TDDE19 Advanced Project Course – AI and Machine Learning

AI Assisted Software Engineering

Mattias Tiger (PhD, AI Researcher)

AI and Integrated Computer Systems (AIICS),
Department of Computer Science

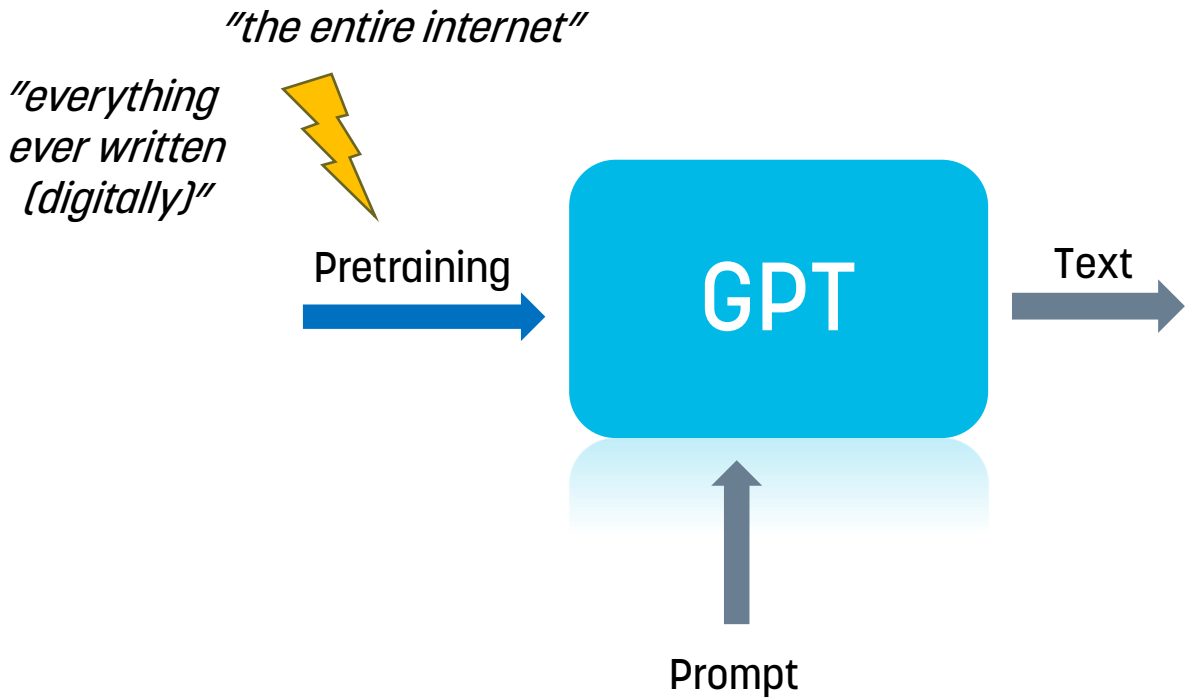
mattias.tiger@liu.se

LINKÖPINGS UNIVERSITET

ARTIFICIAL INTELLIGENCE AND INTEGRATED COMPUTER SYSTEMS
AIICS

ChatGPT

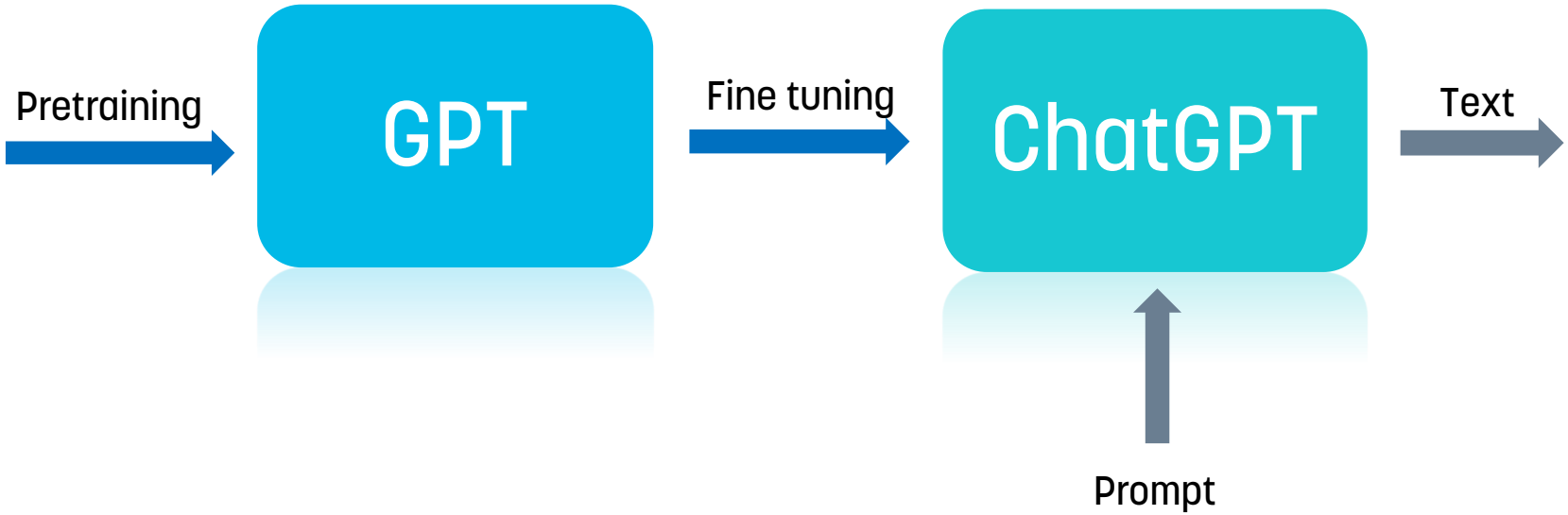
- Large Language Model (LLM)



Funded by the European Union

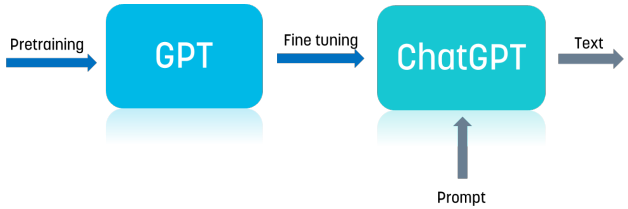
ChatGPT

- Large Language Model (LLM)
- Instruction-tuned LLM



Funded by the European Union

Generative AI (e.g. LLMs)



Large Language Models (LLMs) are trained on enormous bodies of text. *Hard to grasp what is in it!*

> The result is a lossy compression of commonly occurring sentence structures and word-chains (next-word-prediction: N-gram for really large N)

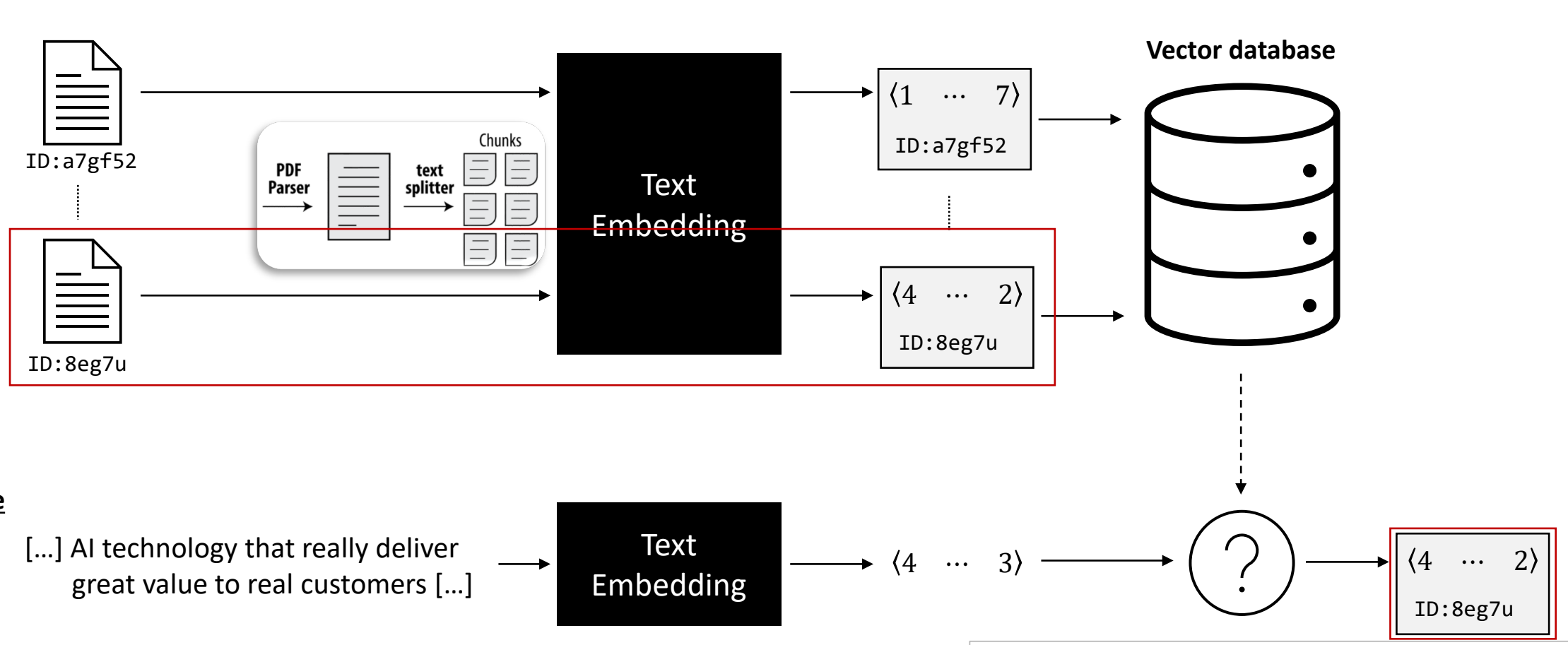
This (approximation) allow an LLM to generate **new text** not part of the training data, but which is **similar** to the text it was trained on.

> **Easy:** Create text with specific style (or style transfer)
>
> **Hard:** Create text that is factful

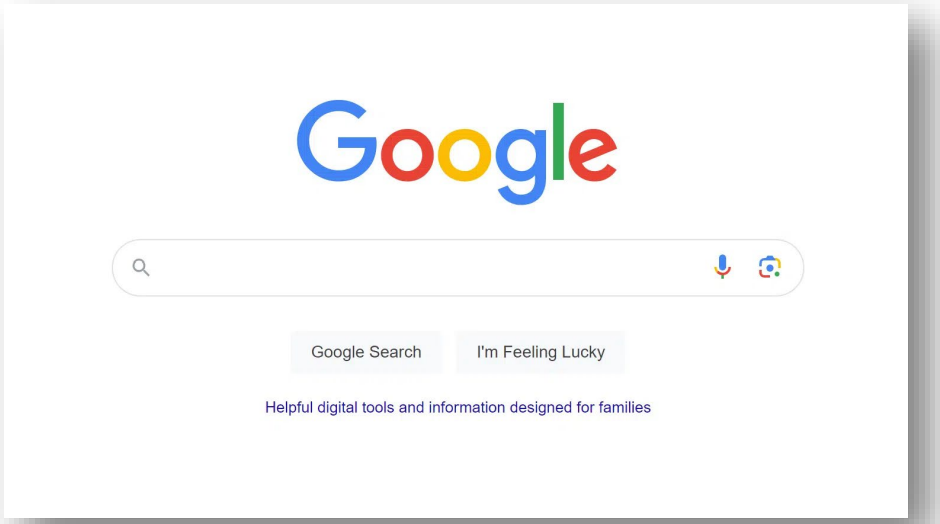
General rule:

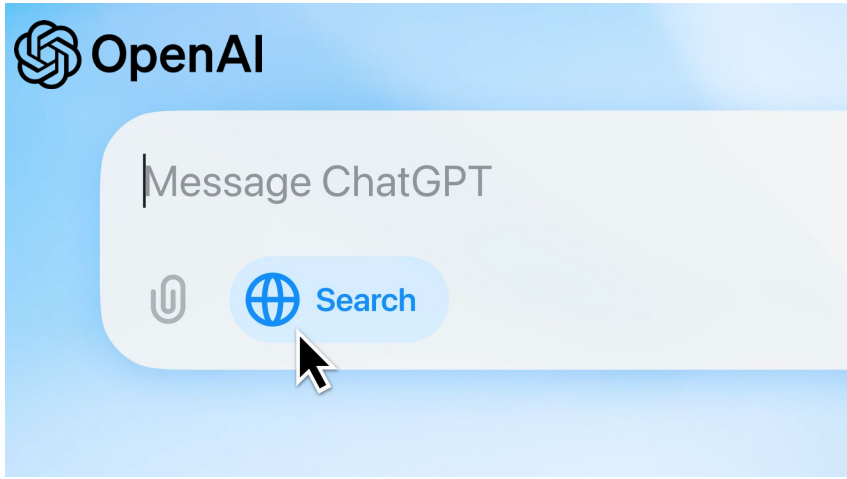
The output must be verified by the user!

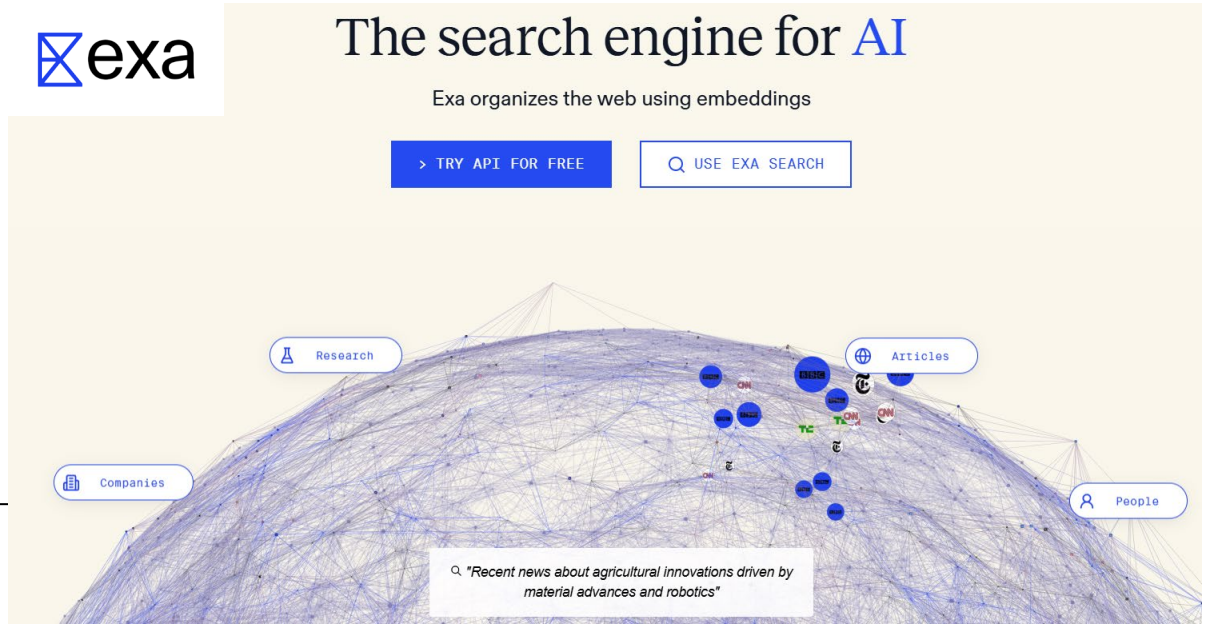
Retrieval (information retrieval)



Retrieval | Exempel: Re-inventing Search

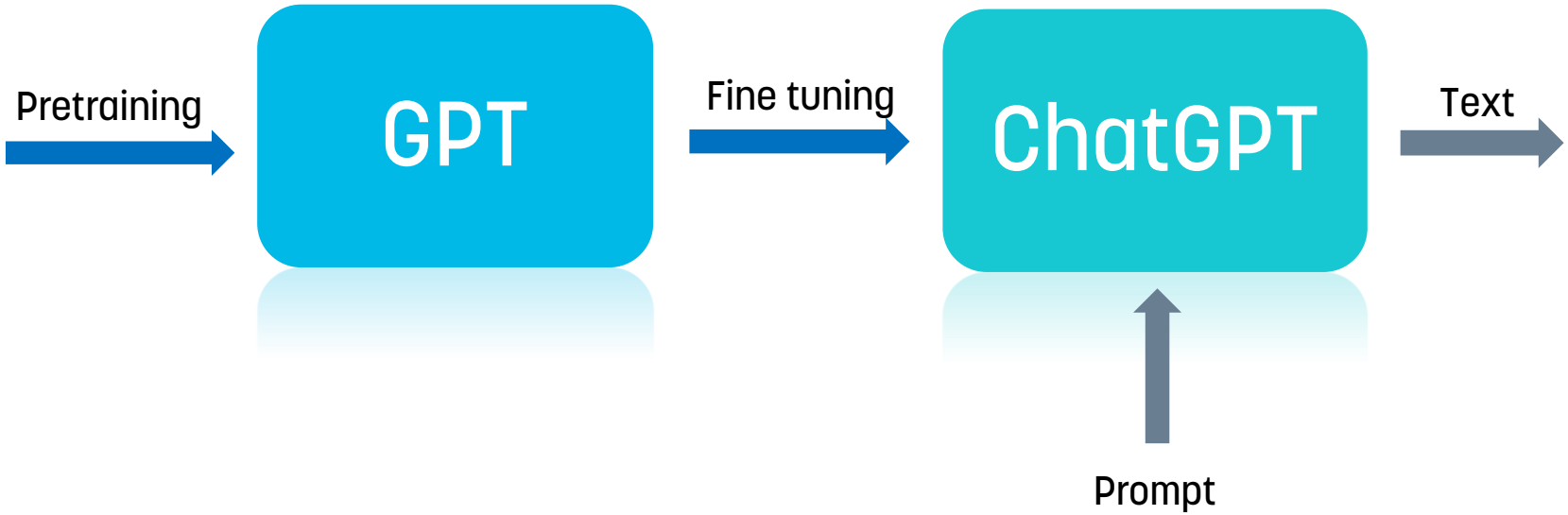






ChatGPT

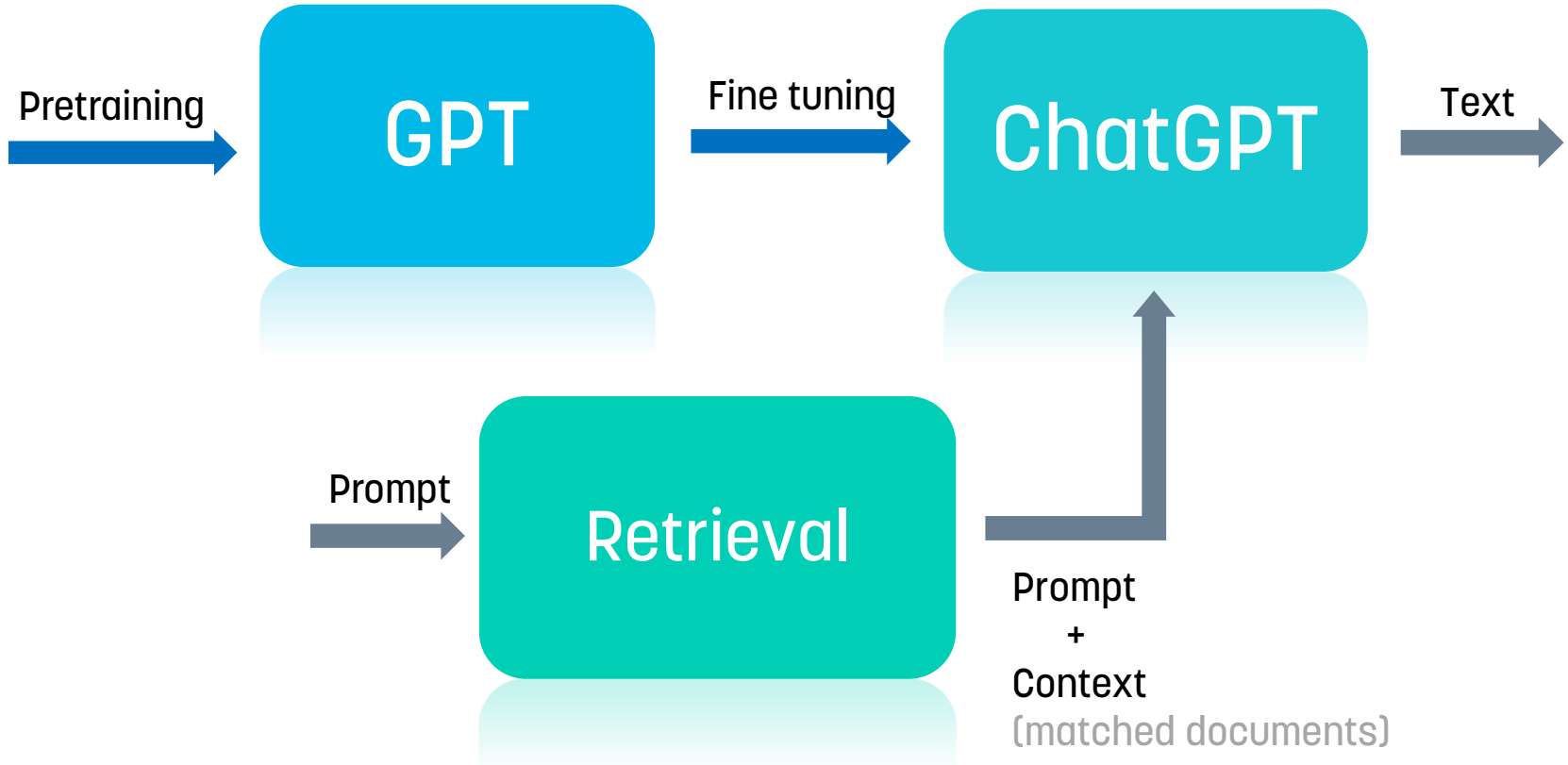
- Large Language Model (LLM)
- Instruction-tuned LLM
- Retrieval Augmentation Generation (RAG)



Funded by the European Union

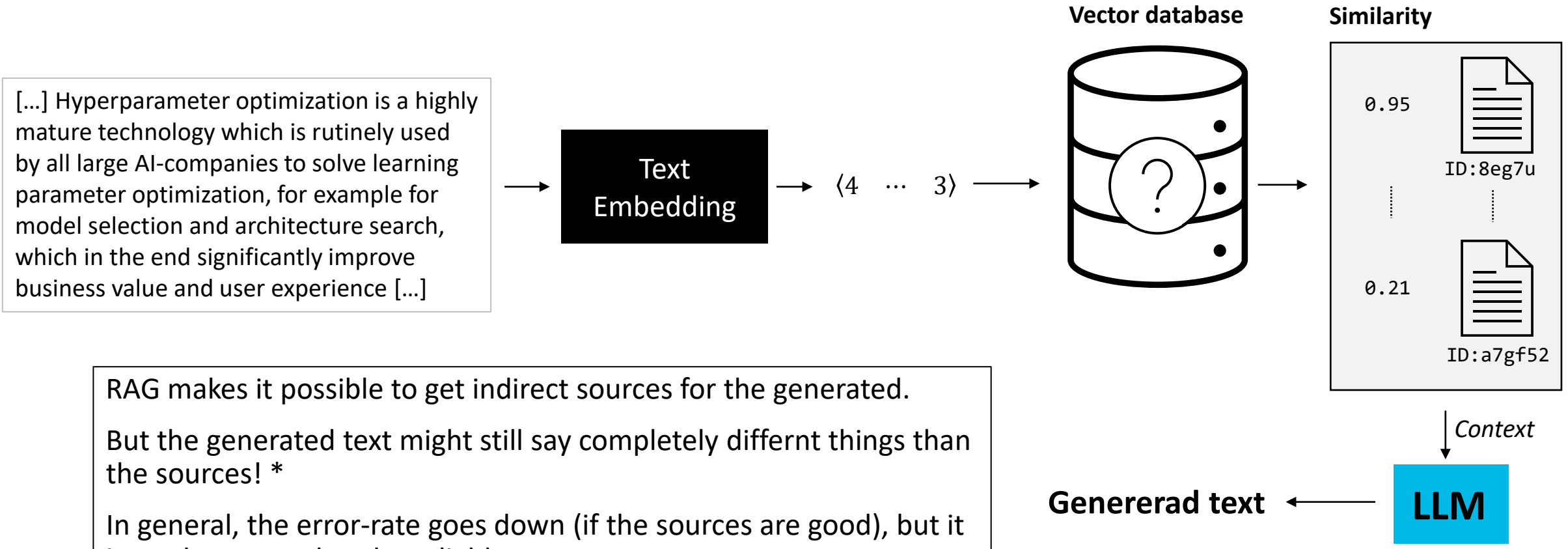
ChatGPT | GPT-4o

- Large Language Model (LLM)
- Instruction-tuned LLM
- Retrieval Augmentation Generation (RAG)



Funded by
the European Union

Retrieval Augmentation Generation (RAG)



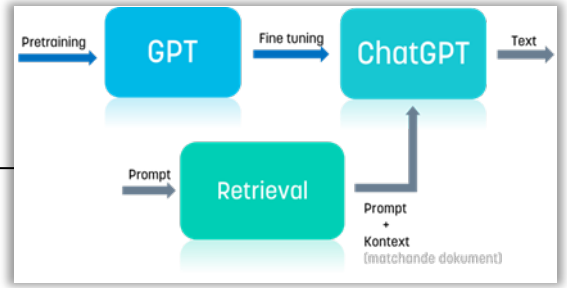
RAG makes it possible to get indirect sources for the generated.

But the generated text might still say completely differnt things than the sources! *

In general, the error-rate goes down (if the sources are good), but it is not low enough to be reliable.

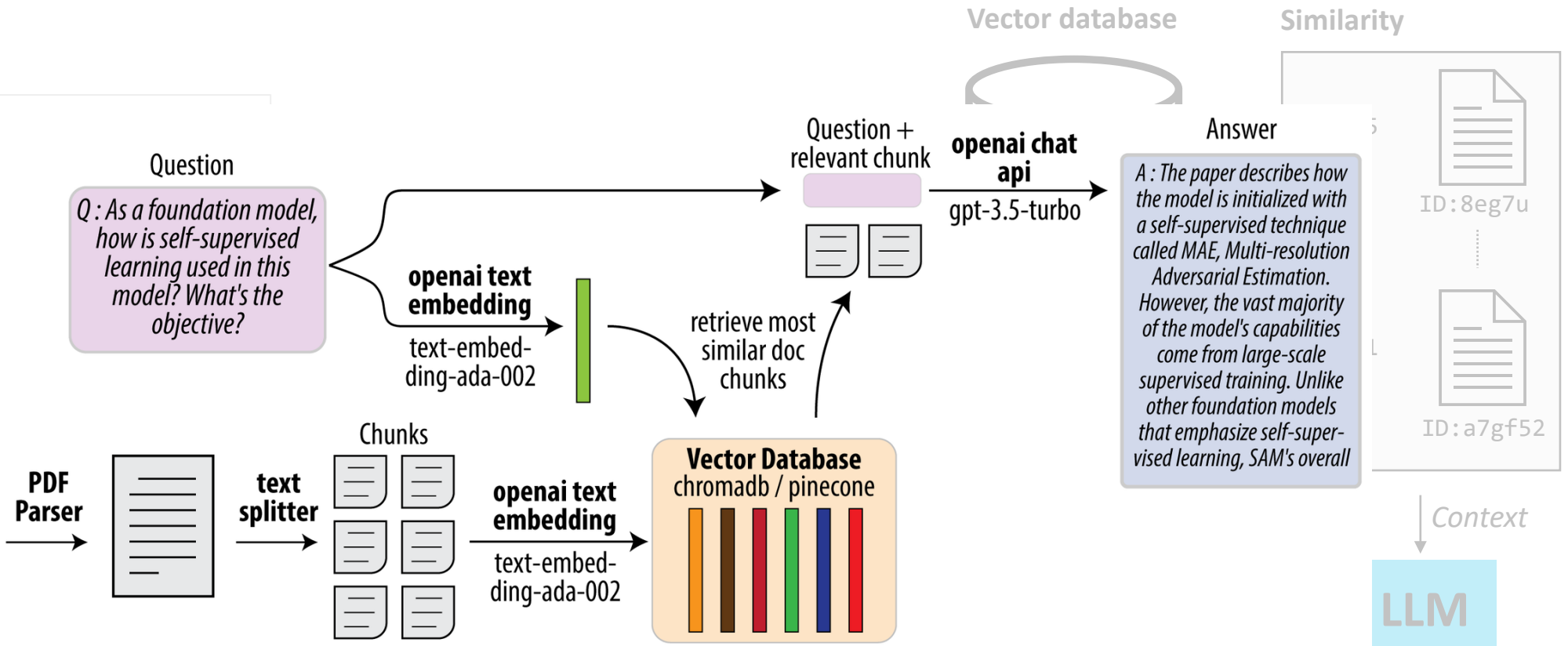
Also: Relevant sources can be missed (text embedding limitations).

* Modern retrieval can often be more useful than full RAG.



Retrieval Augmentation Generation (RAG)

[...] Hyperparameter o
mature technology wh
by all large AI-compani
parameter optimizatio
model selection and ar
which in the end signif
business value and use



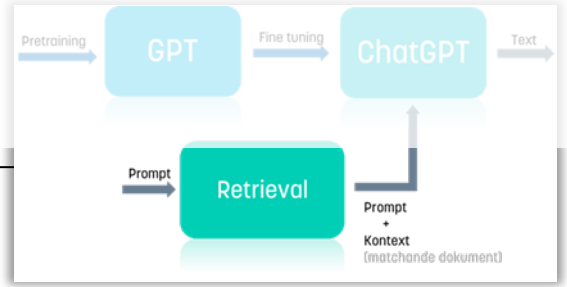
RAG makes

But the gen
the sources

In general, t
is not low enough to be reliable.

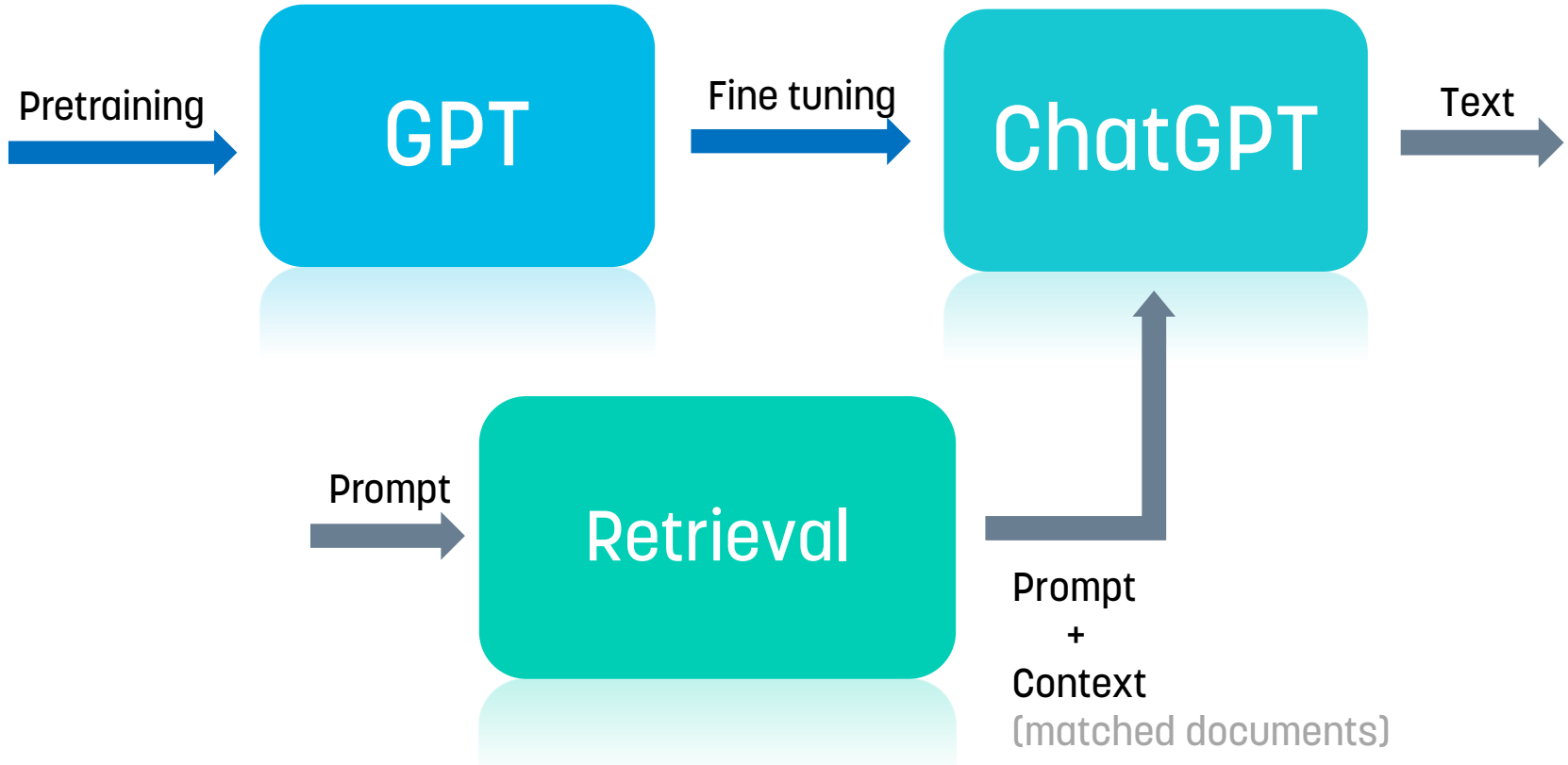
Also: Relevant sources can be missed (text embedding limitations).

* Modern retrieval can often be more useful than full RAG.



ChatGPT | GPT-4o

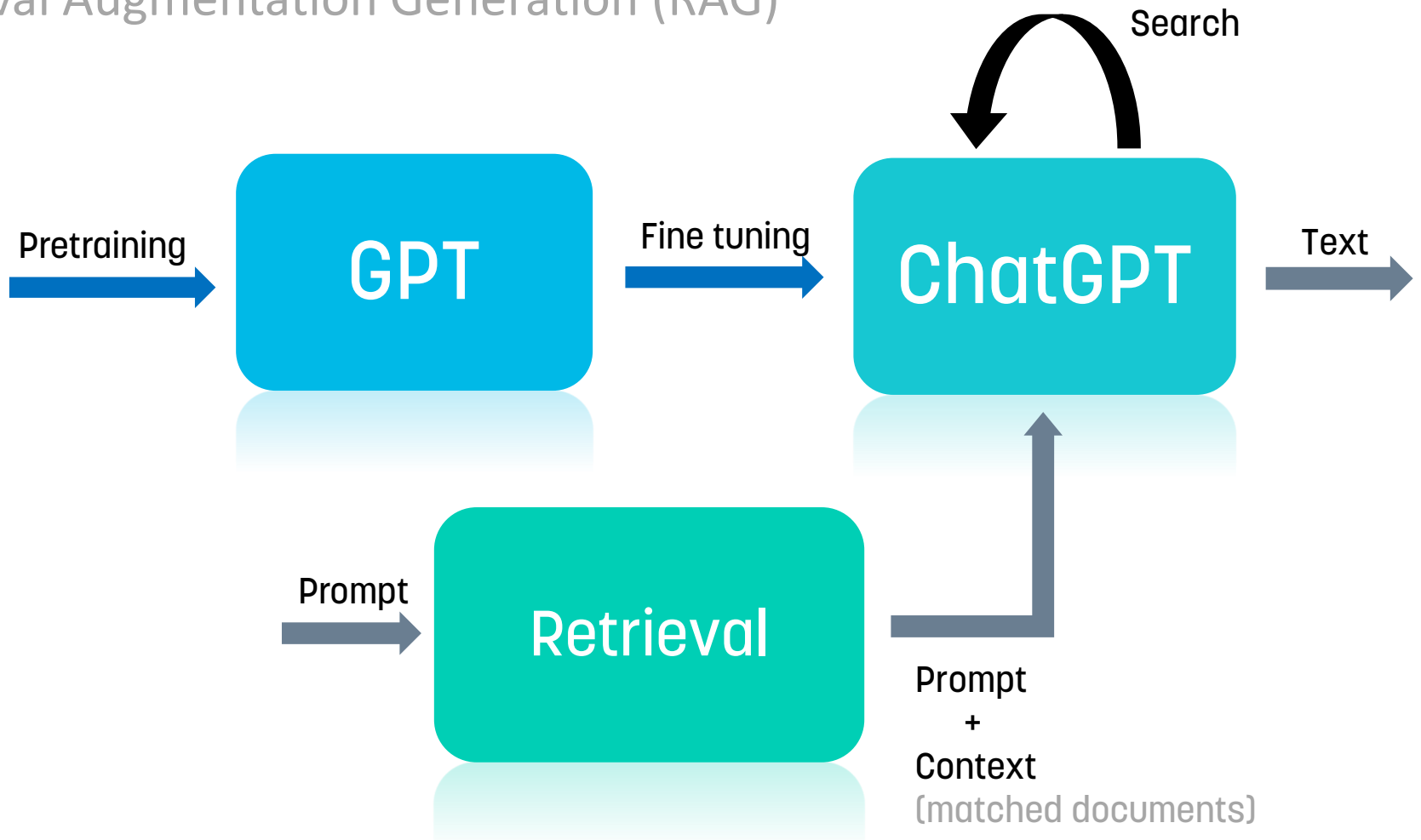
- Instruction-tuned LLM
- Retrieval Augmentation Generation (RAG)
- Search



Funded by
the European Union

ChatGPT | GPT-4o -> o1

- Instruction-tuned LLM
- Retrieval Augmentation Generation (RAG)
- Search



Funded by the European Union

Applied AI – At a glance

Systematic approaches (Solver, Algorithm)

- ✓ Solves the problem exactly
- ❑ Scales to large problem instances
- ✓ Formal language as interface
- ❑ Natural language as interface

Learning-based approaches (Learner)

- ❑ ~~Solves the problem exactly~~ Approximates solving the problem
- ✓ Scales to large problem instances
- ✓ Formal language as interface
- ❑ Natural language as interface

Large Language Models (LLMs)

- ❑ ~~Solves the problem exactly~~ Immitates solving the problem
- ❑ ~~Scale to large problem instances~~ Approximates generality
- ❑ Formal language as interface
- ✓ Natural language as interface

Hybrid AI

A systems perspective on Applied AI:
How can we check the mark for all aspects?

Funded by the European Union

Applied AI | ML approaches

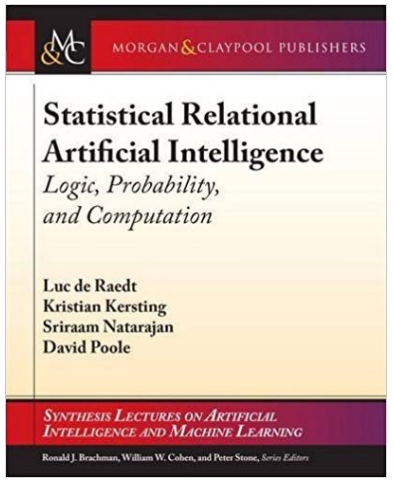
Neurosymbolic AI = Neural + Logic + Probability

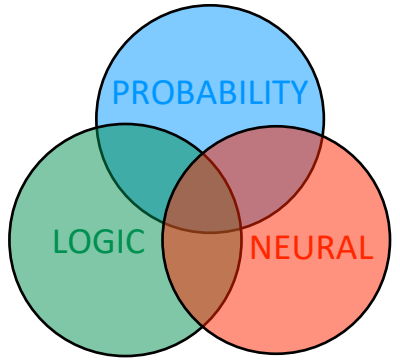
Approximative AI
(Neural networks, GenAI)

Systematic AI
(Search, Solvers)

Probablistic AI
(Bayesian inference/learning)

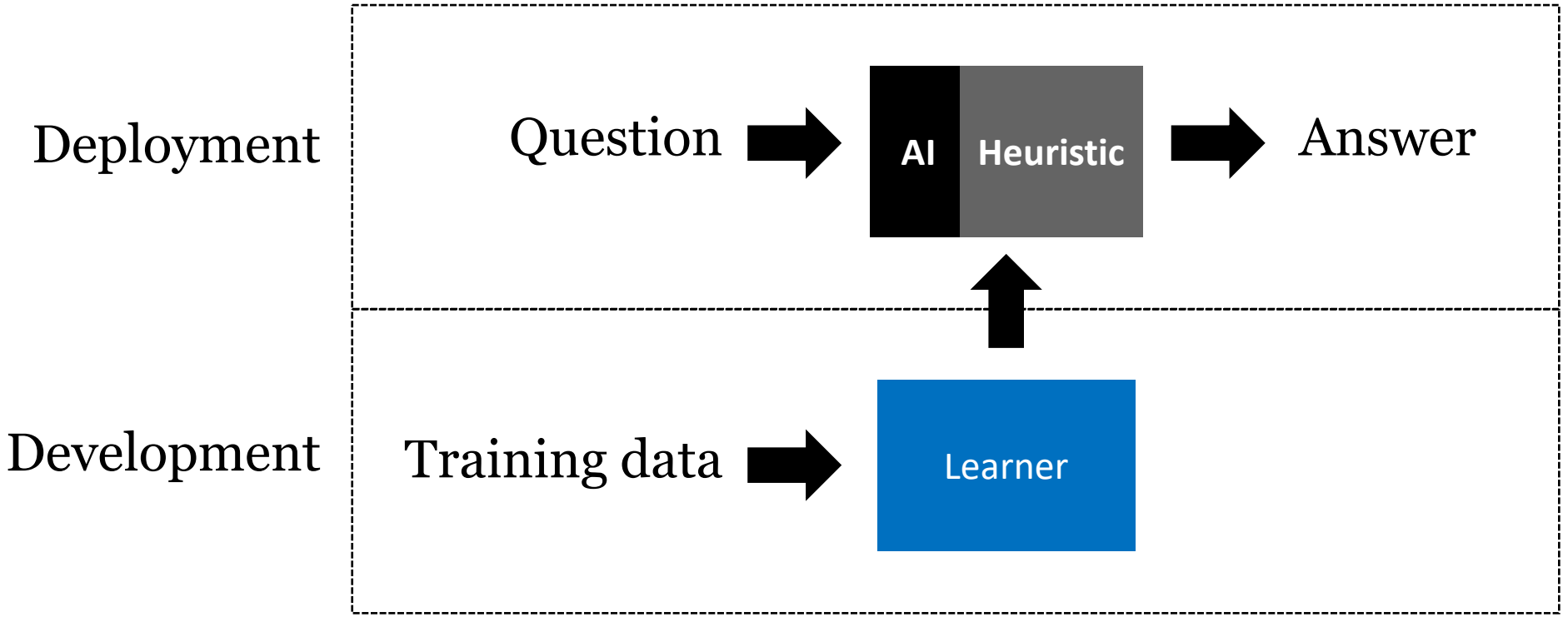
	Neural	Logic	Probability
perception	✓		
scalable	✓		
learning	✓		✓
reasoning		✓	✓
explanations		✓	✓
knowledge		✓	





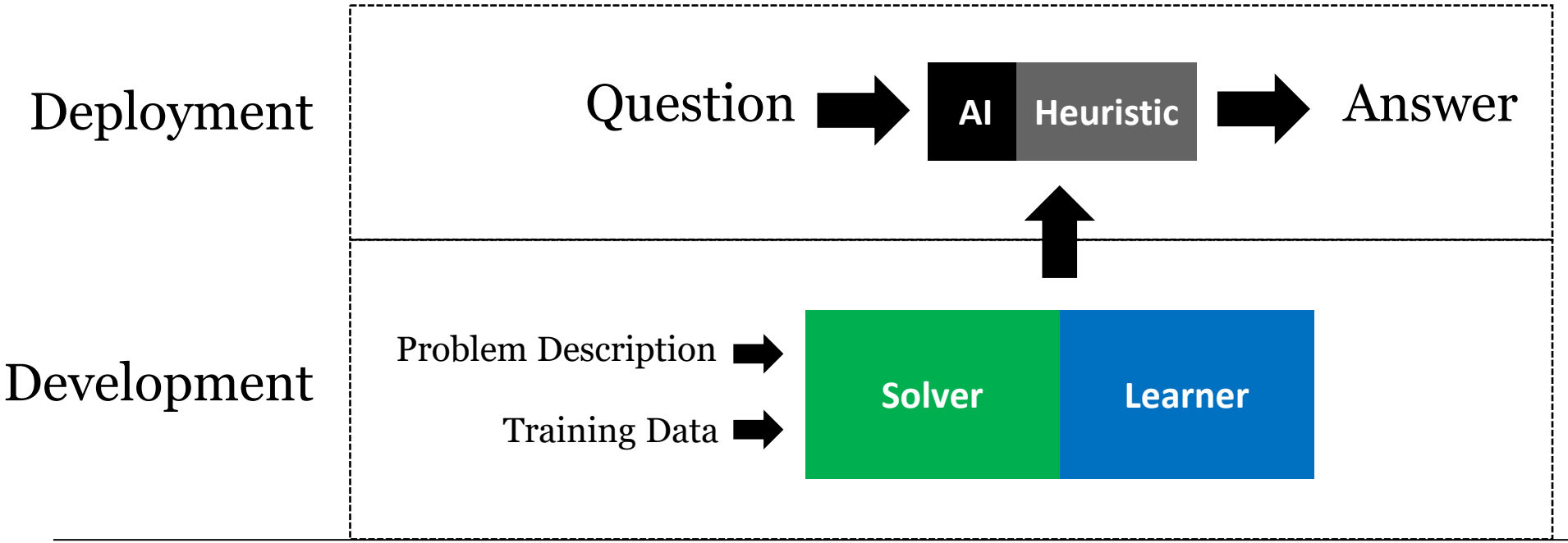
Applied AI | ML (Machine Learning)

- ML in practice (image, audio, text, …):



Applied AI | Hybrid AI with guarantees/quality (offline)

- If you have access to a good Solver/Optimizer
  - *Use it to learn **an efficient approximation** (i.e. finding better solutions faster)*
- If you have access to the data generating function
  - *Collect a **representative dataset***
  - *Use **conformal prediction** to estimate aleatoric uncertainty (outcome variability)*

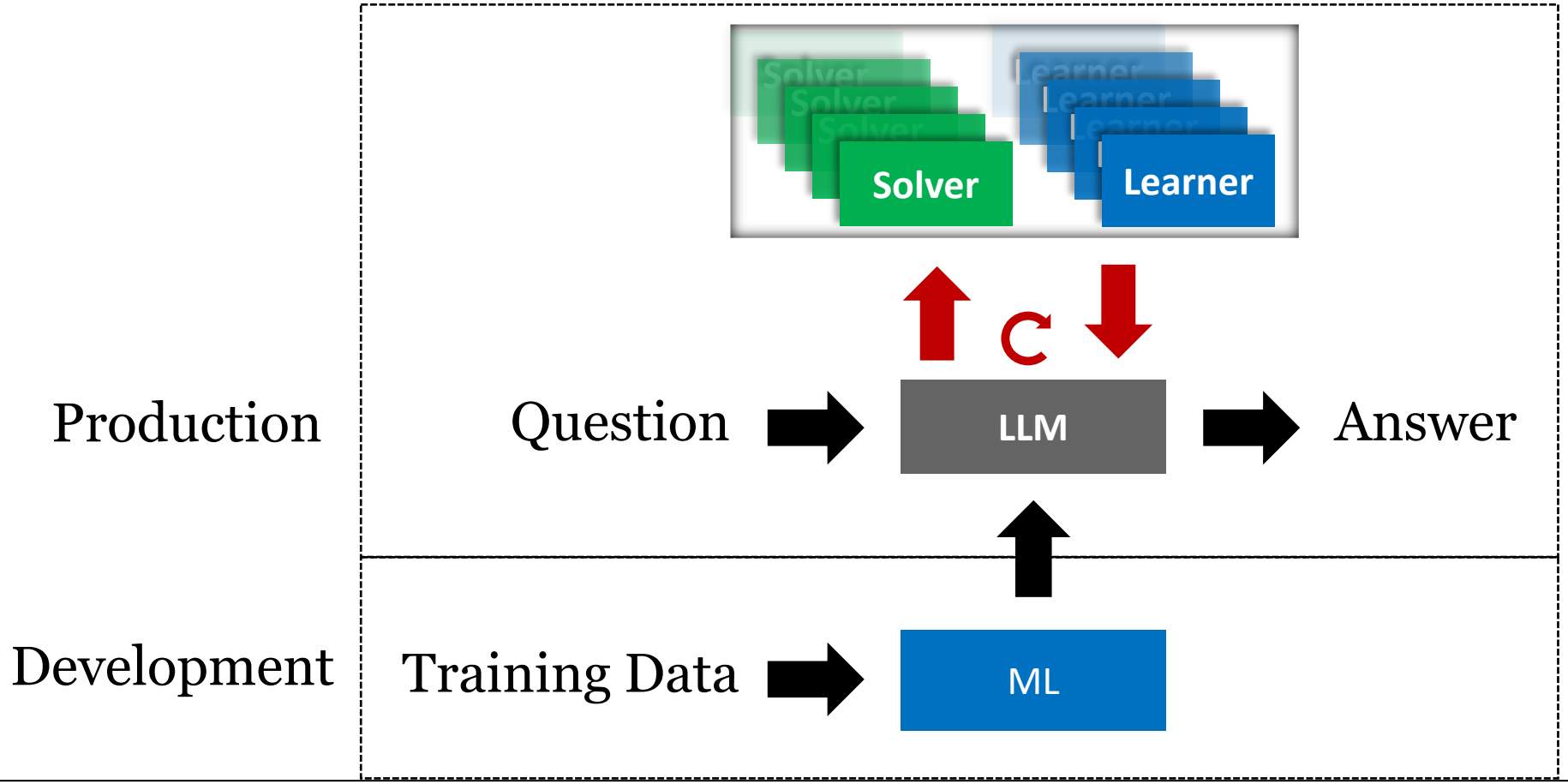


The Next Big Step for LLMs

Speak in your **native language** with the computer

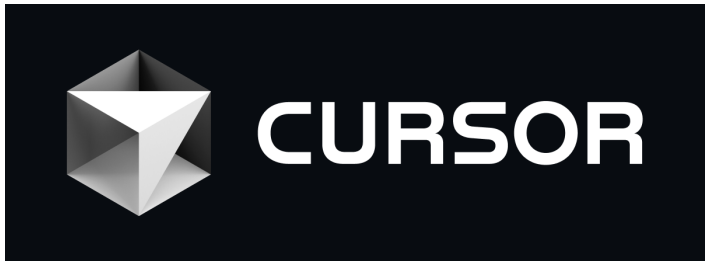
- Standardized "tool interface" (MCP) – Natural Language interface for the user.

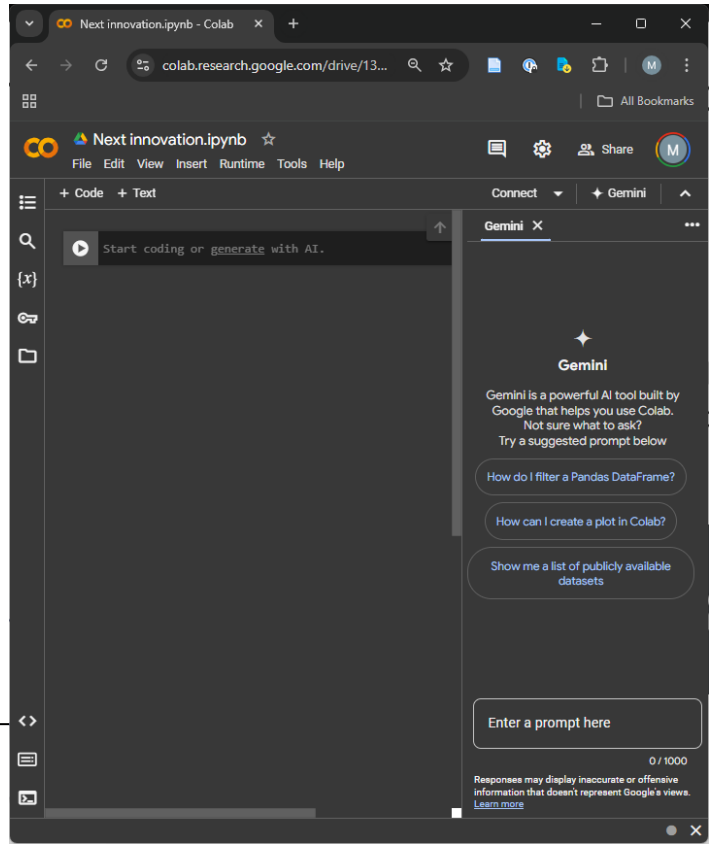
➢ Integration of LLMs with **the rest of the AI field**



Use-case: LLMs for Programming Assistance

"Today, **more than a quarter of all new code at Google is generated by AI**, then reviewed and accepted by engineers. This helps our engineers do more and move faster."





LLM Interaction – Best Practice

- **Don't ask a question without *first building coherent context to the question***
  - ➢ De-ambiguate and focus the answer (home in on the right parts of "everything ever written").
  1. Supply best practices of task X, guidelines for X, API for X, etc.
  2. Ask it to give a detailed summary of 1. focused on a general description of your task
  3. Give examples of what you want (e.g. code or documents) to use as "style transfer"
  4. Ask the actual question (what you want the LLM to do/solve).
- **Ask the LLM to describe/summarize what is good to know or needed to solve a task. Repeatedly ask the LLM for weaknesses or critical details missing.**
  - ➢ LLM-driven Prompt refinement
- **Provide a dump of what you want to achieve.**
  Then ask the LLM to ask you questions until it can convince you that it has "understood" and has specified everything important to know.
  - ➢ Let the LLM guide you in specifying the unknown unknowns

LLM Interaction – Best Practice

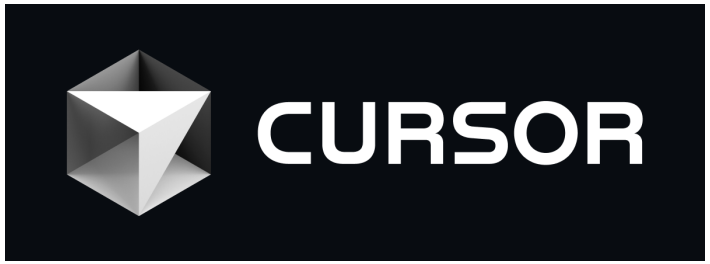
- **Encourage for de-ambiguation with any request:**
  - "Let me know if you have any questions before we start." (E.g. ChatGPT Deep Research)
- **Re-use lessons learns from chat history**
  - "Based on our chat history, create a document which capture common mistakes and how to avoid them or solve them."
  - "Summarize difficulties we have had in understanding each other."
- **Guidance/planning rather than editing**
  - "Provide me a list of suggested changes. Do not update any files."
- **Encourage progress tracking**
  - "Create a progress tracking document and keep it up-to-date in-between every new feature. Make sure that it has verification steps and outline development best-practices."
- **Automated prompt engineering (transfer to new LLM instance)**
  - Write a prompt for X. Assume it is for an LLM with no previous knowledge about our chat history.

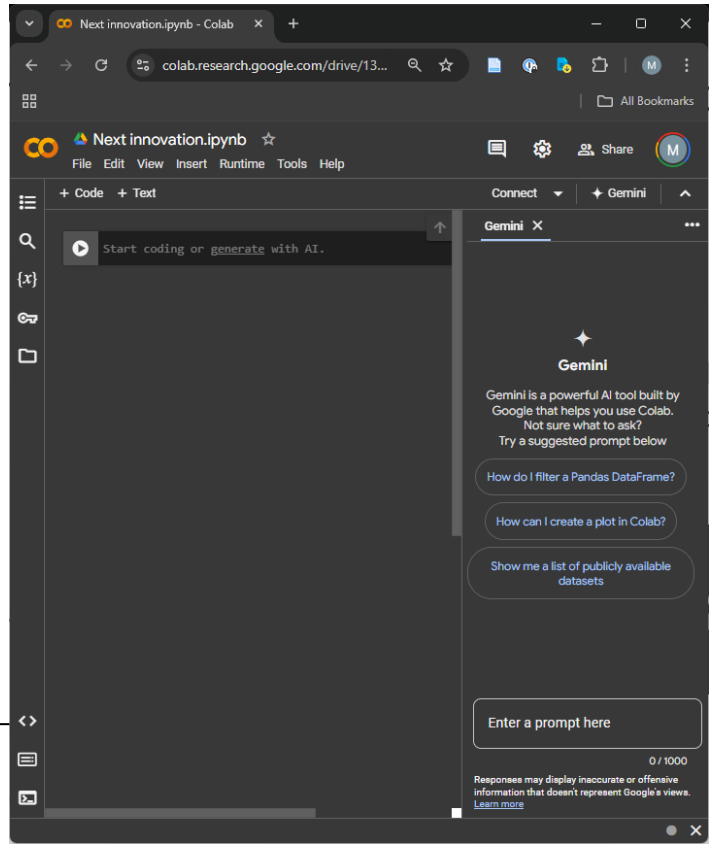
LLM Coding – Best Practice

- **Dump (i.e. train of thoughts) what you want to build. Add as much as you can think about.**
- **Let the LLM guide you to expand and make choices:**
  - "Your mission is to assist me in writing necessary technical documents that describe and enable completion of this software project. Start with iteratively asking me questions to clarify all relevant aspects of the project."
- **Create technical documents:**
  - "Create markdown files of a) requirements specification, 2) high-level project plan and 3) a readme file. Make sure that the project plan contain progress tacking and verification steps. The documents will be used by an LLM to complete the project with minimal human supervision. Do verification tests on your own as much as possible before involving the human user for final verification, at any point during dev."
  - "I want you to: 1) Break down the problem into different phases. 2) Create a detailed plan (markdown file) for each phase. 3) Make sure that all technical documents refer to each other and that all technical documents, and their intended usage, is clearly described in the readme file."
- **Use additional scratch-pads for meta-progress and of-tangents (e.g. debugging or vision-changes)**
  - "Create a memory.md and a progress.md which we can include often in context. They should together capture essential things to keep in mind for the current task at hand, as well as general progress tracking."

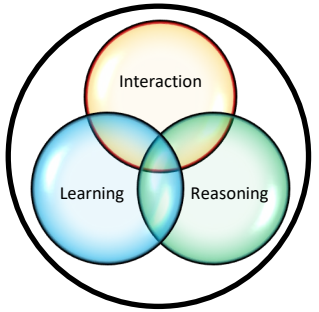
Use-case: LLMs for Programming Assistance

"Today, **more than a quarter of all new code at Google is generated by AI**, then reviewed and accepted by engineers. This helps our engineers do more and move faster."





Example Application | Parameter Tuning



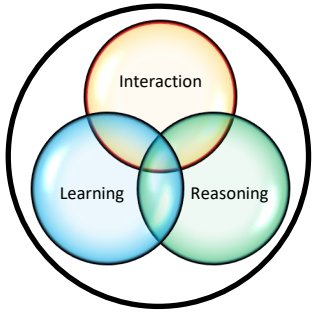
Setting: An objective should be optimized but it is highly expensive (time or cost) to perform each experiments.

Application:

- ML parameter tuning (hyperparameter tuning) [1]
- Tuning algorithms to target applications (domain adaptation) [2,3]
- Drug discovery, A/B-testing, …

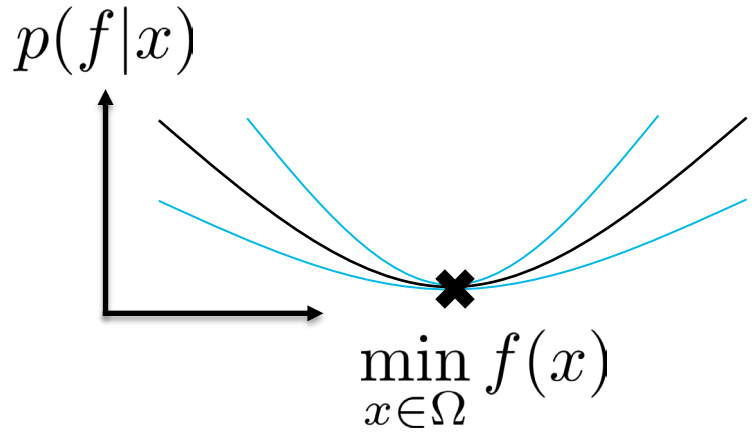
Problem: Search for (global) optima with minimal number of experiments.

Parameter Tuning



Methods

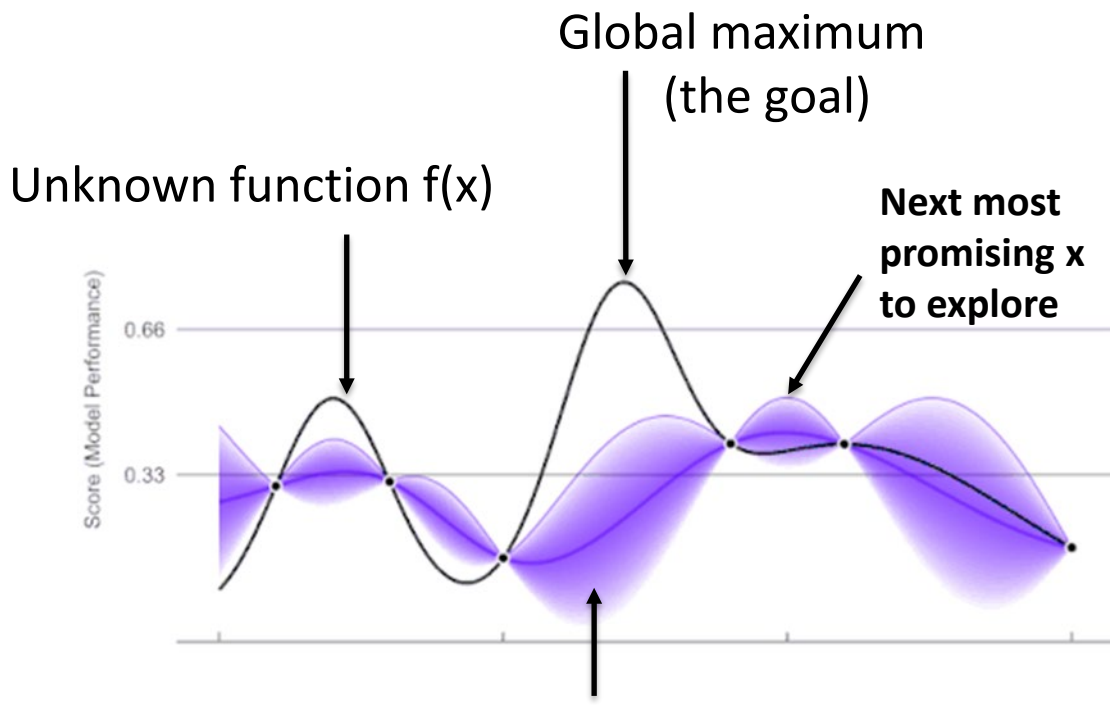
- **Grid search** (evaluate entire grid and save the best grid point)
- **Random search** (sample a distribution and save the best sample)
- **Gradient-based optimization** (follow the gradient to local minima/maxima)
- **Bayesian Optimization** (BO)
  (Update posterior distribution over parameter space after each experiment) [4]

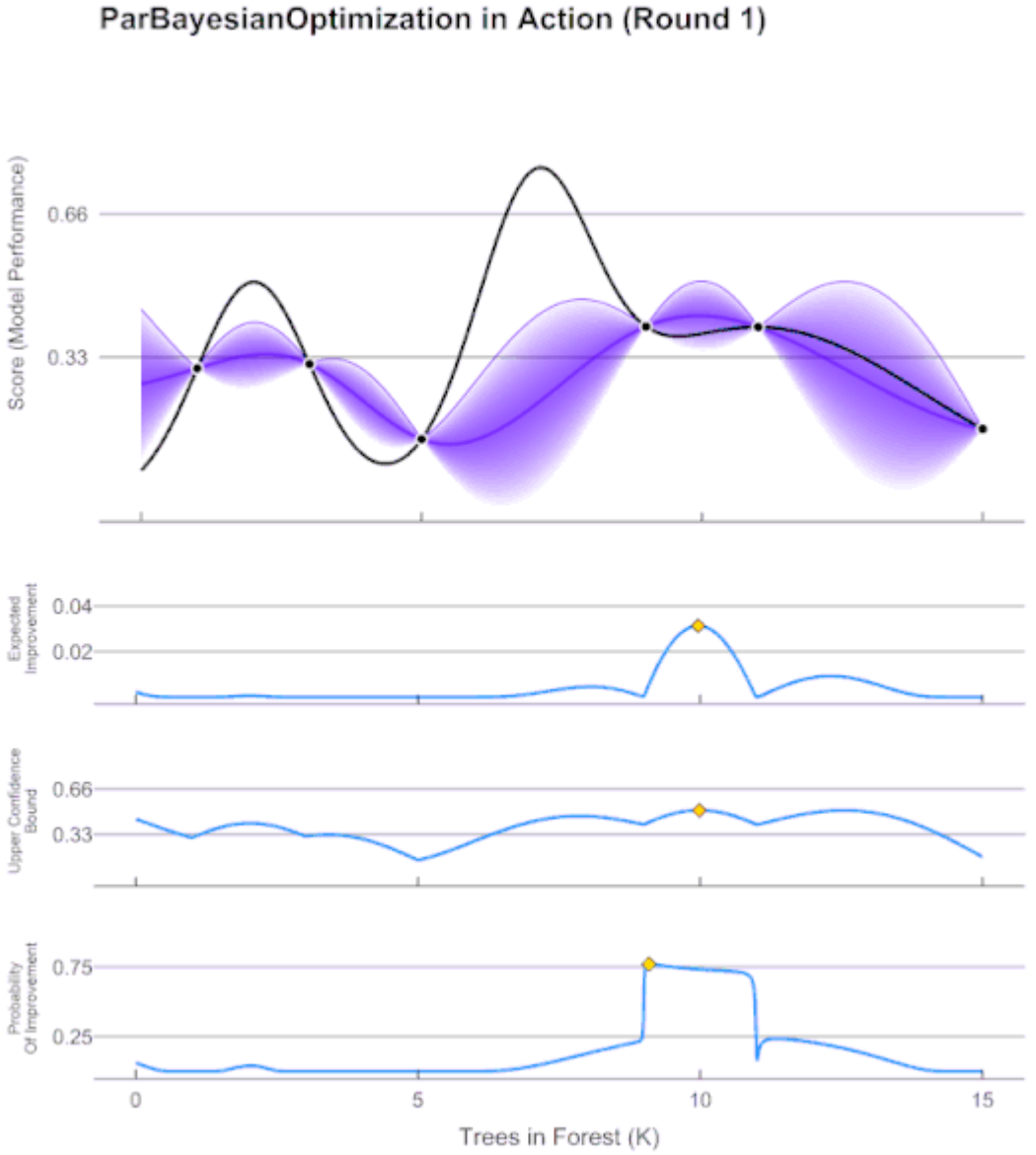


[4] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. *Taking the human out of the loop: A review of Bayesian optimization*. Proceedings of the IEEE, 2016.

Example | Parameter Tuning

Bayesian Optimization for parameter tuning (example)





https://hyperopt.github.io/hyperopt/

https://hyperopt.github.io/hyperopt/tutorials/01.BasicTutorial/

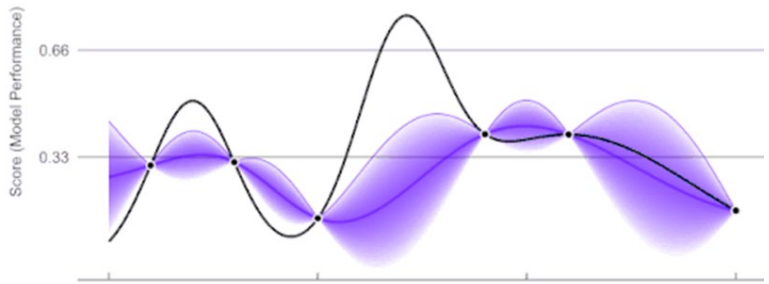
AI Tools | Global Optimization

Bayesian Optimization for parameter tuning (experiment design)

```
1  from hyperopt import fmin, tpe, hp
2  space = hp.uniform( label: 'x',  *args: -10, 10)
3  objective = fn(x)
4  best = fmin(objective, space, algo=tpe.suggest, max_evals=100)
```

Search space

What you want to minimize
(structurally unknown)



Number of tries (experiments)
(50-200 often sufficient)

Try it yourself!

objective = **lambda** x: x ** 2

max_evals = 5, 10, 20, 50

https://hyperopt.github.io/hyperopt/

https://hyperopt.github.io/hyperopt/tutorials/01.BasicTutorial/

New frontiers of software engineering?

Previous paradigm

- Specialize by memorizing and practicing using specific techniques (A*, BO, YOLO).
- Break down problems to solve them piece by piece.
- Divide your code into small increments or features that are easy to grasp and explain to collogues.
- Scrum-ish, since thinking it through in detail is infeasible and beyond our ability.
- Leverage past experience and reference materials (e.g. manuals or blogs).

New paradigm?

- Generalize by knowing what you can ask for: What areas are previously explored, and what algorithms exists. Specialize in QA.
- Break down problems so that the LLM's context does not get overwhelmed.
- Divide your code into separate goal-focused conversations. Global changes that can be hard to grasp. (Hide goal?)
- Repeated "Waterfall", since more context makes the implementation more aligned.
- Approximate context retrieval and interpolation of (almost) all code on Github.

Mattias Tiger

AI och Integrerade Datorsystem (AIICS),
Institutionen för Datavetenskap

www.ida.liu.se/~matti23/mattisite/research/

www.liu.se/ai-academy

www.liu.se/medarbetare/matti23

LINKÖPINGS UNIVERSITET

AIICS

TAILOR
Foundation of Trustworthy AI:
Integrating Learning, Optimisation and Reasoning

WASP
WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM