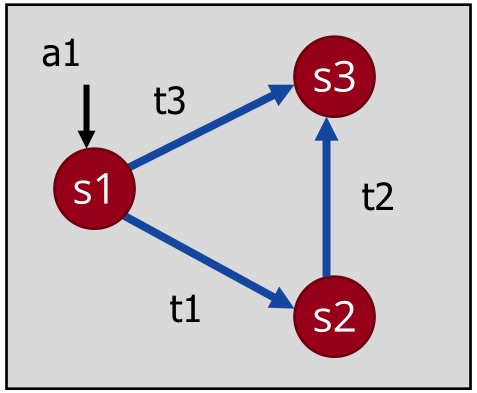
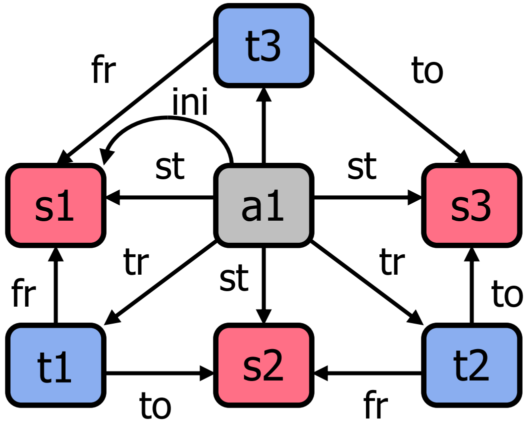
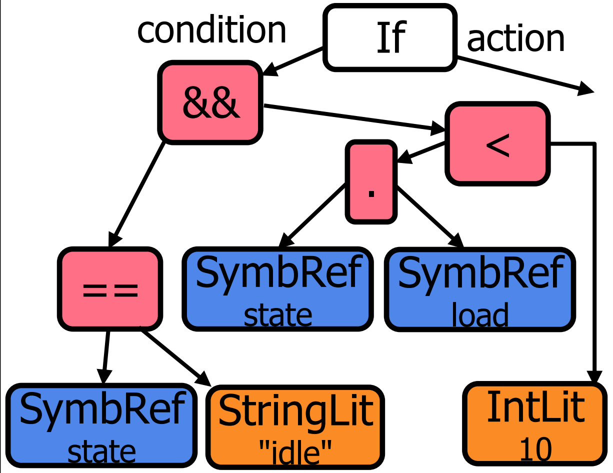
Code and Model Intelligence

Reminder: Concrete and Abstract Syntax

- **Question:** How to capture models?
- **Answer:** graph-based structures!

Concrete syntax	Abstract syntax
	

Concrete syntax	Abstract syntax
see code below	

```
if (
   state ==
   "idle" &&
   this.load<10)
   …
```

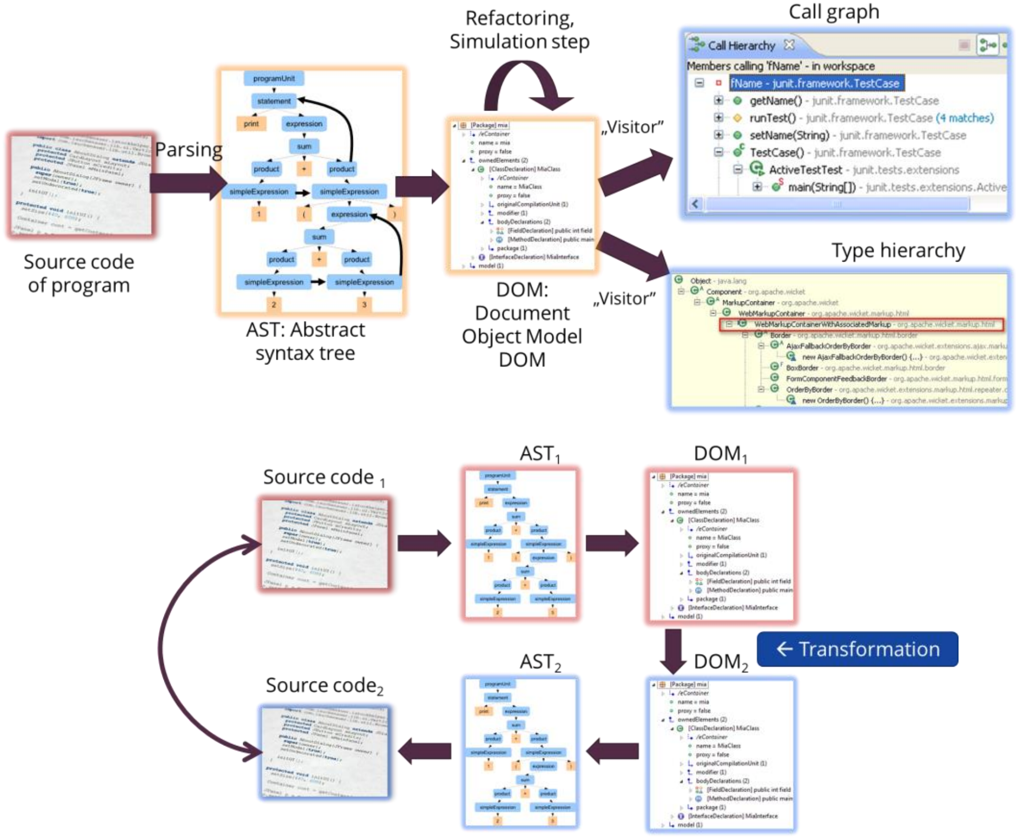

- **Today's Topic:** Intelligent editing features for code and models

Machine Learning models

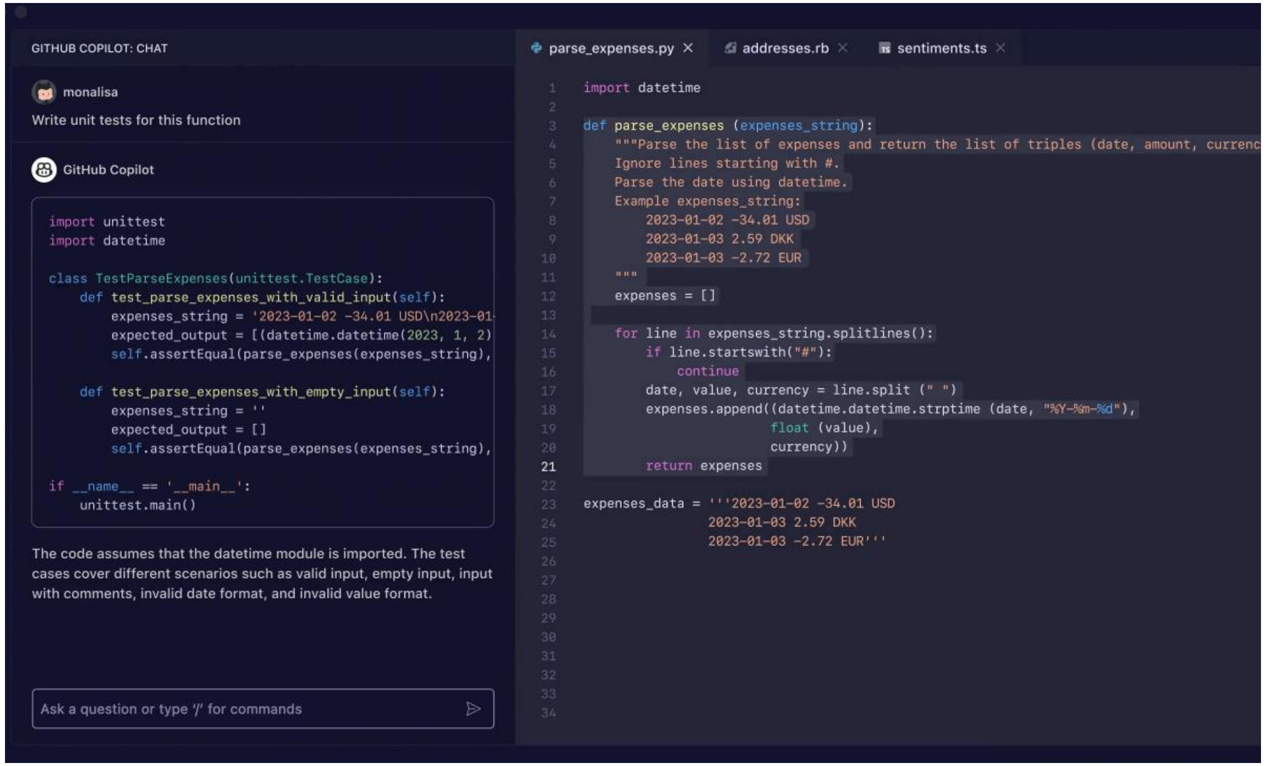
Statistical code intelligence

Code intelligence

AST + DOM based analysis



Machine learning based analysis

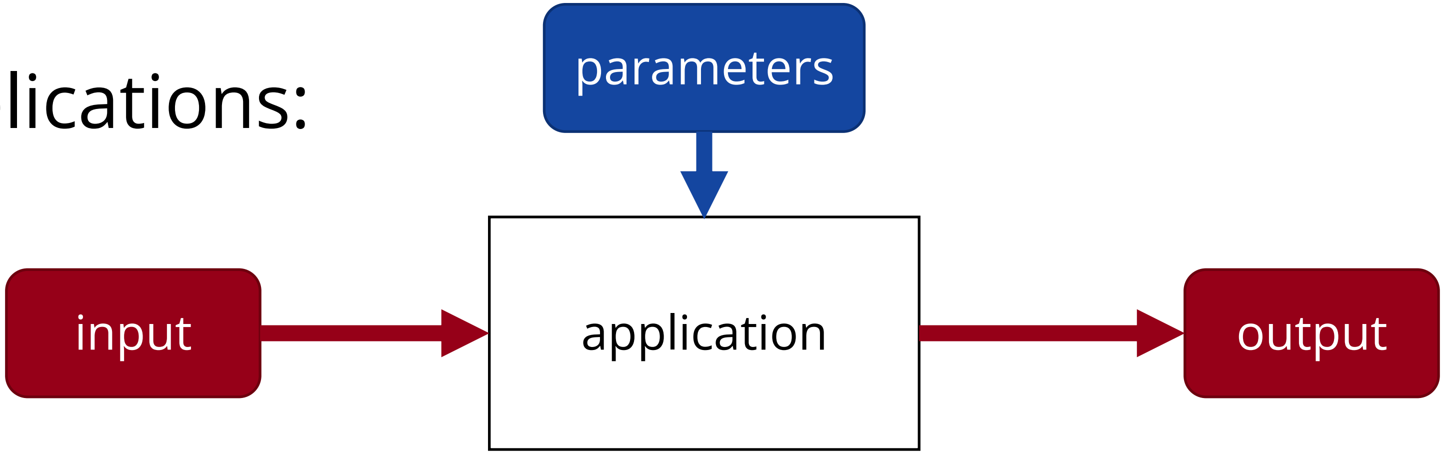


Code intelligence

AST + DOM based analysis	Machine learning based analysis
• **Algorithms** and computations on the parser/linking output • Content assist, call hierarchy, refactoring, … • Can be **systematically tested** • Limited ability to take natural language and common-sense knowledge into account	• Based on **statistical analysis** of a large corpus of examples • Code completion, chat interfaces, agents, … • Provides only **probabilistic guarantees** • Can take natural language input and rely on learned patterns

Machine Learning Applicationgs

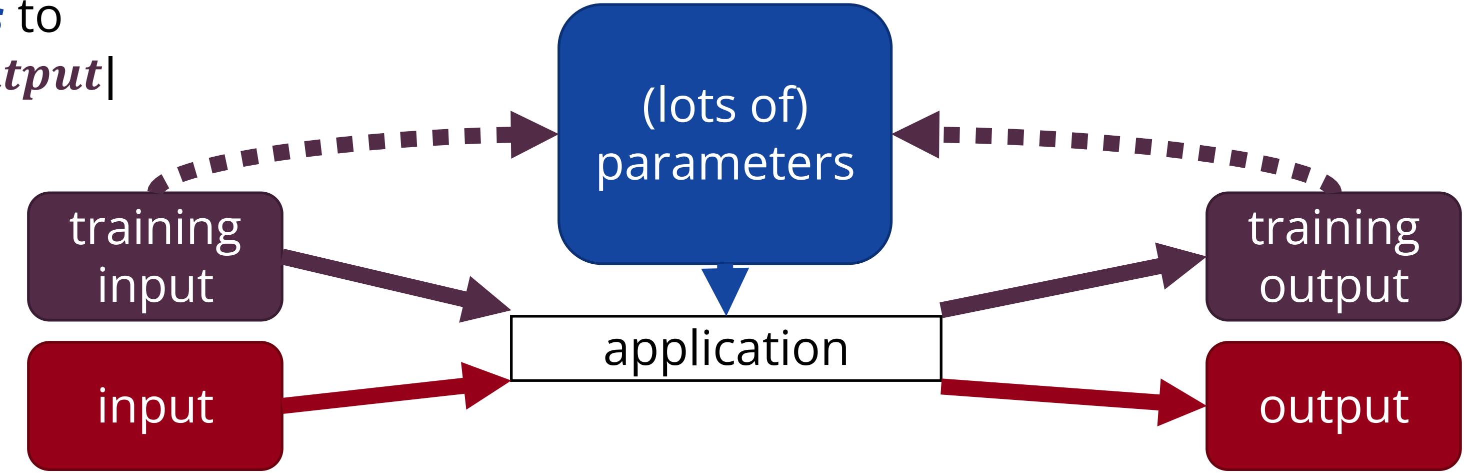
- Normal applications:



- ML apps: **requirements + manual work** → **data + power**

- calculate $\boldsymbol{parameters}$ to
- min $|\boldsymbol{app}(\boldsymbol{input}) - \boldsymbol{output}|$
- wrt. a cost function

If training, validation, and runtime data is similar, it should work.

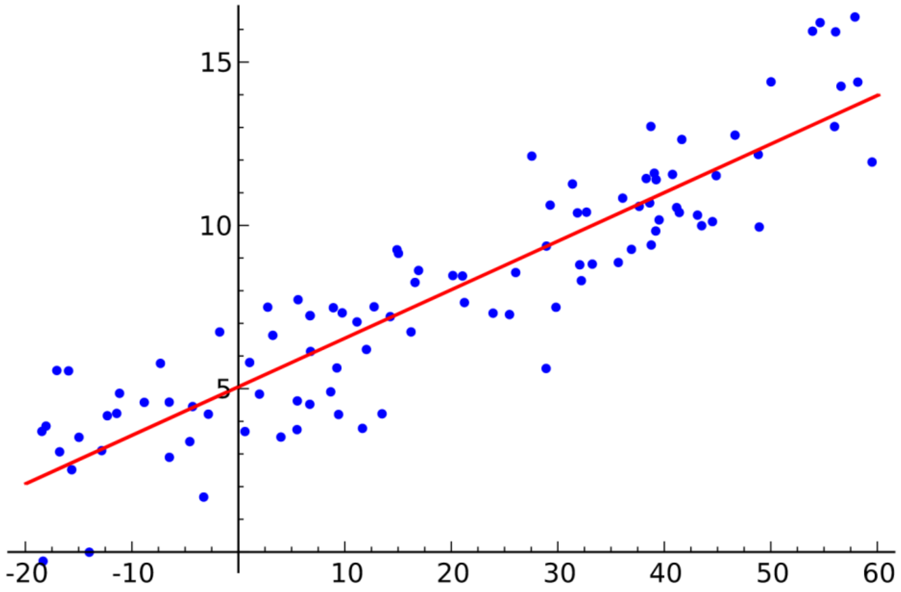


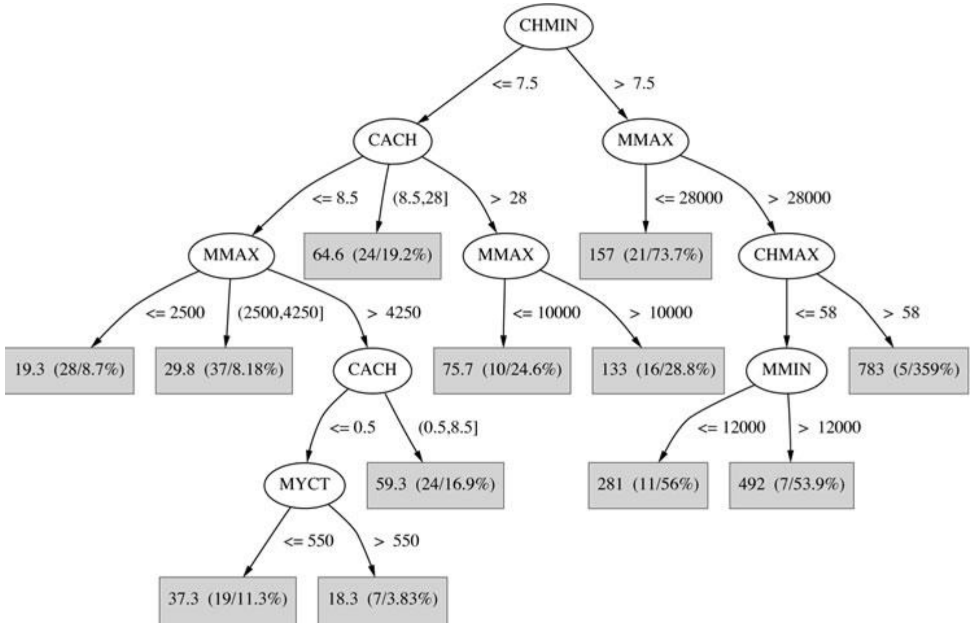
Machine learning tasks

- **Supervised** learning
  - Training input: **labeled** data $\{(x_i, y_i)\}_{i=1}^{n}$, output: $f: x \mapsto \hat{y}$ **predictor**
  - Predict a discrete class label ⇒ **classification**
  - Predict a continuous quantity ⇒ **regression**
- **Unsupervised** learning
  - Input: **unlabeled** data $\{x_i\}_{i=1}^{n}$, output: discovered **structure** of data
  - Example: **clustering**, anomaly detection
- **Reinforcement** learning
  - Input: **environment** model or interactions, output: **controller**
  - Learn to plan and optimize in complex environments

Machine learning models

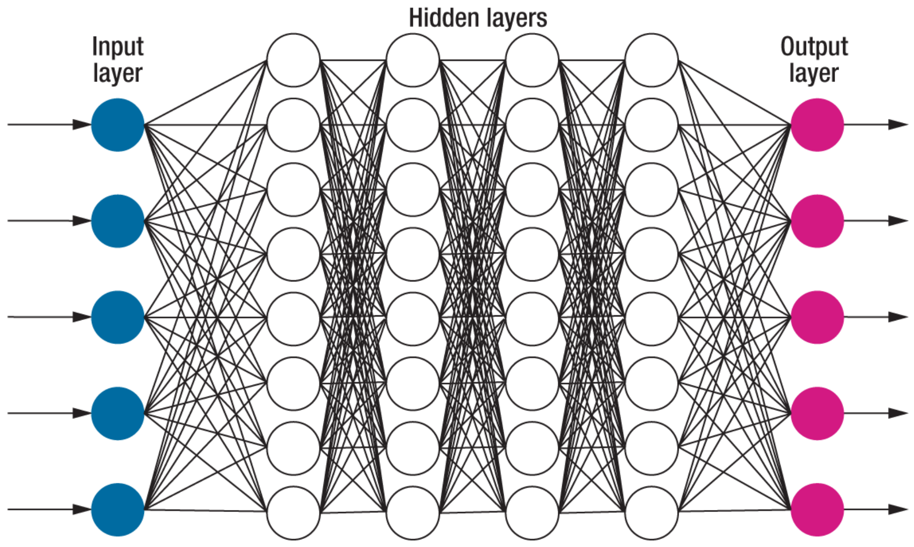
- Classical machine learning and **statistics**
  - E.g., linear regression, decision trees, Support Vector Machines
    - ✓ Easy to interpret the model
    - ✓ Fast learning and inference
    - ✘ Limited expressive power





Machine learning models

- **Deep Neural Networks** (DNNs)
  - Leverages the computation power and modern GPUs and other hardware to build extremely large models
    - ✓ Very high expressive power
    - ✓ Widely available tools
    - × Hard to interpret models
    - × High computational complexity



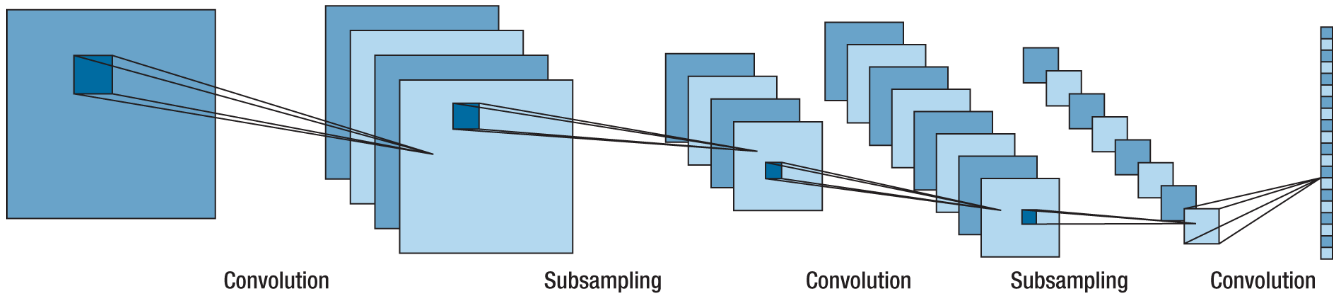
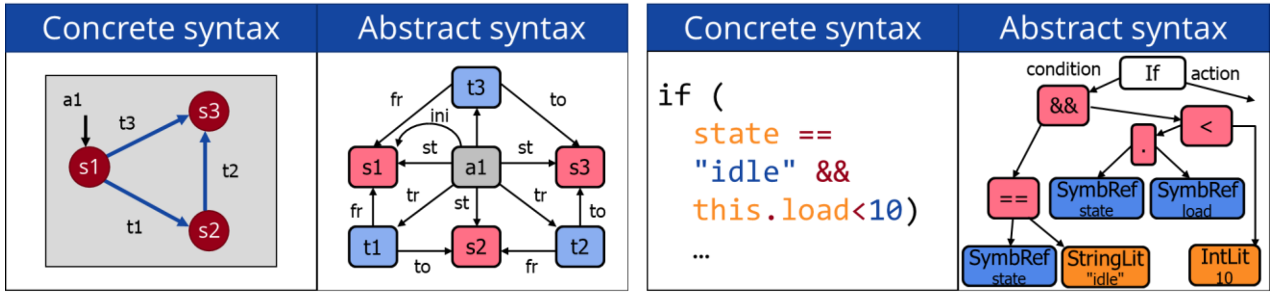


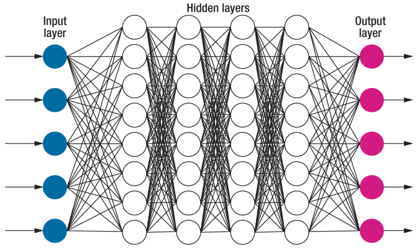
Figure: Falcini et al. (2017). Deep Learning in Automotive Software. In: IEEE Software

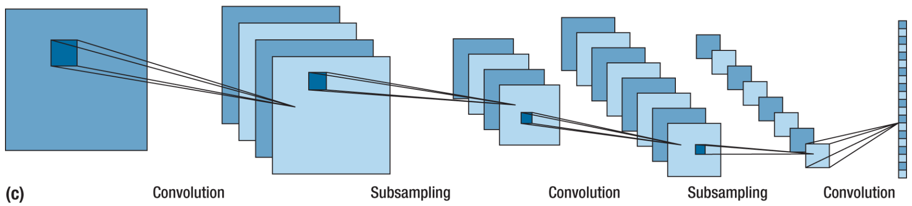
A note on models

- We are using the term *“model”* in two senses
  - A **software model** is a **graph-like** artifact with **formal meaning**



  - A **machine learning model** contains **parameters** for a **statistical technique**

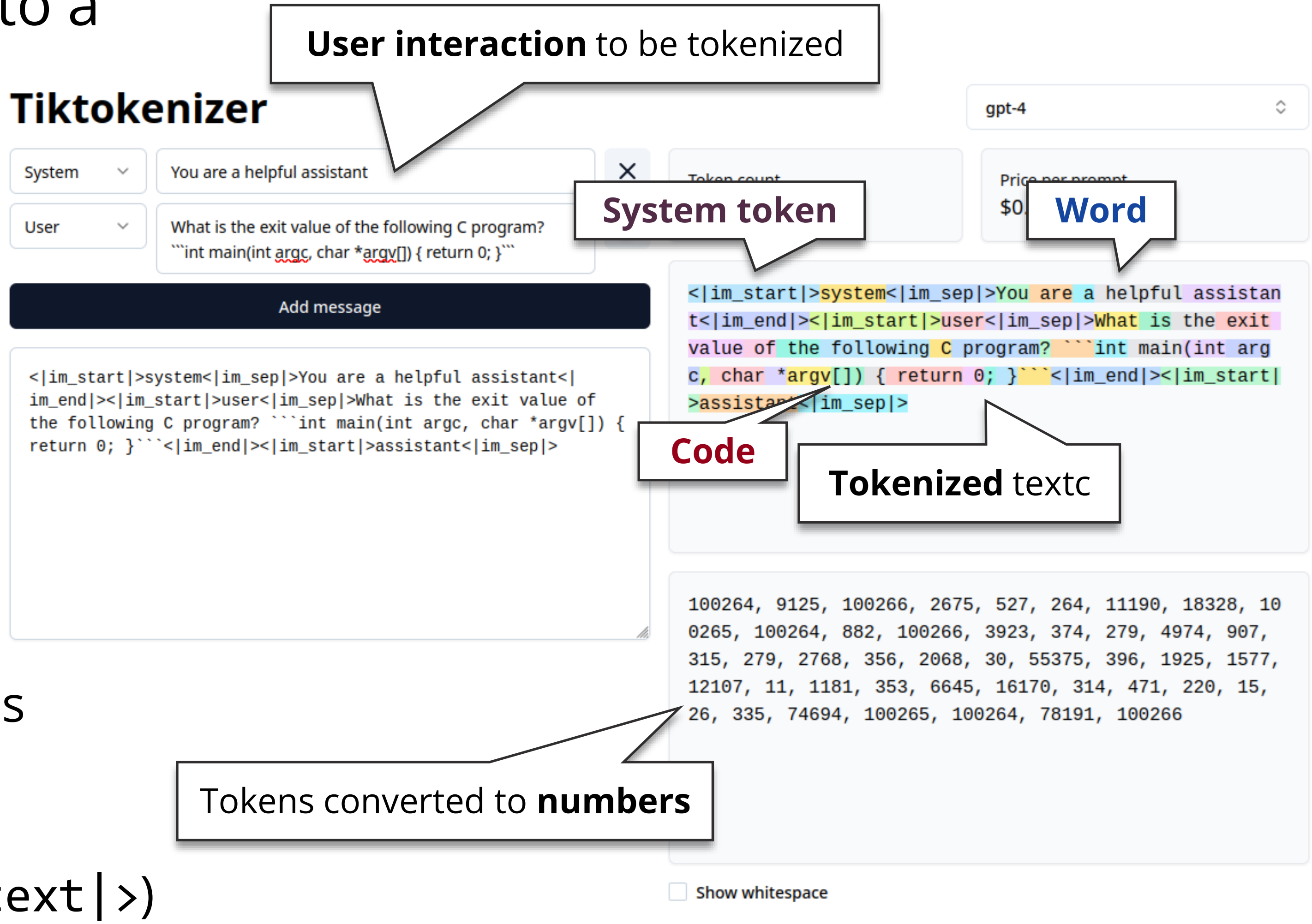




- Both are **abstract descriptions** an aspect of reality, but with different **guarantees** of **correctness** and **accuracy**

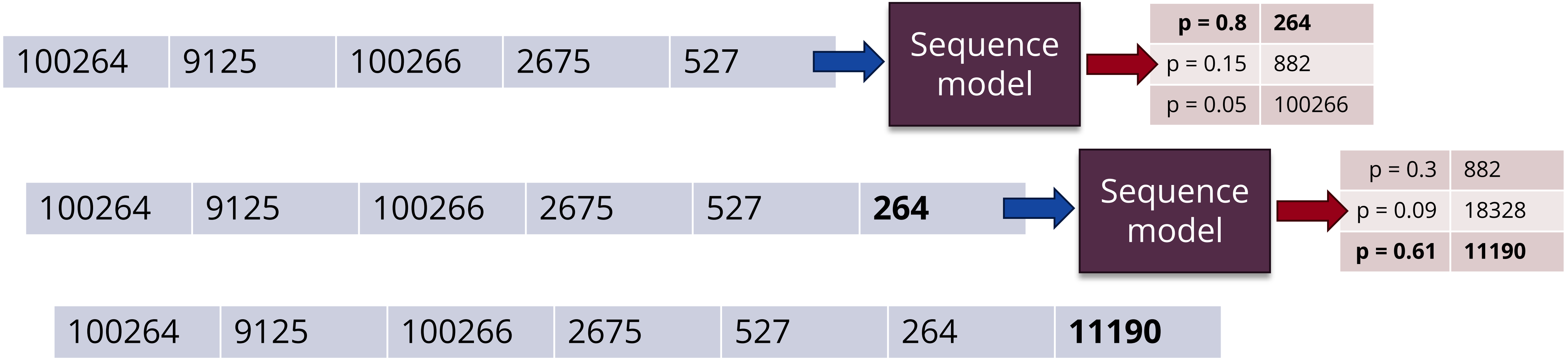
Text processing with machine learning

- **Tokenization:** turn the text into a sequence of numbers
  - High-dimensional vectors are more efficient than single characters
    - e.g., each token can be one of ~100 000 values in the `cl100k_base` encoding used by GPT-4
- Types of tokens
  - **Words** and parts of words (~0.75 English words per token)
  - Common character combinations used in **code**
  - System tokens describing **user interaction** (e.g., `<|endoftext|>`)



Sequence modelling

- **Input:** sequence of tokens (**context window**)
- **Output:** probability distribution over the **next token**
- The sequence model encodes the structure of the language to **predict** the next token with high accuracy
  - Predicting multiple tokens: run the model iteratively (**beam search**)

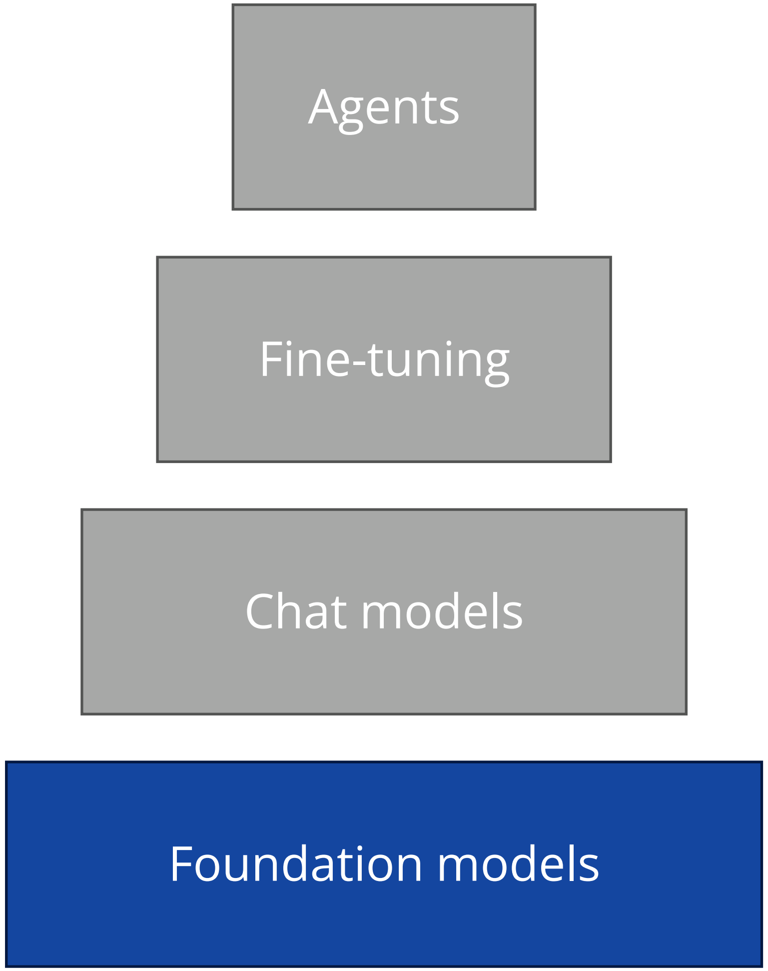


Large Language Models

Foundation and chat models

Foundation models

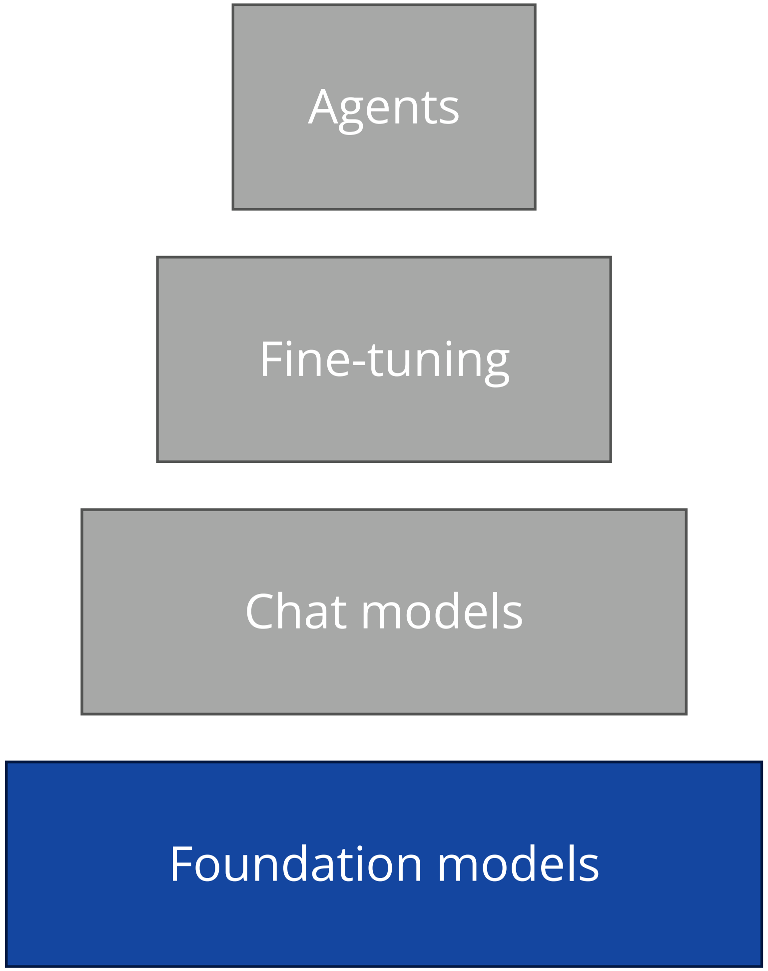
- **Supervised learning** (parameter optimization) on a large corpus of text
  - E.g., the *Pile* dataset used in the training of open-source models is 825 GiB
    - Books, academic articles
    - Patents and law
    - **Source code** from GitHub and Stack Exchange
  - Proprietary datasets may be much larger
- Minimize **perplexity** (probability of failed prediction of the next token)
- Fixed **context window** size
  - Larger context windows required quadratically larger computational resources



Foundation models

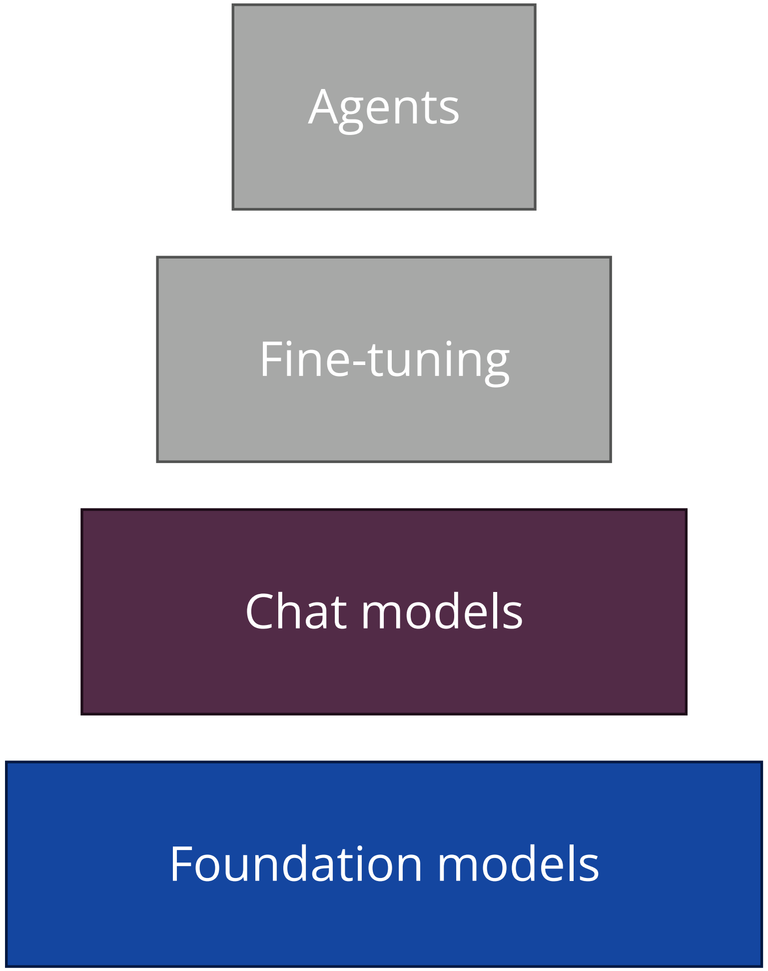
- **Large Language Models:**
  parameter counts measured in billions
  - Models downloadable at no cost, e.g., *Llama3.1:* 8B, 70B, 405B variants
  - *ChatGPT:* allegedly 1.8 trillion (1800B)
  - Each parameter is a 16-bit floating point number!
- **Inference** requires specialized hardware
  - For the smallest models: **neural accelerators** in laptops, phon
  - Models runna large amounts
  - The largest and mo ate models run on numerous data ce ter GPUs and are only available as **APIs** with a per token cost

Send an **API token** identifying the user along with each query for authentication and billing purposes



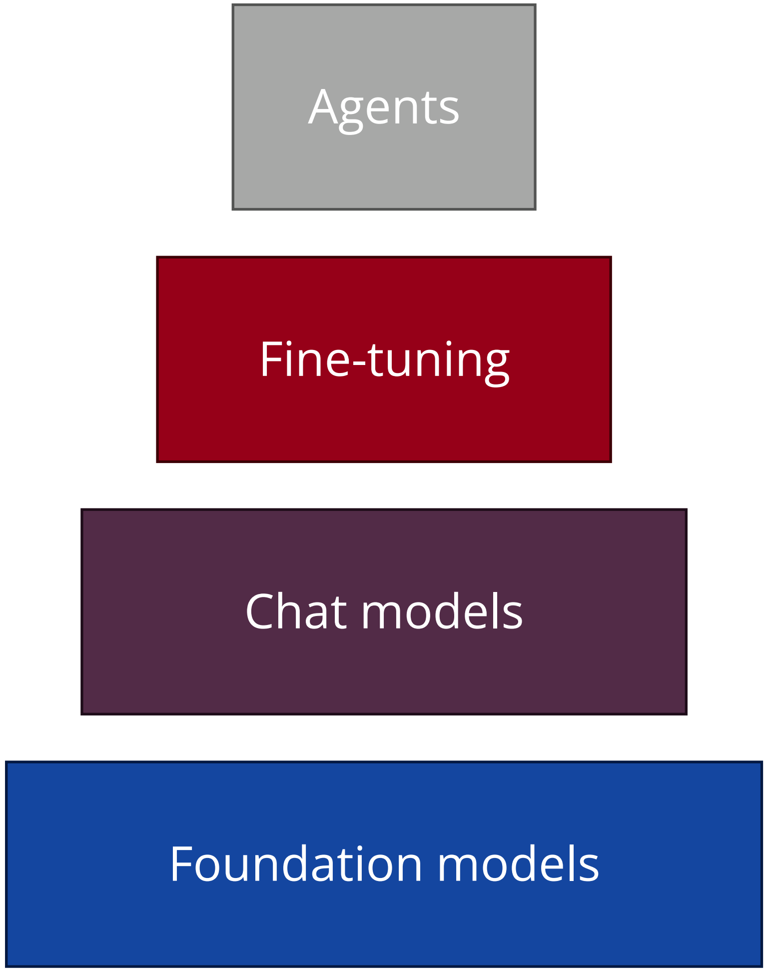
Chat models

- **Reinforcement Learning from Human Feedback** (RLHF) to **align** the model to follow textual instructions
  - Requires costly, manual data annotation
- Turns the foundation model into a chatbot
- **System prompt:** general description of the chat system (e.g., *"You are a helpful agent."*)
- **User prompt:** instructions to be followed



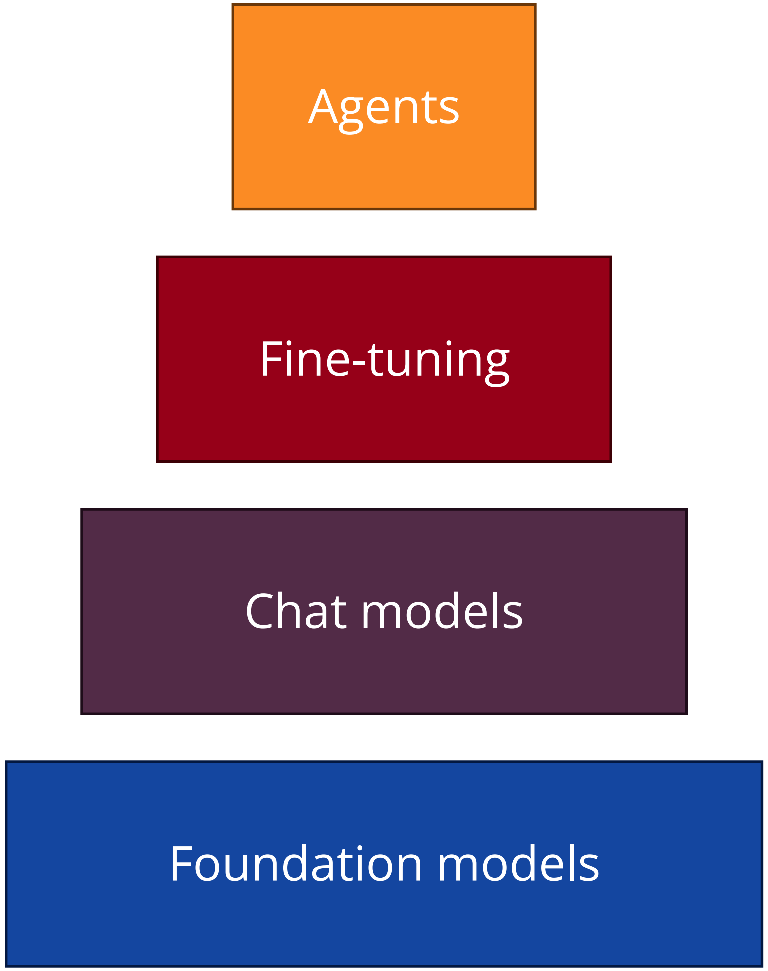
Fine-tuning

- Further supervised learning based on a corpus of **specialized text**
- Examples
  - Fine-tune on a set of natural `language <-> SQL query` pair for SQL query generation
  - Fine-tune of code from a specific project to incorporate project structure and API into the model
- May require a very large dataset
  - Possible solution: synthetic generation with another, larger LLM (knowledge **distillation**)
  - But still more efficient than training from scratch



Agent systems

- Specific ways of **calling** the model with prompts and other input data and **parsing** the output
  - Often involves **feedback loops,** where output from the model is fed into another prompt
  - Allows **combining** ML with classical, AST and DOM based algorithms in the loop
- May use a fine-tuned or a base chat model
  - Can be an effective alternative to fine-tuning when only a low amount of data is available
  - Agent development is **less computationally intensive** than training or fine-tuning



Building code and modelling agents

Common architectures and patterns

Building coding agents

Information retrieval → Query composition → LLM inference → Output parsing → Tool execution

- **Problem:** a full source code repository does not fit into the context window of the language model
- **Retrieval-Augmented Generation:** augment the input query with relevant parts of the repository
  - **Textual proximity:** code around the requested edit point
  - **Scope-based:** definitions (or natural-language documentation!) of symbols available in the current scope
  - **Vector search:** use another model to estimate the similarity of code snippets, and return the most similar ones

Building coding agents

Information retrieval → **Query composition** → LLM inference → Output parsing → Tool execution

- **Prompt engineering:** assembling the context and the request into an LLM prompt so that the answer is likely to be accurate
  - "More of an art than science," effective prompts depend on the specific training data and training process of the LLM
- **Few-shot learning:** include correct solutions to **similar problems** in the prompt so that the LLM is likely to answer with a similar solution
- **Chain-of-Thought** (CoT): include instructions like *"Think step-by-step!"* to make the LLM generate a longer, more detailed response
  - May improve accuracy even if all but the final output is discarded

Building coding agents

Information retrieval → Query composition → **LLM inference** → Output parsing → Tool execution

- Inference may happen on-device (neural accelerator or GPU) or by calling an LLM API (e.g., OpenAI, Claude)
  - **API token** provided for authentication and billing
  - **Inference costs** depend on the amount of tokens used and the selected LLM
  - **Data confidentiality** concerns when calling an external API
  - Providers offer **enterprise agreements** for confidential code bases

Building coding agents

Information retrieval → Query composition → LLM inference → **Output parsing** → Tool execution

- **Challenge:** chat LLMs output natural language text by default
  - May be suitable for user-facing chat (e.g., Copilot Chat)
  - But hard to use in automated tools
- Use of **few-shot learning** to make the model more likely to conform to a prescribed template
  - Still need to repeat inference if parsing fails
- Some LLMs support constraining **beam search** with a **context-free grammar**
  - Easier to implement with local inference (modify the inference code)
  - But, e.g., OpenAI offers constraining output to valid JSON syntax
  - Still need to ensure that the output is semantically correct

Building coding agents

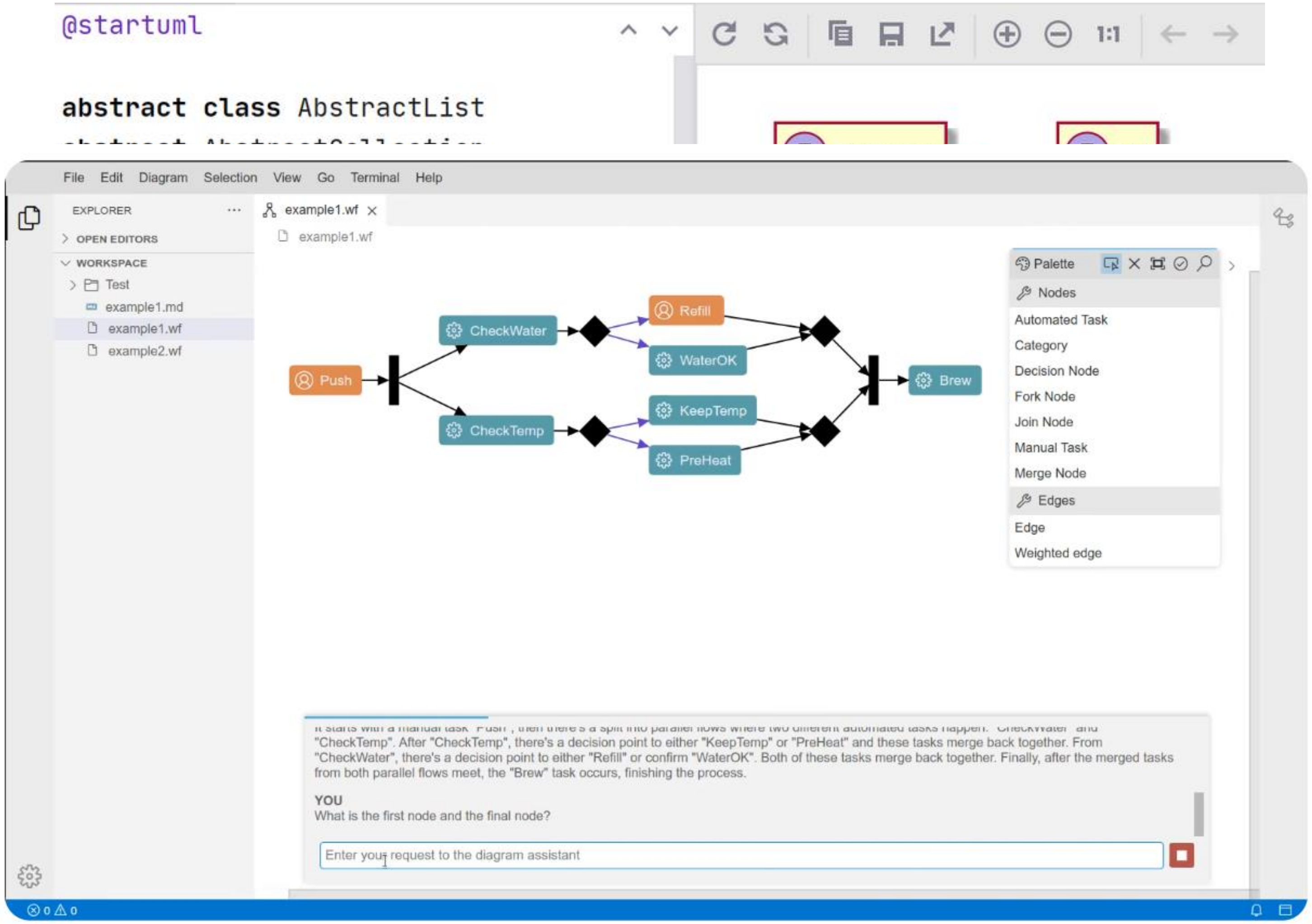
Information retrieval → Query composition → LLM inference → Output parsing → **Tool execution**

- Model output may be simply taken as generated code
- **More complex actions** can be automated with tools and function
  - The model generates JSON describing the action to take based on a few-shot prompt describing possible **tools** and their **actions**
    - E.g., create new file, rename function, use tes
  - The agent system **parses** the JSON, then **v**
  - **Function** calling: **feed back** the output of
    - E.g., search for files matching a given pattern in the r sitory
    - Hallucinations about computations (e.g., 2 + 2 = 5) may be remediated by generating source code (e.g., Python) and executing it in a tool instead of obtain the answer

Tools and functions must be **sandboxed** to avoid accidentally generating and performing destructive actions

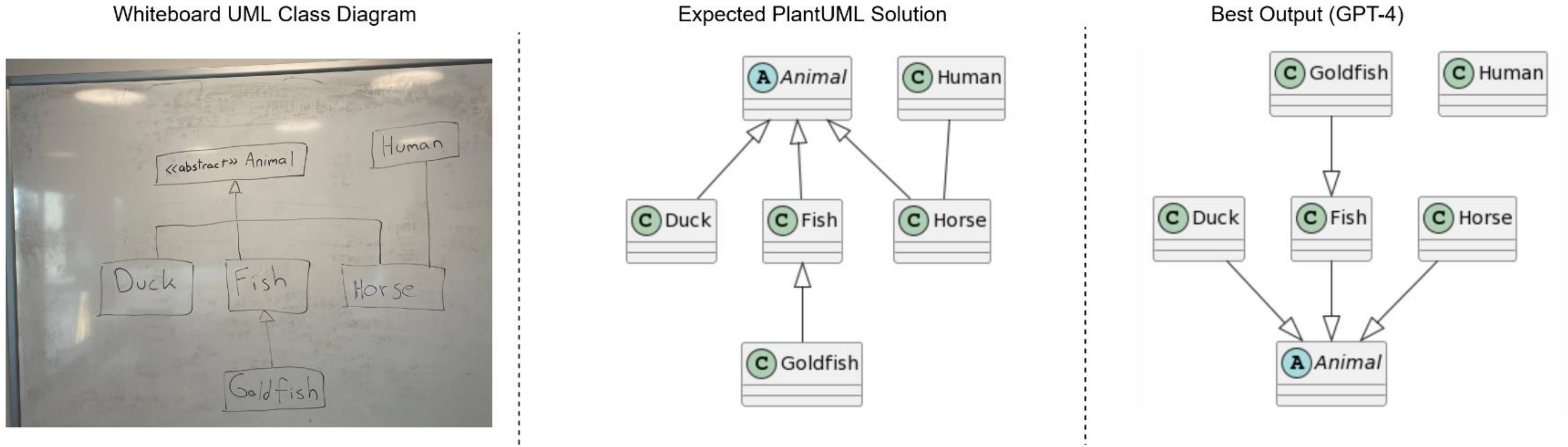
Building modeling agents from text

- **Challenge:** graphs cannot be directly fed to sequence models
- May use a **textual notation** (e.g., PlantUML) for the graph
  - Works best if this was already present in the training set
- Or turn the graph into **natural language sentences** (e.g., Eclipse Theia GLSP AI)
  - **String templates** and M2T generation
  - **Tool use** to create and edit graph model elements



Building modeling agents from text+image

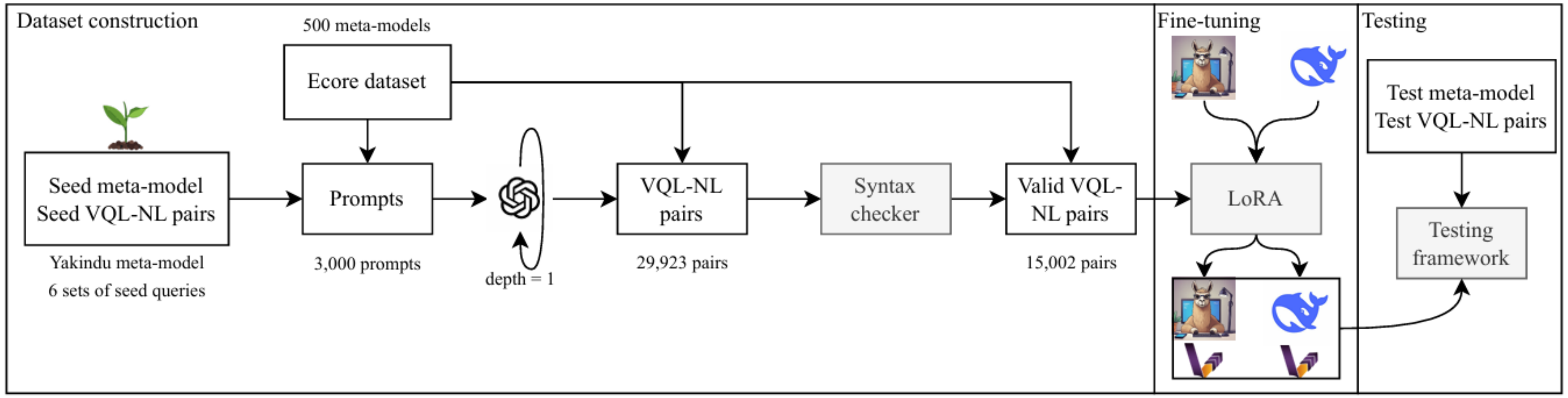
- **Multi-modal** models can take input other than text
  - E.g., images, sound
  - Internally, other modality inputs are often turned into special tokens
- Possible use-case: **image to model** (via PlantUML output)



Conrady and Cabot (2024). From Image to UML: First Results of Image-Based UML Diagram Generation using LLMs

Modeling agents with reasoning capacity

- ML applications: ✓ Fast learning and inference
  ✗ Limited expressive power
- **Idea:** mix ML applications with logic reasoning
  - ML application creates the design from **requirements** and **best practices**
  - Logic reasoning ensures that the design is **valid** and **consistent**



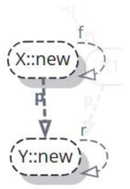
JAH López, M Földiák, D Varró: Text2VQL: Teaching a Model Query Language to Open-Source Language Models with ChatGPT. MODELS'24

Vision: Neuro-Symbolic Reasoning

complex structure

consistency guarantees

challenging domains



Refinery

graph transformation

logic reasoning

neural networks

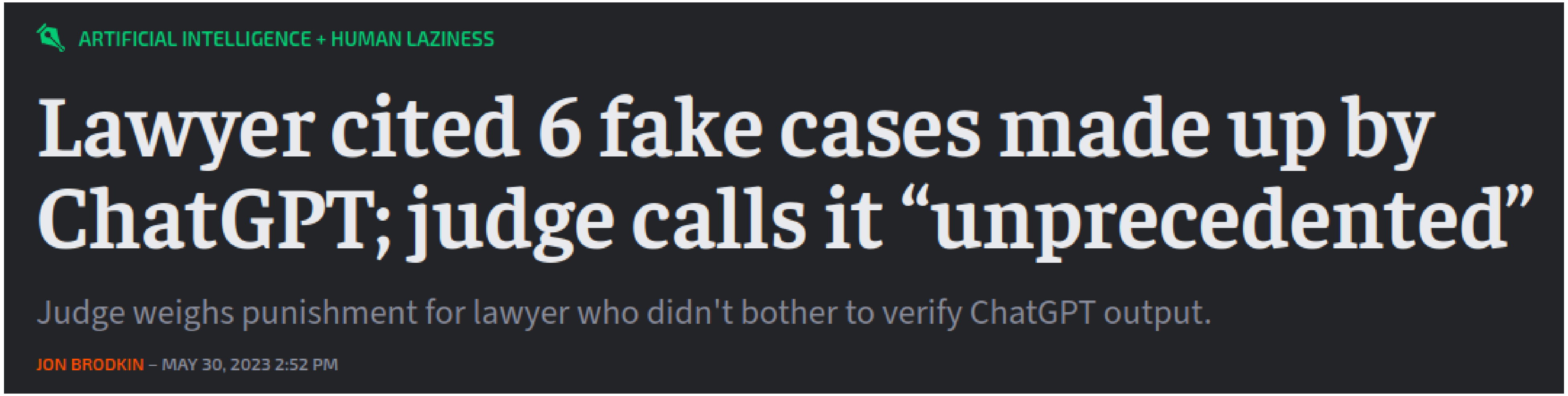
Ethical and legal aspects

What to watch out for when using LLMs for automation

Hallucinations and incorrect output

- Language models only offer **probabilistic guarantees** of correctness
  - Most of the time, output looks **statistically s**
- **Hallucination:** the LLM coming up with but incorrect output
- It is the duty of the model user to **verify** th output and avoid inadvertently introducing **serious errors** into the codebase

> - Simple case: the output uses a **library that does not exist** in reality
> - Harder case: the output **misses edge cases** or **error handling**


ARTIFICIAL INTELLIGENCE + HUMAN LAZINESS

Lawyer cited 6 fake cases made up by ChatGPT; judge calls it "unprecedented"

Judge weighs punishment for lawyer who didn't bother to verify ChatGPT output.

JON BRODKIN – MAY 30, 2023 2:52 PM

Copyright and attribution

- LLMs were trained on large amounts of open source code
  - But provide **no specific attribution** of sources
  - No guidelines for complying with the **licenses** of the code
- **Small amounts** of code are usually not subject to copyright issues
- But for reproducing **non-trivial portions** from the training data, the duty for **attribution** and **license compliance** falls on the model user
  - **Look up sources** manually


The Verge / Tech / Reviews / Science / Entertainment / AI / More +

The Verge / Tech / Reviews / Science / Entertainment / AI / More +

ARTIFICIAL INTELLIGENCE / TECH / POLICY

The developers suing over GitHub Copilot got dealt a major blow in court

/ A California judge dismissed nearly all claims laid out in a lawsuit that accuses GitHub, Microsoft, and OpenAI of copying code from developers.

By Emma Roth, a news writer who covers the streaming wars, consumer tech, crypto, social media, and much more. Previously, she was a writer and editor at MUO.

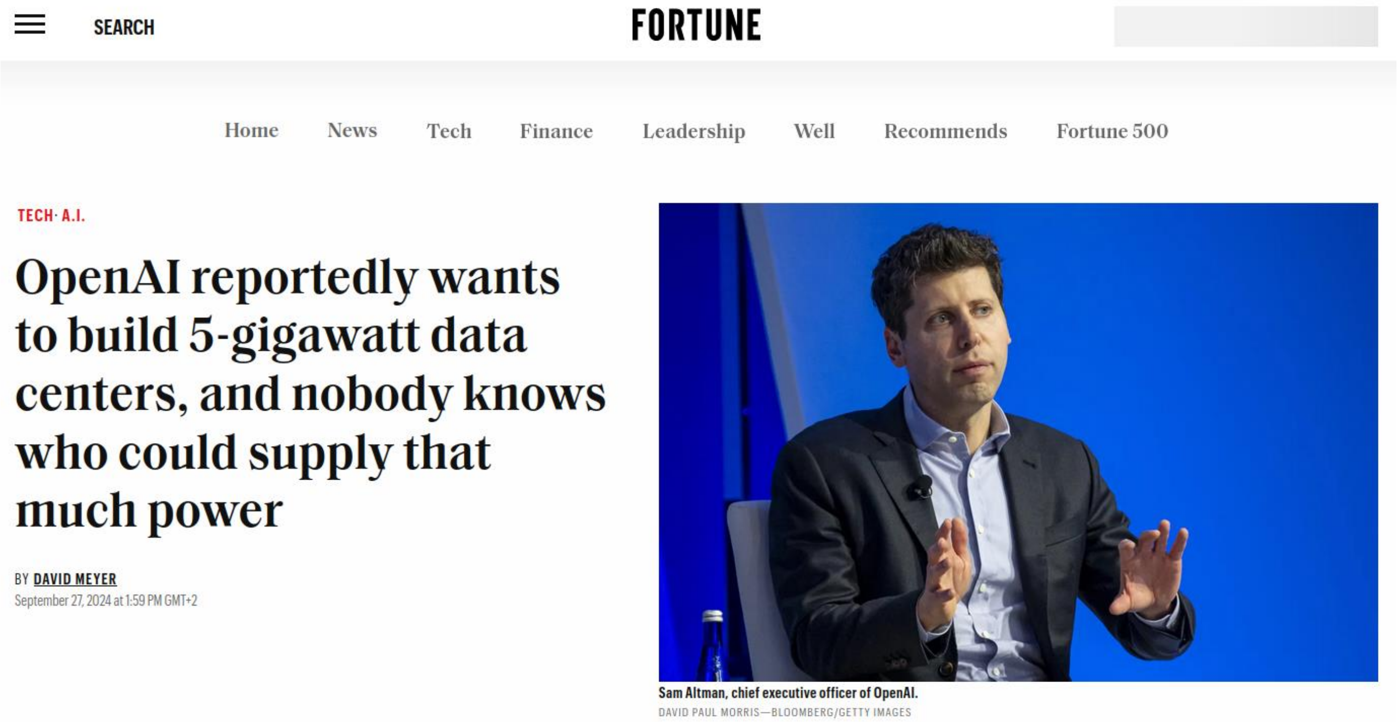
Jul 9, 2024, 10:09 PM UTC

Illustration by Cath Virginia / The Verge | Photos from Getty Images

ftsrg

Sustainability

- The largest language models require a disproportionate amount of **energy** and **natural resources** to run
- While further advances in inference hardware may reduce this in the future, we should be mindful of the efficiency of queries
- Often, code intelligence that does not involve natural language processing can be more effectively implemented with AST and DOM based method
  - **Lower** resource use
  - **Stronger** formal guarantees


SEARCH FORTUNE

Home News Tech Finance Leadership Well Recommends Fortune 500

TECH · A.I.

OpenAI reportedly wants to build 5-gigawatt data centers, and nobody knows who could supply that much power

BY DAVID MEYER

September 27, 2024 at 1:59 PM GMT+2

Sam Altman, chief executive officer of OpenAI.

DAVID PAUL MORRIS—BLOOMBERG/GETTY IMAGES