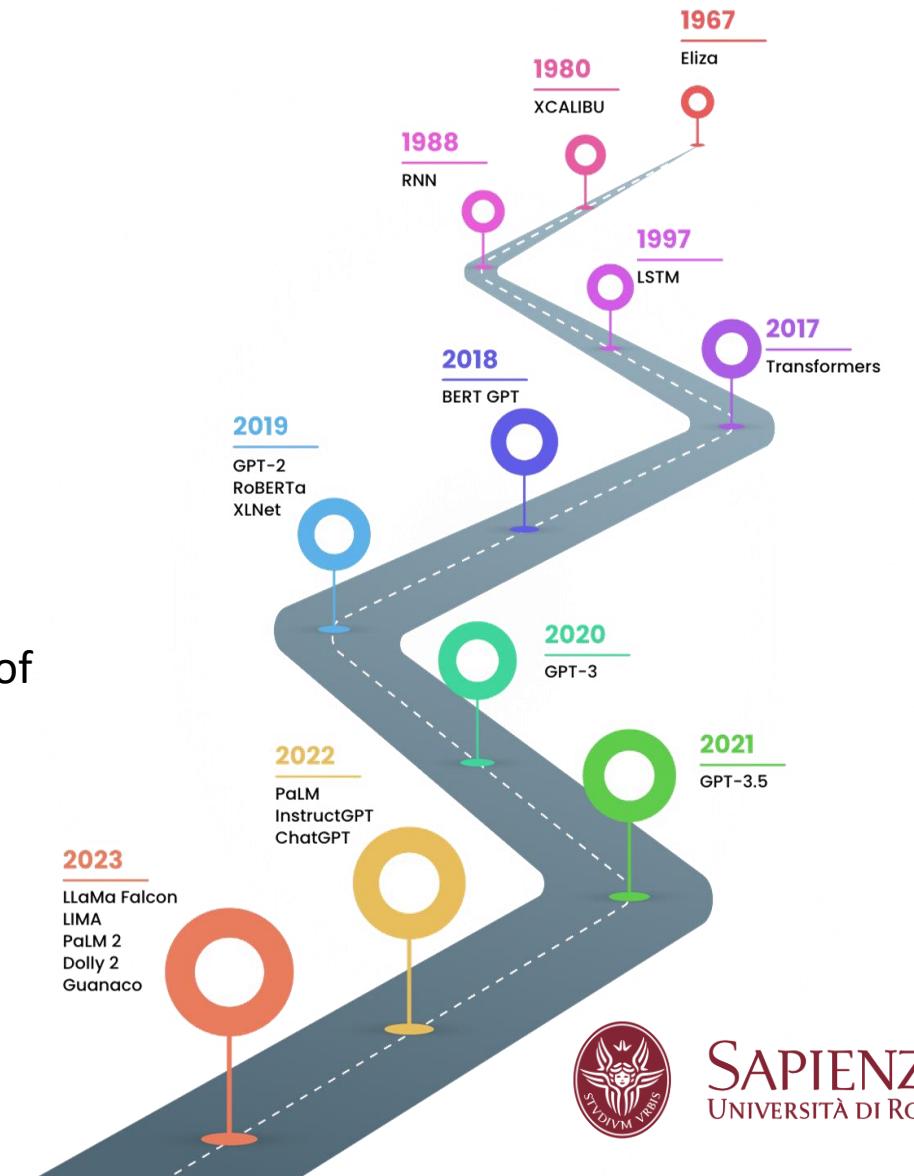
A language model is a probabilistic model of a natural language^[1]. Since the 60s researchers in the field of Artificial Intelligence have tried to *teach* the natural language to the machine.

A few milestones:

- **Eliza**: is the world's first chatbot. It simulated conversation by using a pattern matching and substitution methodology that gave users an illusion of understanding on the part of the program (*Symbolic AI*)
- **RNN**: is a type of neural network characterized by direction of the flow of information between its layers (*Subsymbolic AI*)
- **LSTM**: is a type of RNN aimed at dealing with the vanishing gradient problem present in traditional RNNs^[2] (*Subsymbolic AI*)
- **Transformers**: the major breakthrough in the field that made possible large language models (*Subsymbolic AI*)

[1] Jurafsky, Dan; Martin, James H. "N-gram Language Models". (2024)

[2] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.



Large Language Models - Basics

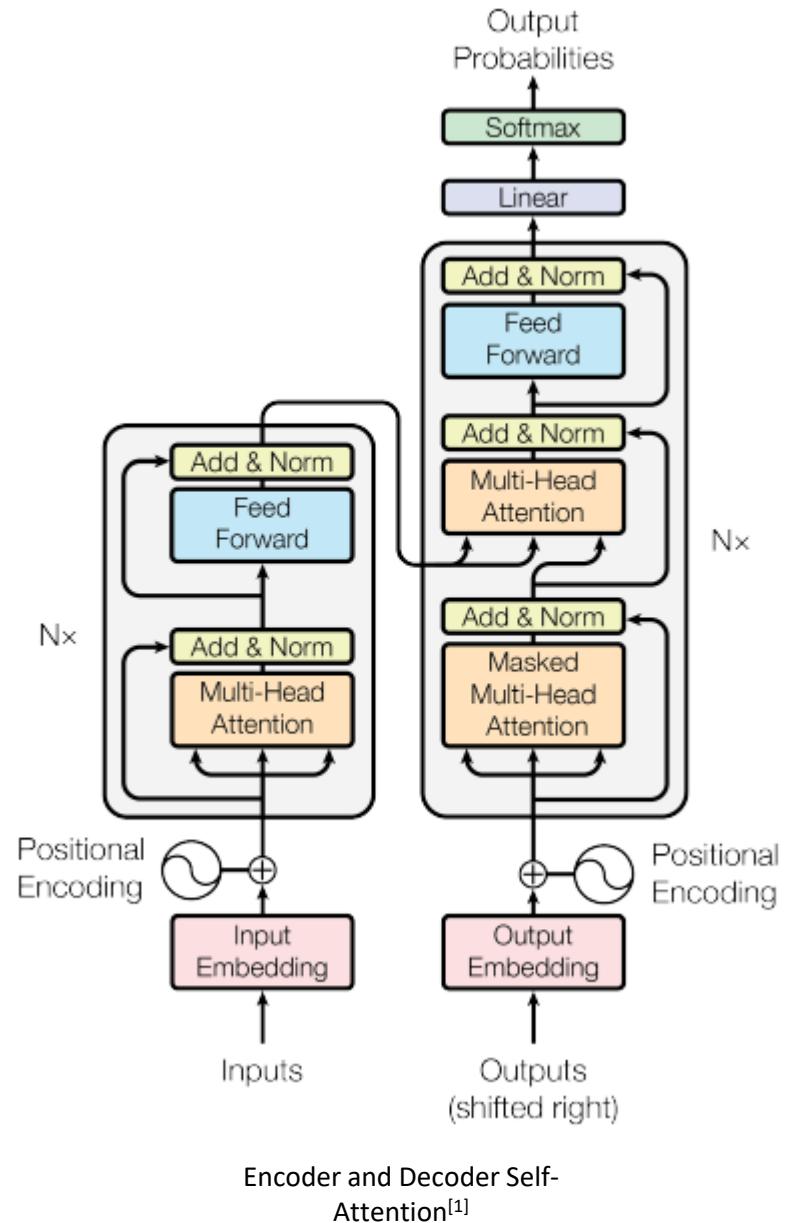
Transformers

A **transformer** is a deep learning architecture developed by Google and based on the **multi-head self attention mechanism**^[1]. They have the advantage of having no recurrent units. It is originally composed by an **encoder** and a **decoder** part.

What makes a model “large”

The introduction of the transformer architecture made possible the training of models with **billions of parameters**. Bigger models as of today are reaching one trillion parameters.

Even though the transformer architecture has been improved over the years (mostly because of $O(n^2)$ computation complexity), **it is still the base of modern Large Language Models**.

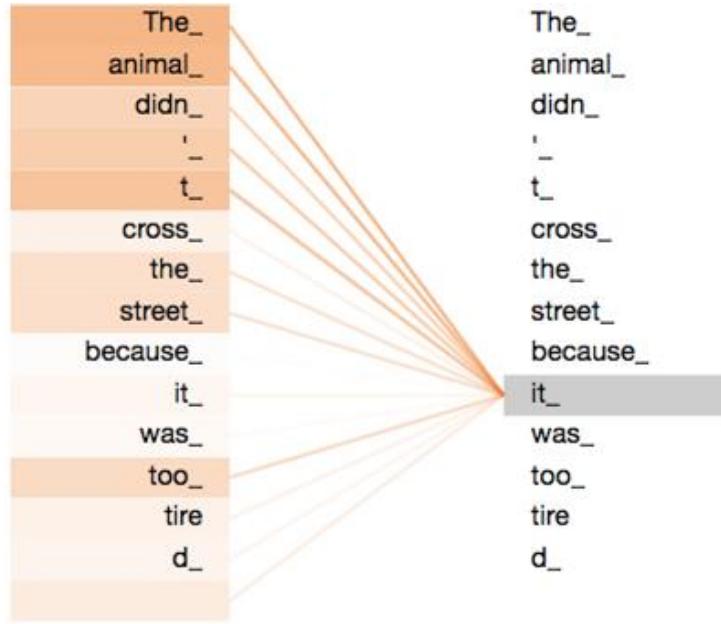


Encoder and Decoder Self-
Attention^[1]

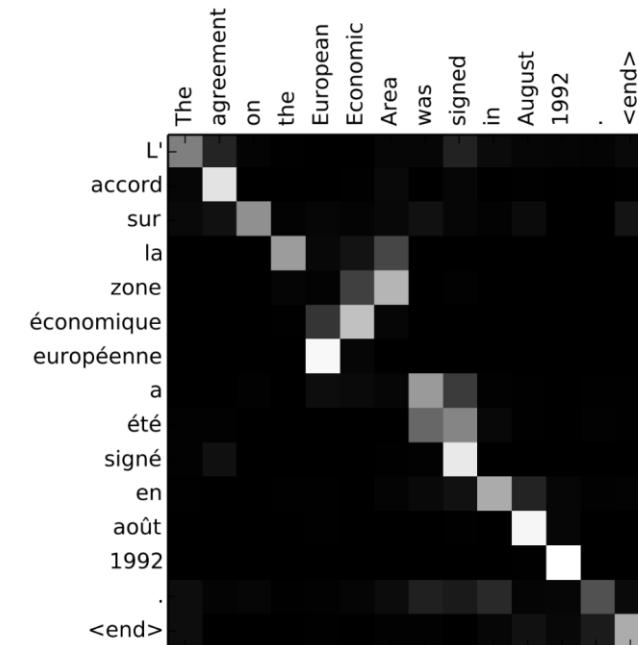
[1] Vaswani, Ashish, et al. "Attention is all you need." (2017)



Visualizing Attention



Self-attention



Attention-matrix heatmap

Bahdanau, et al. 2015. Neural machine translation by jointly learning to align and translate. In Proc. ICLR.

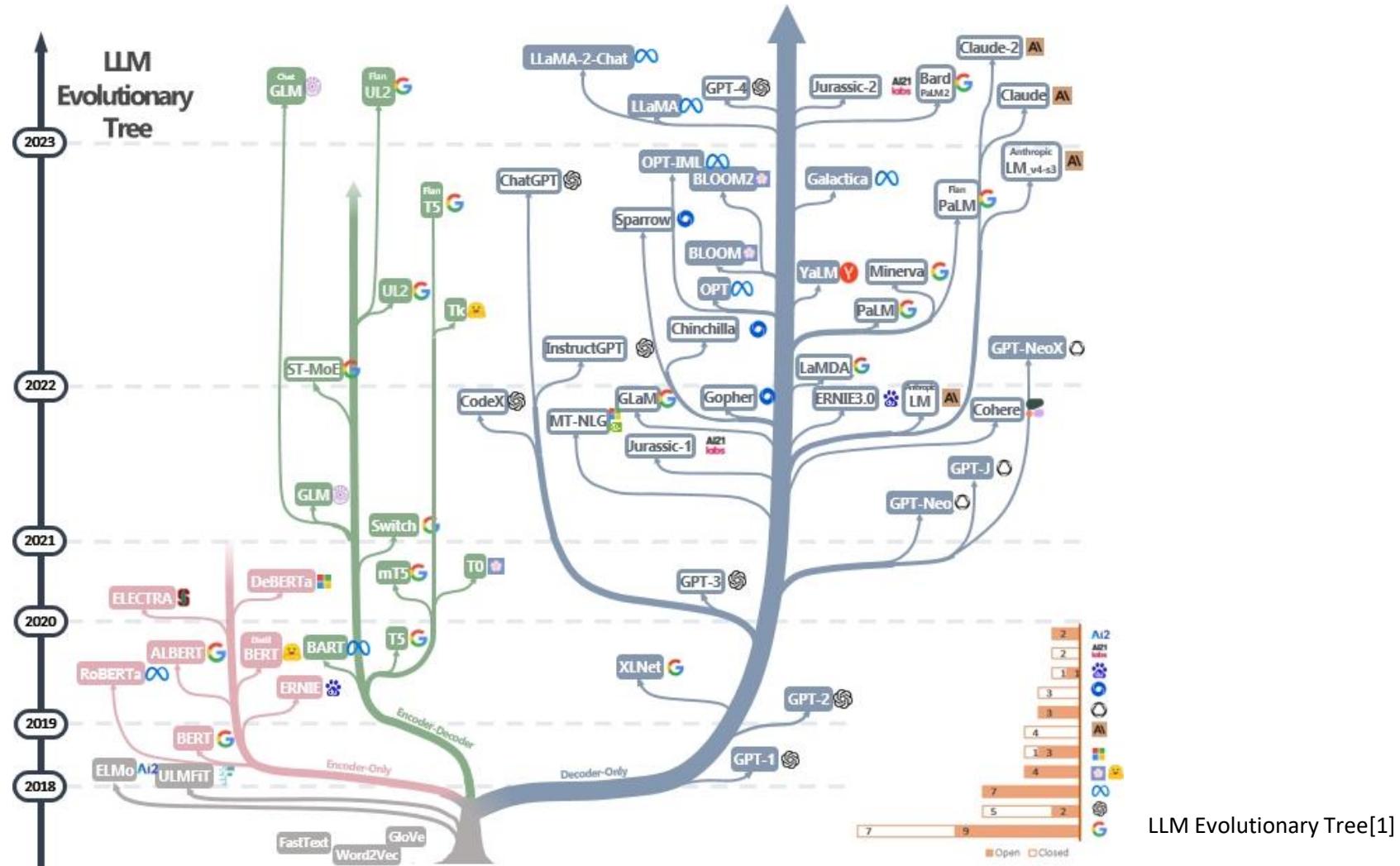
Cross-attention

Source: <https://jalammar.github.io/illustrated-transformer/>



SAPIENZA
UNIVERSITÀ DI ROMA

LLM - Architecture and Evolution



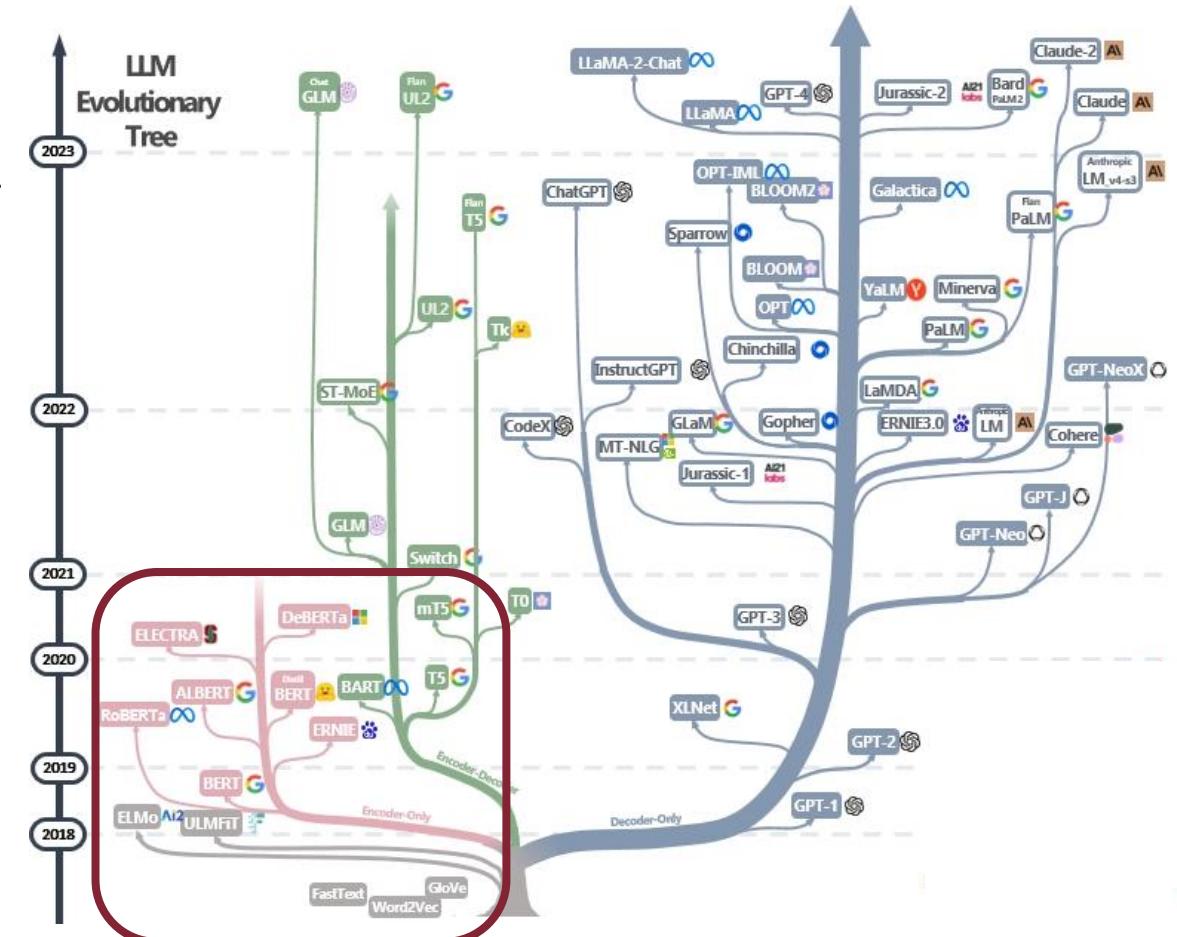
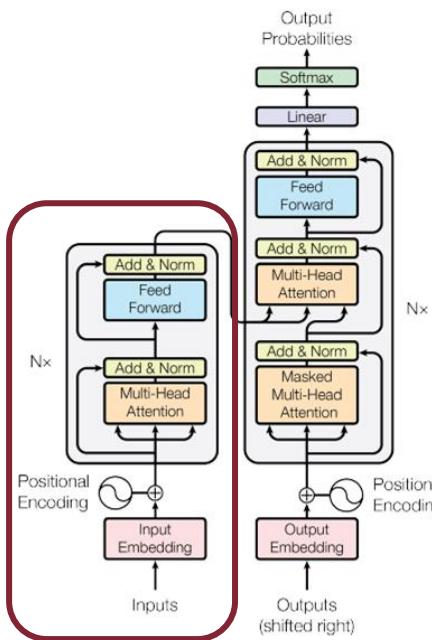
[1] [Yang, Jingfeng, et al. "Harnessing the power of llms in practice: A survey on chatgpt and beyond \(2024\)"](#)



LLM - Architecture and Evolution

Encoder-Only Architecture

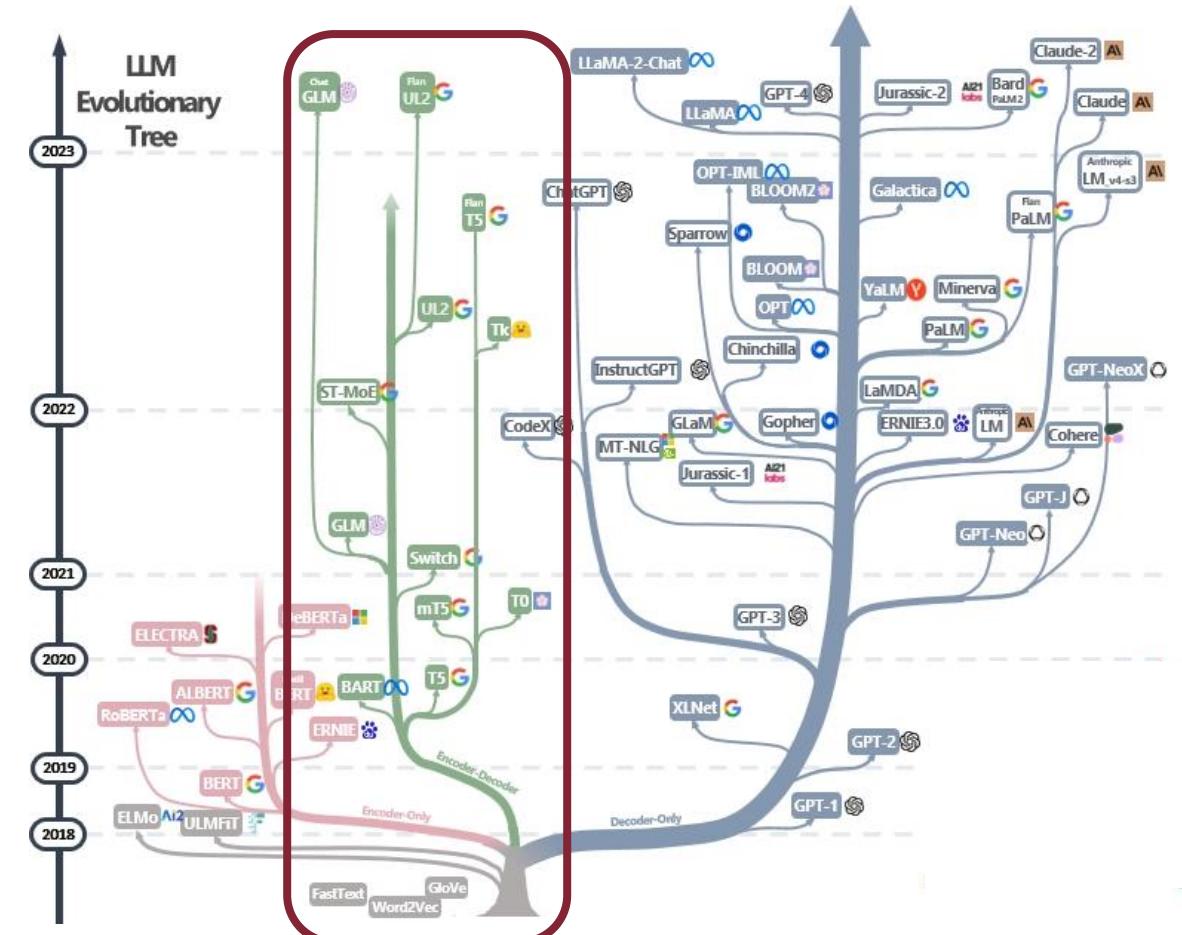
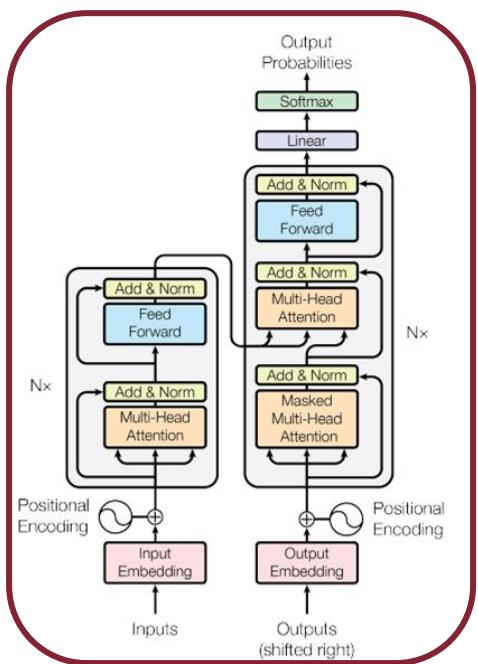
The transformer encoder architecture takes in a sequence of tokens and produces a fixed-size vector representation of the entire sequence, which can then be used for **classification**



LLM - Architecture and Evolution

Encoder-Decoder Architecture

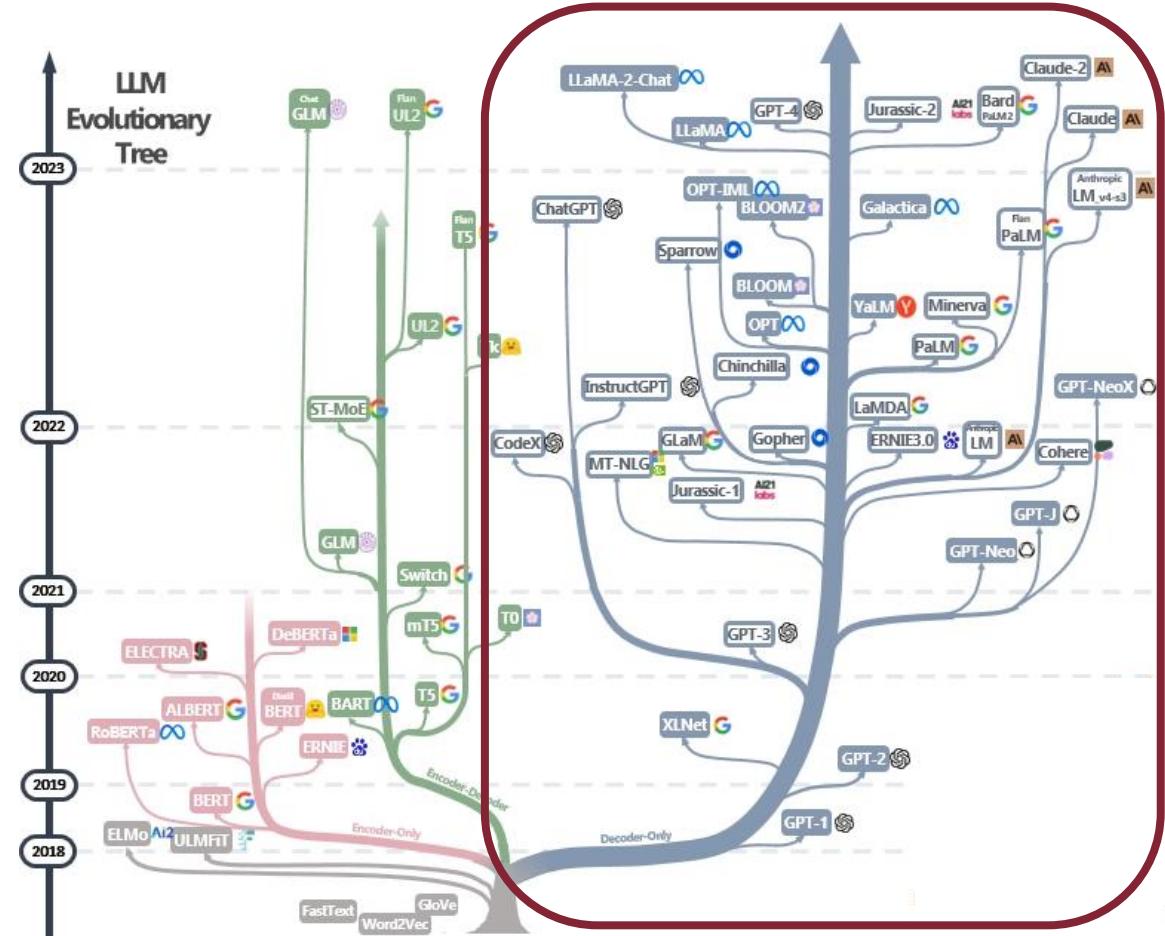
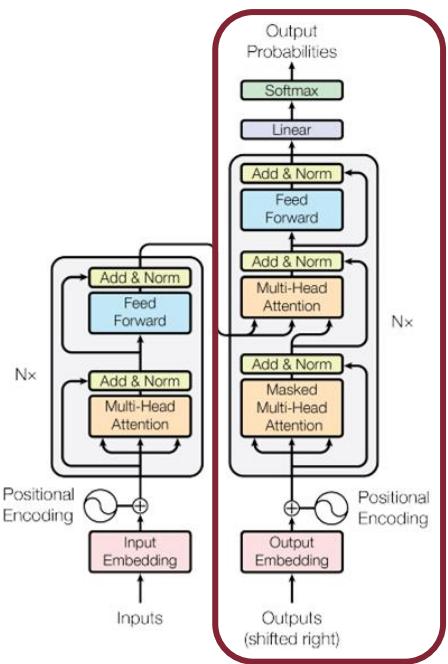
The transformer encoder-decoder architecture is used for tasks like **language translation**, where the model must take in a sentence in one language and output a sentence in another language



LLM - Architecture and Evolution

Decoder-Only Architecture

The transformer decoder architecture is used for tasks like **language generation**. The decoder takes in a fixed-size vector representation of the context and uses it to generate a sequence of words, with each word being conditioned on the previously generated words



Large Language Models - Pre-Training

Pre-Training or Training?

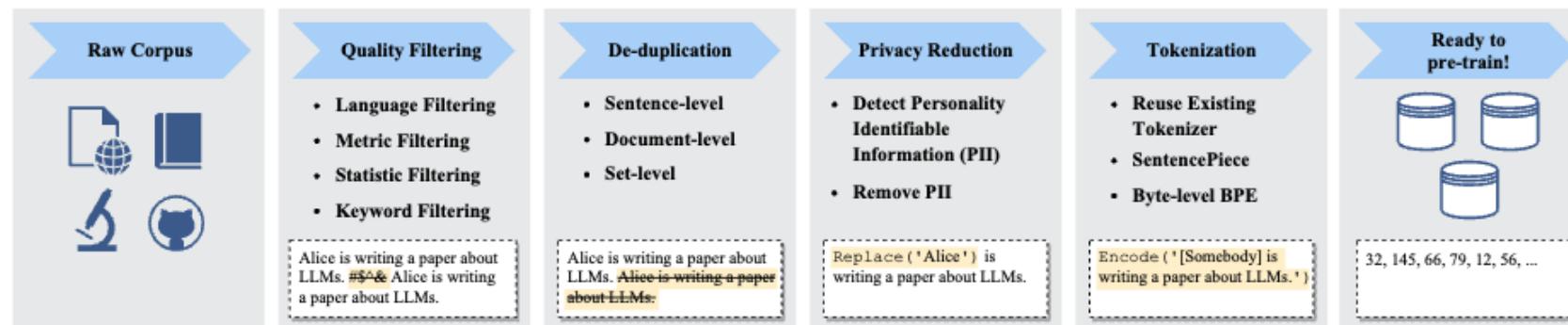
Pre-training is a foundational step in the LLM training process, where the model gains a general **understanding of language** by exposure to vast amounts of text data, in an unsupervised manner. It is called pre-training (instead of training) because after this stage the model is generally **not capable of solving specific problems**. Most pre-training common task are:

Masked Language Modeling (MLM): “The quick brown fox jumps over the _____ (lazy) dog.”

Next Sentence Prediction (NSP): “The scientist made a groundbreaking discovery. ____ (She was awarded the Nobel Prize).”

Contrastive Learning: “Hope is the thing with feathers — / That perches in the soul — / And sings the tune without the words — / And never stops at all — (Similar)”

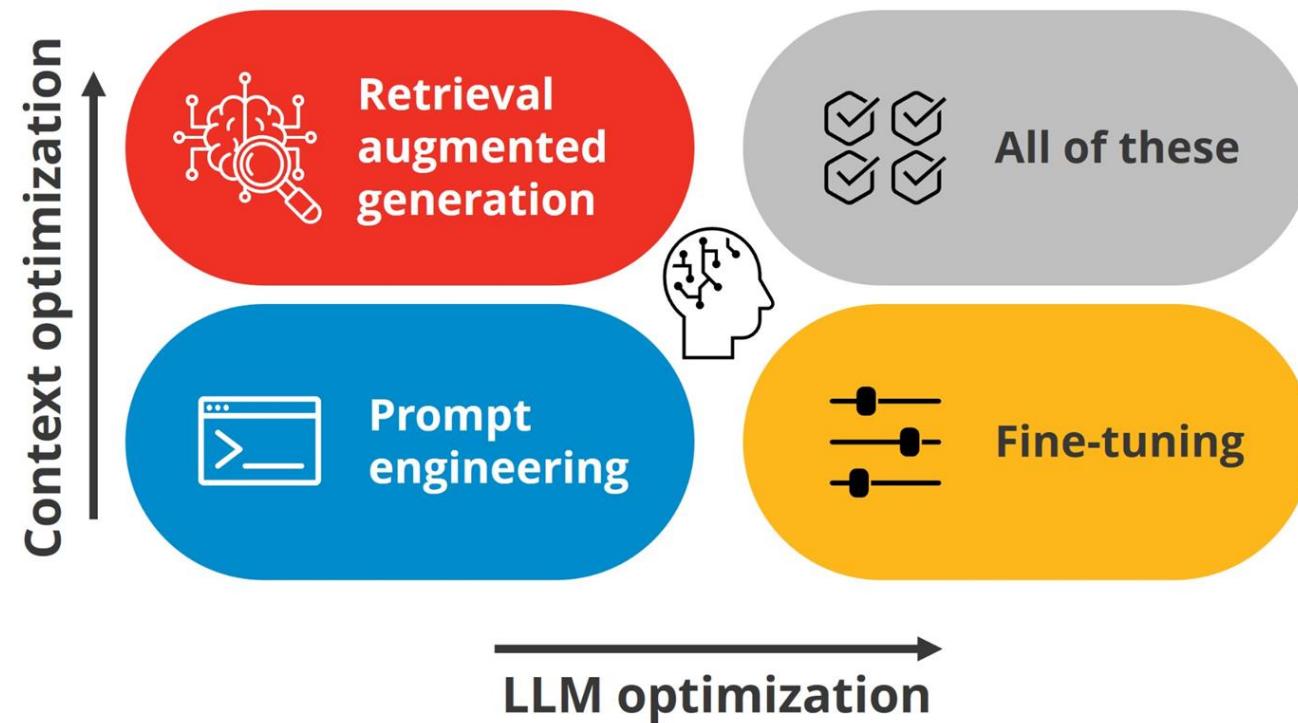
Permutation Language Modeling (PLM): “the sky blue is so” -> “so blue is the sky”



[1] Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023)

Pre-Training Pipeline[1]

Specializing Large Language Models



Optimizing a Language Model for a specific task

Source: <https://fractal.ai/blog/governing-langs-through-enhancing-techniques/>

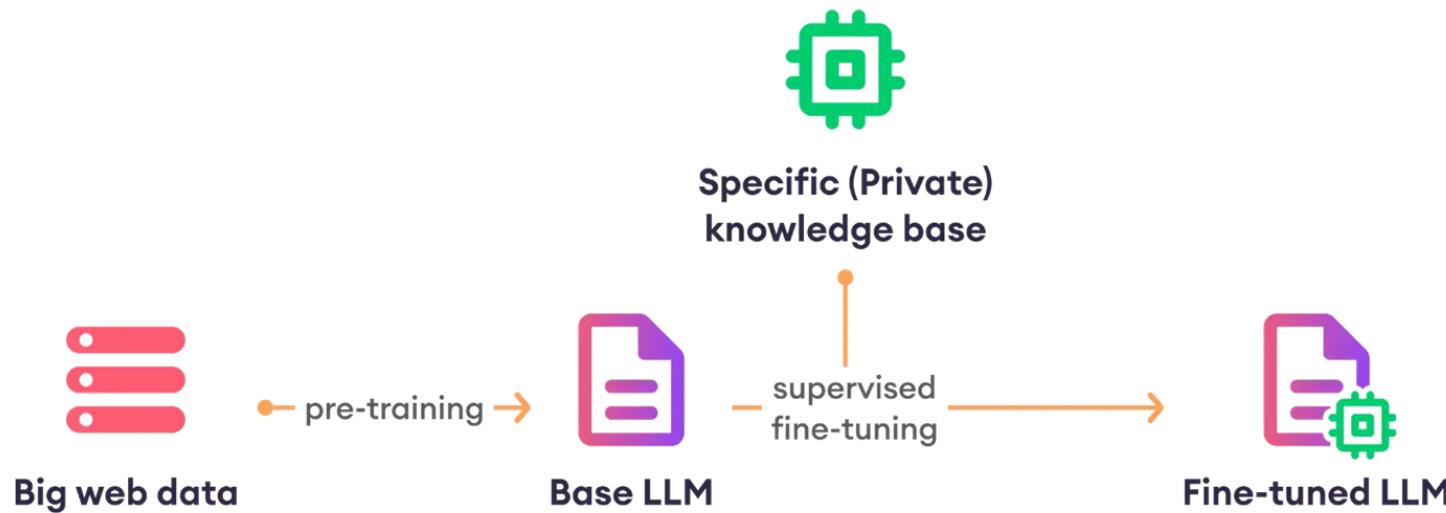


SAPIENZA
UNIVERSITÀ DI ROMA

Fine-Tuning

After pre-training, LLMs can acquire the general abilities for solving various tasks. However, an increasing number of studies have shown that LLM's abilities can be further adapted according to specific goals with **fine tuning**.

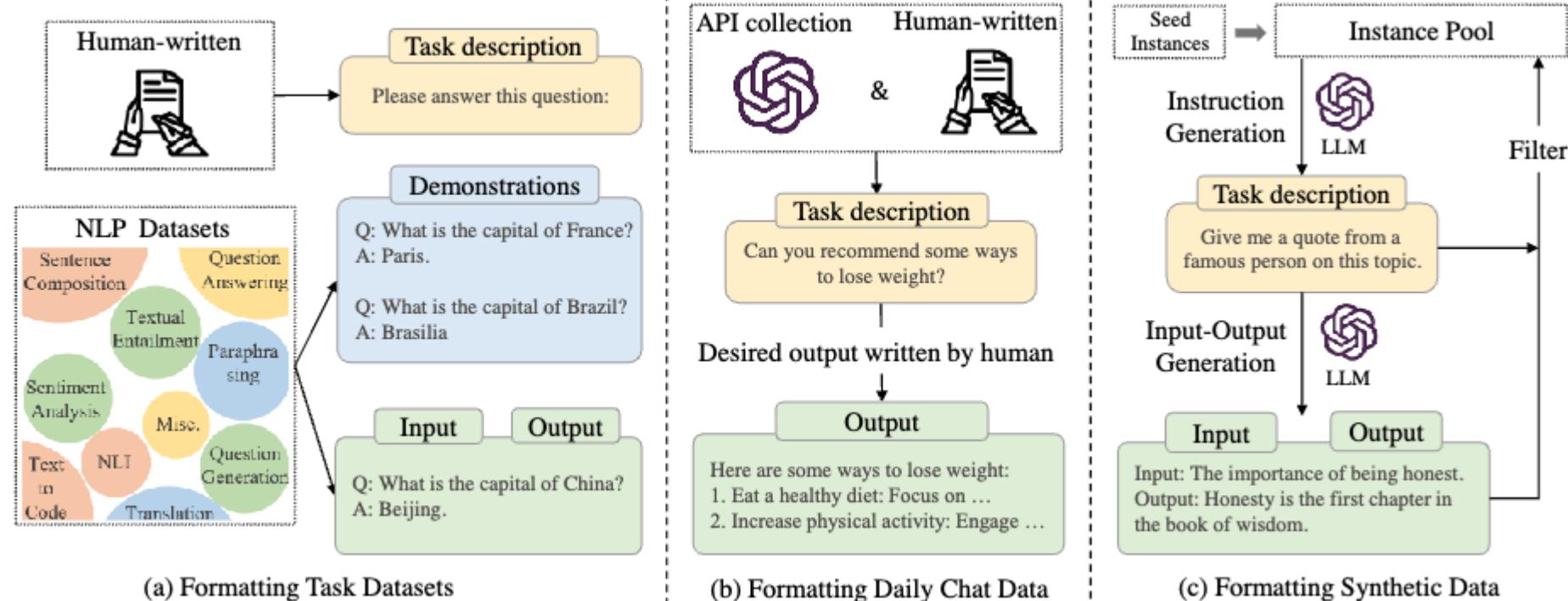
Several optimization strategies can make the fine-tuning process **parameter efficient** and/or **memory efficient**



Source: <https://www.superannotate.com/blog/llm-fine-tuning>

Fine Tuning - Instruction

Instruction tuning is the approach to fine-tuning pre-trained LLMs on a collection of formatted instances in the form of instructions in natural language. Even with a small dataset it can be very efficient



Instruction Fine tuning^[1]

[1] Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023)



SAPIENZA
UNIVERSITÀ DI ROMA

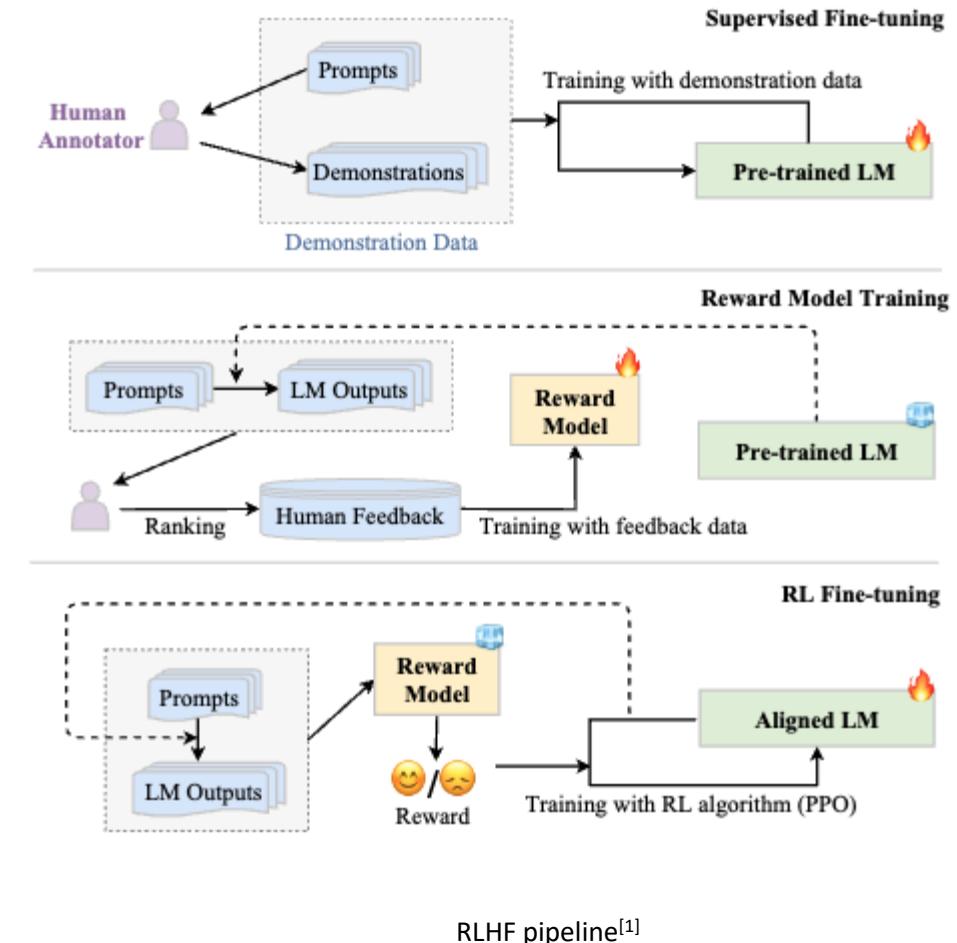
Fine Tuning - Alignment Tuning/RLHF

Alignment Tuning is used to avert these unexpected behaviors (e.g., fabricating false information, pursuing inaccurate objectives, and producing harmful, misleading, and biased expression)

Reinforcement Learning from Human Feedback (RLHF)

It is used to align LLMs with human values. It is a fine-tuned LLMs with the collected human feedback data, which is useful to improve the alignment criteria (e.g., helpfulness, honesty, and harmlessness). employing reinforcement learning (RL) algorithms to adapt LLMs to human feedback by learning a reward model.

- ChatGPT is an example



[1] Zhao, Wayne Xin, et al. "A survey of large language models." *arXiv preprint arXiv:2303.18223* (2023)



Fine Tuning limitations

Specialization is expensive

Fine-tuning is the **most effective** technique for specializing an LLM. However, it is **not always viable**. The major drawbacks of fine-tuning of LLM are **costs** in terms of:

- human resources (for both supervised and unsupervised training)
- time
- power
- CO₂ emissions

Model	GPT-3	BLOOM	LLaMA	LLaMA-2	T5	PaLM
Developer	OpenAI	BigScience	Meta	Meta	Google	Google
Model Size (# parameters)	175B	175B	7B, 13B, 33B, 65B	7B, 13B, 34B, 70B	11B	540B
Training Data (# tokens)	300B	350B	1.4T	2T	34B	795B
Training Compute (FLOPs)	3.2E+23	3.7E+23	9.9E+23	1.5E+24	2.2E+21	2.6E+24
Processor	Manufacturer	Nvidia	Nvidia	Nvidia	Nvidia	Google
	Type	GPU	GPU	GPU	GPU	TPU
	Model	V100	A100	A100	A100	TPU v3
Processor Hours	3,552,000	1,082,990	1,770,394	3,311,616	245,760	8,404,992
Grid Carbon Intensity (kgCO ₂ e/KWh)	0.429	0.057	0.385	0.423	0.545	0.079
Data Center Efficiency (PUE)	1.1	1.2	1.1	1.1	1.12	1.08
Energy Consumption (MWh)	1,287	520	779	1,400	86	3,436
Carbon Emissions (tCO ₂ e)	552	30	300	593	47	271

$KWh = \text{Hours to train} \times \text{Number of Processors} \times \text{Average Power per Processor} \times PUE \div 1000$

$tCO_2e = KWh \times \text{kg CO}_2e \text{ per KWh} \div 1000$

Source: <https://lajavaness.medium.com/llm-large-language-model-cost-analysis-d5022bb43e9e>



SAPIENZA
UNIVERSITÀ DI ROMA

Fine Tuning - Parameter Efficient

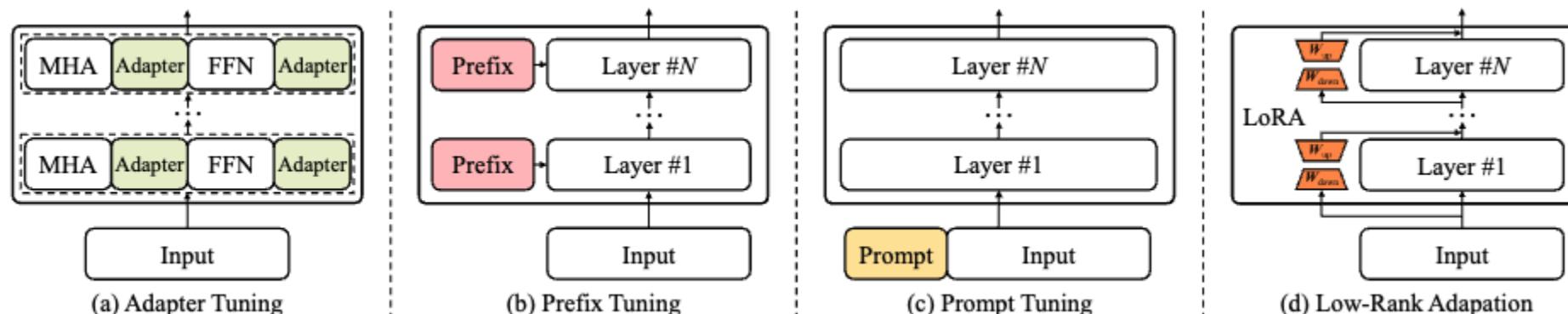
Parameter-efficient fine-tuning has been an important topic that aims to reduce the number of trainable parameters while retaining a good performance as possible. Several techniques can be used:

Adapter Tuning: incorporates **small neural network modules** (called adapter) into the Transformer models. Only adapter parameters are updated during fine-tuning

Prefix Tuning: prepends a sequence of prefixes, which are a set of **trainable continuous vectors**, to each Transformer layer in language models. Only prefix parameters are updated during fine-tuning

Prompt Tuning: incorporates **trainable prompt vectors** at the input layer

Low-Rank Adaptation (LoRA): imposes the **low-rank constraint** for approximating the update matrix at each dense layer, to reduce the trainable parameters for adapting to downstream tasks



Parameter efficient fine tuning[1]

[1] Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023)



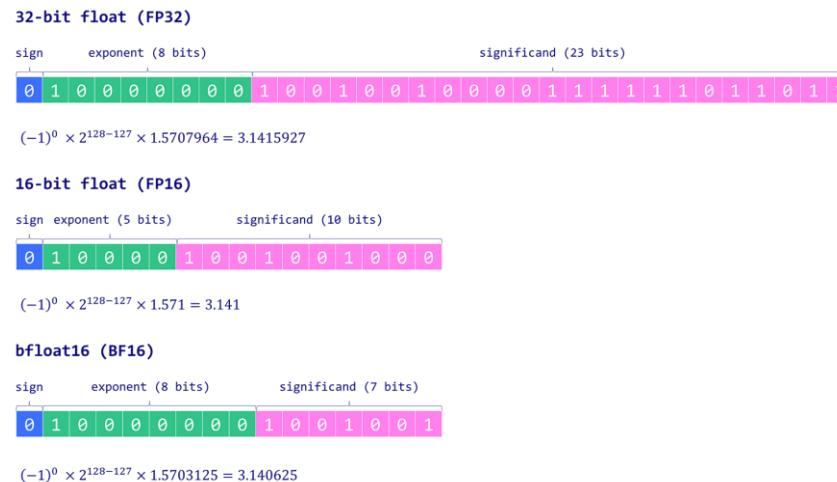
SAPIENZA
UNIVERSITÀ DI ROMA

Fine Tuning - Memory Efficient

The most used techniques to reduce a model's memory footprint are either reducing floating points weight resolution from the full 32-bit down to 16-bit or **quantization**

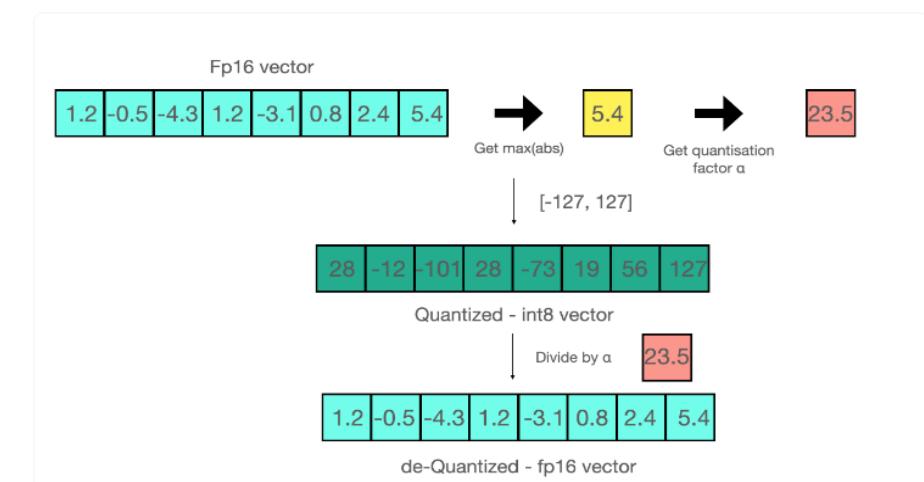
Quantization

It maps weights from floating-point numbers to integers, of different resolutions, from 8-bit down to 2-bit



Source:

<https://towardsdatascience.com/introduction-to-weight-quantization-2494701b9c0c>



Source: <https://levelup.gitconnected.com/cant-afford-the-expensive-gpu-for-your-ai-development-try-4-bit-quantization-b26108229b75>



SAPIENZA
UNIVERSITÀ DI ROMA

Prompt Engineering

Prompt engineering is the process of structuring an instruction that can be interpreted and understood by a generative AI model^[1]. It enables:

Zero-Shot

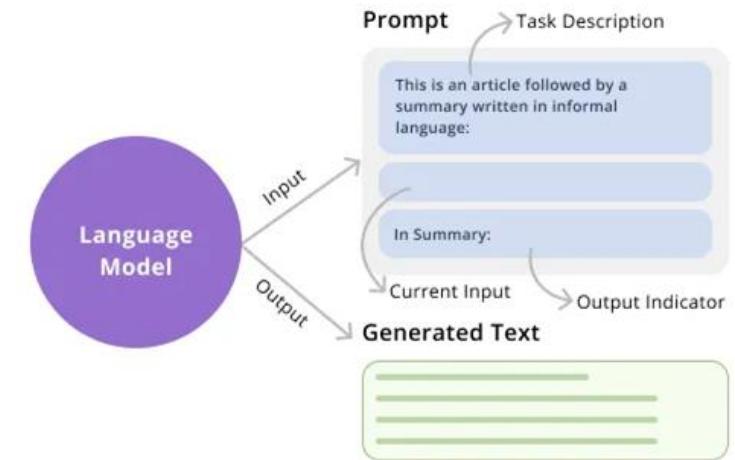
The prompt used to interact with the model **won't contain examples** or demonstrations

In-Context Learning or Few-Shot Learning

Demonstrations of the task are provided to the model as part of the prompt

Chain of thought/reasoning

Incorporates intermediate **reasoning steps**, which serve as the bridge between inputs and outputs

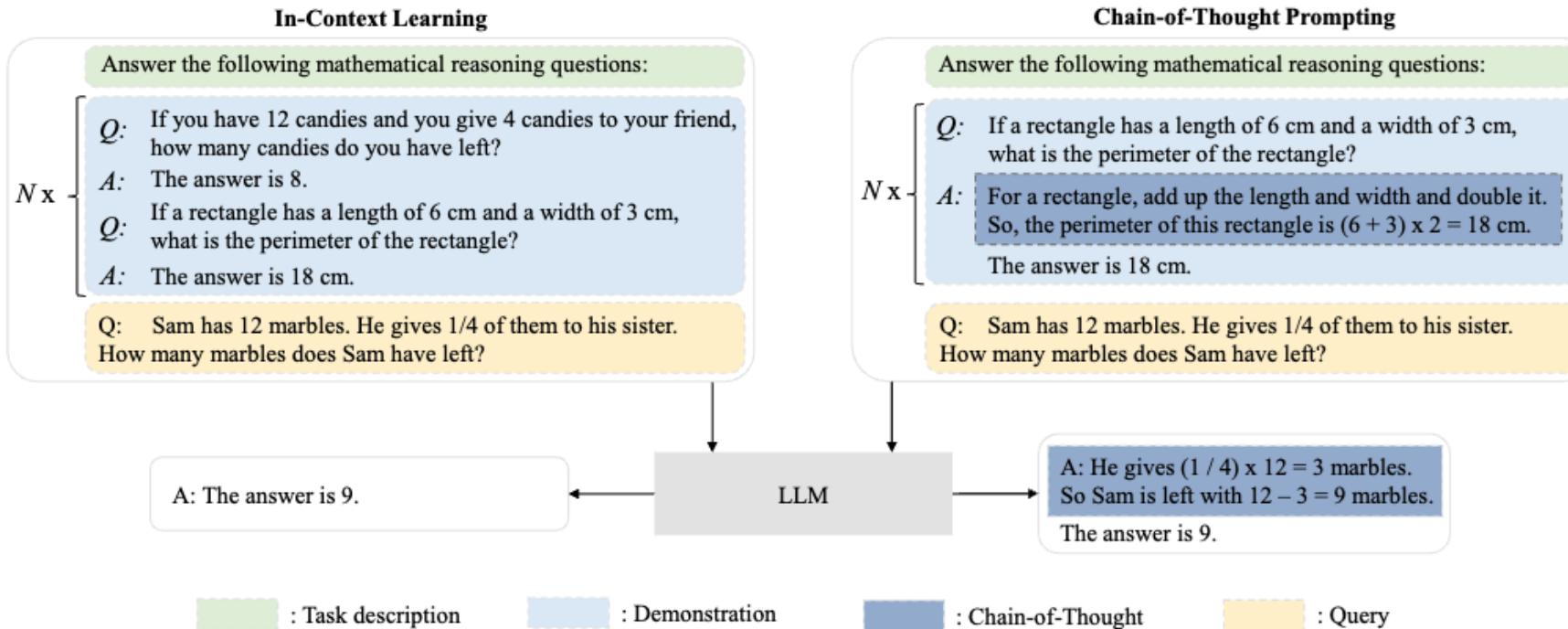


Source: <https://botpenguin.com/glossary/prompt-engineering>



SAPIENZA
UNIVERSITÀ DI ROMA

Prompt Engineering at Work



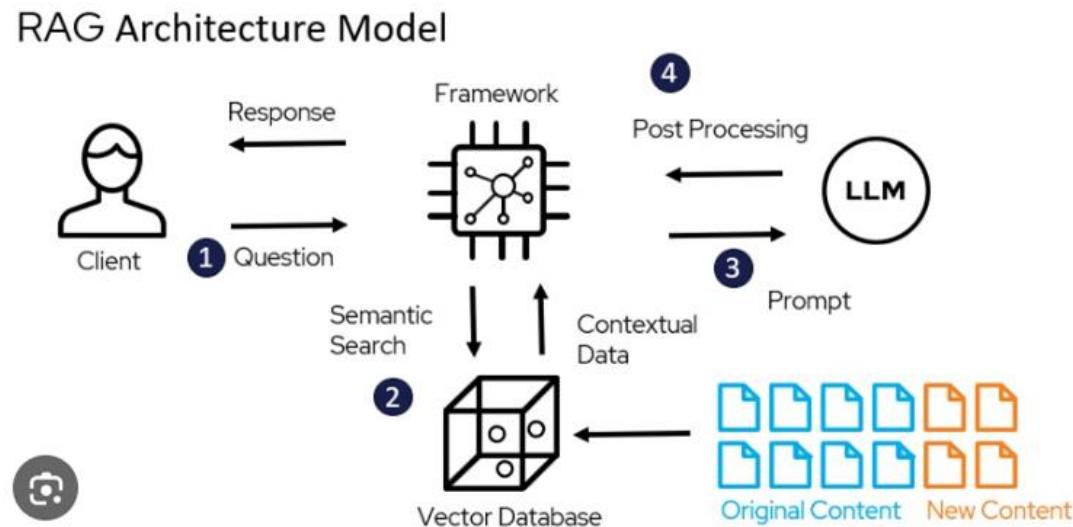
Source: [Zhao, Wayne Xin, et al. "A survey of large language models."](#) arXiv preprint arXiv:2303.18223 (2023)



SAPIENZA
UNIVERSITÀ DI ROMA

Retrieval Augmented Generation (RAG)

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



Source:

<https://medium.com/@bijit211987/advanced-rag-for-langs-slms-5bcc6fbba411>



SAPIENZA
UNIVERSITÀ DI ROMA

RAG at Work

Who is Massimo Mecella?

User Prompt

LLM
(zero shot)

Are we sure about that? 😳

Massimo Mecella is a well-known **Italian photographer** and visual artist who has gained fame for his stunning and often surreal photographs of people, landscapes, and still-life compositions.

Final Output

Tested with [meta-llama/Meta-Llama-3-8B-Instruct](#)



MASSIMO MECELLA


 Professore ordinario
 stanza: B211
 telefono: +39 0677274028

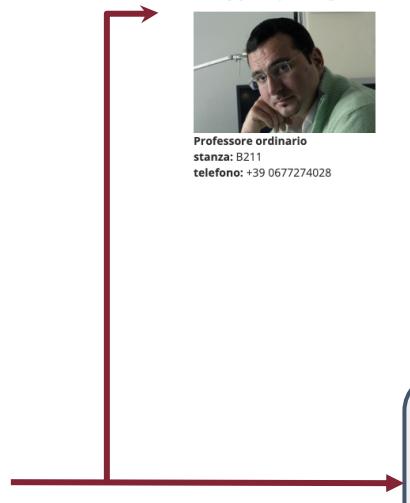
Biography

Massimo Mecella (born in Rome, Italy in 1973) holds a PhD in Computing Science and Engineering from SAPIENZA (earned on 21 January 2002). Currently he is associate professor in the School of Information Engineering, Computer Science and Statistics of SAPIENZA, in the Department of Computer, Control and Management Sciences & Engineering ANTONIO RUBERTI (DIAG, formerly DIS - Department of Computer and Systems Sciences). He has recently achieved (April 2017) the Italian National Qualification (ASN) to Full Professor. From March 2005 to August 2006 he has been a Postdoc on *Service Oriented Computing: Synthesis and Orchestration of Composite Services* at DIS. In the winter semester 2005, he has been a temporary research assistant at CERIAS (Center for Education and Research in Information Assurance and Security) and Department of Computer Science, Purdue University, USA, hosted by Elisa Bertino. From November 2002 to October 2004 he has been a research fellow in SAPIENZA (in the Latina campus). During the PhD period, in 1999 he has been *Internship student* at Telcordia Applied Research, Telcordia Technologies (formerly Bellcore), Morristown, New Jersey, USA, hosted by Amjad Umar, Paolo Missier and Francesco Caruso.

RAG at Work

Who is Massimo Mecella?

User Prompt



LLM + RAG



You are an assistant for question-answering tasks. Use the following pieces of retrieved context to answer the question. If you don't know the answer, just say that you don't know. Use three sentences maximum and keep the answer concise.

Question: Who is massimo mecella?

Context: Massimo Mecella | Department of Computer, Control and Management Engineering [...]

Rag generated prompt

Retrieval from the internet

Much better

Massimo Mecella is a **Professor of computer science** at the Sapienza University of Rome, Italy, and leader of the Processes, Services and Software Management group.

Final Output

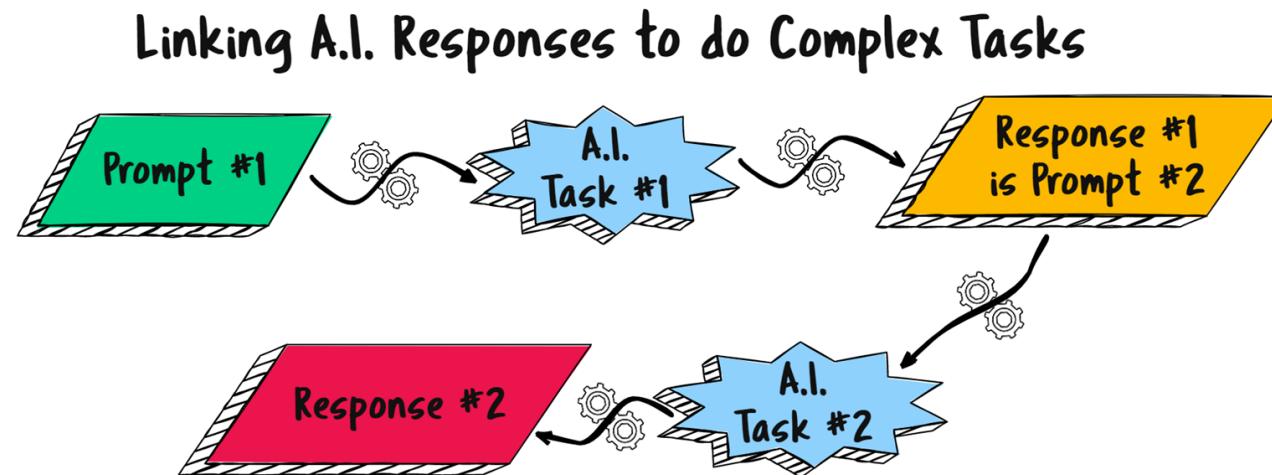
Tested with [meta-llama/Meta-Llama-3-8B-Instruct](#) and [langchain v0.2](#)



SAPIENZA
 UNIVERSITÀ DI ROMA

Prompt Chaining

To improve the reliability and performance of LLMs, one of the important prompt engineering techniques is **to break tasks into its subtasks**. Once those subtasks have been identified, the LLM is prompted with a subtask and then its response is used as input to another prompt.



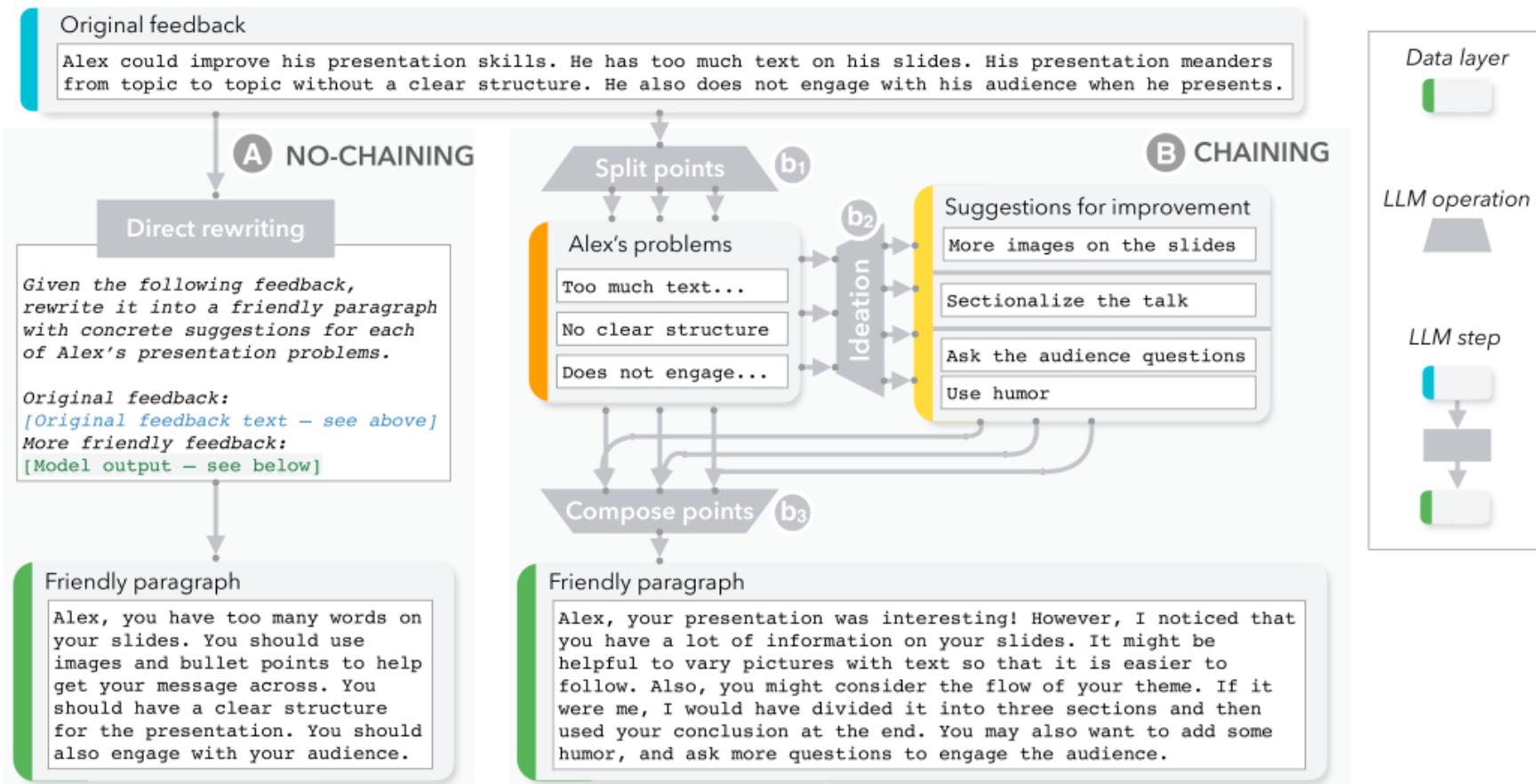
Source:

<https://aigenealogyinsights.com/2023/06/13/the-power-of-a-i-genealogical-prompt-chaining/>



SAPIENZA
UNIVERSITÀ DI ROMA

Prompt Chaining at Work



Source:

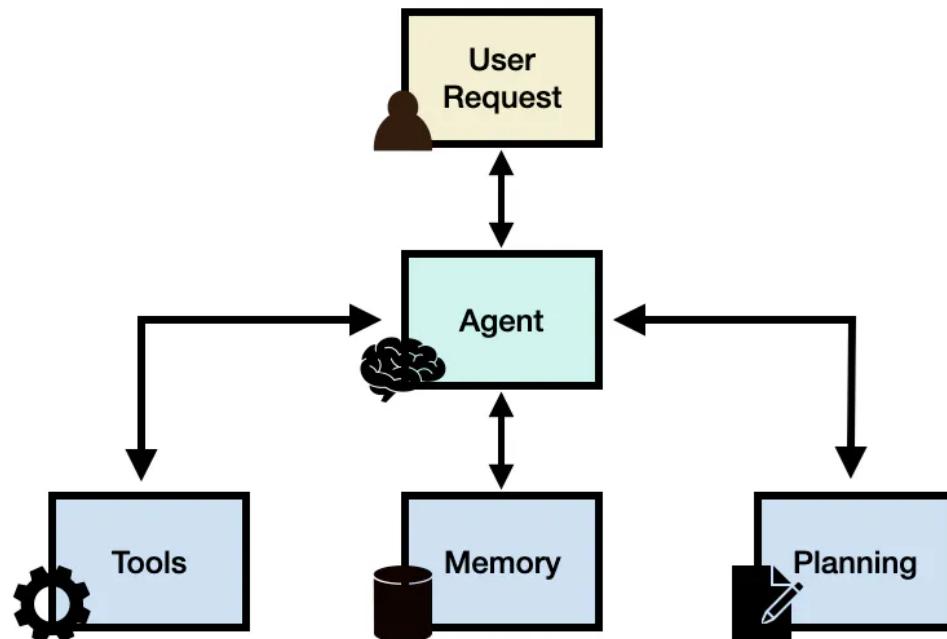
<https://cobusgreyling.medium.com/chaining-large-language-model-prompts-81091daccad>



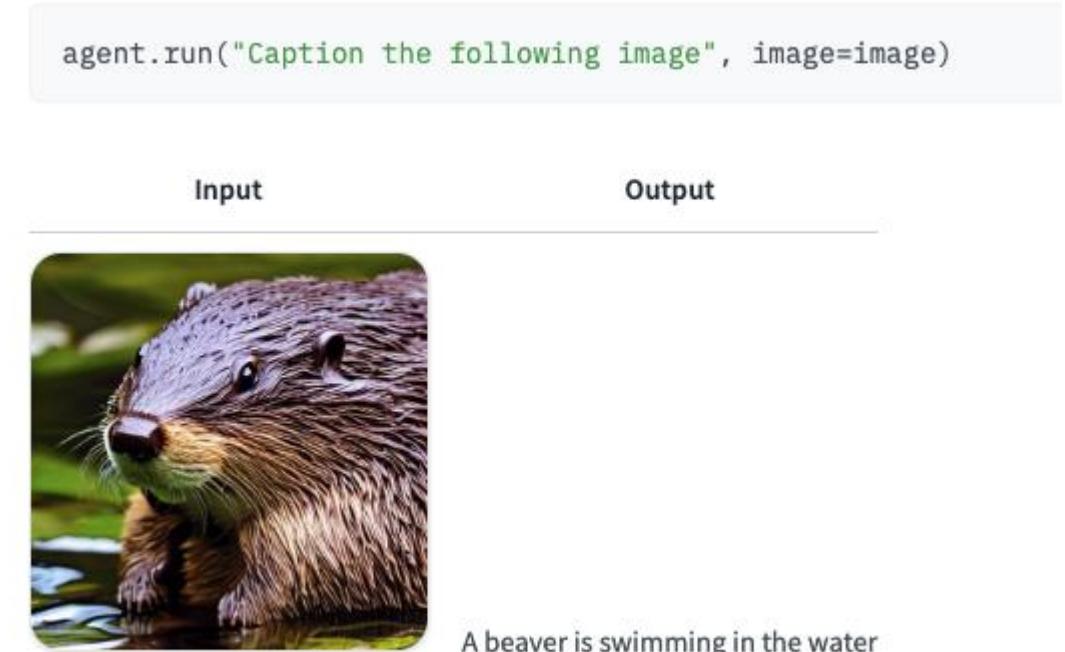
SAPIENZA
UNIVERSITÀ DI ROMA

LLM Agent

LLM agents involve LLM applications that can execute **complex tasks** using an architecture that combines LLMs with key modules like **planning and memory**. When building LLM agents, an LLM serves as the **main controller** or "brain" that controls a flow of operations needed to complete a task or user request

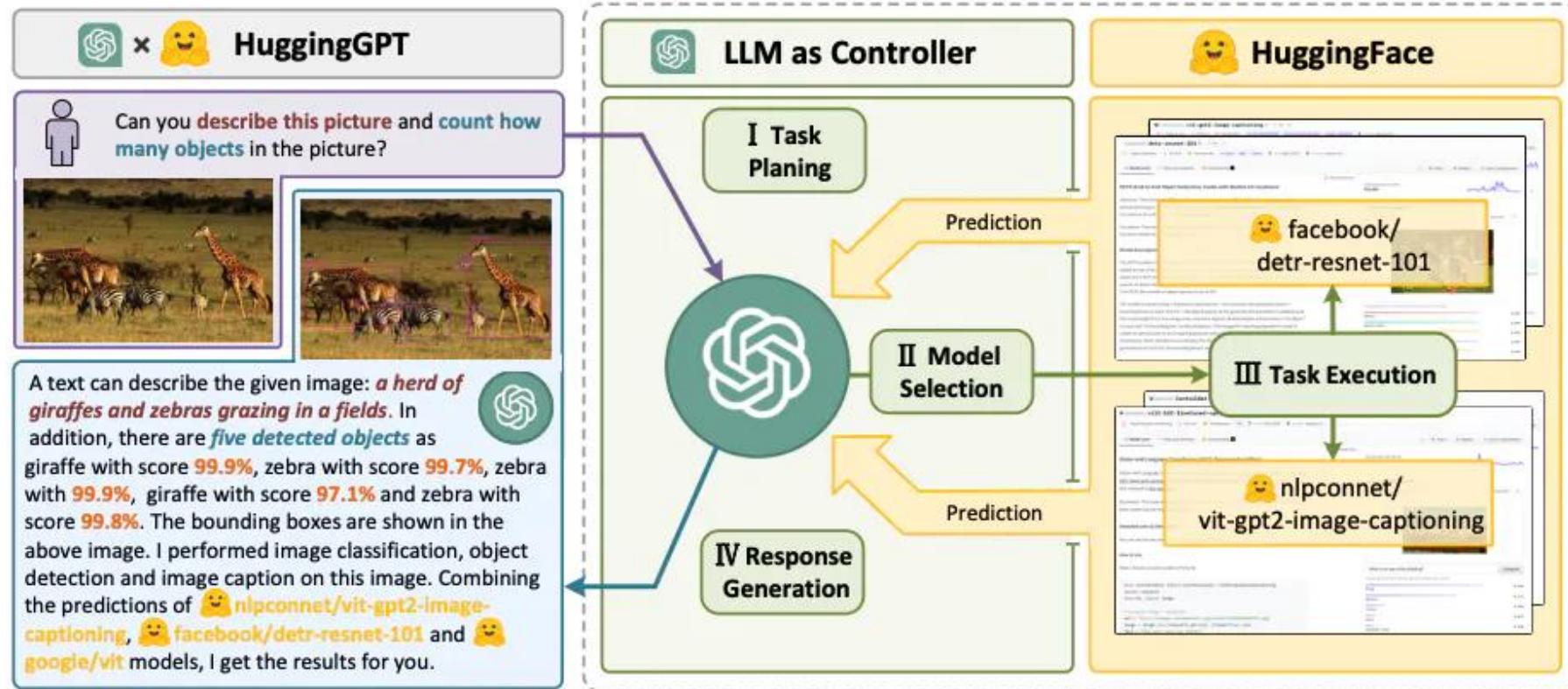


Source: <https://www.promptingguide.ai/research/llm-agents>



Source: https://huggingface.co/docs/transformers/transformers_agents

LLM Agent at Work



Source: <https://www.promptingguide.ai/research/llm-agents.en>



SAPIENZA
UNIVERSITÀ DI ROMA

Libraries for Development

Transformers (Fine-Tuning/Inference)

Is an open-source Python library for building models using the Transformer architecture, which is developed and maintained by Hugging Face. It is the *de-facto* reference for LLMs development, also thanks to Hugging Face HUB



Transformers

DeepSpeed (Pre-Training/Fine-Tuning/Inference)

Is a deep learning optimization library(compatible with PyTorch) developed by Microsoft. It provides the support of various optimization techniques for distributed training, such as memory optimization (ZeRO technique, gradient checkpointing), and pipeline parallelism. Well integrated with Transformers



deepspeed

Megatron-LM (Pre-Training/Fine-Tuning)

Is a deep learning library developed by NVIDIA for training large-scale language models. It also provides rich optimization techniques for distributed training, including model and data parallelism, mixed precision training, and FlashAttention. Well integrated with Transformers

NVIDIA/Megatron-LM



SAPIENZA
UNIVERSITÀ DI ROMA

Libraries for Development

JAX (Pre-Training/Fine-Tuning/Inference)

Is a Python library for high-performance machine learning algorithms developed by Google, allowing users to easily perform computations on arrays with hardware acceleration (e.g., GPU or TPU). It enables efficient computation on various devices and also supports several featured functions, such as automatic differentiation and just-in-time compilation.



Colossal-AI (Pre-Training/Fine-Tuning)

Is a deep learning library developed by HPC-AI Tech for training large-scale AI models. It is implemented based on PyTorch and supports a rich collection of parallel training strategies.



Langchain (Pre-Training/Fine-Tuning/Inference)

Is a framework for developing applications powered by large language models (LLMs). It is an integrated resource from the development to the deployment stage of an LLM. Supports natively third party tool, templates and RAG. Well integrated with Transformers



SAPIENZA
UNIVERSITÀ DI ROMA

Libraries specific for local Inference/Quantization

Llama.cpp (GGUF Backend)



Enables LLM inference with minimal setup and state-of-the-art performance on a wide variety of hardware - locally and in the cloud. It uses the [gguf](#) format and is optimized for faster cpu inference

GPTQ (Backend)

GPTQ is a one-shot weight quantization method based on approximate second-order information, optimized for GPU inference

Activation Aware Quantization (AWQ Backend)

AWQ is an activation-aware weight quantization approach developed by the MIT-HAN lab. It protects salient weights by observing activations rather than the weights themselves

LM-Studio/GPT4all (GGUF Front End)

Graphical front ends for LLama.cpp library



Source: <https://huggingface.co/docs/hub/en/gguf>



SAPIENZA
UNIVERSITÀ DI ROMA

OpenAI API Parameters

messages

A list of messages comprising the conversation so far.

model

ID of the model to use. See the model endpoint compatibility table for details on which models work with the Chat API.

frequency_penalty

Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim.

logit_bias map

Accepts a JSON object that maps tokens (specified by their token ID in the tokenizer) to an associated bias value from -100 to 100. Mathematically, the bias is added to the logits generated by the model prior to sampling. The exact effect will vary per model, but values between -1 and 1 should decrease or increase likelihood of selection; values like -100 or 100 should result in a ban or exclusive selection of the relevant token.

logprobs

Whether to return log probabilities of the output tokens or not. If true, returns the log probabilities of each output token returned in the content of message.

top_logprobs

An integer between 0 and 20 specifying the number of most likely tokens to return at each token position, each with an associated log probability. logprobs must be set to true if this parameter is used.

max_tokens

The maximum number of tokens that can be generated in the chat completion.

n

How many chat completion choices to generate for each input message. Note that you will be charged based on the number of generated tokens across all of the choices. Keep n as 1 to minimize costs.

presence_penalty

Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics.

seed

This feature is in Beta. If specified, our system will make a best effort to sample deterministically, such that repeated requests with the same seed and parameters should return the same result. Determinism is not guaranteed, and you should refer to the system_fingerprint response parameter to monitor changes in the backend.

temperature

The sampling temperature, between 0 and 1. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic. If set to 0, the model will use log probability to automatically increase the temperature until certain thresholds are hit.



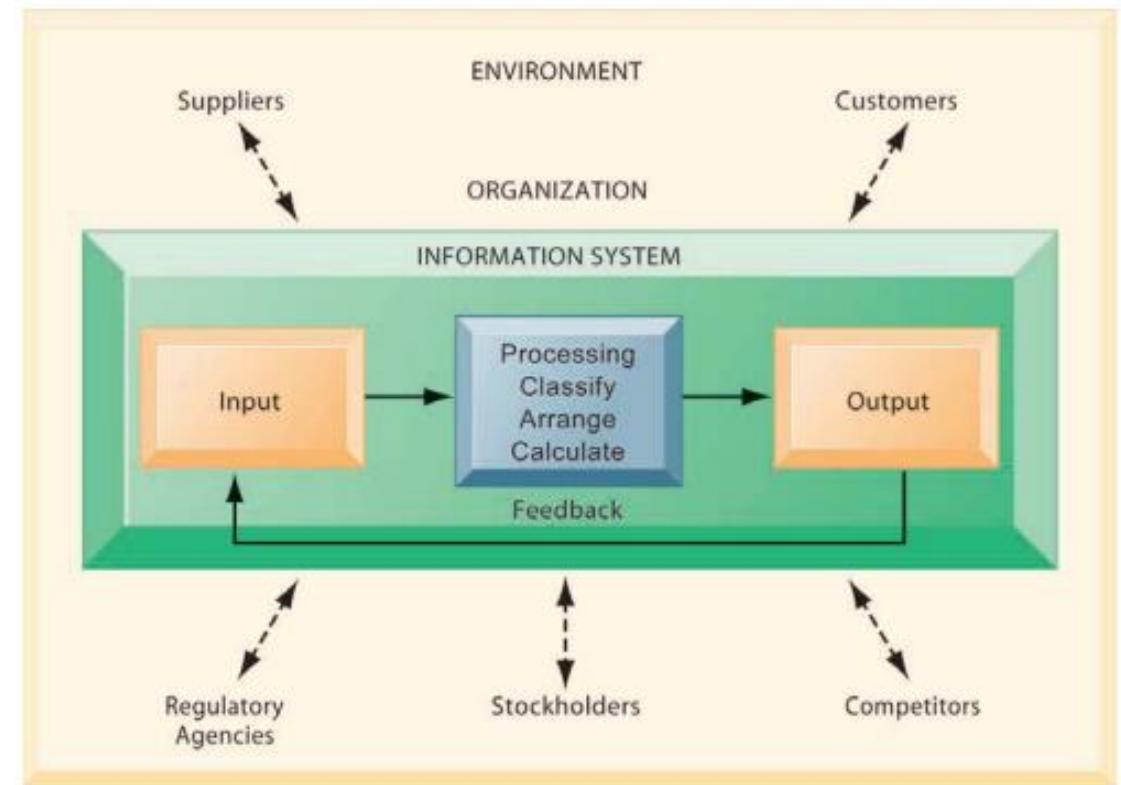
Once upon a time ...

The traditional methods for IS engineering

Classical view of an IS

An information system contains information about an organization and its surrounding environment

- Three basic activities—input, processing, and output—produce the information organizations need
- Feedback is output returned to appropriate people or activities in the organization to evaluate and refine the input.
- Environmental actors, such as customers, suppliers, competitors, stockholders, and regulatory agencies, interact with the organization and its information systems.

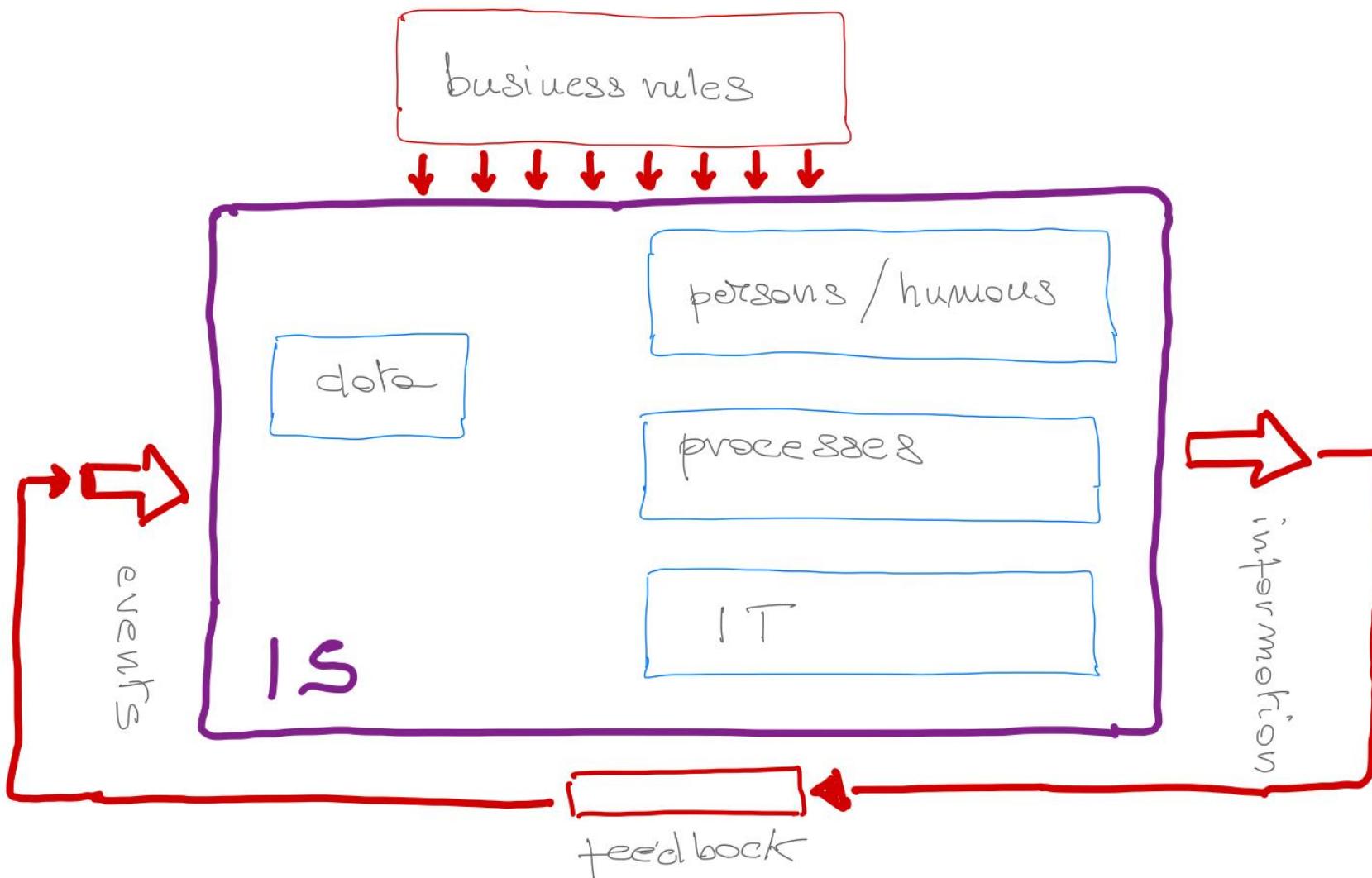


Source: Laudon, K., & Laudon, J. (2009). Management Information Systems: International Edition, 11/E. London, UK: Pearson Higher Education.



SAPIENZA
UNIVERSITÀ DI ROMA

System-view of an IS



Information System Lifecycle

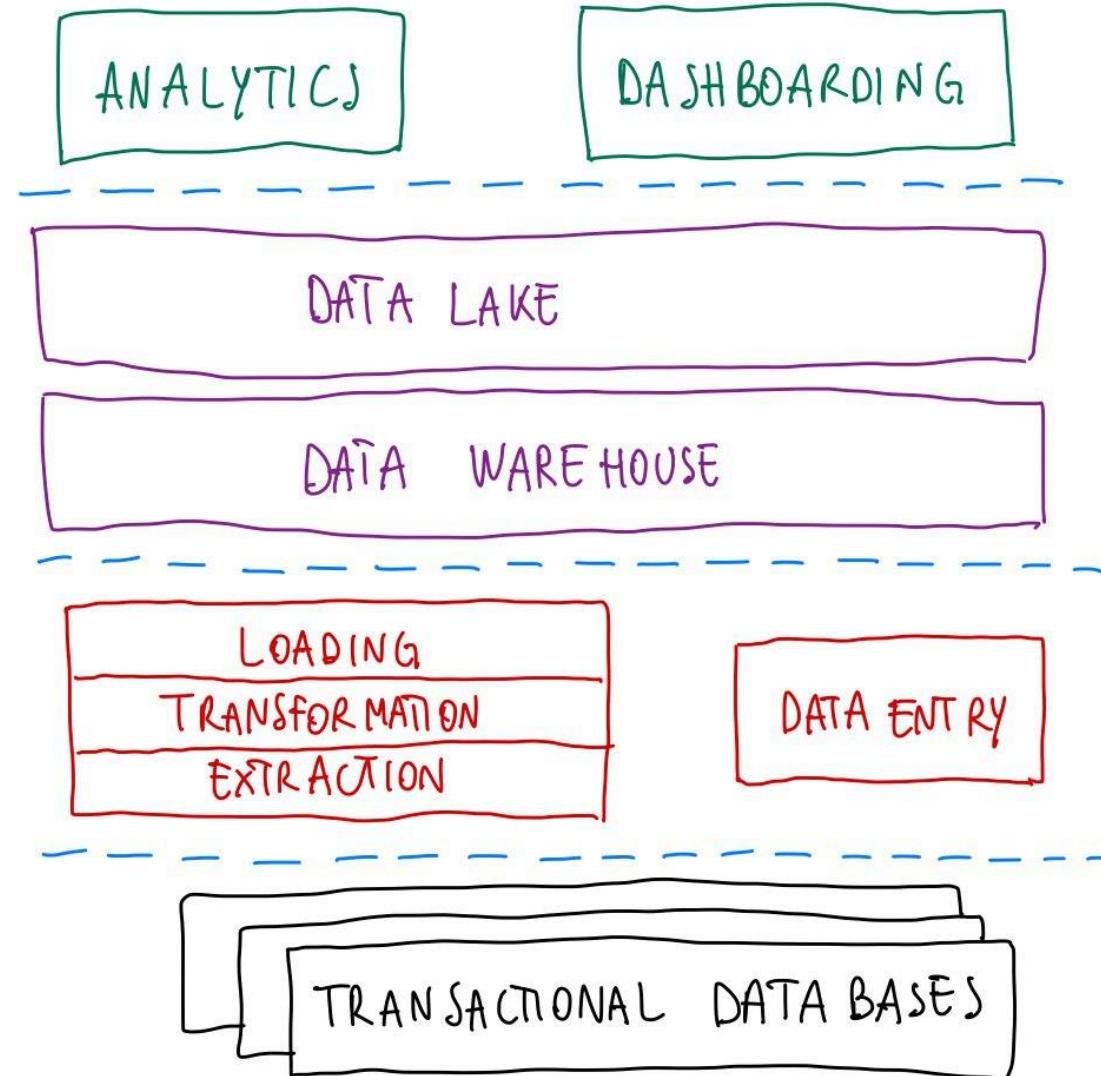


Source: <https://www.smartsheet.com/system-development-life-cycle-guide>



SAPIENZA
UNIVERSITÀ DI ROMA

The information & knowledge management layer



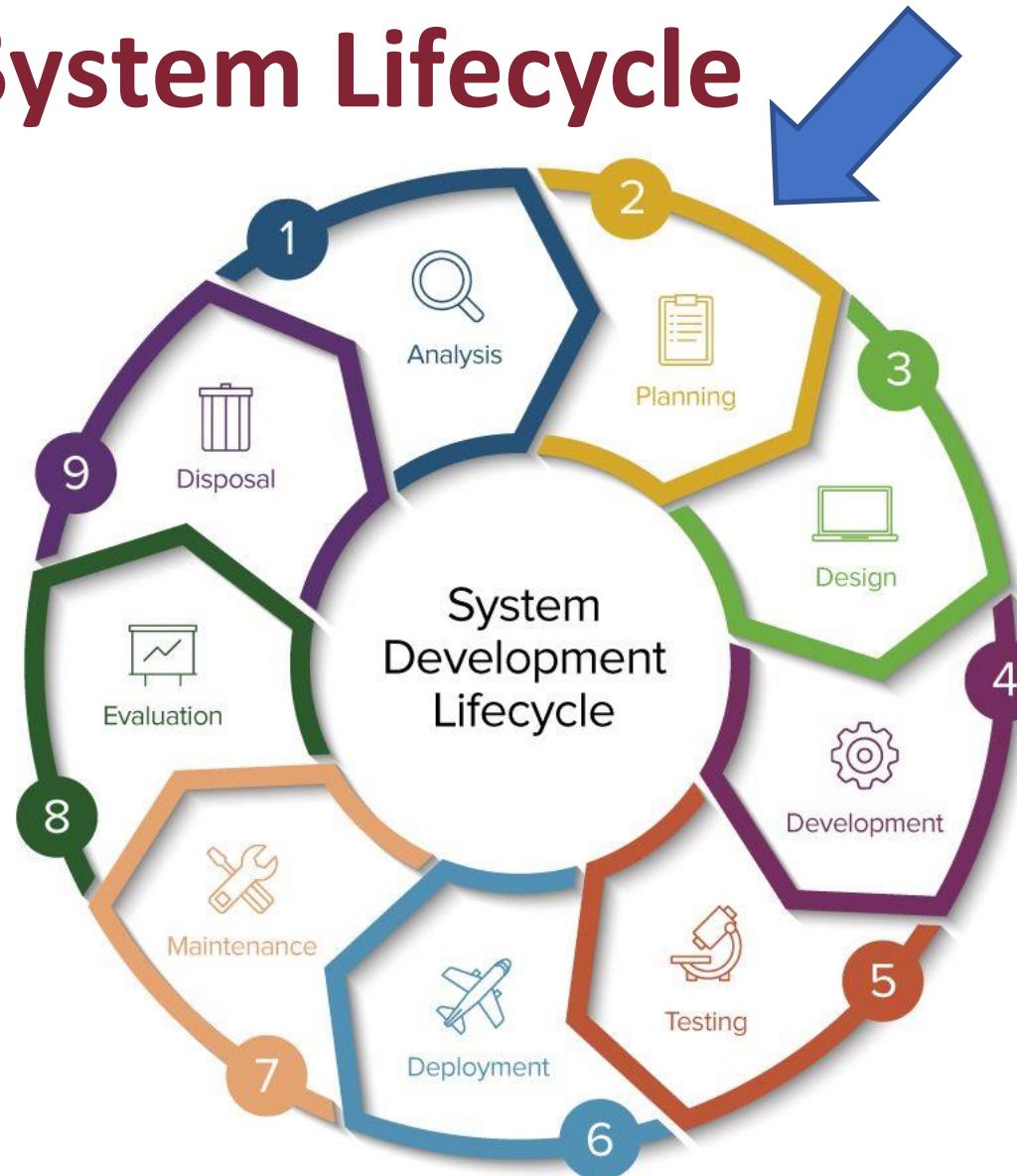
Surveying LLMs in IS engineering phases

What has been done in the last few months ... a paper about LLMs is published every 8 minutes



SAPIENZA
UNIVERSITÀ DI ROMA

Information System Lifecycle



Source: <https://www.smartsheet.com/system-development-life-cycle-guide>



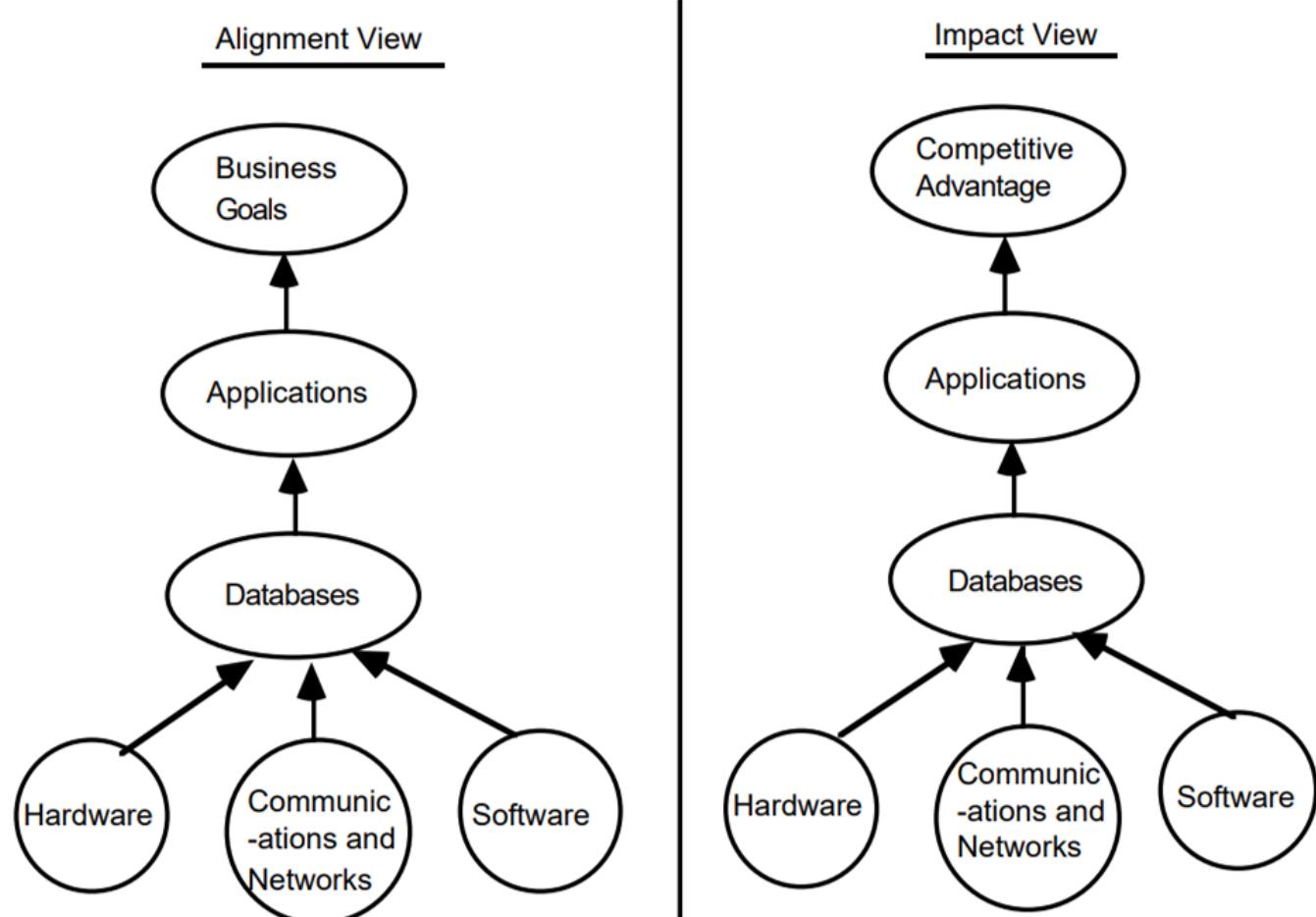
SAPIENZA
UNIVERSITÀ DI ROMA

Strategic IS Planning

Two categories of methodologies

An information system contains information about an organization and its surrounding environment

- Impact view: these methodologies help the organization to create and justify new uses of IT
- Alignment view: the aim of these methodologies is aligning IS objectives with organizational goals

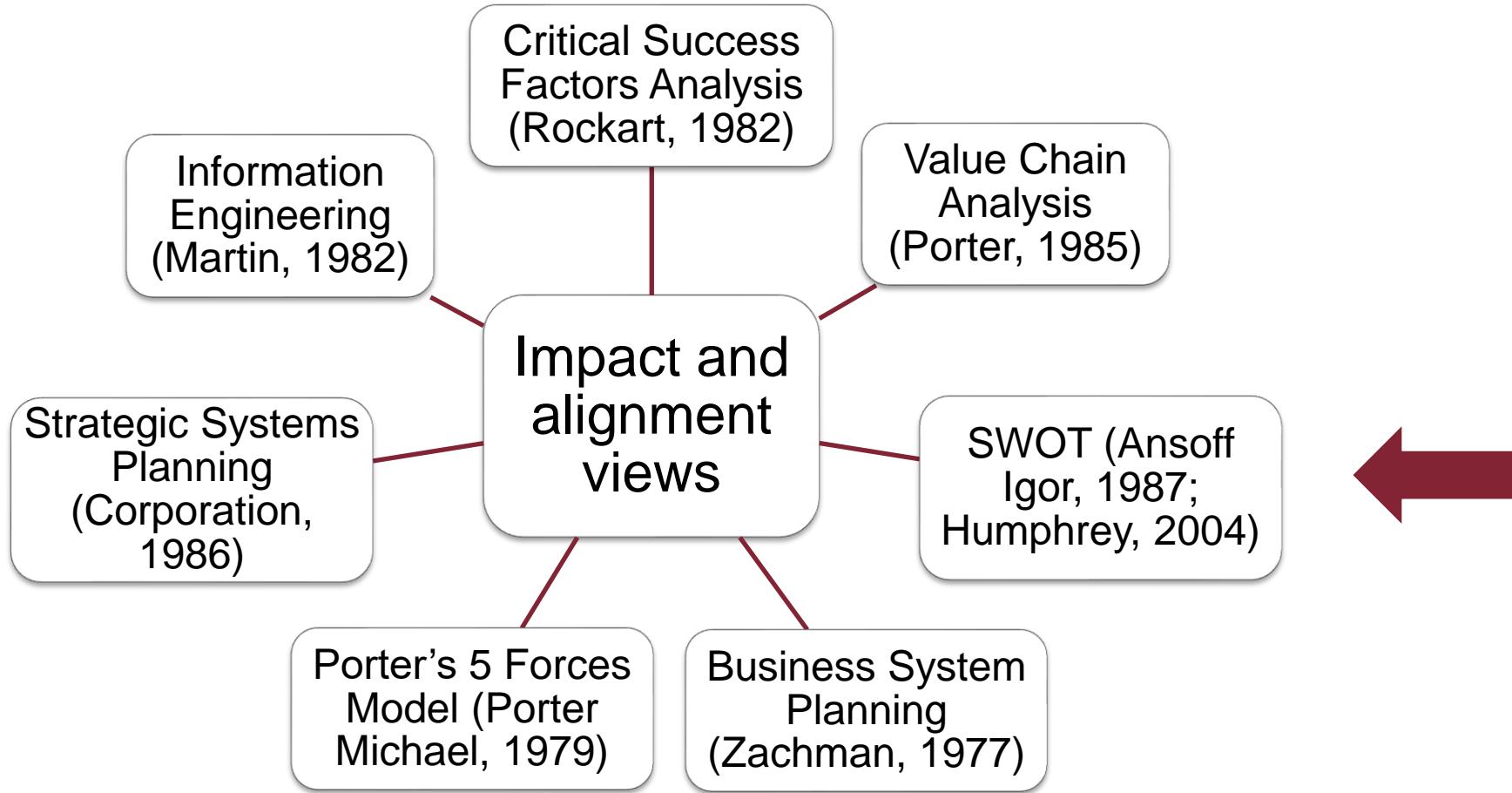


Source: Vitale, M., Ives, B. and Beath, C., "Identifying Strategic Information Systems," Proc. 7th Int'l Conf. Inf. Sys., San Diego, December 1986, pp. 265-276.



SAPIENZA
UNIVERSITÀ DI ROMA

SISP Methodologies



- Automated SWOT Analysis Using Massive Language Models
- AI-generated SWOT Analysis

Source: Pant, S., & Hsu, C. (1995, May). Strategic information systems planning: a review. In Information Resources Management Association International Conference

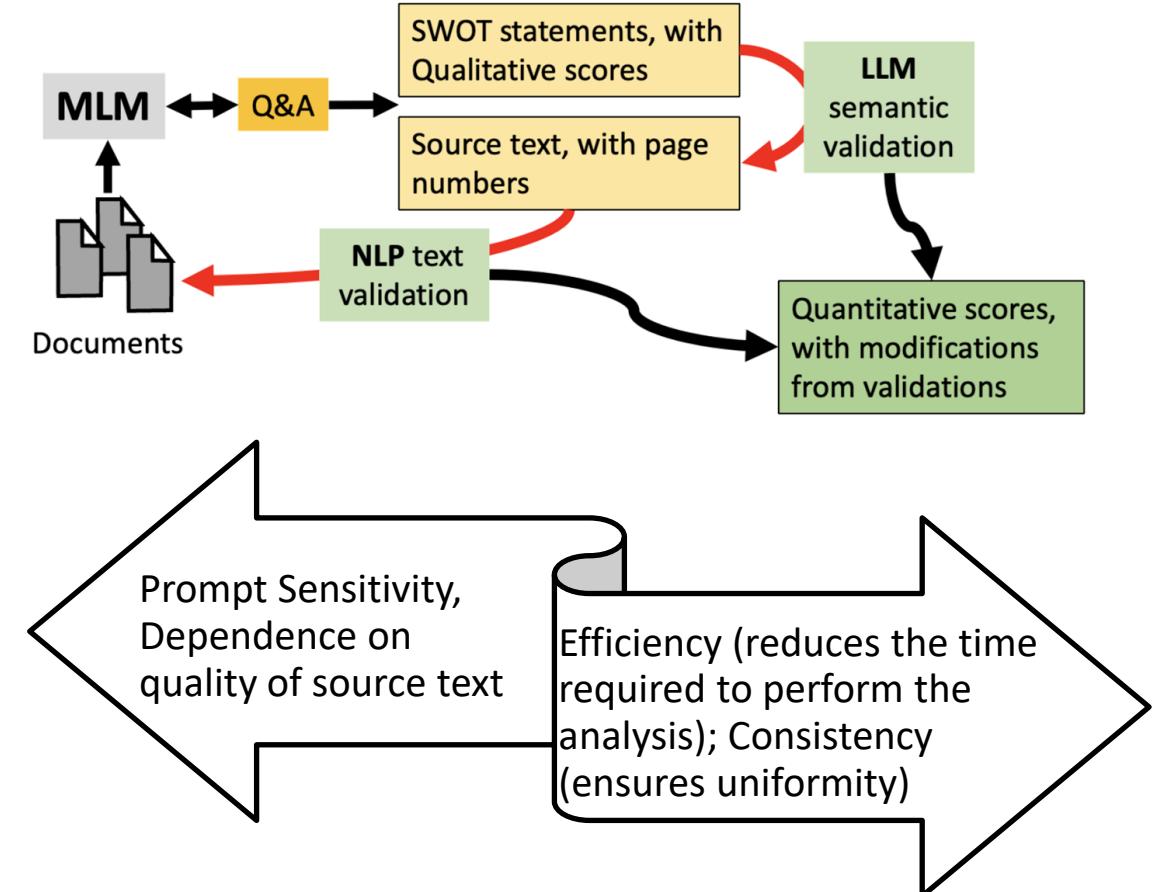


SAPIENZA
UNIVERSITÀ DI ROMA

Automated SWOT Analysis Using Massive Language Models

NLP techniques and MLMs to automate SWOT (Strengths, Weaknesses, Opportunities, Threats) analysis

- Data Extraction: Utilize MLMs to extract SWOT elements from annual reports
- Validation: Validate extracted information using LLMs to ensure accuracy and reduce hallucinations
- Scoring: Implement prompt engineering for precise SWOT scoring, supported by quotes from the source text
- Scalability: Analyze large datasets, such as the S&P500, efficiently within minutes, providing a robust foundation for strategic portfolio construction



AI-generated SWOT Analysis

SWOT analysis is well-suited for automated generation by AI rather than being manually created by humans.

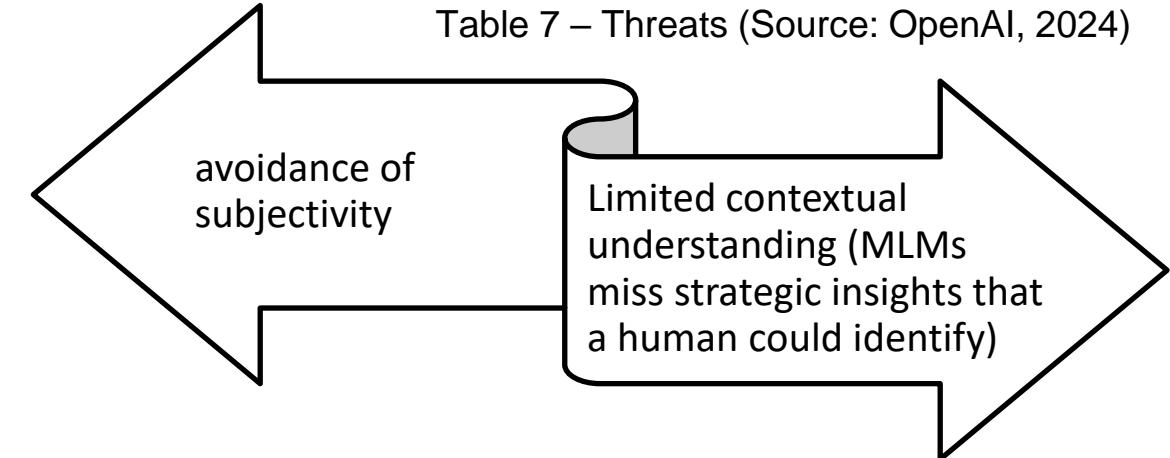
- The present study employs the conversational AI language model ChatGPT

Three drawbacks associated with manually conducted SWOT analyses:

- Inadequate definition of factors
- lack of prioritization of factors
- over-subjectivity in the generation of factors

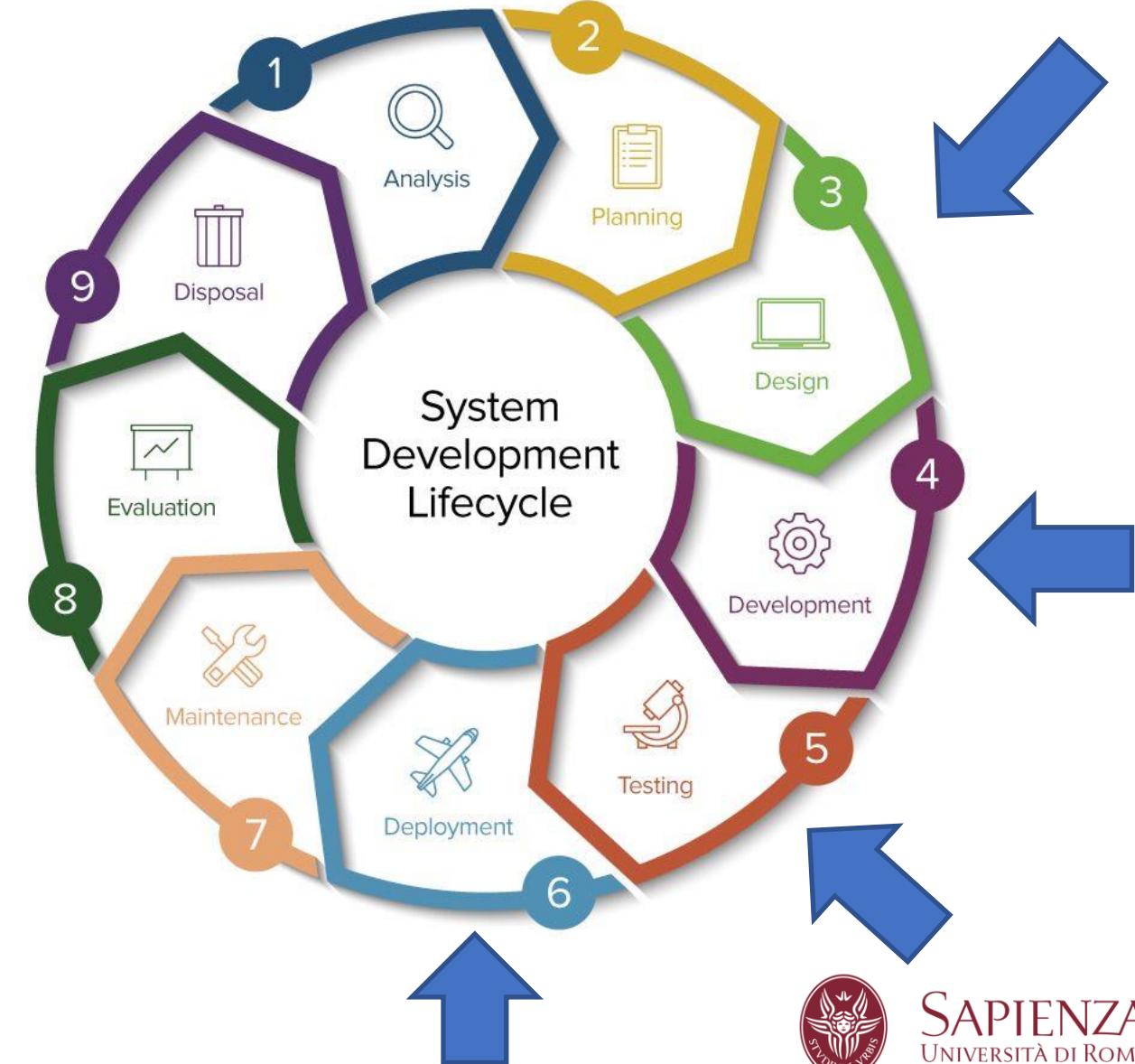
Urban air mobility (UAM)	Unmanned aerial vehicles (UAVs)	Supersonic travel	Electric aircraft	Sustainable aviation fuels (SAFs)
Safety Concerns: Accidents or incidents may lead to public distrust and regulatory setbacks.	Regulatory Restrictions: Stringent regulations limiting operations.	Regulatory Challenges: Stringent regulations affecting adoption.	Regulatory Challenges: Complexities in safety standards and certification processes.	Fluctuating Oil Prices: Volatility impacts economic viability.
Competing Technologies: Competition with other advanced transportation technologies.	Security Concerns: Misuse for malicious purposes and attacks.	Competitive Landscape: Competition from other transportation modes.	Competition from Conventional Aircraft: Advances in traditional aviation technology.	Competing Technologies: Emerging alternatives like electric aviation.
Airspace Congestion: Increased usage may lead to airspace congestion.	Competitive Landscape: Intense competition affecting profitability.	Economic Downturn: Impact of economic uncertainties on premium travel demand.	Material Dependency: Reliance on critical materials subject to supply chain issues.	Economic Conditions: Downturns may reduce investments in SAFs.
Security Risks: Vulnerability to cybersecurity threats, including hacking.	Technological Risks: Rapid advancements requiring constant innovation.	Public Perception: Negative views on noise, safety, or environmental concerns.	Public Perception: Building trust in safety and reliability.	Policy Uncertainty: Changes in regulations impact industry growth.
Economic Viability: Dependent on pricing, operational costs, and consumer adoption.	Public Resistance: Concerns about privacy, safety, and job displacement.	Technological Risks: Unforeseen challenges during development and operations.	Economic Factors: Downturns impact industry investment in new technologies.	Consumer Perception: Safety concerns may hinder widespread adoption.

Table 7 – Threats (Source: OpenAI, 2024)



Design of an IS

- **Conceptual modeling**
- Software architecture design
- Software development
- Software and system testing



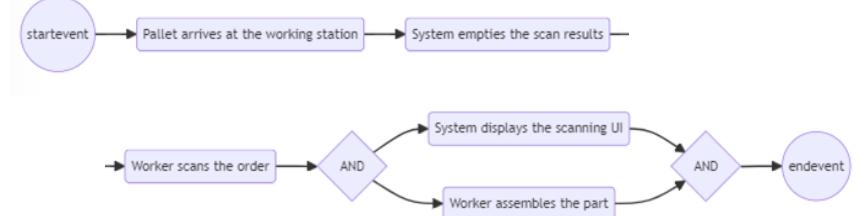
Process Modeling

- Mermaid.js model generation from textual process description
 - Prompt engineering
 - description of the task to be performed
 - type and format of the desired output
 - The model is not executable

When a pallet arrives at the working station, the system empties the scan results. Then the worker scans the order. Afterwards the system displays the scanning UI to the worker and in parallel, the worker assembles the part.



```
graph LR
1:startevent((startevent)) --> 2:task:(Pallet arrives at the working station)
2:task: --> 3:task:(System empties the scan results)
3:task: --> 4:task:(Worker scans the order)
4:task: --> 5:parallelgateway:{AND}
5:parallelgateway: --> 6:task:(System displays the scanning UI)
5:parallelgateway: --> 7:task:(Worker assembles the part)
6:task: --> 8:parallelgateway:{AND}
7:task: --> 8:parallelgateway:{AND}
8:parallelgateway: --> 9:endeevent((endeevent))
```



Process Modeling

- BPMN generation from textual process description
 - Python code that generates POWL converted in BPMN/Petri nets
 - Self-refinement to fix code from errors
 - Prompt engineering
 - Role
 - Knowledge injection about POWL
 - Few-shot learning
 - Negative prompting (common mistakes to be avoided)
 - The model is not executable



Process Model Generator with Generative AI

Enter the OpenAI model name: Enter your OpenAI API key:

Enter the process description:

When a pallet arrives at the working station, the system empties the scan results. Then the worker scans the order. Afterwards the system displays the scanning UI to the worker and in parallel, the worker assembles the part.

Model generated successfully!

Feedback:

Export Model

Select a view:
BPMN

View Image

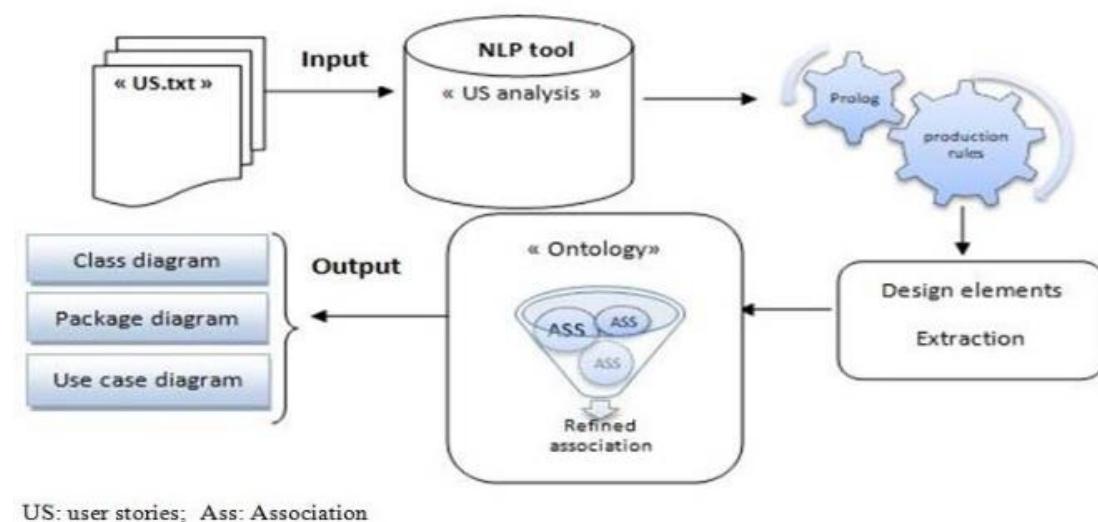
```
graph LR; Start(( )) --> A[Empty scan results]; A --> B[Scan the order]; B --> C[Display scanning UI]; C --> D[Assemble part]; D --> End(( ));
```



SAPIENZA
UNIVERSITÀ DI ROMA

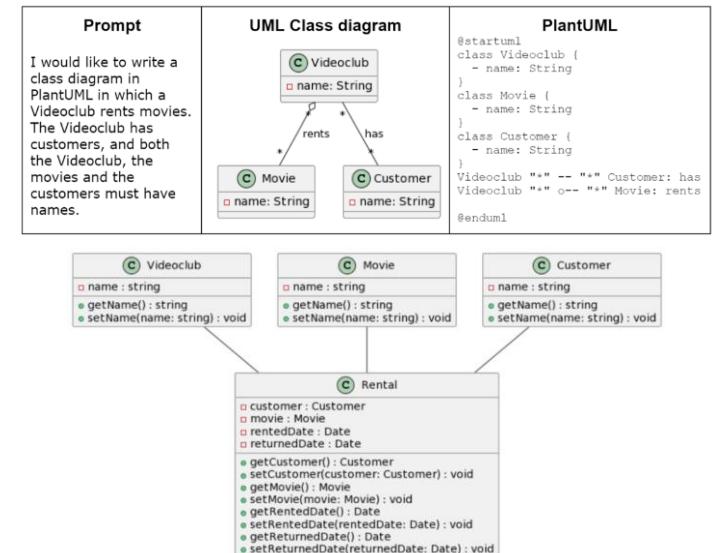
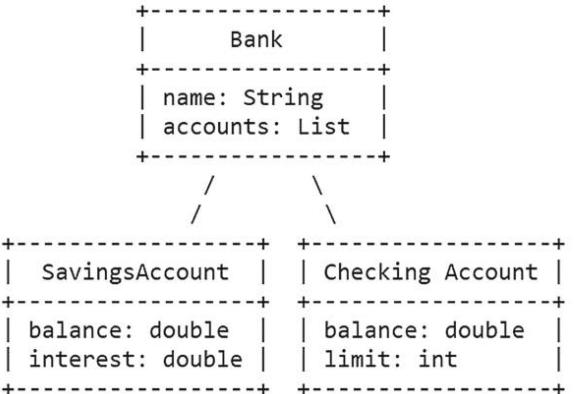
System Modeling

- From user stories to UML diagrams
 - User stories → analysis → prolog and production rules → ontology → class diagram, package diagram, use case diagram
 - Relies on
 - Prolog rules
 - Based on Stanford NLP tool
 - Ontology



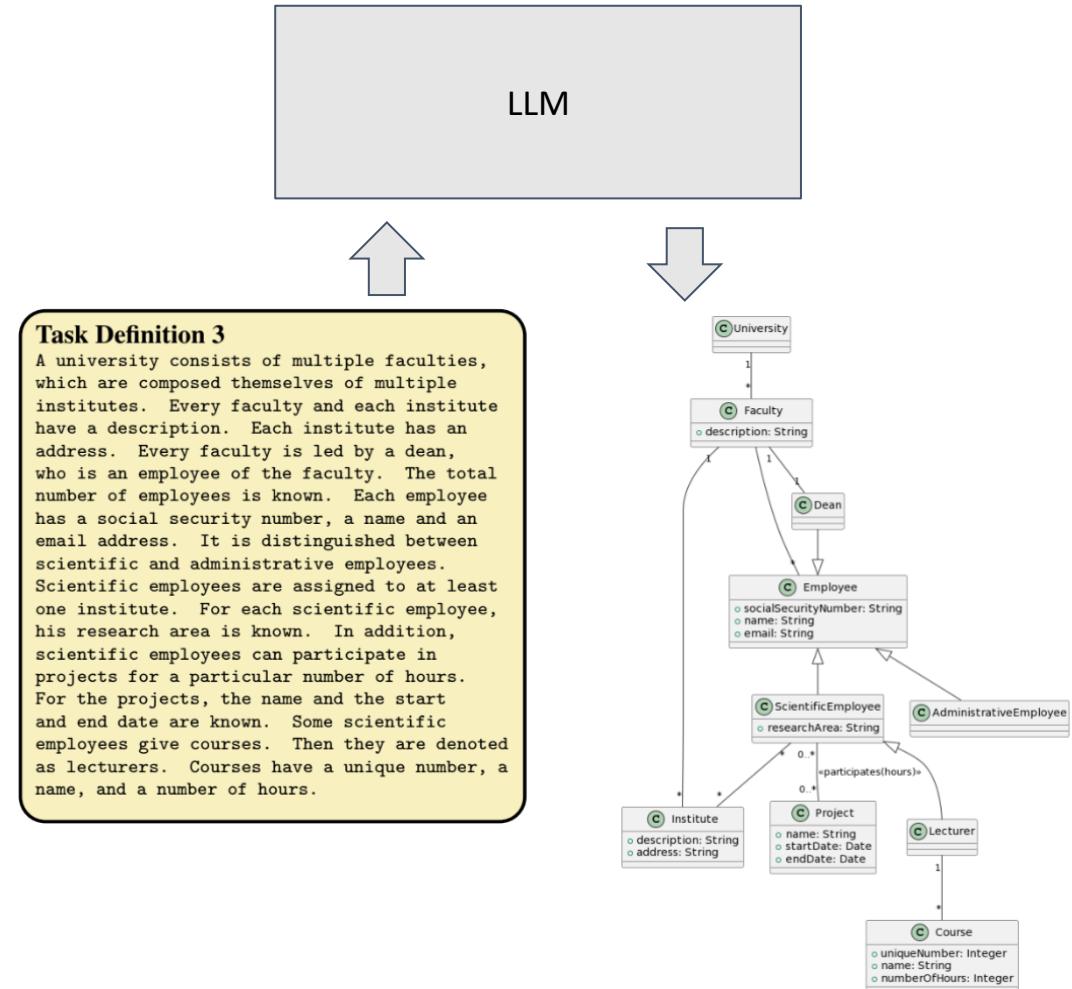
System Modeling

- Conversational experimental analysis of ChatGPT and PlantUML
 - Responses
 - Plain text
 - PlantUML
 - They produce different results repeating the experiments in different sessions
 - Problems in defining classes associations
 - In contrast to code generation and completion, the performance of the current version of ChatGPT for software modeling is still quite limited



System Modeling

- Experimental analysis using ChatGPT
 - Prompt engineering
 - Explanation of ER diagrams
 - Range of possible cardinalities
 - Output format
 - Few-shot
 - Generation of UML
 - PlantUML format
 - Despite some impressive results that are fully correct, it still requires experience in modeling to validate the results



System Modeling (other works)

- Translation of images to UMLs
 - From Image to UML: First Results of Image Based Class Diagram Generation Using LLMs [1]
 - Img2UML: A system for extracting UML models from images [2]
- Classification of contents in images with UML diagrams
 - Automatically recognizing the semantic elements from UML class diagram images [3]

[1] <https://modeling-languages.com/image-to-uml-with-lm/>

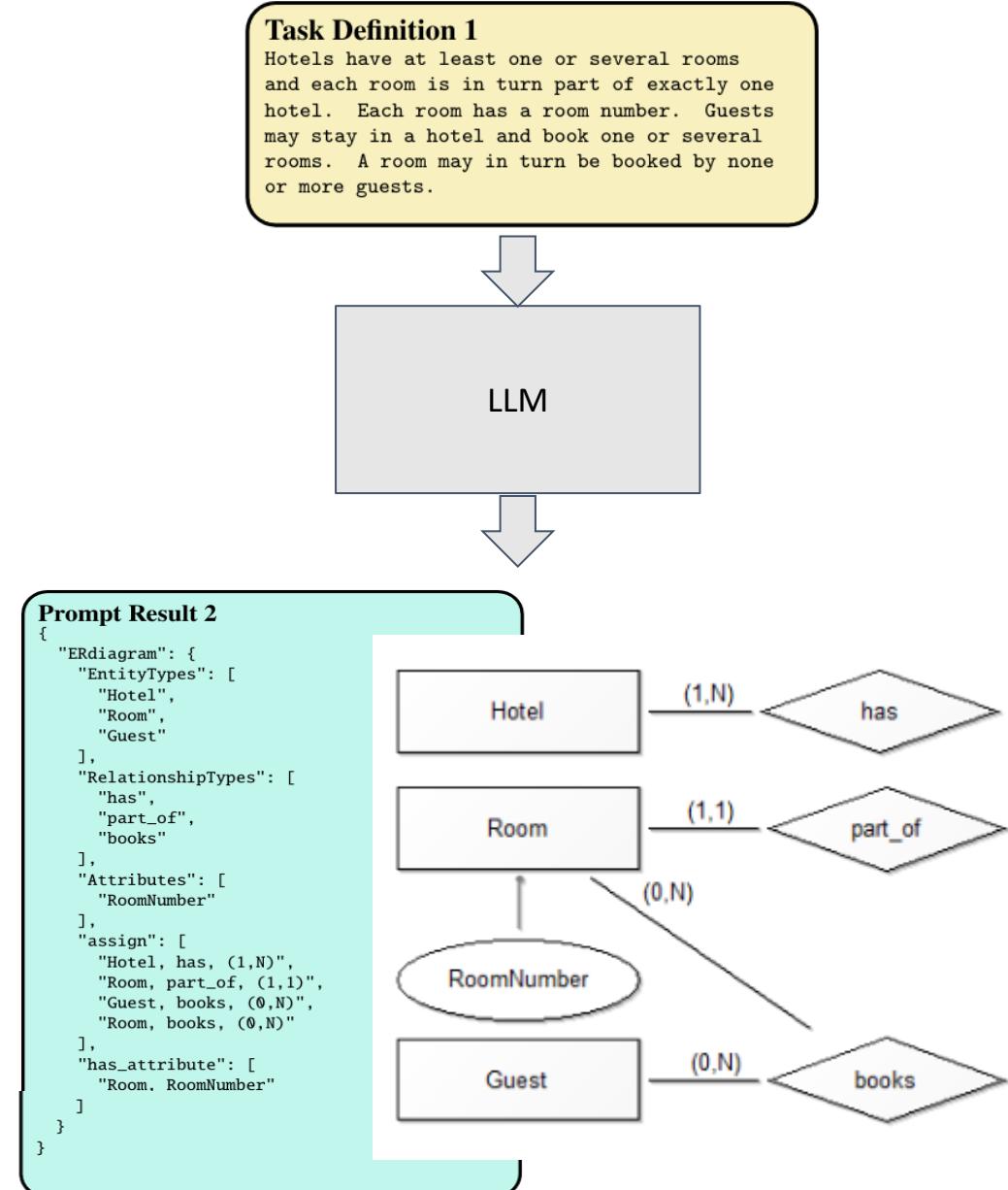
[2] https://www.researchgate.net/publication/261085408_Img2UML_A_system_for_extracting_UML_models_from_images

[3] <https://www.sciencedirect.com/science/article/pii/S0164121222001340>



ER Diagrams

- Experimental analysis using ChatGPT
 - Prompt engineering
 - Explanation of ER diagrams
 - Range of possible cardinalities
 - Output format
 - Few-shot
 - Despite some impressive results that are fully correct, it still requires experience in modeling to validate the results



Design of an IS

- Conceptual modeling
- **Software architecture design**
- **Software development**
- Software and system testing

Prompt Engineering Techniques in Code Dev

ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design

- Collection of simple prompt patterns to easily improve response quality in Software Engineering related tasks

Requirements Elicitation	Requirements Simulator Specification Disambiguation Change Request Simulation
System Design and Simulation	API Generator API Simulator Few-shot Example Generator Domain-Specific Language (DSL) Creation Architectural Possibilities
Code Quality	Code Clustering Intermediate Abstraction Principled Code Hidden Assumptions
Refactoring	Pseudo-code Refactoring Data-guided Refactoring

API Generator Pattern
1. Using system description X
2. Generate an API specification for the system
3. The API specification should be in format Y
Data-guided Refactoring Pattern
1. Refactor the code
2. So that its input, output, or stored data format is X
3. Provide one or more examples of X



Prompt Engineering Techniques in Code Dev

Chain of Thoughts (CoT)

- Creates an intermediate pseudo-code from the task description, then transforms it in real code

Self-Planning Code Generation

- Based on CoT
- Instead of using pseudo-code as an intermediate step, generates a list of more generic instructions
- Each instruction corresponds to a sub-task

Structured Chain of Thought

- Categorizes pieces of code into loops, branches and sequences
- Uses these structures to generate a less discursive pseudo-code

```
CoT & Code  
def prime_fib(n: int):  
    Define a function to check if a number is prime.  
    def is_prime(n: int):  
        Check if the number is divisible by any number between 2 and n-1.  
        if n < 2:  
            return False  
        Check if the number is divisible by any number between 2 and n-1.  
        for i in range(2, n):  
            if n % i == 0:  
                return False  
        If the number is not divisible by any number between 2 and n-1, it's prime.  
        return True  
  
Initialize a list "fib_seq" with the first two Fibonacci numbers 1, 1.  
fib_seq = [1, 1]  
Initialize a counter "counter" with the value of n.  
counter = n  
While the counter is greater than 0, extend the list "fib_seq" with the  
next Fibonacci number.  
while counter > 0:  
    fib_seq.append(fib_seq[-1] + fib_seq[-2])  
Check if the last number in the list "fib_seq" is prime.  
If it's prime, decrement the counter.  
    if is_prime(fib_seq[-1]):  
        counter -= 1  
Return the last number in the list "fib_seq".  
return fib_seq[-1]
```

An example of CoT pseudo-code and python code



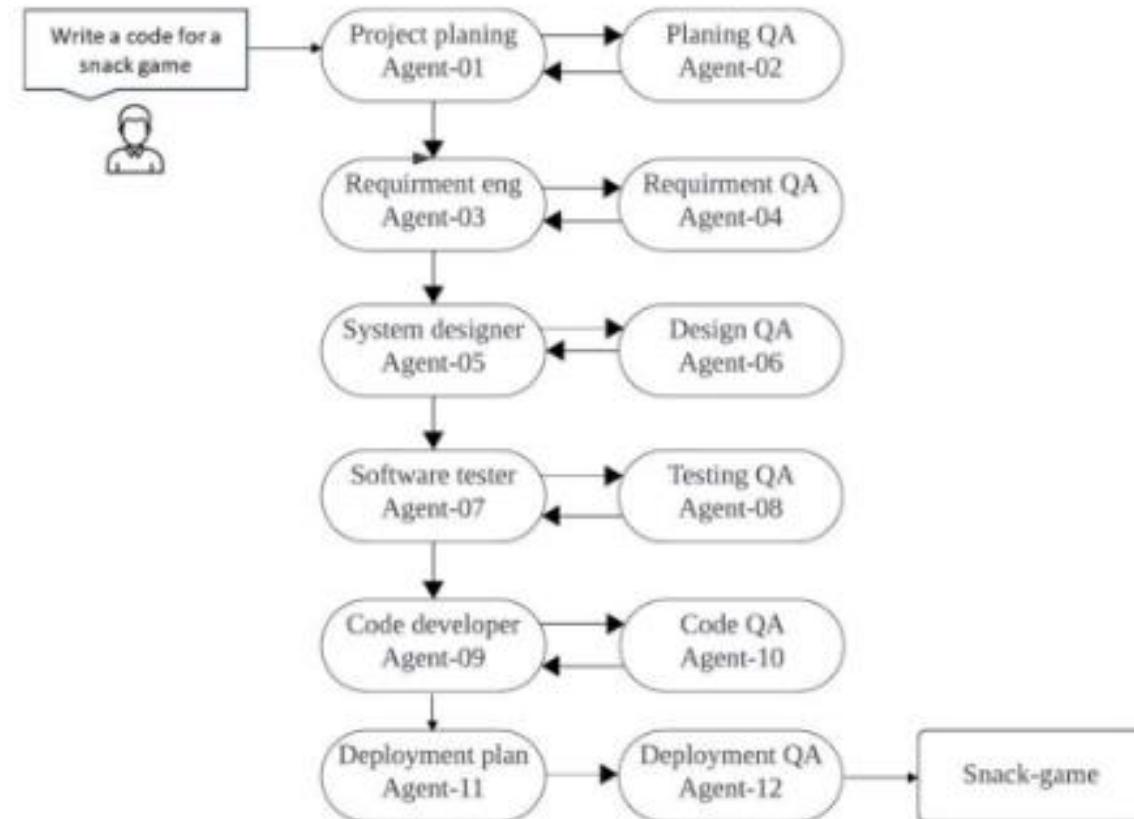
Prompt Engineering Techniques in Code Dev

Self collaboration Code Generation

- Multiple LLMs act as distinct “experts”, each responsible for a specific subtask within a complex task
- Three different roles for code generation:
 - analyst
 - programmer
 - tester

Autonomous Agents in Software Development

- Multiple LLMs assume different roles based on the specific SW lifecycle task
- Each role is covered by two LLMs, the one in charge of the product generation, the other acts as a “senior” and answers field-related questions



Example of an autonomous agents pipeline



Code Evaluation

SPoC: Search-based Pseudocode to Code

- k code samples are generated per problem
- A problem is considered solved if any sample passes the unit tests, and the total fraction of problems solved is reported

Autonomous Agents in Software Development (HumanEval)

- Evaluates functional correctness on a set of 164 handwritten programming problems
- Each problem includes a function signature, docstring, body, and several unit tests, with an average of 7.7 tests per problem
- Generates $n \geq k$ samples per task, counts the number of correct samples $c \leq n$ which pass unit tests, and calculate the unbiased estimator



Code Evaluation

ClassEval: A Manually-Crafted Benchmark for Evaluating LLMs on Class-level Code Generation

- Defines a Class Skeleton, the blueprint of a class in which each subsequence must have the correct structure and contents
- Correctness of the solution is based on the presence and correctness of each subsequence (class constructor, method signatures...)

from datetime import datetime	Import Statements
class VendingMachine:	Class Name
"""This is a class to simulate a vending machine, including adding products, inserting coins, purchasing products, viewing balance, replenishing product inventory, and displaying product information. """	Class Description
def __init__(self):	Class Constructor
"""	
Initializes the vending machine's inventory and balance.	
"""	
self.inventory= []	Method Signature
self.balance= {}	Functional Description
def purchase_item(self, item_name):	Parameter/Return
""" Purchases a product from the vending machine and returns the balance after the purchase.	Description
:param item_name: str, the name of the product to be purchased, which should be in the vending machine.	Example Input/Output
:return: If successful, returns the balance of the vending machine after the product is purchased, float, if the product is out of stock, returns False.	Method Signature
>>> vendingMachine.inventory = {'Coke': {'price': 1.25, 'quantity': 10}}	Functional Description
>>> vendingMachine.balance = 1.25	
>>> vendingMachine.purchase_item('Coke')	
0.0	
>>> vendingMachine.inventory	
{'Coke': {'price': 1.25, 'quantity': 9}} """	
def restock_item(self, item_name, quantity):	Parameter/Return Description
"""	Description
Replenishes the inventory of a product already in the vending machine.	Example Input/Output
:param item_name: The name of the product to be replenished, str, which should be in the vending machine.	Method Signature
:param quantity: The quantity of the product to be replenished, int, which is greater than 0.	Functional Description
:return: If the product is already in the vending machine, returns True, otherwise, returns False.	
>>> vendingMachine.inventory = {'Coke': {'price': 1.25, 'quantity': 10}}	
>>> vendingMachine.restock_item('Coke', 10)	
True	
>>> vendingMachine.inventory	
{'Coke': {'price': 1.25, 'quantity': 20}} """	
...	Example Input/Output



Design of an IS

- Conceptual modeling
- Software architecture design
- Software development
- **Software and system testing**

LLMs applied to software testing

LLMs are successfully applied in a wide range of testing task:

- Program repair
- Unit test
- System test
- Bug analysis
- Debug

Source:

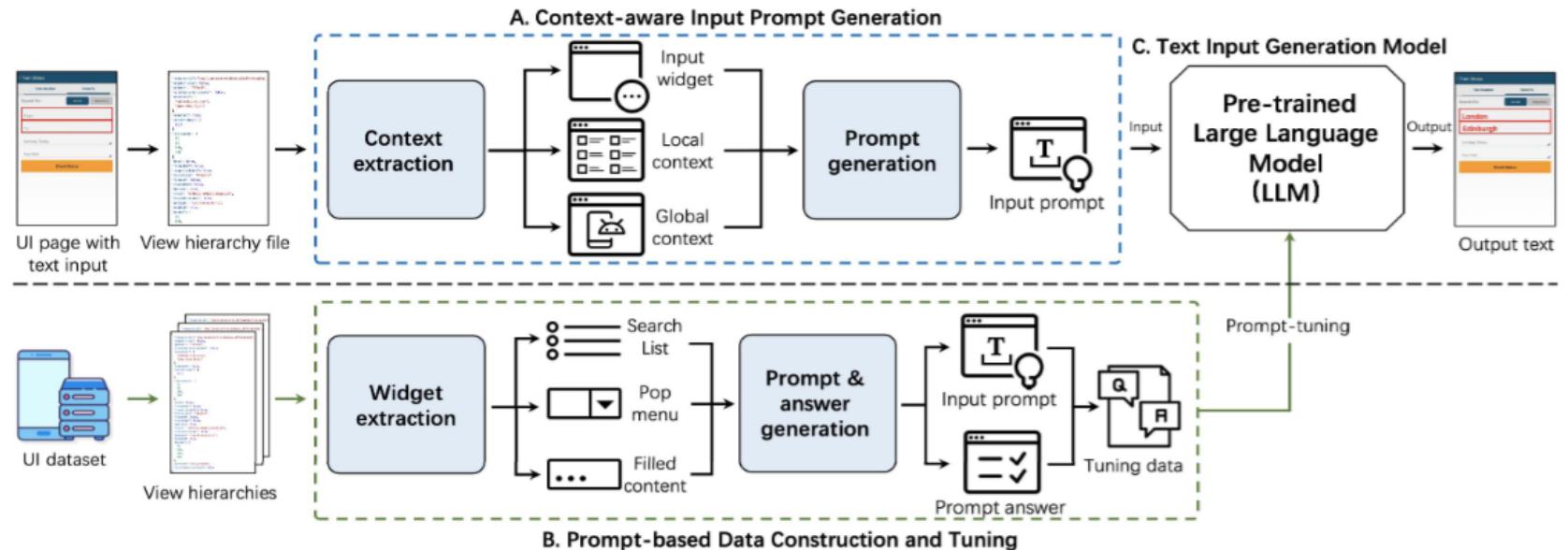
[1] Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*.



SAPIENZA
UNIVERSITÀ DI ROMA

Fill in the Blank: Context-aware Automated Text Input Generation for Mobile GUI Testing

Qtypist is an Automated GUI testing approach, which uses a pre-trained Large Language Model (LLM) to automatically generate input text.



Source:

[2] Liu, Z., Chen, C., Wang, J., Che, X., Huang, Y., Hu, J., & Wang, Q. (2023, May). Fill in the blank: Context-aware automated text input generation for mobile gui testing. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 1355-1367). IEEE.

Fill in the Blank: Context-aware Automated Text Input Generation for Mobile GUI Testing

Methodology:

- The context-aware input prompt generation method enhance LLM performance by describing input widget information using extracted context and linguistic patterns.
- An automated way to construct a prompt-based dataset from view hierarchy files, leveraging widgets with candidate input contents for effective prompt-tuning.
- QTypist leverages GPT-3 to enhance automated testing through smart text generation.

Source:

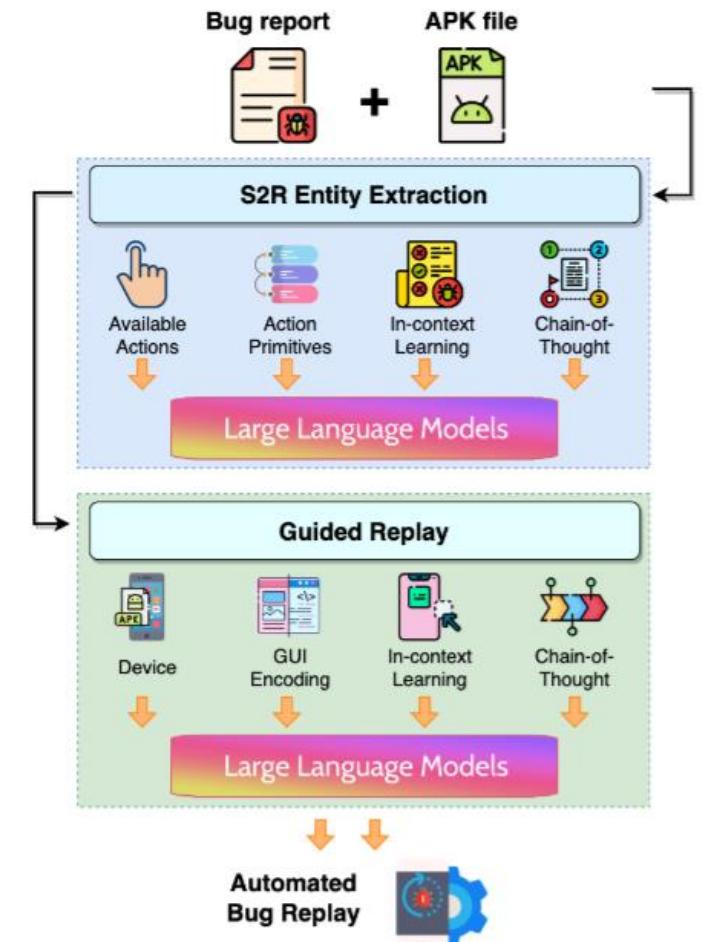
[2] Liu, Z., Chen, C., Wang, J., Che, X., Huang, Y., Hu, J., & Wang, Q. (2023, May). Fill in the blank: Context-aware automated text input generation for mobile gui testing. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 1355-1367). IEEE.

Automated Android Bug Replay with Large Language Models

AdbGPT is a lightweight approach for automatically reproducing bugs using prompt engineering.

Methodology:

- Prompt LLMs with entity specifications, representative exemplars, and developers' reasoning.
- Extract entities from bug reports.
- Provide current GUI screen and use few-shot learning with chain-of-thought reasoning.
- Dynamically guide probable target components to reproduce bugs.



Source:

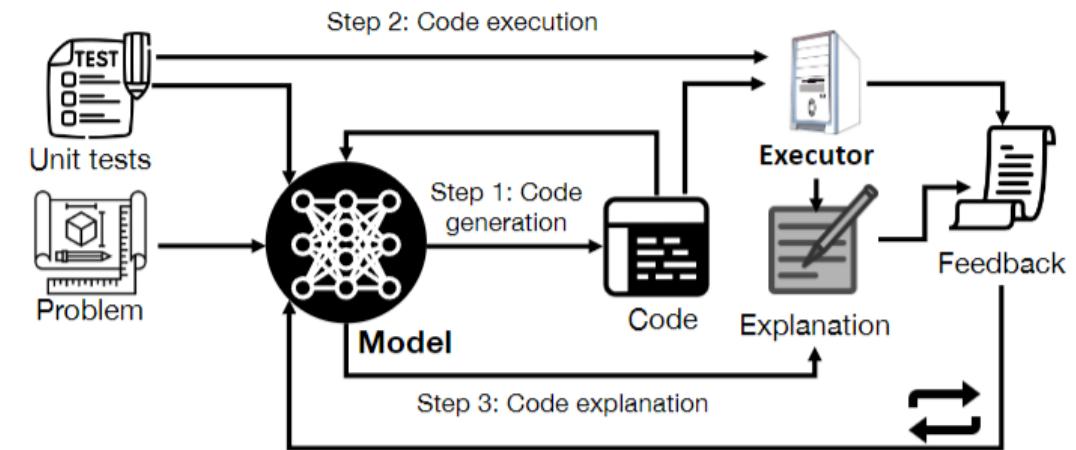
[3] Feng, S., & Chen, C. (2024, February). Prompting Is All You Need: Automated Android Bug Replay with Large Language Models. In Proceedings of the 46th IEEE/ACM International Conference on Software Engineering (pp. 1-13).

Teaching Large Language Models to Self-Debug

It enables large language models to debug code generated by themselves allowing the model to perform rubber duck debugging, so to identify and fix bugs without human intervention.

Results:

- Highlights the potential to improve coding performance by teaching models to iteratively debug their predictions.
- Focuses on understanding code, identifying errors, and following error messages to fix bugs.



Source:

[4] Chen, X., Lin, M., Schärli, N., & Zhou, D. (2023). Teaching large language models to self-debug. arXiv preprint arXiv:2304.05128.

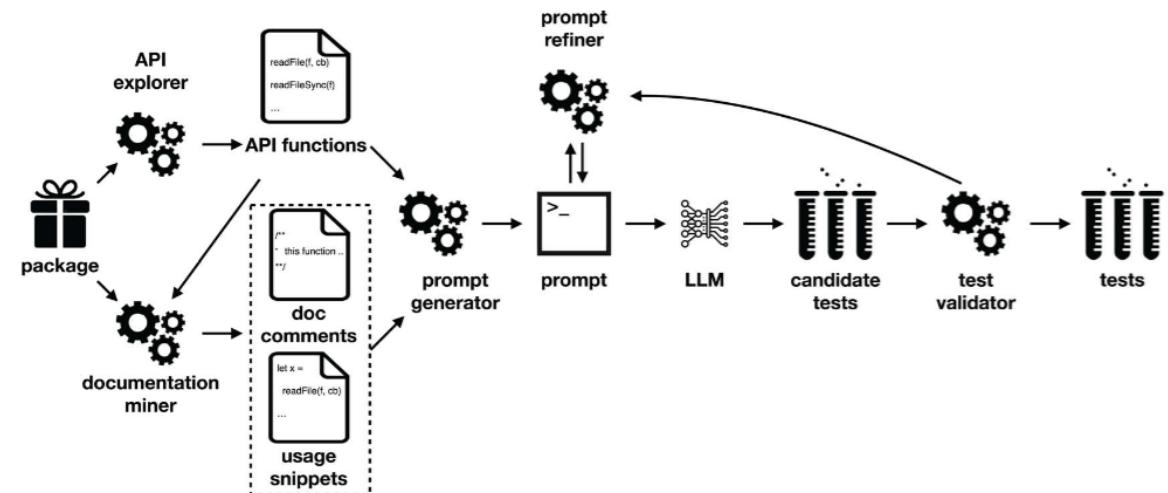
An empirical evaluation of using large language models for automated unit test generation.

TESTPILOT is an innovative approach for generating unit tests adaptively using large language models (LLMs). It does not require fine-tuning but it embeds contextual information about the function under test into the prompt.

There is a combination of feedback-directed test generation, like Nessie, with LLM-based techniques.

It is an approach adaptable to any programming language that lead to:

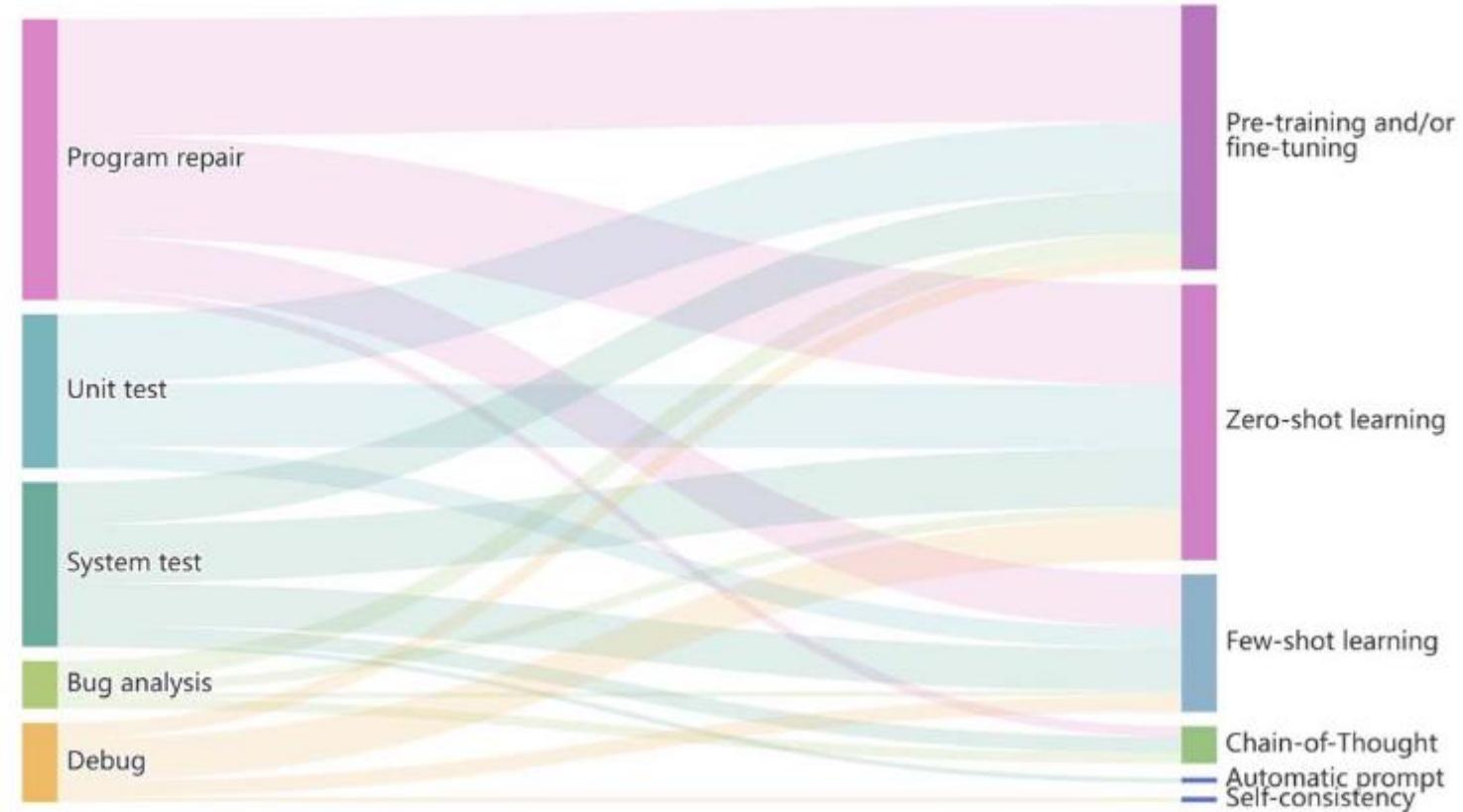
- Adjust prompts to match language syntax and testing framework.
- Adapt mining of documentation and usage examples.



Source:

[6] Schäfer, M., Nadi, S., Eghbali, A., & Tip, F. (2023). An empirical evaluation of using large language models for automated unit test generation. IEEE Transactions on Software Engineering.

Mapping between testing tasks and how LLMs are used.



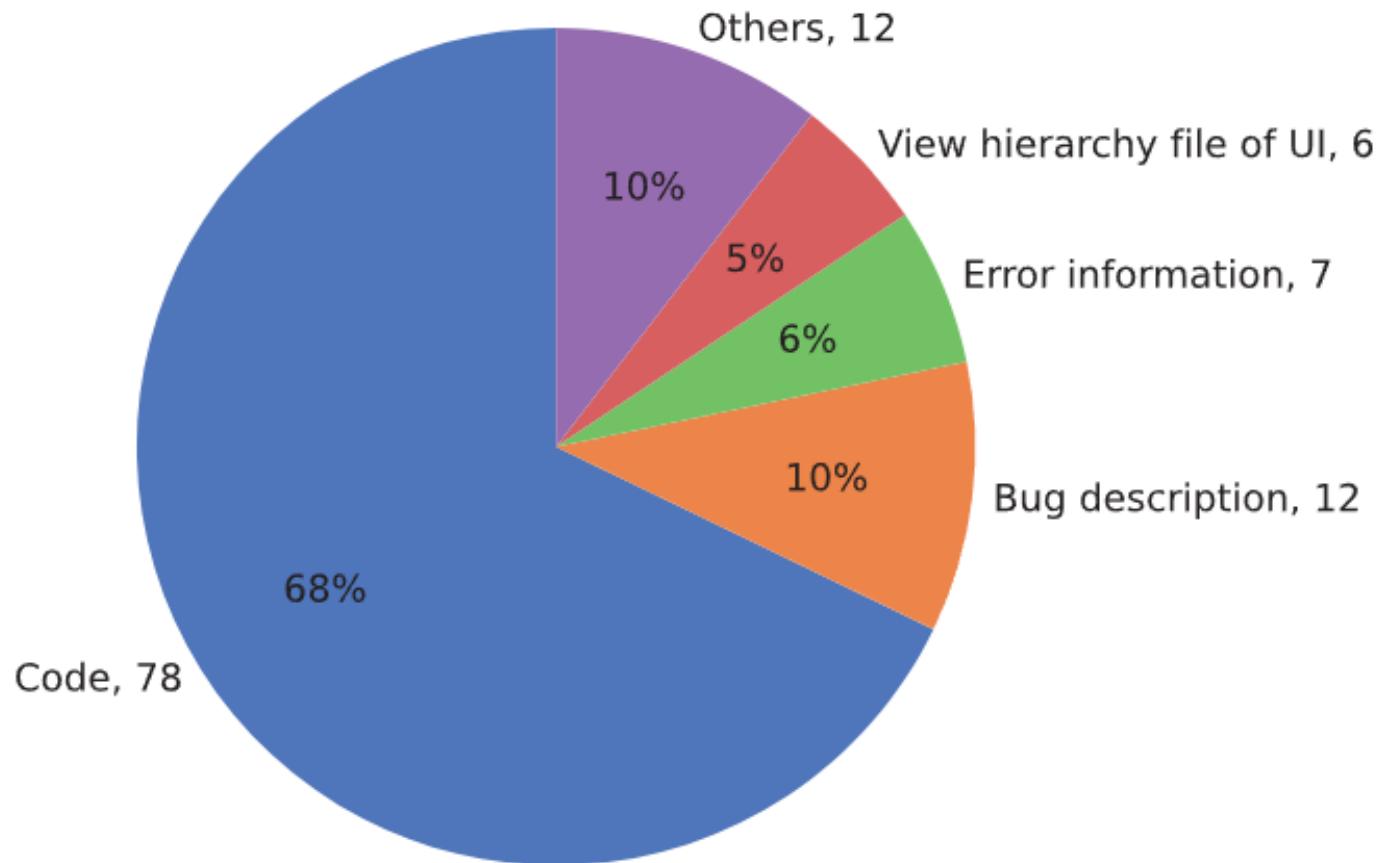
Source:

[1] Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*.



SAPIENZA
UNIVERSITÀ DI ROMA

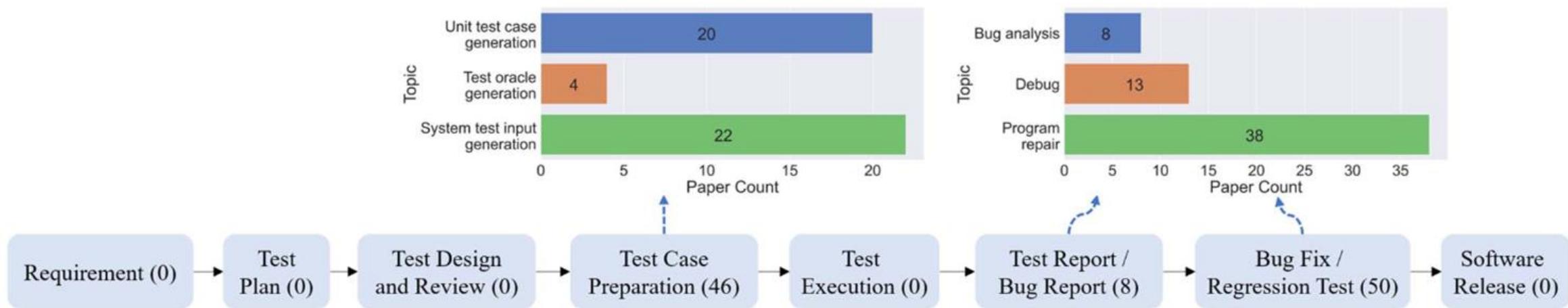
LLMs testing input



Source:

[1] Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*.

Testing task distribution along software lifecycle



Source:

[1] Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*.



SAPIENZA
UNIVERSITÀ DI ROMA

Limitations (so far)

LLMs used in a subset of the entire testing lifecycle: middle and later stages of testing, only serving the unit and system testing phases, and only for functional testing.

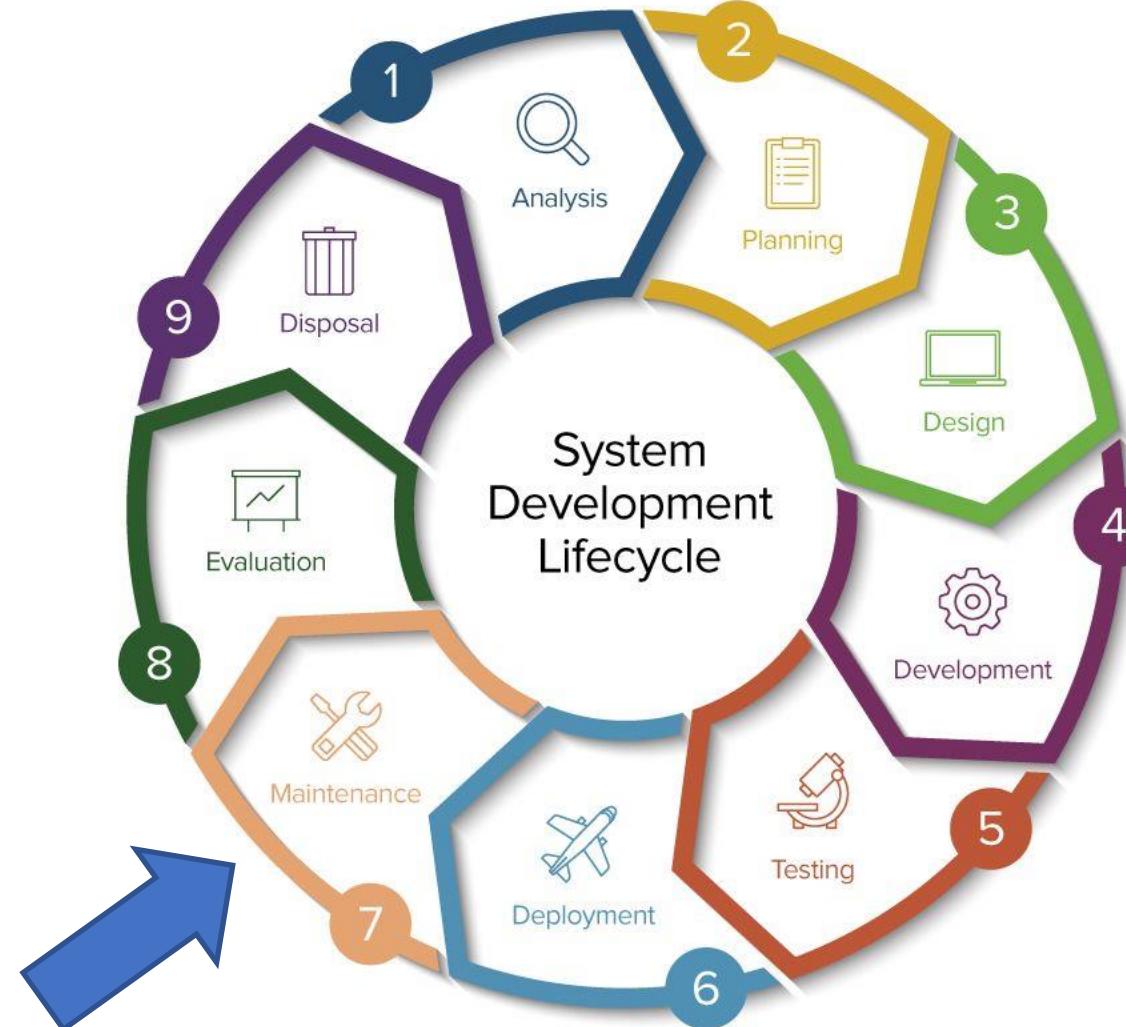
Source:

- [1] Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*.



SAPIENZA
UNIVERSITÀ DI ROMA

Information System Lifecycle



Source: <https://www.smartsheet.com/system-development-life-cycle-guide>



SAPIENZA
UNIVERSITÀ DI ROMA

Maintenance

Swanson's (1976) "Maintenance consists of: (A) **Corrective Maintenance**: actions to fix processing failures, performance issues, or potential implementation problems; (B) **Adaptive maintenance**: adjustments to accommodate changes in the data or processing environment; (C) **Perfective maintenance**: enhancements to improve processing efficiency, performance, add new features, or improve software maintainability [1]".

Riggs (1978) "Systems maintenance is the activity associated with keeping operational computer systems continuously in tune with the requirements of users, data processing operations, associated clerical functions, and external demands from government and other agencies [2]".



Edwards (1984):

"Maintenance is an alteration to any systems element necessary to eliminate a fault in that element, adapt to a change in that or any other element, or to improve upon the performance characteristics of an element [3]".

[1] E. Burton Swanson. (1976). *The dimensions of maintenance*. In Proceedings of the 2nd international conference on Software engineering (ICSE '76). IEEE Computer Society Press, Washington, DC, USA, 492–497.

[2] B. P. Lientz, E. B. Swanson, and G. E. Tompkins. (1978). *Characteristics of application software maintenance*. Commun. ACM 21, 6 (June 1978), 466–471.
<https://doi.org/10.1145/359511.359522>

[3] Edwards, C. (1984). *Information Systems Maintenance: An Integrated Perspective*. MIS Quarterly, 8(4), 237–256. <https://doi.org/10.2307/249094>



LLM for monitoring IS: literature review



(*"AI"* OR *"LLM"* OR *"Large Language Model"* OR *"Artificial Intelligence"*) AND (*"Information System"*) AND (*"Monitoring"* OR *"Maintenance"*)

(*"AI"* OR *"LLM"* OR *"Large Language Models"* OR *"Artificial Intelligence"*) AND (*"Information System"*) AND (*"Monitoring"* OR *"Maintenance"* OR *"Evaluation"*)

(*"AI"* OR *"LLM"* OR *"Large Language Models"* OR *"Artificial Intelligence"* OR *"ML"* OR *"Machine Learning"*) AND (*"Information System"*) AND (*"Monitoring"* OR *"Maintenance"*)

(*"AI"* OR *"LLM"* OR *"Large Language Models"* OR *"Artificial Intelligence"* OR *"ML"* OR *"Machine Learning"*) AND (*"Information System"*) AND (*"Monitoring"* OR *"Maintenance"* OR *"Evaluation"* OR *"Check"*)



No relevant results!



Let's focus on monitoring techniques for information systems.



SAPIENZA
UNIVERSITÀ DI ROMA

LLM for monitoring IS

Log Analysis

- Process of **examining, interpreting, and deriving insights from log files** generated by various components of a system. This involves identifying patterns, detecting anomalies, diagnosing issues, and improving system performance and security by analyzing records of system activities, user actions, and events [4].
- To extract relevant information, **log parsing is necessary**, and log format detection plays a crucial role in this process.

By leveraging the advanced representation learning capabilities of LLMs, log analysis can become even more efficient and effective!



BERTOps: LLM that can effectively generalize to multiple downstream tasks of log analysis [5]

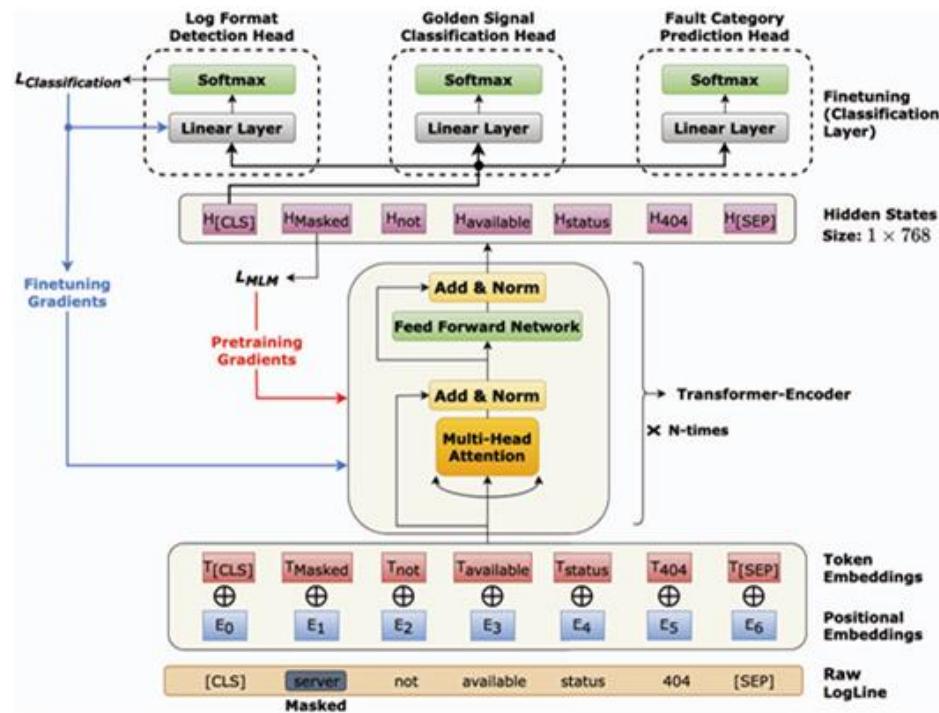
[4] Adam Oliner, Archana Ganapathi, and Wei Xu. (2012). *Advances and challenges in log analysis*. Commun. ACM 55, 2 (February 2012), 55–61.

<https://doi.org/10.1145/2076450.2076466>

[5] Gupta, P., Kumar, H., Kar, D., Bhukar, K., Aggarwal, P., & Mohapatra, P. (2023). *Learning Representations on Logs for AIOps*. 2023 IEEE 16th International Conference on Cloud Computing (CLOUD), 155-166.



LLM for monitoring IS: BERTOps



Log Format Detection (LFD) [6]:

Understanding log formats is crucial for log analysis, aiding in tasks like parsing, structure extraction, and anomaly detection. However, dealing with multiple variations within each format poses a challenge. **To address this, a multi-class classification model is trained to differentiate logs from 16 formats.**

Golden Signal Classification (GSC) [7]:

Golden Signals as key performance indicators (KPIs) used for monitoring applications hosted on the cloud. Monitoring these signals, which include latency, traffic, errors, and saturation, allows for the efficient detection and troubleshooting of errors. This proactive approach enables teams to identify anomalies and trends before they escalate into critical issues, ensuring smooth and efficient system operation.

Fault Category Prediction (FCP) [8]:

Fault categories in logs help detect abnormal system behaviors and diagnose failures, providing insight into why a service fails. They aid in isolating root causes and directing faults to the right teams for resolution. **The goal is to create a fault categorization model for classifying log messages into seven categories:** Memory, Network, Authentication, I/O, Device, Application, and Other.

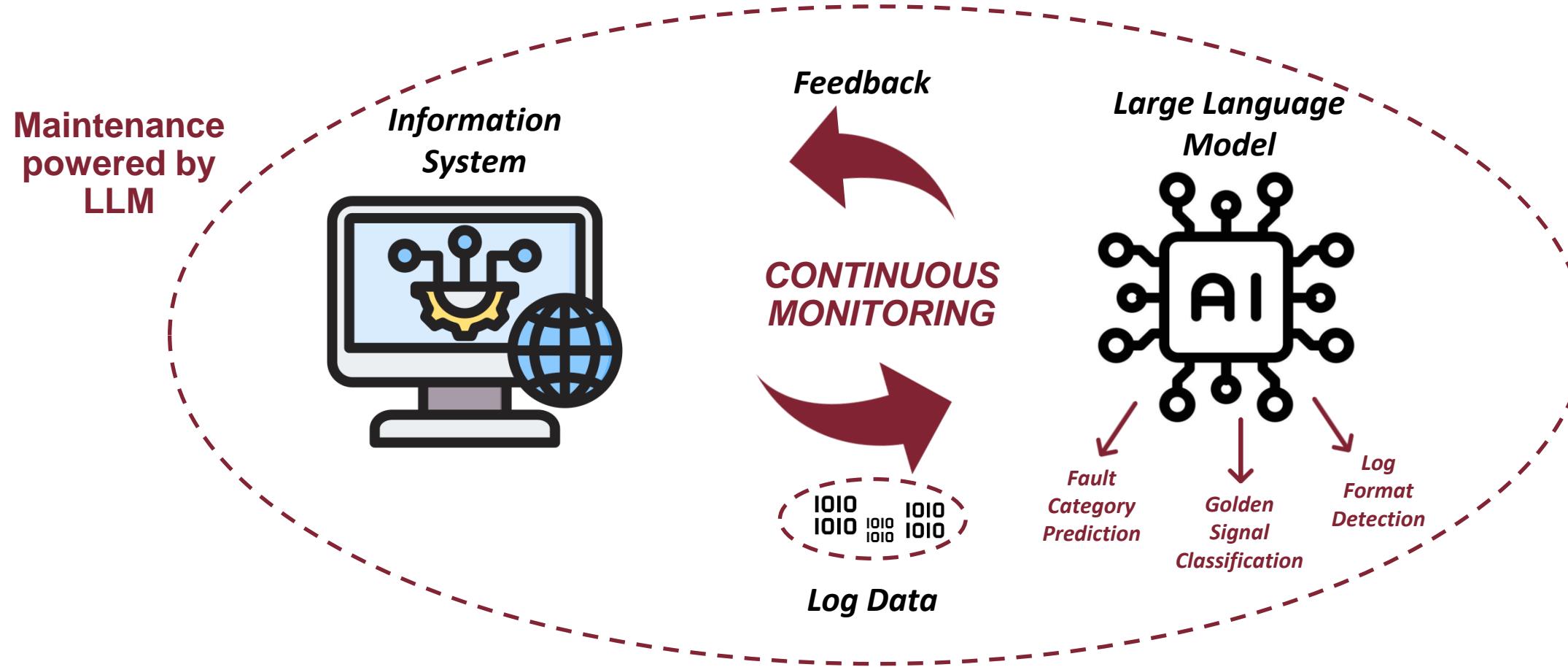
[6] S. Messaoudi, A. Panichella et al., *A Search-Based Approach for Accurate Identification of Log Message Formats*, 2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC), Gothenburg, Sweden, 2018, pp. 167-16710.

[7] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, *Site Reliability Engineering: How Google Runs Production Systems*, 2016. Available: <http://landing.google.com/sre/book.html>

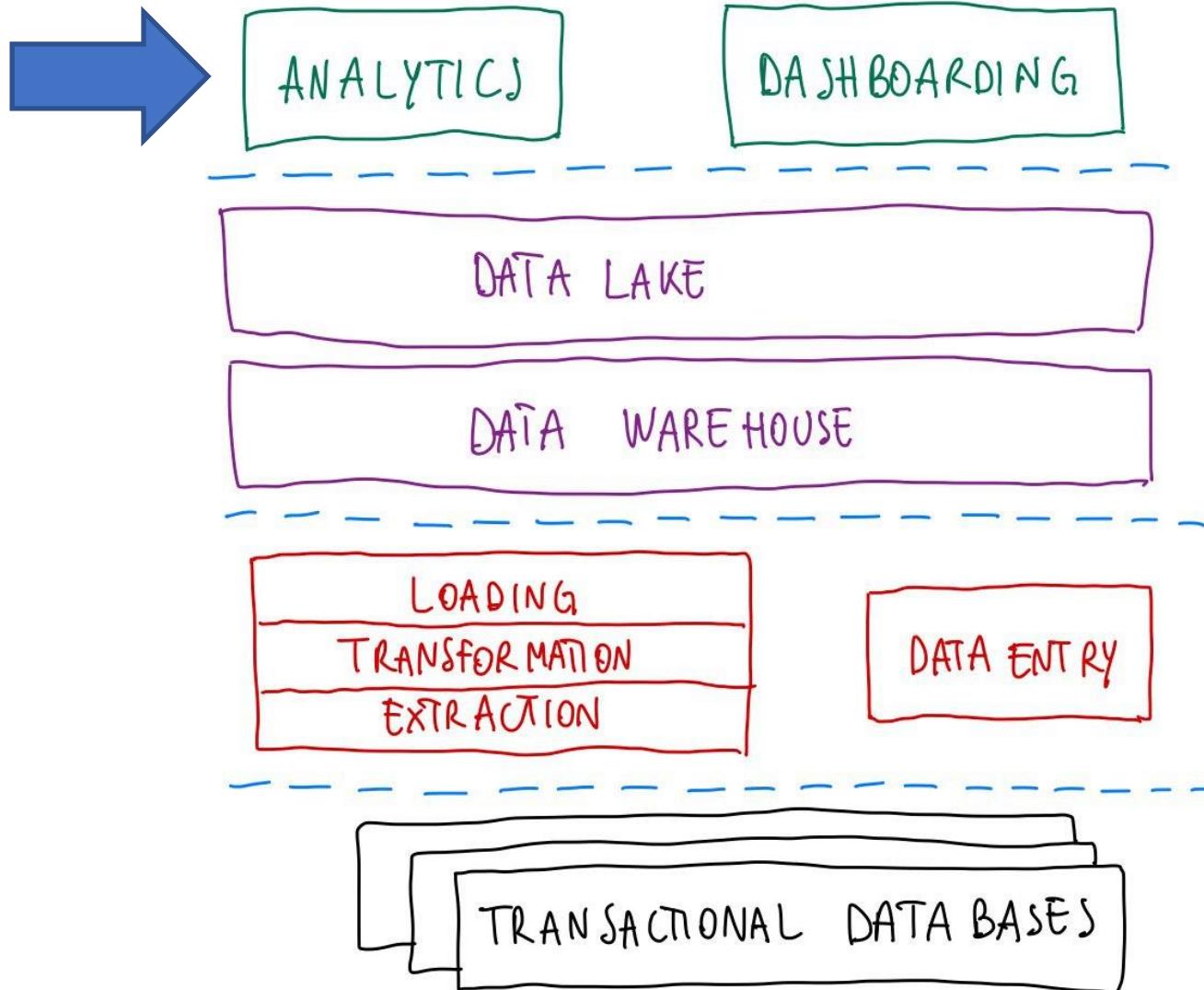
[8] Zou, D., Qin, H., Jin, H., Qiang, W., Han, Z., Chen, X. (2014). *Improving Log-Based Fault Diagnosis by Log Classification*. In: Hsu, CH., Shi, X., Salapura, V. (eds) Network and Parallel Computing. NPC 2014. Lecture Notes in Computer Science, vol 8707. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-44917-2_37



LLM for monitoring IS: summary



The information & knowledge management layer



LLMs and Data Analytics

- Pretrained Language Models (PLM) and LLMs have shown remarkable capabilities at processing and wrangling data
- Have the potential of lowering the barrier in analysing and running analytics on data

Data Overview

Year	Brand	Category	Model	Sales
1/1/2007	BMW	Compact	BMW 3-Series	142490
1/1/2008	BMW	Compact	BMW 3-Series	112464
1/1/2009	BMW	Compact	BMW 3-Series	90960
1/1/2010	BMW	Compact	BMW 3-Series	100910
1/1/2011	BMW	Compact	BMW 3-Series	94371

User Inquiry
I want to know the overall trend of Toyota.

User Inquiry

Report Generated by LLM

Based on my analysis of the CarsSales data, the overall trend of Toyota can be summarized as follows. Over the years, Toyota's sales have decreased rapidly and steadily, with the trend being most noticeable for their subcompact cars, especially the Corolla and Camry models. Despite this trend, Camry continues to be a top-selling model for Toyota, with Corolla following closely behind. Additionally, Toyota's overall sales show a strong positive correlation with the sales of their Camry model, indicating that Camry remains a key driver of Toyota's sales.

Prioritize Relevant Insight

It is interesting to note that among all the brands, Toyota and Honda consistently emerge as the top two brands for subcompact cars, with Toyota leading the way. Also, for each year, Camry and Corolla are the top-selling models for Toyota.

Furthermore, the data analysis highlights specific years, such as 2007-01-01 and 2008-01-01, where certain categories such as compact and subcompact cars have

Insights Discovered by Insight Engine

Basic Insight: Brand=Toyota(44.32%) and Brand=Honda(41.43%) are outstanding top two of 'Category'='Subcompact' in Sales among all Brand(s).

Brand	Sum(Sales)
Toyota	~1,800,000
Honda	~1,600,000
Hyundai	~200,000
Ford	~100,000
Volkswagen	~50,000
BMW	~50,000
Mazda	~50,000

RUN

From [1]

[1] Ma, Pingchuan, et al. "InsightPilot: An LLM-Empowered Automated Data Exploration System." *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2023.

LLMs and Data Analytics

- We can distinguish between three main categories of flavours of LLMs for data analytics and data analysis:
 - Data analytics on sample data
 - Tabular QA
 - Data analytics on external data sources:
 - Relational databases: Text-to-SQL models
 - Knowledge graphs: Text-to-SPARQL
 - Data analytics on multiple external data sources:
 - LLM Agents

Data analytics on sample data: Tabular QA

- Requires:
 - a dataset (e.g. a table)
 - a natural language question
- The model then reasons over the table to generate the response
 - PLMs: TAPAS [1], TAPAX [2]
 - LLMs: Chain-of-Tables [3]

Table				Example questions			
Rank	Name	No. of reigns	Combined days	#	Question	Answer	Example Type
1	Lou Thesz	3	3,749	1	Which wrestler had the most number of reigns?	Ric Flair	Cell selection
2	Ric Flair	8	3,103	2	Average time as champion for top 2 wrestlers?	AVG(3749,3103)=3426	Scalar answer
3	Harley Race	7	1,799	3	How many world champions are there with only one reign?	COUNT(Dory Funk Jr., Gene Kiniski)=2	Ambiguous answer
4	Dory Funk Jr.	1	1,563	4	What is the number of reigns for Harley Race?	7	Ambiguous answer
5	Dan Severn	2	1,559	5	Which of the following wrestlers were ranked in the bottom 3?	{Dory Funk Jr., Dan Severn, Gene Kiniski}	Cell selection
6	Gene Kiniski	1	1,131		Out of these, who had more than one reign?	Dan Severn	Cell selection

From [1]

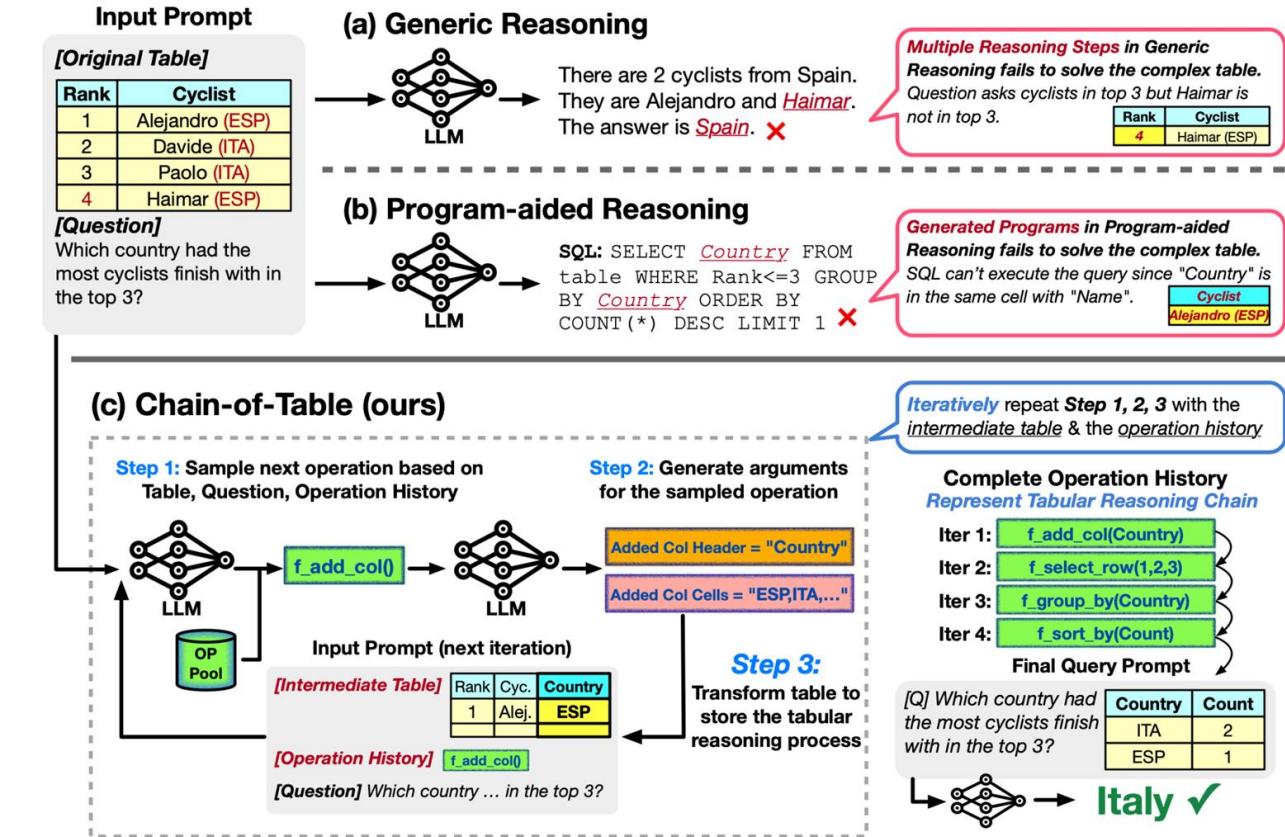
[1] Herzig, Jonathan, et al. "TaPas: Weakly Supervised Table Parsing via Pre-training." *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020.

[2] Liu, Qian, et al. "TAPEX: Table Pre-training via Learning a Neural SQL Executor." *International Conference on Learning Representations*. 2021.

[3] Wang, Zilong, et al. "Chain-of-Table: Evolving Tables in the Reasoning Chain for Table Understanding." *The Twelfth International Conference on Learning Representations*. 2023.

Tabular QA example: Chain of Tables [1]

- Uses LLMs and in-context learning to dynamically plan a chain of operations over a table in response to a question
 - e.g. add column, select row, etc.
- This reasoning is done iteratively by modifying the table step-by-step
- Each intermediate operation and resulting table are put into the model prompt



From [1]

[1] Wang, Zilong, et al. "Chain-of-Table: Evolving Tables in the Reasoning Chain for Table Understanding." *The Twelfth International Conference on Learning Representations*. 2023.



Data Analytics on external data sources

- Models falling in this category are trained to convert natural language to a query language (e.g. SQL, SPARQL)
 - e.g. [1,2,3]
- The generated query can then be runned against a data source (e.g. RDBMS, KG, etc.)
- Usually need additional information besides the query
 - e.g. database schema

Complex input sentence:

What are the name and lowest instructor salary of the departments with average salary greater than the overall average?

Database:

Table 1 **instructor**

ID	name	department_name	salary	...
----	------	-----------------	--------	-----

Columns

foreign key

Table 2 **department**

name	building	budget
------	----------	--------	-------

primary key

Table *n*

Correct SQL translation:

```
SELECT min(salary), department_name
FROM instructor
GROUP BY department_name
HAVING avg(T1.salary) >
       (SELECT avg(salary) FROM instructor)
```

From [1]

[1] Yu, Tao, et al. "Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task." arXiv preprint arXiv:1810.05237 (2018).

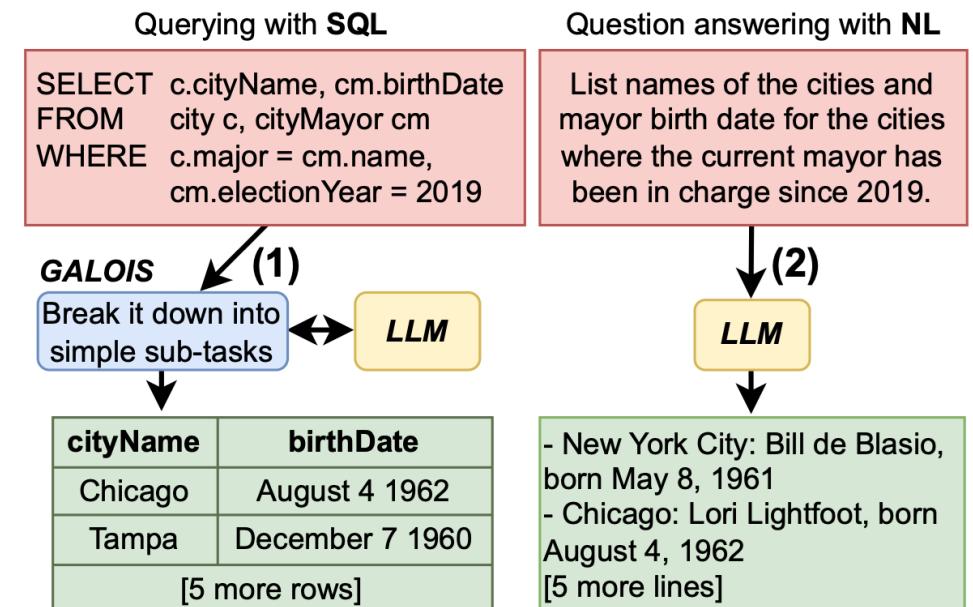
[2] Ochieng, Peter. "PAROT: Translating natural language to SPARQL." Expert Systems with Applications: X 5 (2020): 100024.

[3] Gu, Zihui, et al. "Few-shot text-to-sql translation using structure and content prompt learning." Proceedings of the ACM on Management of Data 1.2 (2023): 1-28.



An example: GALOIS [1]

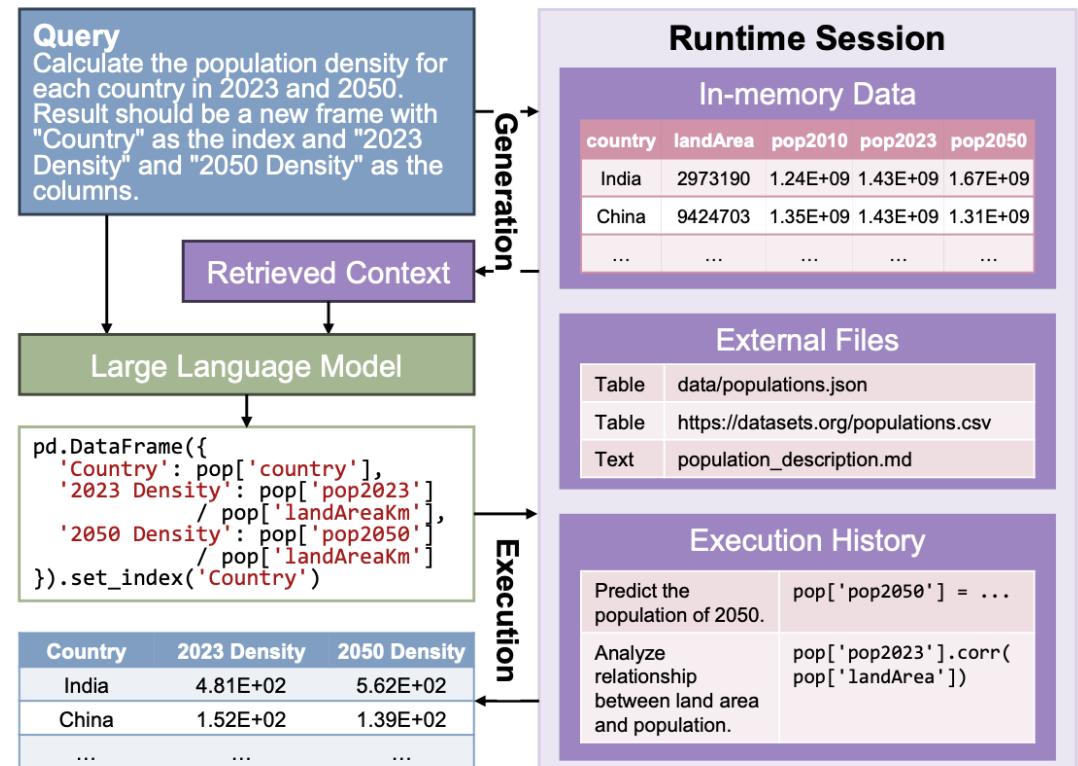
- Sometimes even LLMs can be queried as databases to answer questions or run analytics, e.g. Galois [1]
 - Leverages the data on which the LLM was pretrained on to produce tables
- Galois takes an SQL query and breaks into a set of subtasks (or logical plan)
- These subtasks can be fed to an LLM, producing intermediate tuples
- Finally Galois integrates the intermediate tuples to generate the final table



From [1]

Data Analytics over multiple data sources

- Previous approaches were limited in the data simple (e.g. sample data) or the use of a single data source (e.g. a database)
- Modern LLM-based approaches can work with multiple data sources and at a higher level of abstraction
 - e.g. generating plots
- Based on the LLM Agents [2]



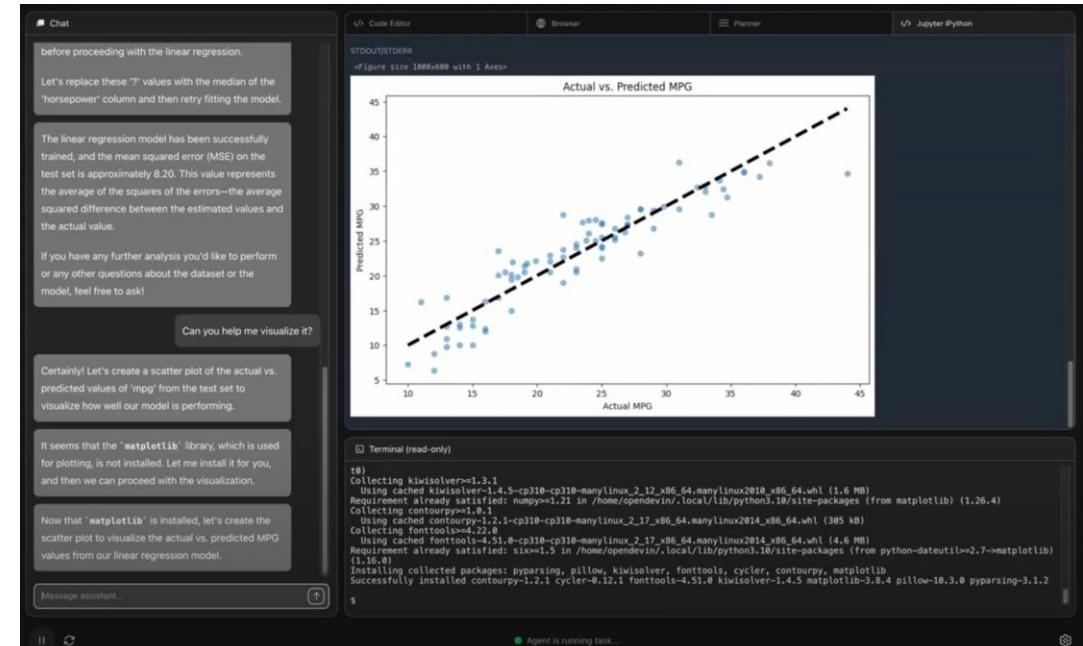
From [1]

[1] Zhang, Yuge, et al. "Benchmarking Data Science Agents." *arXiv preprint arXiv:2402.17168* (2024).

[2] Parisi, Aaron, Yao Zhao, and Noah Fiedel. "Talm: Tool augmented language models." *arXiv preprint arXiv:2205.12255* (2022).

SWE Agents

- Software Engineering (SWE) agents
 - e.g. Devin [1], OpenDevin [2] swe-agent [3] and Devika [4]
 - Primarily used for SWE related tasks, e.g. code completion, solving issues and bugs, pull requests
- They can be also used for data analysis (and analytics) related tasks, e.g. running a linear regression and plotting the predictions



From [5]

[1] <https://www.cognition.ai/blog/introducing-devin>

[2] <https://opendevin.github.io/OpenDevin/>

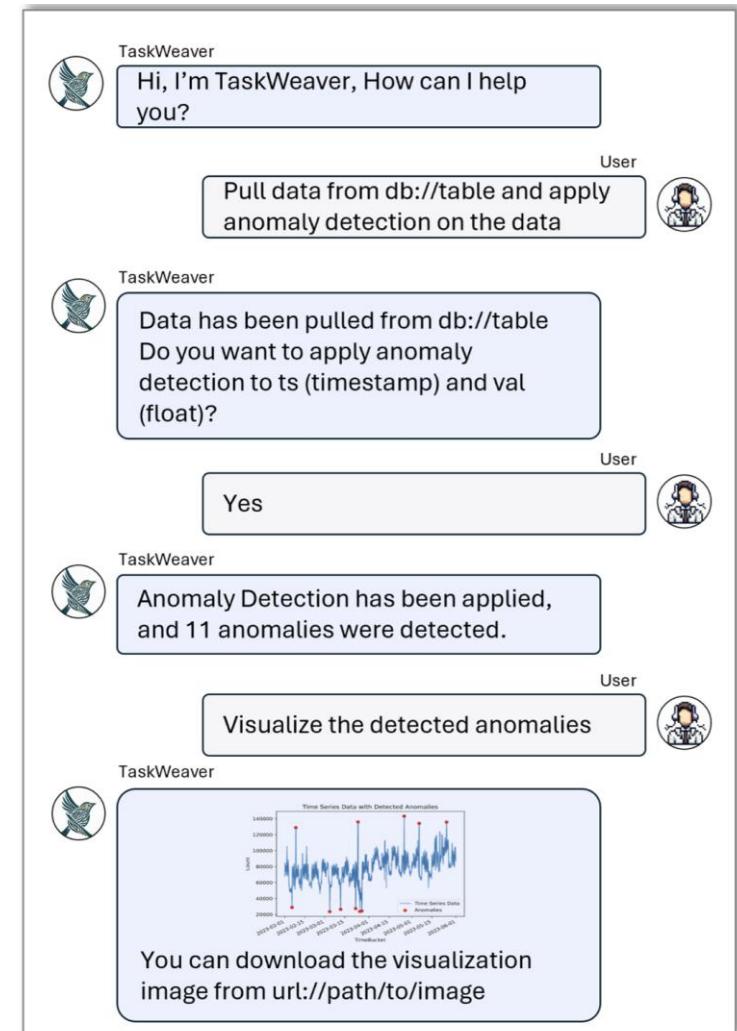
[3] Yang, John, et al. "SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering", arXiv preprint arXiv:2405.15793 (2024).

[4] <https://github.com/stitionai/devika>

[5] https://x.com/xingyaow_/status/1785737674609004714

Data Science Agents

- LLM Agents that address data-centric tasks
 - data analysis, machine learning , etc.
- e.g. MetaGPT Data Interpreter [1], Task Weaver [2], Autogen [3]Can be used also to run analytics at a higher level of abstraction compared to Tabular QA and Text-to-SQL models



From [2]

[1] Hong, Sirui, et al. "Data Interpreter: An LLM Agent For Data Science." *arXiv preprint arXiv:2402.18679* (2024).

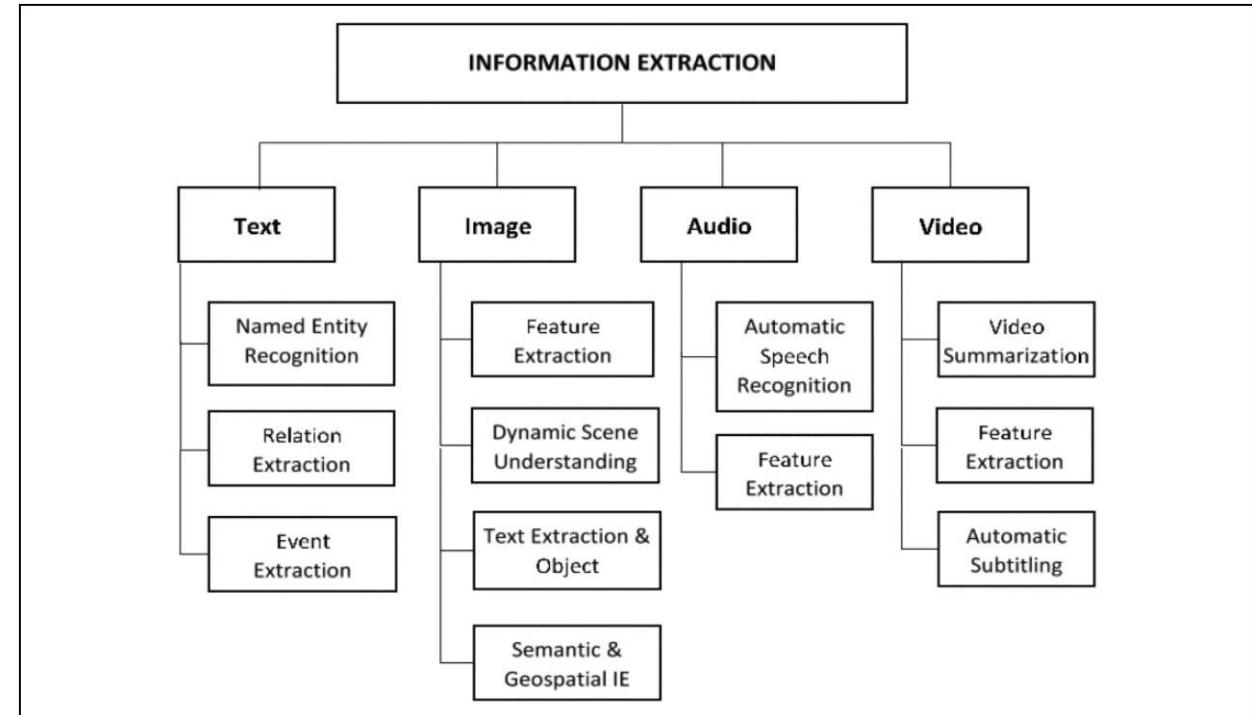
[2] Qiao, Bo, et al. "Taskweaver: A code-first agent framework." *arXiv preprint arXiv:2311.17541* (2023).

[3] Wu, Qingyun, et al. "Autogen: Enabling next-gen llm applications via multi-agent conversation framework." *arXiv preprint arXiv:2308.08155* (2023).

What is Information Extraction

Information Extraction (IE) refers to the automatic extraction of structured information such as entities, relationships between entities, and attributes describing entities from unstructured sources^[1].

Unstructured sources can include (i) text, (ii) images, (iii) audio, and (iv) video, each with its own specific tasks.



IE tasks grouped by data source^[2].

[1] Sarawagi, Sunita. "Information extraction." (2008).

[2] Adnan, Kiran, and Rehan Akbar. "Limitations of information extraction methods and techniques for heterogeneous unstructured big data." (2019).

Textual Information Extraction

Typical textual Information Extraction tasks consists of Named Entity Recognition (NER), Relation Extraction (RE) and Event Extraction (EE) (Wang et al., 2023c)^[1,2,3].

Elon Musk apparently wasn't aware that his company Space X had a Facebook page. The company's CEO has responded to a comment on Twitter acknowledging he didn't know one existed, and then following up with promised that he will take them down today (following the screenshot taken around 12:10 PM ET).

(a)

Sentence: Elon Musk is the founder of Space X.
[Entity 1] [Relationship] [Entity 2]
Extract relation roleInCompany
Tuple form: < Elon Musk, founder, Space X >

(b)

Passage: Indonesia will delay the execution of six convicts including an Indian on death row after five of them appealed to the Supreme Court for a second review.

Agent: Indonesia
Person: convicts
Place: None
Justice:Execute: Adjudicator: Supreme Court
Plaintiff: None
Justice:Appeal: Plaintiff: None

Justice:Execute	
Agent :	Indonesia
Person :	convicts
Place :	None

Justice:Appeal	
Plaintiff :	five
Prosecutor :	None
Adjudicator :	Supreme Court
Place :	Indonesia

(c)

Illustrations of NER (a)^[2], RE (b)^[2] and EE (c)^[4] tasks on unstructured text.

[1] [Adnan, Kiran, and Rehan Akbar. "Limitations of information extraction methods and techniques for heterogeneous unstructured big data."](#) (2019).

[2] [Abdullah, Mohd Hafizul Afifi, et al. "Systematic literature review of information extraction from textual data: recent methods, applications, trends, and challenges."](#) (2023).

[3] [Xu, Derong, et al. "Large language models for generative information extraction: A survey."](#) (2023).

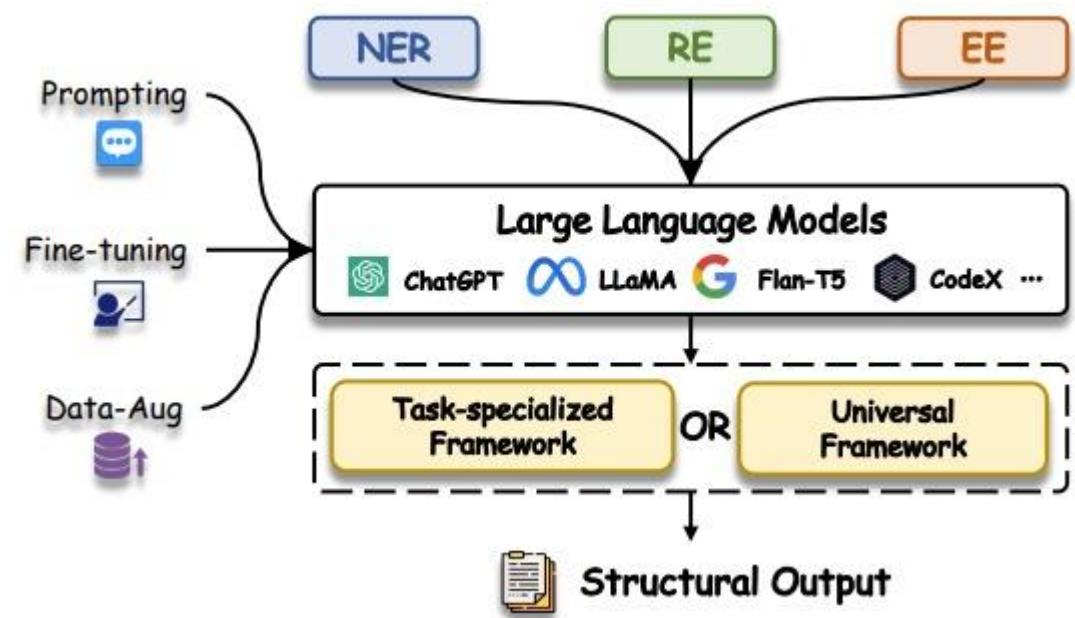
[4] [Hsu, I., et al. "DEGREE: A data-efficient generation-based event extraction model."](#) (2021).



LLMs & IE

The emergence of LLMs, with their exceptional capabilities in text understanding, has led to a **surge of interest in IE methods that adopt LLMs** to extract generate structural information from text^[1].

Studies have explored **different learning paradigms**, LLM architectures, and both specialized and universal **frameworks** for handling multiple IE tasks^[1].



Textual IE tasks can be performed using various LLM paradigms and frameworks^[1].

[1] Xu, Derong, et al. “Large language models for generative information extraction: A survey.” (2023).

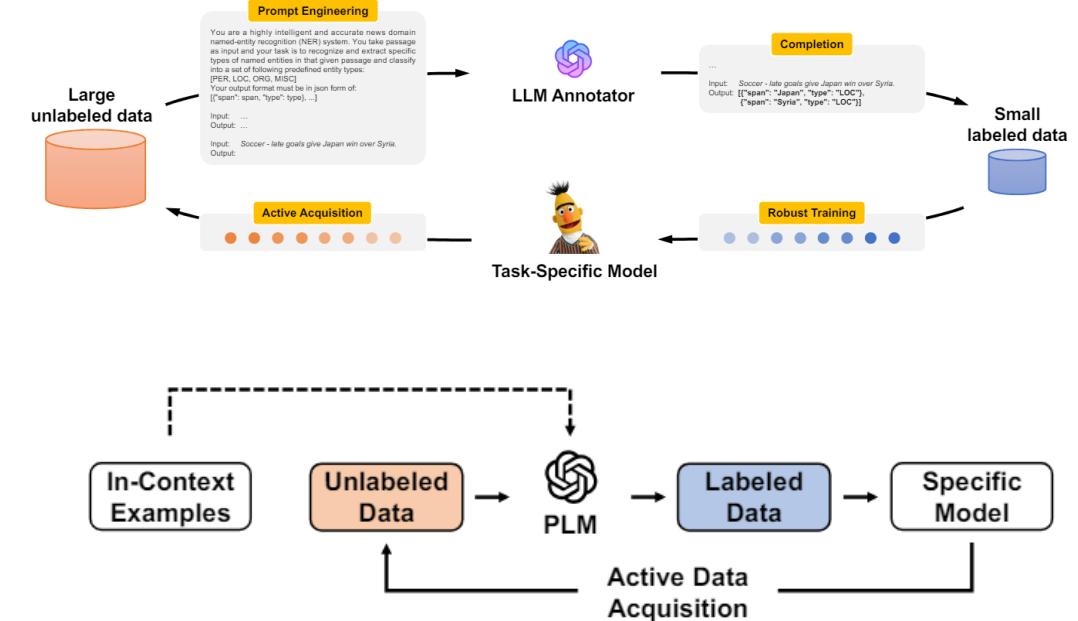


Data Augmentation & IE

Performing IE tasks often necessitates **substantial amounts of annotated data** to train LLMs or assess their abilities. The advancements in generative LLMs have driven the exploration of data augmentation techniques **to generate — partially or entirely — synthetic data** using both supervised and unsupervised methods^[1].

Prevalent data augmentation techniques for generating datasets suitable for IE tasks include (i) Data Annotation, (ii) Knowledge Retrieval, and (iii) Inverse Generation^[2].

In this scenario, the **main challenge** lies in avoiding the introduction of unrealistic, misleading, or biased patterns while synthesizing the data^[2].



Example of a framework that takes LLMs as annotators and puts them into an active learning loop for text labeling^[2].

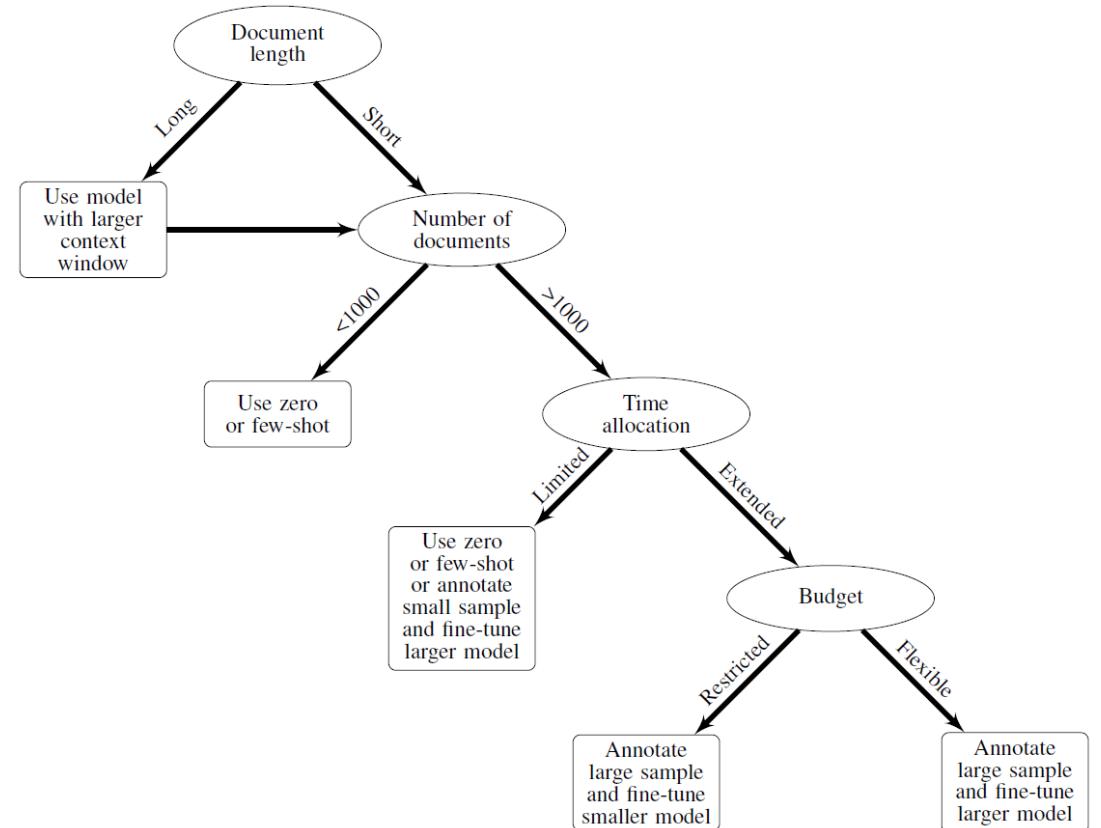
[1] [Zhang, Ruoyu, et al. "Llmaaa: Making large language models as active annotators."](#) (2023).

[2] [Xu, Derong, et al. "Large language models for generative information extraction: A survey."](#) (2023).

Prompting vs Fine-tuning for IE

The **most commonly used paradigms for performing IE tasks** are prompting and fine-tuning.

Neither approach is ideal for IE tasks, as each involves **trade-offs** and compromises that must be accepted^[2].



[1] Chae, Youngjin, and Thomas Davidson. "Large language models for text classification: From zero-shot learning to fine-tuning." (2023).

[2] Trad, Fouad, and Ali Chehab. "Prompt Engineering or Fine-Tuning? A Case Study on Phishing Detection with Large Language Models." (2024).

An example of a trade-off approach for performing a text classification task with LLMs^[1].



Prompting for IE

Recommended for:

- tasks that require a **general knowledge** and **reasoning**^[1,2]

Pros:

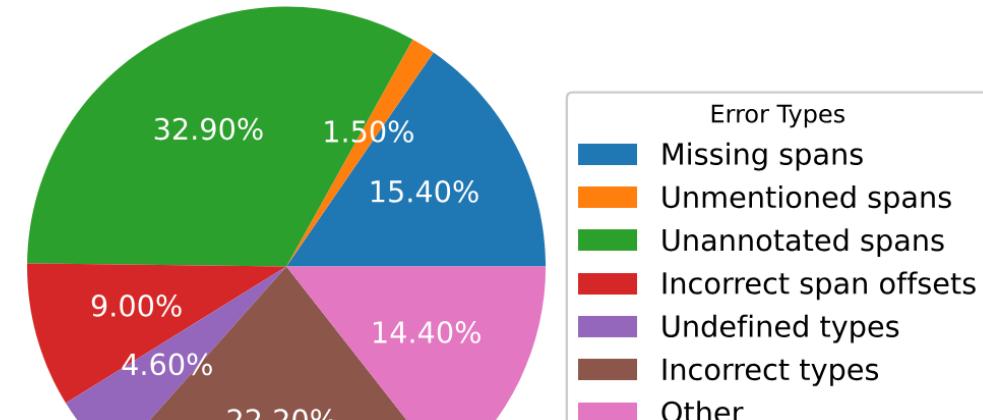
- **no dataset or expertise** needed for training

Cons:

- **prone to hallucination**^[2]
- effectiveness heavily depends on the **prompt formulation**

[1] Törnberg, Petter. “Chatgpt-4 outperforms experts and crowd workers in annotating political twitter messages with zero-shot learning.” (2023).

[2] Han, Ridong, et al. “Is information extraction solved by chatgpt? an analysis of performance, evaluation criteria, robustness and errors.” (2023).



The figure shows the distribution of errors made by prompting GPT 3.5 in a NER task. The type errors “Unmentioned Spans,” “Incorrect span offsets,” and “Undefined Types” are related to the hallucinatory nature of the model^[2]



Prompting for IE

Recommended:

- standard IE tasks^[1]

Pros:

- Fine-tuned LLMs achieve good performance with **smaller models** compared to the prompting paradigm.

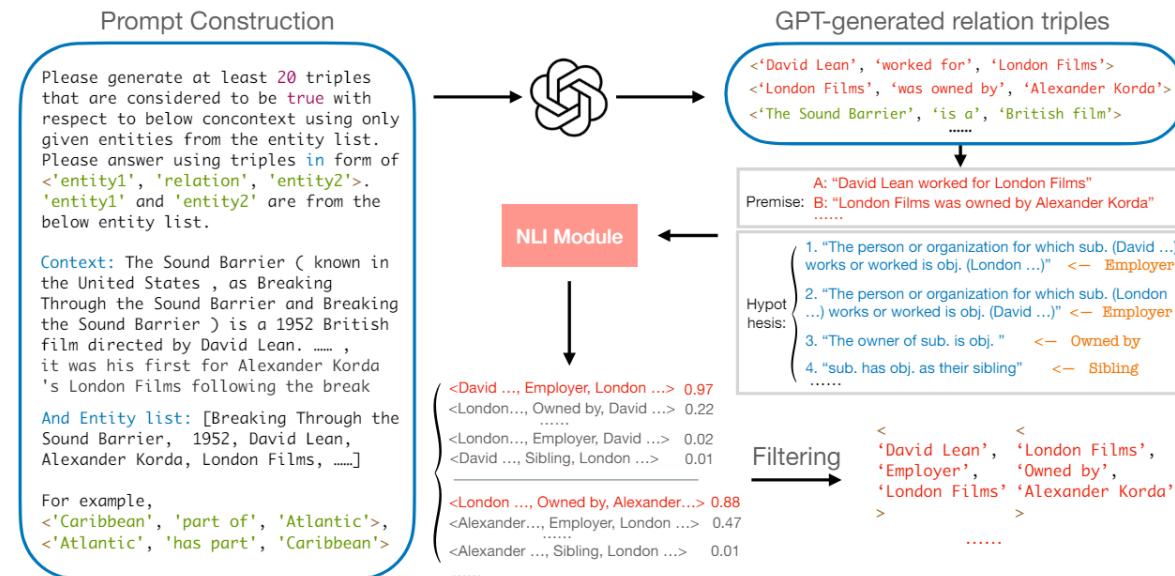
Cons:

- annotated datasets^[2,3]
- computational power needed for training

[1] Li, Bo, et al. "Evaluating ChatGPT's Information Extraction Capabilities: An Assessment of Performance, Explainability, Calibration, and Faithfulness." (2023).

[2] Li, Junpeng, Zixia Jia, and Zilong Zheng. "Semi-automatic data enhancement for document-level relation extraction with distant supervision from large language models." (2023).

[3] Zhang, Ruoyu, et al. "Llmaaa: Making large language models as active annotators." (2023).

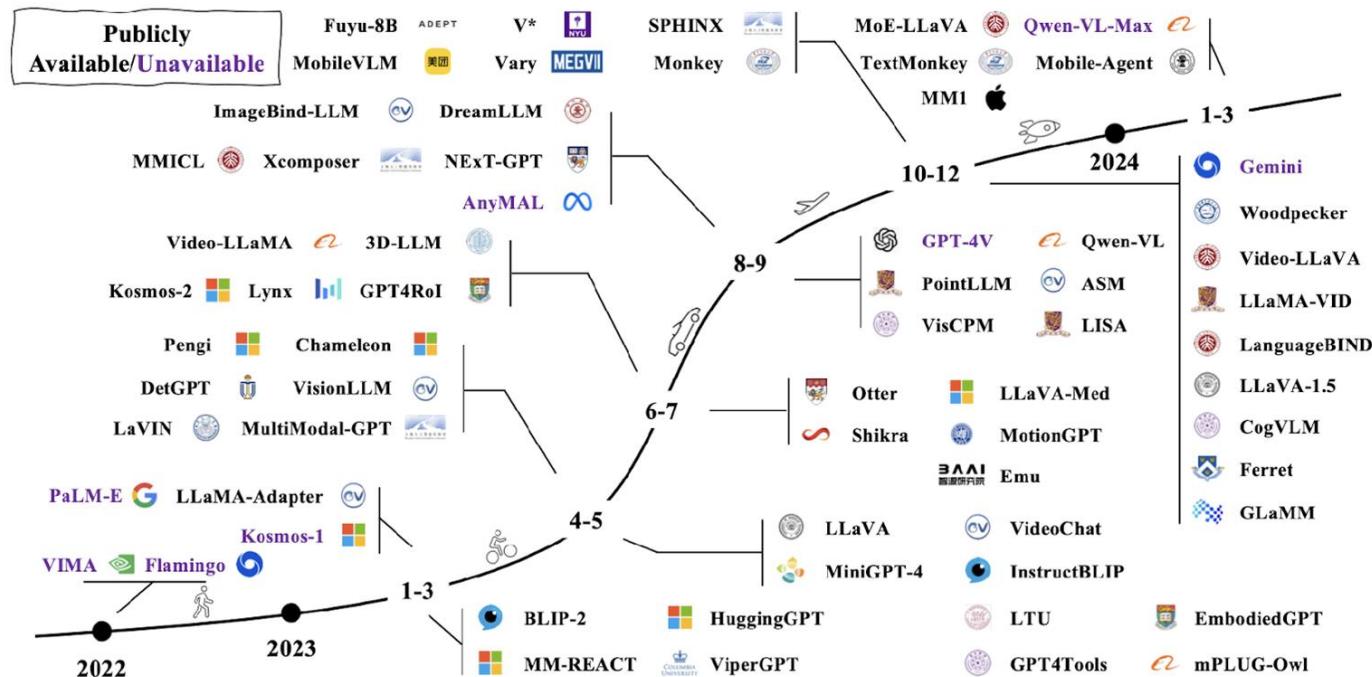


The figure shows a framework for synthesizing customized specific triples. The increasing focus on generating synthetic datasets for IE tasks is a clear indication of the general scarcity of such datasets^[2].



Multimodal Large Language Models

Large Language Models (LLMs) excel in tasks like text generation and encoding but they lack in understanding other data types. To address this issue, **Multimodal LLMs** (MLLMs) integrate multiple data types, overcoming the limitations of pure text models. They are LLMs-based that are able to receive, reason, and output with multimodal information. [2,3]

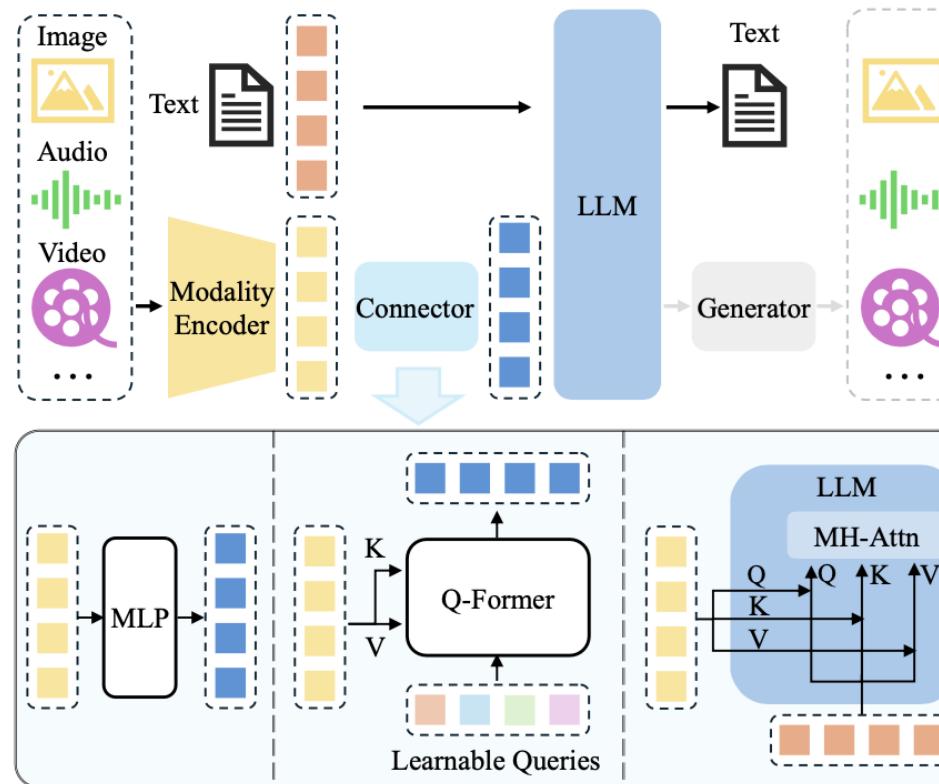


[2] Wu, Jiayang, et al. "Multimodal large language models: A survey." *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 2023.

[3] Yin, Shukang, et al. "A survey on multimodal large language models." *arXiv preprint arXiv:2306.13549* (2023).

Multimodal Large Language Models

A typical MLLM can be abstracted into three modules, *i.e.* a pre-trained modality encoder, a pre-trained LLM, and a modality interface to connect them. Some MLLMs also include a generator to output other modalities apart from text. [3]



[3] Yin, Shukang, et al. "A survey on multimodal large language models." arXiv preprint arXiv:2306.13549 (2023).



Image

In [4] has been conducted a comprehensive MLLM Evaluation benchmark MME. It measures both perception and cognition abilities on a total of 14 subtasks.

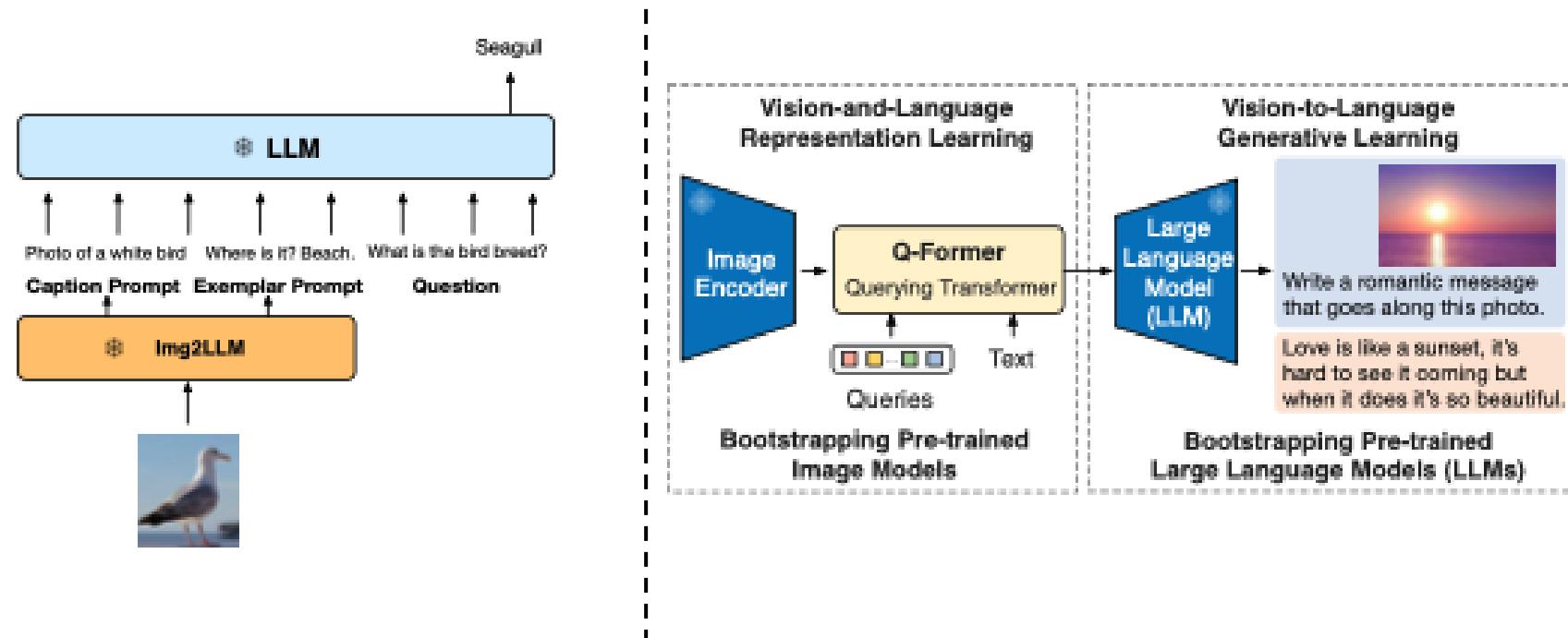
Perception (Coarse-Grained Tasks)		Perception (Fine-Grained Tasks)		
Existence	[Y] Is there a elephant in this image? [N] Is there a hair drier in this image?	[Y] Is there a refrigerator in this image? [N] Is there a donut in this image?	[Y] Is this movie directed by francis ford coppola ? [N] Is this movie directed by franklin j. schaffner ? [Y] Is this movie titled twilight (2008) ? [N] Is this movie titled the horse whisperer (1998) ?	
Count	[Y] Is there a total of two person appear in the image? [N] Is there only one person appear in the image?	[Y] Are there two pieces of pizza in this image? [N] Is there only one piece of pizza in this image?	[Y] Is the actor inside the red box called Audrey Hepburn ? [N] Is the actor inside the red box called Chris April ? [Y] Is the actor inside the red box named Jim Carrey ? [N] Is the actor inside the red box named Jari Kinnunen ?	
Position	[Y] Is the motorcycle on the right side of the bus? [N] Is the motorcycle on the left side of the bus.	[Y] Is the baby on the right of the dog in the image? [N] Is the baby on the left of the dog in the image?	[Y] Does this image describe a place of moat water ? [N] Does this image describe a place of marsh ? [Y] Is this picture captured in a place of gallery ? [N] Is this picture captured in a place of physics laboratory ?	
Color	[Y] Is there a red coat in the image? [N] Is there a yellow coat in the image?	[Y] Is there a red couch in the image? [N] Is there a black couch in the image?	[Y] Is this an image of Beijing Guozijian ? [N] Is this an image of Klinikkirche (Pfafferode) ? [Y] Is this a picture of Church of Saint Giles in Prague ? [N] Is this a picture of Pfarrkirche St. Martin an der Raab ?	
Perception (OCR Task)		Artwork		
OCR	[Y] Is the phone number in the picture "0131 555 6363" ? [N] Is the phone number in the picture "0137 556 6363" ?	[Y] Is the word in the logo "high time coffee shop" ? [N] Is the word in the logo "high tite cofeee shop" ?	[Y] Does this artwork belong to the type of still-life ? [N] Does this artwork belong to the type of mythological ? [Y] Is this artwork displayed in musée du louvre ? [N] Is this artwork displayed in galleria nazionale d'arte moderna e contemporanea ?	
Cognition (Reasoning Tasks)				
Commonsense Reasoning	[Y] Should I stop when I'm about to cross the street? [N] When I see the sign in the picture, can I cross the street?	[Y] Is there one real cat in this picture? [N] Is there two real cats in this picture?	[Y] Appropriate to translate into English ' classic taste '? [N] Appropriate to translate into English ' strawberry flavor '? 共同努力	[Y] Appropriate to translate into English ' work hard together '? [N] Appropriate to translate into English ' be filled with intrigue '?
Numerical Calculation	[Y] Is the answer to the arithmetic question in the image 65 ? [N] Is the answer to the arithmetic question in the image 56 ?	[Y] Should the value of "a" in the picture equal 3 ? [N] Should the value of "a" in the picture equal 2 ?	Code Reasoning	[Y] Python code. Is the output of the code ' Hello '? [N] Python code. Is the output of the code ' World '? [Y] Python code. Is the output of the code ' O '? [N] Python code. Is the output of the code ' T '?
Text Translation				

Common Problems

1. not following instructions;
2. lack of perception;
3. lack of reasoning;
4. object hallucination following instructions.

Image

In [5,6] is shown how it is possible to leverage on pre-trained large language models to enable zero-shot capabilities and computational efficiency. In the first one are used frozen, pre-trained LLMs to perform zero-shot VQA tasks. In the second one is used a frozen image encoder to learn visual features most relevant to text, and such visual features are connected to a frozen LLM for text generation.

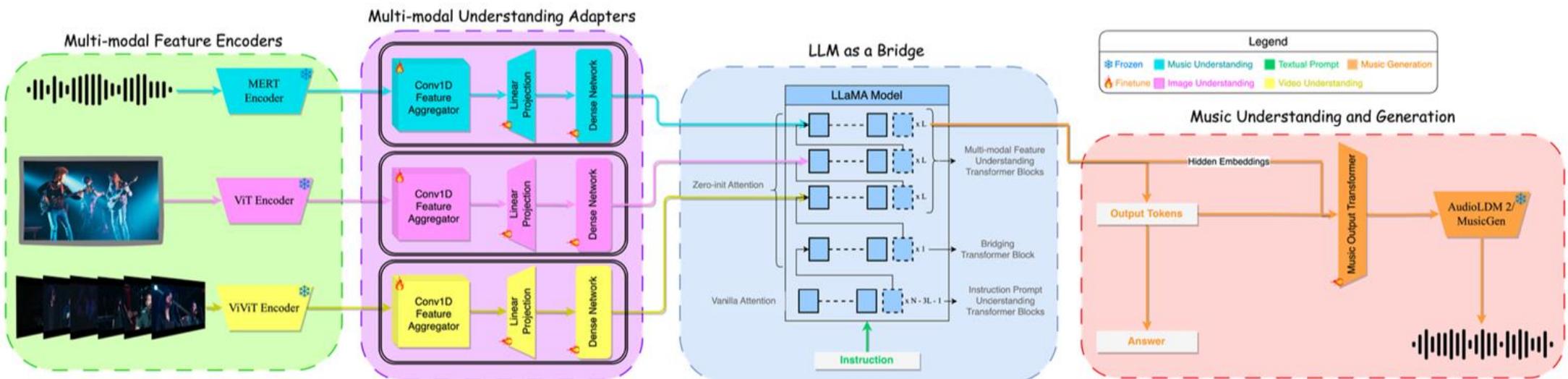


[5] Guo, Jiaxian, et al. "From images to textual prompts: Zero-shot vqa with frozen large language models." *arXiv preprint arXiv:2212.10846* (2022).

[6] Li, Junnan, et al. "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models." *International conference on machine learning. PMLR*, 2023.

Audio

In [7], it is demonstrated how pre-trained large language models can be utilized for audio information extraction, enabling multi-modal music understanding and generation. This approach uses pre-trained encoders to extract features from music, images, and videos, which are then processed by LLaMA 2 to generate contextually relevant music outputs. This method integrates various modalities to improve the relevance and quality of music generation, achieving superior performance compared to state-of-the-art models.

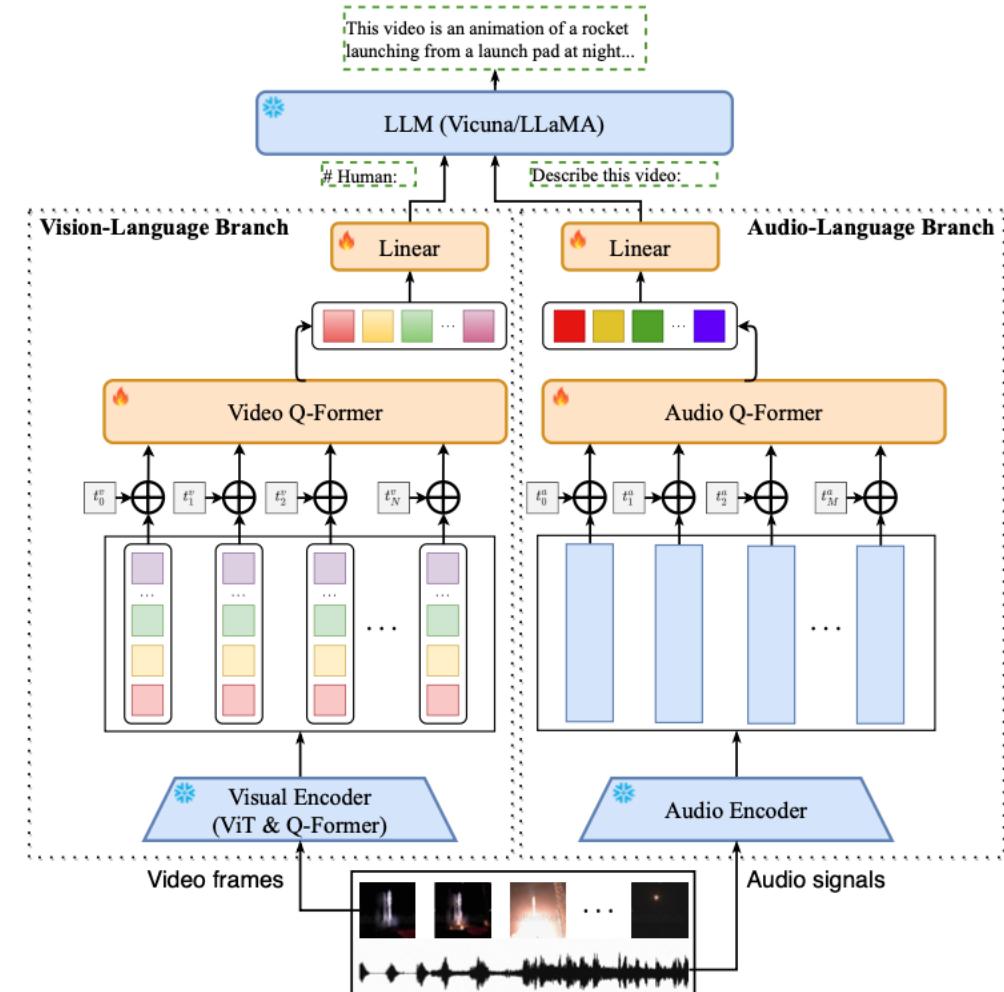


[7] Hussain, Atin Sakkeer, et al. "M \$^{2} \\$ UGen: Multi-modal Music Understanding and Generation with the Power of Large Language Models." arXiv preprint arXiv:2311.11255 (2023).



Video

In [8], the focus is on enhancing video comprehension using pre-trained large language models. Video-LLaMA, a multi-modal framework, empowers LLMs to understand both visual and audio content in videos. By leveraging frozen pre-trained visual and audio encoders alongside frozen LLMs, Video-LLaMA addresses the challenges of capturing temporal changes in visual scenes and integrating audio-visual signals. A Video Q-former and Audio Q-former are introduced to align visual and audio features with the LLM's embedding space, respectively. The model is pre-trained on massive video/image-caption pairs and fine-tuned with high-quality visual-instruction datasets, resulting in superior video understanding and meaningful response generation based on both visual and audio information present in videos.



[8] [Zhang, Hang, Xin Li, and Lidong Bing. "Video-llama: An instruction-tuned audio-visual language model for video understanding." arXiv preprint arXiv:2306.02858 \(2023\).](#)

A vision for the future

Transforming IS engineering with low design approaches and design assistants



SAPIENZA
UNIVERSITÀ DI ROMA

The traditional approach

- IS Engineering has been developed as discipline of design
 - Principled approaches to develop artefacts, in which refinements from abstract (conceptual) till concrete (executable/runnable) artefacts are carried out in order to achieve a system
 - Waterfall, then iterative, then agile
 - (practically) the quality of the designer (team) partly impact on the quality of the system
 - Methodologies and practices support the designer (team), but the skills and capabilities and knowledge is solely in his/her/their minds
 - Heavily design-based approach (design is crucial)



Low design and assistants

- With GenAI/LLMs we can move to a different approach
 - A design has a *smart twin* (not a digital twin, that's another story ☺) which can iteratively support his/her work
 - A precise design is no more crucial
 - The designer can quickly try and iterate the production of the artefact s/he is in charge of, being supported by the assistant
- In analogy with low code approaches to software development, we define this approach «low design»
- Still far away from there, but initial symptoms are here (see the survey and our research)

Implications

IS Engineering can become:

- faster
- more standardized
 - A potential issue for innovation and disruptive ideas
- proprietary (it depends on which LLMs we are using)
 - Openness of AI models is an issue per se
- still more costly and environmental impactfull (cf. the footprint of LLMs and their costs if used via paid APIs)

Three current researches

NL2ProcessOps

NL2DataGen

ArchiGPT

NL2ProcessOps

Monti, F., Leotta, F., Mangler, J., Mecella, M & Rinderle-Ma, S. (2024). NL2ProcessOps: Towards LLM-guided Code Generation for Process Execution. Accepted in BPM Forum 2024.



SAPIENZA
UNIVERSITÀ DI ROMA

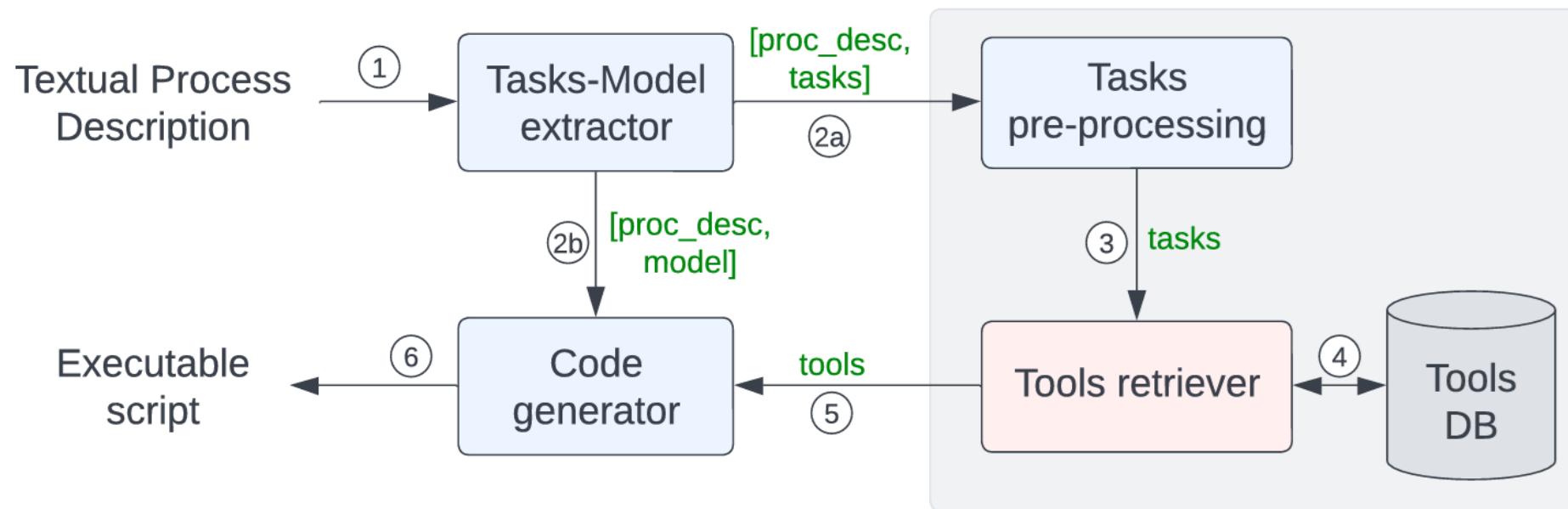
ProcessOps

- Challenge: adapt rapidly and effectively to changes that may require the need of new processes poorly specified and defined
- ProcessOps
 - Derived from *process* and *operations* (like DevOps ☺)
 - agile approach in which new processes can be easily deployed and modified to accommodate changing requirements
 - process models are in the form of **scripts** and manually **deployed** into PEE with minimal interventions transforming abstract primitives in the script into runnable runtime calls

NL2ProcessOps

- Work with textual process descriptions
- Goals:
 - Extract the control flow → Mermaid.js process model
 - Retrieve tools' operations associated with tasks → services
 - Generate executable script (*process script*) → Python code
- Leverages RAG and tool to generate the process script
 - RAG-based approach to retrieve necessary tools' operations
 - Tools offers operations (considered as services) executing tasks of the process

Architecture





Prototype

- Built on top of Langchain
- Uses OpenAI GPT-4
 - gpt-4-turbo-preview
 - Temperature set to 0
- Tools DB relies on Chroma DB
 - text-embedding-ada-002 OpenAI model to embed tools operations descriptions
 - Tools DB containing 62 tools operations based on a set of processes contained in [1][2]
- Prompt Engineering
 - Role
 - Task description
 - Knowledge injection
 - Few-shot
 - Output format

```
SetSpeedDieMachine = {  
    "description": "Set the speed of the die-cutting machine.",  
    "more_details": "It takes as input the speed. It returns a  
        boolean value, True if the speed has been set,  
        False otherwise.",  
    "input_parameters": ["speed:int"],  
    "output_parameters": ["speed_set:bool"],  
    "tool": "die_machine" }
```



[1] <https://zenodo.org/records/7783492>

[2] <https://huggingface.co/datasets/patriziobellan/PET>



Demo

1. Launch NL2ProcessOps
2. Prompt the textual process description
 - The calibration process of cardboard production consists of continuously capturing a photo of the cardboard being produced. Each photo is analyzed to check if all the markers identified are ok. If markers are not ok, the calibration process continues. If the markers are ok, the speed of the die-cutting machine is set to 10000 RPM and the process ends.
3. Invoke the LLM and wait for the response 😊

```
(pyllm) C:\Users\flavi\Desktop\CAISE24TutorialLLM\NL2ProcessOps\src>I
```

Launch NL2ProcessOps



SAPIENZA
UNIVERSITÀ DI ROMA

1. Launch NL2ProcessOps

1

(pyllm) > python ProcessLLM.py

2. Prompt the textual process

2

Enter a process description:

The calibration process of cardboard production consists of continuously capturing a photo of the cardboard being produced. Each photo is analyzed to check if all the markers identified are ok. If markers are not ok, the calibration process continues. If the markers are ok, the speed of the die-cutting machine is set to 10000 RPM and the process ends.

3. Invoke the LLM and wait for the response 😊



3

-----PROCESS SCRIPT-----

```
import numpy as np

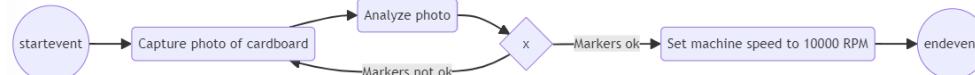
def calibration_process():
    while True:
        image = CaptureImage.call()
        markers_ok = CheckMarkers.call(image=image)

        if markers_ok:
            speed_set = SetSpeedDieMachine.call(speed=10000)
            if speed_set:
                print("Machine speed set to 10000 RPM successfully.")
            else:
                print("Failed to set machine speed.")
            break
        else:
            print("Markers not ok, capturing another photo.")

    if __name__ == "__main__":
        calibration_process()
```

-----MODEL-----

```
graph LR
1:startevent((startevent)) --> 2:task:(Capture photo of cardboard)
2:task --> 3:task:(Analyze photo)
3:task --> 4:exclusivegateway:{x}
4:exclusivegateway{x} -- "Markers not ok" --> 2:task
4:exclusivegateway{x} -- "Markers ok" --> 5:task:(Set machine speed to 10000 RPM)
5:task --> 6:endevent((endevent))
```



-----TOOLS-----

```
{"name": "CaptureImage", "description": "Capture of a photo of the cardboard from the camera.This tool takes no input and returns an image captured from the the camera. The image is returned as a numpy matrix.", "input_parameters": [], "output_parameters": ["image:np.matrix"], "actor": "camera"}
{"name": "StoreCardboardRoll", "description": "Worker stores the cardboard roll in the warehouse.It takes as input the location. It returns no output.", "input_parameters": ["location:str"], "output_parameters": [], "actor": "worker"}
{"name": "CheckMarkers", "description": "Analysis of the markers on a cardboard image.This tool takes as input an image depicting a cardboard. It returns a boolean indicating if markers are present on the cardboard.", "input_parameters": ["image:np.matrix"], "output_parameters": ["markers_ok:bool"], "actor": "vision_is"}
{"name": "SetSpeedDieMachine", "description": "set the speed of the die cutting machine.It takes as input the speed. It returns a boolean value, True if the speed has been set, False otherwise.", "input_parameters": ["speed:int"], "output_parameters": ["speed_set:bool"], "actor": "die_machine"}
```

NL2DataGen



SAPIENZA
UNIVERSITÀ DI ROMA

Introduction

- Enterprises are increasingly becoming data-centric and data intensive
 - They produce and consume large amounts of data
- Understanding (analytics) such vast amount of data is difficult
 - technical knowledge required (e.g. coding skills)
 - might require to analyze several data sources
 - time consuming
- LLMs are good at using external tools!!
- **Can we use LLMs to answer analytics-related questions by generating structured data from these data sources?**



Our solution:

- Key points
 - The user submits a natural language query
 - Produce a pipeline
 - executable code that generates a tables
 - Leverages data services (i.e. tools) to generate the pipeline
 - APIs or other kind of interfaces to interact with the data sources
 - User feedback
 - The user can interact with the solution to correct or refine the pipeline

Workflow

- **Step 0:** The user submits a query
- **Step 1:** Retrieve relevant data can help the LLM in answering the query (e.g. similar pipelines, APIs to access data sources, etc.)
- **Step 2:** Use an LLM to generate a pipeline
- **Step 3:** The pipeline is iteratively refined until it satisfies the user requirements
- **Step 4:** The pipeline is stored in the repository

Step 0: Submit query

Query: "Number of cardboards in each stack, percentage of defected cardboards and average speed of the diecutting machine 3"

Step 1: Retrieve similar pipelines from the repository

```
def count_cardboards_in_stack()  
def count_errors_in_stack()  
...
```

Step 2: Generate pipeline



Cardboards are produced in batches called "stacks" ...

```
def production_info(...)  
...  
for cardboard in stack:  
    img = get_card_image()  
    errors = find_errors(img)  
...
```

Step 3: Generate intermediate table

Feedback: Errors should be reported as ratios. Missing column related to the number of cardboards.



stack id	errors	speed
7	4	1.8
8	7	2.1

✓

stack id	num card	errors	speed
7	40	10%	1.8
8	35	20%	2.1

Step 4: Save pipeline to the repository

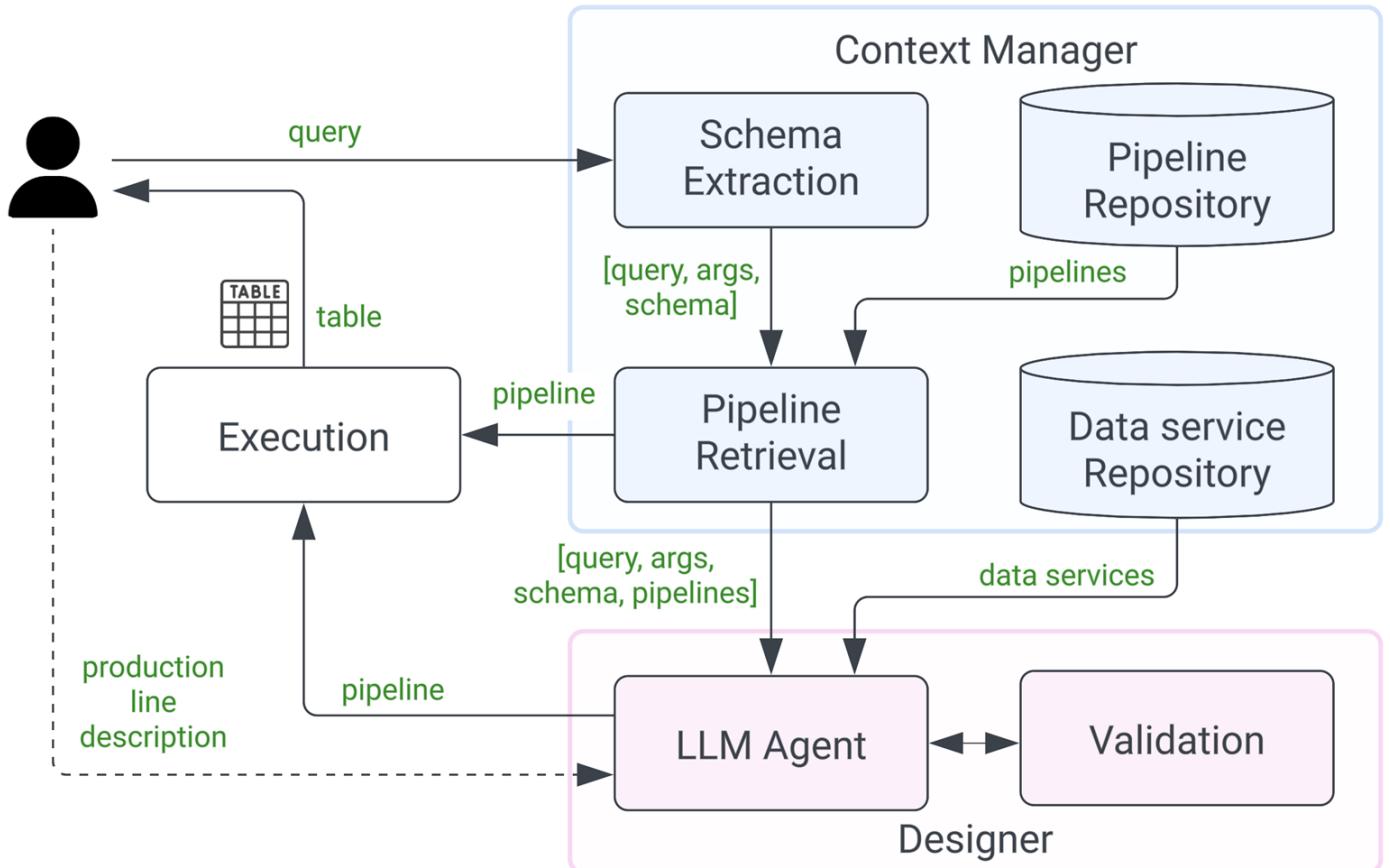
```
def count_cardboards_in_stack()  
def count_errors_in_stack()  
def production_info(..)  
...
```



SAPIENZA
UNIVERSITÀ DI ROMA

Proof of Concept

- Simplified version
 - with no feedback
 - Pipeline and Data service repository populated with APIs related to the cardboard manufacturing domain
- Uses GPT4-turbo as the LLM Agent to generate a pipeline
 - represented as a Python script



Demo

1. Launch NL2DataGen
2. Prompt the query
 - Generate a table containing the max speed of the diemachine with id 25 over a time span of 30 seconds.
3. Invoke the LLM and wait for the pipeline to be generated and executed

I

Launch NL2DataGen

```
(pyllm) C:\Users\flavi\Desktop\CAISE24TutorialLLM\NL2DataGen\src\nl2datagen>
```

1. Launch NL2DataGen → 0 (pyllm) > python nl2datagen.py
2. Prompt the query → 1 Provide the query:
Generate a table containing the max speed of the die machine with id 25 over a time span of 30 seconds.
3. Invoke the LLM and wait for the pipeline to be generated and executed ☺

2

```
-----PIPELINE-----
import time
from threading import Thread
from nl2datagen.documentation.database import GetDiecutterChipIdFromDiecutterId
from nl2datagen.documentation.database import GetDiecutterIdFromDiemachineId
from nl2datagen.documentation.diecutter import GetCurrentDataFromChip

def get_max_speed(chip_id, time_span_seconds):
    max_speed = 0
    start_time = time.time()
    while time.time() - start_time < time_span_seconds:
        current_data = GetCurrentDataFromChip.call(chip_id=chip_id)
        current_speed = current_data["speed"]
        if current_speed > max_speed:
            max_speed = current_speed
        time.sleep(1) # Ensuring the data is refreshed
    return max_speed
```

```
def pipeline_function(diemachine_id, time_span_seconds):
    # Get the diecutter id from the diemachine id
    diecutter_id = GetDiecutterIdFromDiemachineId.call(diemachine_id=diemachine_id)
    # Get the chip id from the diecutter id
    chip_id = GetDiecutterChipIdFromDiecutterId.call(diecutter_id=diecutter_id)

    # Create a thread to get the max speed over the time span
    max_speed_thread = Thread(target=get_max_speed, args=(chip_id, time_span_seconds))
    max_speed_thread.start()
    max_speed_thread.join()

    max_speed = get_max_speed(chip_id, time_span_seconds)
    return [{"Max Speed": max_speed}]
```



```
-----RESULT-----
+---+-----+
|   |     Max Speed |
|---+-----|
| 0 |         9 |
+---+-----+
```



ArchiGPT



SAPIENZA
UNIVERSITÀ DI ROMA

Goal

- Starting from User Stories and a short description of the system,
generating a design of the architecture of the system
- The final outcome is achieved through a combination of **task-specific sub-models**, crafted using prompt engineering techniques and trained on university-grade projects
- **University-grade projects** have been sanitized and documented to be used **to train the model**



Main Concepts

Project Standard

- Template for the specifications of the system, ranging from an overview of the architecture to details of the single services
- Serves as a template for both the documentation of the projects used for training and for the architecture design documents produced by the Architectural Design model

Projects Pre-Processing Module

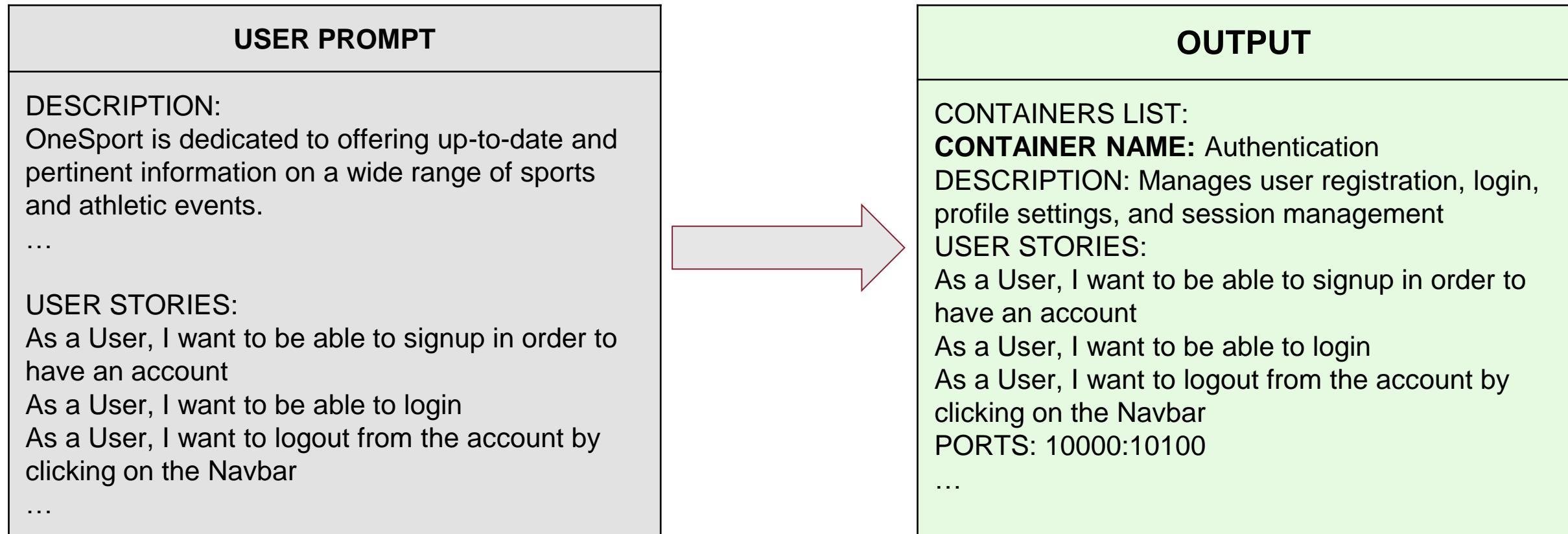
Generates documentation in Project Standard format for the university projects, in order to use them to train the Architectural Design model

Architectural Design Module

Starting from User Stories, the model generates the architectural design in Project Standard format

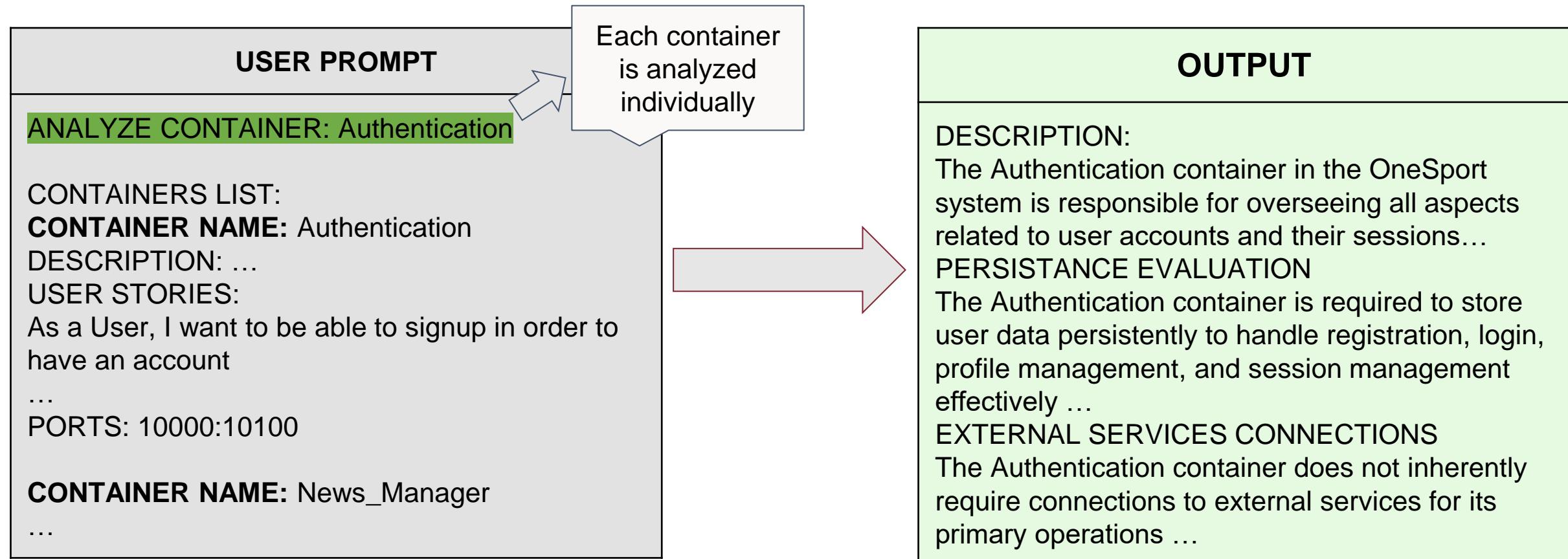


First Step - From User Stories to containers



Starting from a brief description and User Stories, the LLM crafts a **list of the system's containers**, each one with a brief description of its role, User Stories to fulfill and a ports range

Second Step - Containers' details enrichment



Using the result of the first step as the User Prompt for the second step, the LLM crafts **more details on the container's behaviour** to enhance precision and avoid hallucination in the following steps

Third Step – Containers microservices generation

USER PROMPT
CONTAINER NAME: Authentication
USER STORIES:
...
PORTS: ...
DESCRIPTION:
...
PERSISTANCE EVALUATION
...
EXTERNAL SERVICES CONNECTIONS
...



OUTPUT
MICROSERVICES LIST:
MICROSERVICE: authService
TYPE: backend
DESCRIPTION: Manages user registration, login, logout, and session handling. This service provides endpoints for ...
PORT: 10010
...
MICROSERVICE: sessionDB
TYPE: database
DESCRIPTION: Handles persistent storage of user credentials, session tokens ...
PORTS: 10020
...

Filtering only the informations previously generated on one container and using them as User Prompt, the LLM generates a **list of all the microservices** within the container

Concluding remarks

Do not beat us up now ☺



SAPIENZA
UNIVERSITÀ DI ROMA

The tutorial by ChatGPT (1)



Assume you are a professor who need to give a tutorial for a conference about engineering information systems through LLMs and AI. Which outline do you propose ?



Certainly! Below is a comprehensive outline for a tutorial on "Engineering Information Systems through LLMs and AI" suitable for a conference setting. This outline aims to cover the fundamental concepts, technical aspects, practical applications, and future trends in the field.

Tutorial Outline: Engineering Information Systems through LLMs and AI

1. Introduction

- 1.1 Welcome and Objectives
 - Brief introduction of the speaker
 - Overview of tutorial goals and structure
- 1.2 Importance of Information Systems in Engineering
 - Historical perspective
 - Current trends and challenges

2. Foundations of LLMs and AI in Information Systems



- 2.1 Basics of Large Language Models (LLMs)

2. Foundations of LLMs and AI in Information Systems

• 2.1 Basics of Large Language Models (LLMs)

- What are LLMs?
- Key concepts: tokens, embeddings, transformers
- Popular models: GPT-4, BERT, T5, etc.

• 2.2 Fundamentals of AI and Machine Learning

- Definitions and key concepts
- Types of machine learning: supervised, unsupervised, reinforcement learning
- Overview of neural networks and deep learning

3. Designing and Building Information Systems with LLMs and AI

• 3.1 System Architecture and Components

- Overview of typical system architecture
- Integrating LLMs into information systems
- Key components: data pipeline, model integration, user interface

• 3.2 Data Management and Preprocessing

- Data collection and sourcing
- Data cleaning and preprocessing techniques
- Handling large datasets and ensuring data quality

The tutorial by ChatGPT (2)

- **3.3 Model Training and Fine-Tuning**
 - Selecting the right model for the task
 - Training and fine-tuning LLMs
 - Transfer learning and domain adaptation
- **3.4 Deployment and Integration**
 - Model deployment strategies
 - Integration with existing systems
 - Scalability and performance optimization
- **4. Practical Applications and Case Studies**
 - **4.1 Natural Language Processing Applications**
 - Text generation and summarization
 - Sentiment analysis and opinion mining
 - Information retrieval and question answering
 - **4.2 Intelligent Automation in Engineering**
 - Predictive maintenance and anomaly detection
 - Process optimization and decision support systems
 - Automated documentation and reporting
- **4.3 Real-World Case Studies**
 - Case study 1: Implementing an AI-powered customer support system
 - Case study 2: Enhancing engineering workflows with AI-driven insights
 - Case study 3: Building a knowledge management system using LLMs
- **5. Challenges and Ethical Considerations**
 - **5.1 Technical Challenges**
 - Model interpretability and explainability
 - Handling biased and imbalanced data
 - Ensuring robustness and reliability
 - **5.2 Ethical and Societal Implications**
 - Data privacy and security concerns
 - Ethical AI and responsible use of technology
 - Addressing bias and fairness in AI systems



The tutorial by ChatGPT (3)

6. Future Trends and Research Directions

- 6.1 Emerging Technologies and Innovations

- Advances in LLMs and AI algorithms
- Integration of AI with IoT and edge computing
- Role of AI in Industry 4.0 and smart manufacturing

- 6.2 Research Opportunities

- Open research problems in LLMs and AI for information systems
- Collaboration and interdisciplinary research
- Funding and resources for research in this field

7. Hands-on Session

- 7.1 Setting up the Environment

- Tools and frameworks: TensorFlow, PyTorch, Hugging Face, etc.
- Data sources and preparation

- 7.2 Building a Simple Information System with LLMs

- Step-by-step guide to building and deploying a basic application
- Interactive coding session and Q&A

- 7.3 Troubleshooting and Best Practices

- Common issues and solutions



- 7.1 Setting up the Environment

- Tools and frameworks: TensorFlow, PyTorch, Hugging Face, etc.
- Data sources and preparation

- 7.2 Building a Simple Information System with LLMs

- Step-by-step guide to building and deploying a basic application
- Interactive coding session and Q&A

- 7.3 Troubleshooting and Best Practices

- Common issues and solutions
- Best practices for developing robust and scalable systems

8. Conclusion and Q&A

- 8.1 Recap of Key Takeaways

- Summary of main points covered
- Final thoughts and recommendations

- 8.2 Open Q&A Session

- Addressing audience questions
- Providing additional resources and references



LLMs for IS Eng

- GenAI/LLMs have been proposed as a powerful tool for mostly all phases of IS Engineering (from strategic planning to monitoring and analytics)
 - Still scattered works
 - Not yet a clear methodology
 - Not yet a unified vision
- Our claim is that LLMs can become a powerful assistant for designers
 - *Human-in-the-loop* is essential
 - *Openess* is a crucial point