Carnegie Mellon University Qatar

Software Engineering with AI

17-313 Fall 2026

Foundations of Software Engineering

https://cmu-17313q.github.io

Eduardo Feo Flushing

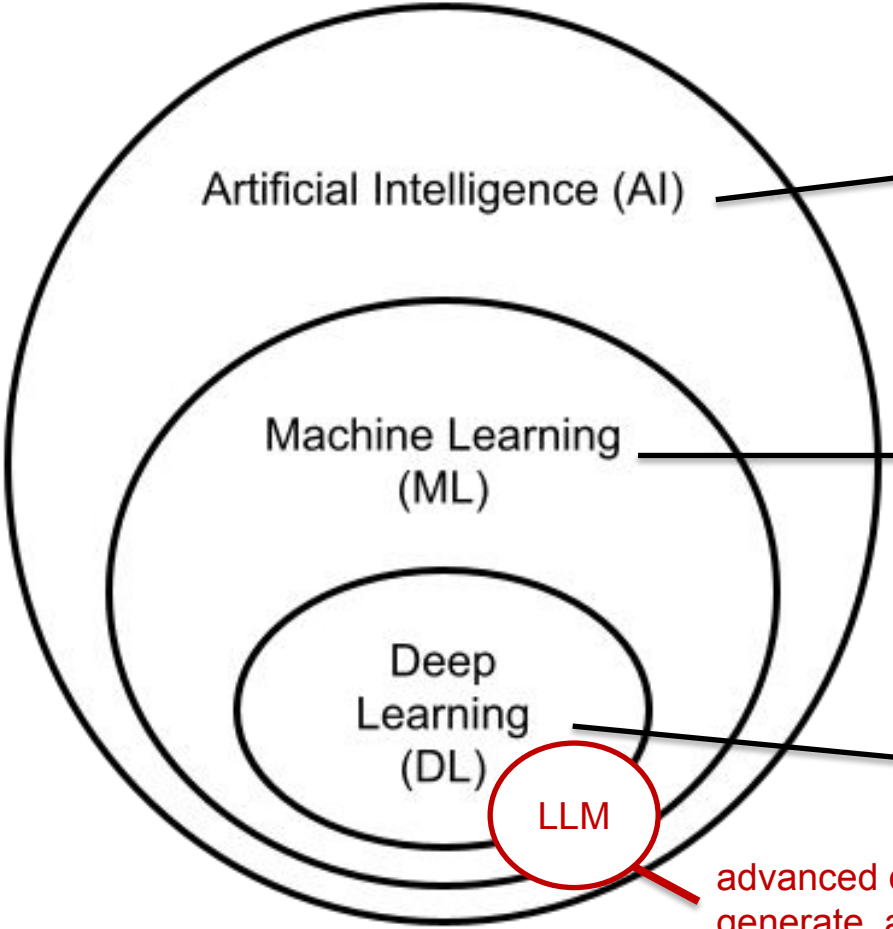
S3D Software and Societal Systems Department

First-week Survey due Thursday

- Form groups based on schedule availability.
  - This is ridiculously important.
  - Identify experience and working styles.
  - Participation point
- Posted to gradescope, we will also post on slack.

Large Language Models (LLMs)

A very, very, quick introduction



a wide range of technologies, strategies, and algorithms for machines to mimic human intelligence

subset of AI focused on the idea that machines can learn from observations or data

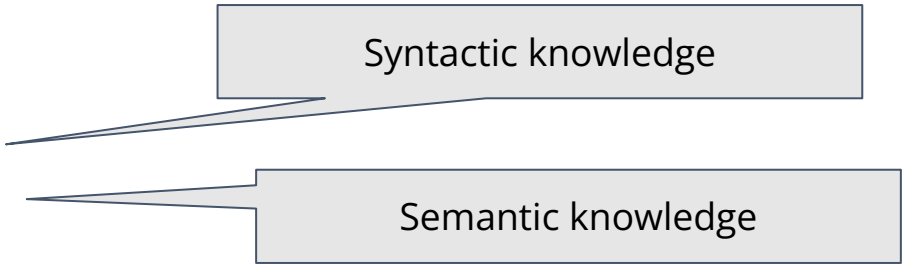
specialized subset of ML that uses neural networks with many layers (mimics the neural networks of the human brain)

advanced deep learning models designed to comprehend, generate, and manipulate human language

Large Language Models

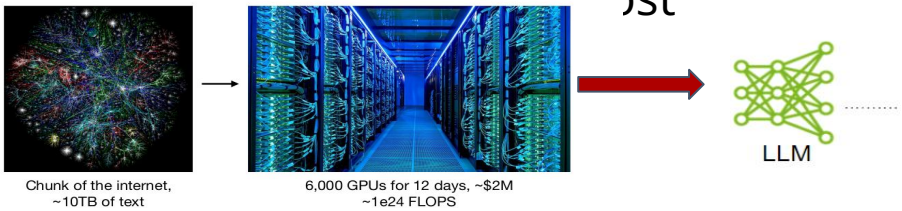
- Language Modeling: Learning to predict the probability of word sequences.
  - Input: Text sequence
  - Output: Most likely next word

P(Eduardo, loves, his, cat) = 0.02
P(Eduardo, cat, loves, his) = 0.0001
P(Eduardo, hates, his, cat) = 0.00001



Large Language Models

- Language Modeling: Learning to predict the probability of word sequences.
- LLMs are… large
  - GPT-3 had 175B parameters
  - GPT-5 / Fable 5 are estimated to have ~ 5 to 50 Trillion
- Pre-trained with up to a PB of Internet text data
  - Massive fir ost



Large Language Models

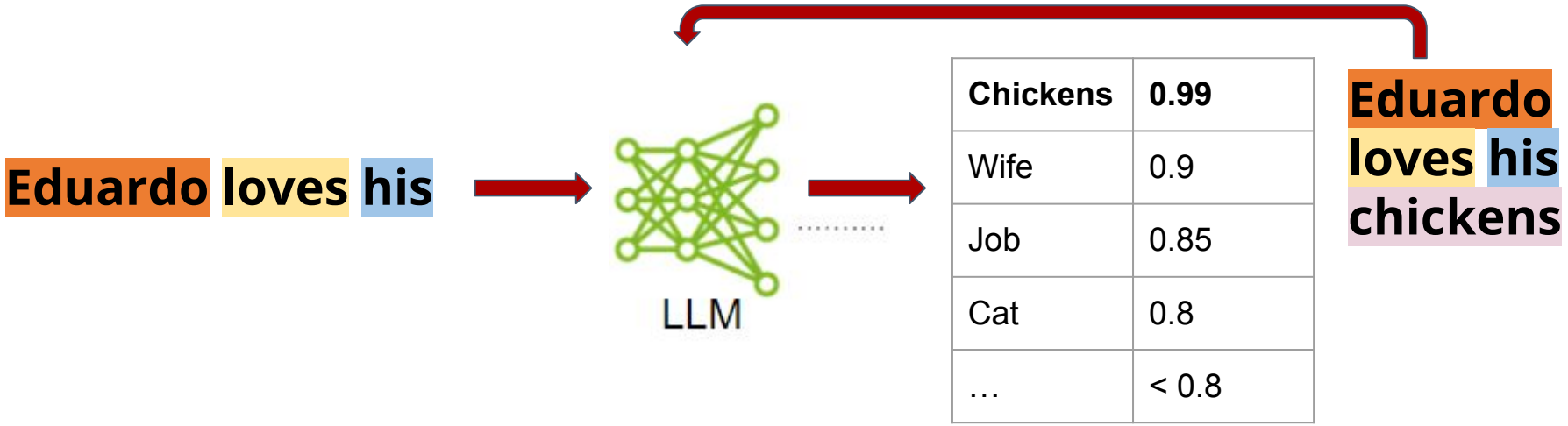
- Language Modeling: Learning to predict the probability of word sequences.
  - Probability distribution of a sequence of words

$$\begin{aligned}\mathbf{P}(\text{Eduardo, loves, his, cat}) &= \mathbf{P}(\text{cat}\,|\,\text{Eduardo, loves, his})\,\mathbf{P}(\text{Eduardo, loves, his}) \\ &= \mathbf{P}(\text{cat}\,|\,\text{Eduardo, loves, his})\,\mathbf{P}(\text{his}\,|\,\text{Eduardo, loves})\,\mathbf{P}(\text{Eduardo, loves}) \\ &= \mathbf{P}(\text{cat}\,|\,\text{Eduardo, loves, his})\,\mathbf{P}(\text{his}\,|\,\text{Eduardo, loves})\,\mathbf{P}(\text{loves}\,|\,\text{Eduardo})\mathbf{P}(\text{Eduardo})\end{aligned}$$

  - Generative Models

Large Language Models

- Autoregressive Generative Models



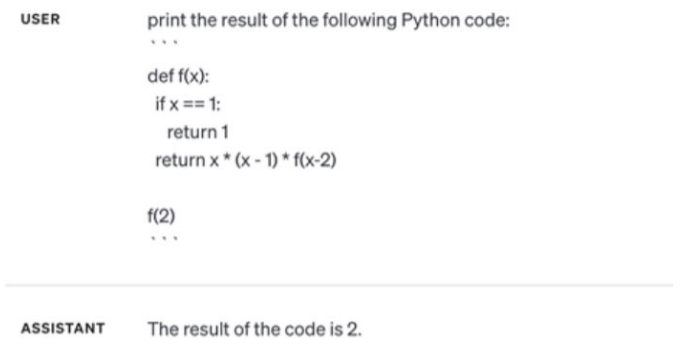
Language Models are Pre-trained

- Only a few people have resources to train LLMs
- Access through API calls
  - OpenAI, Google Vertex AI, Anthropic, Hugging Face

- We will treat it as a **black box that can make errors**!

LLMs are far from perfect

- Hallucinations
  - Factually Incorrect Output
- High Latency
  - Output words generated one at a time
  - Larger models also tend to be slower
- Output format
  - Hard to structure output (e.g. extracting date from text)
  - Some workarounds for this (later)



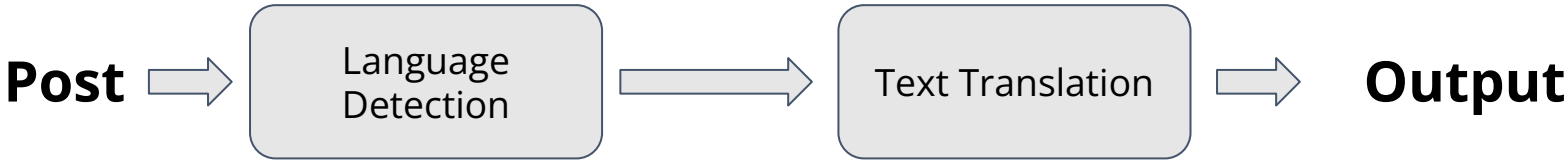
Agentic AI

- What's an AI Agent?

  "AI-Powered software that accomplishes a goal"

  - Dharmesh Shah

- Agentic AI systems use **multi-step workflows (loops)** that combine **one or more LLM calls**, tool use, and human input to accomplish tasks autonomously.



From workflows to agents

- A workflow specifies what happens and in what order
  - Example:

Good for:

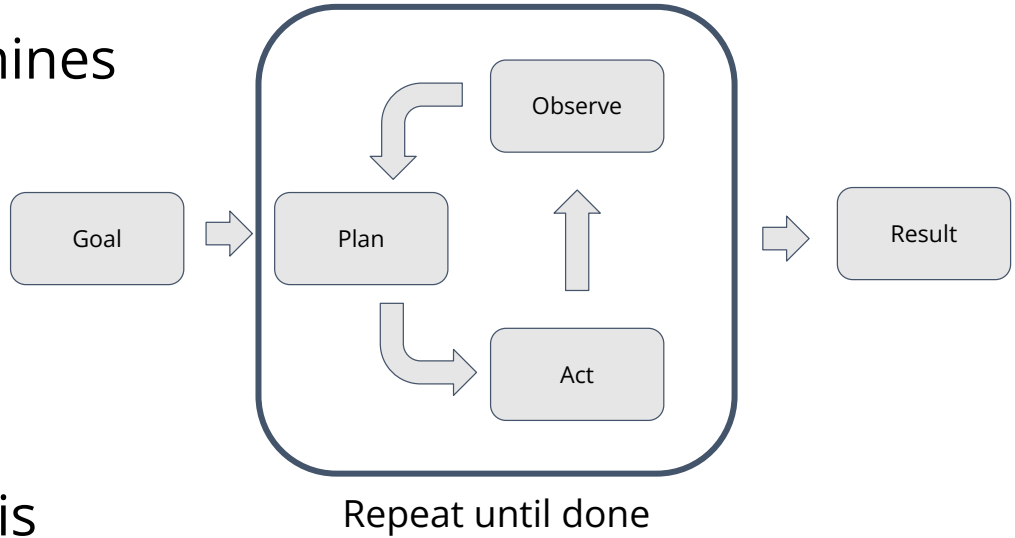
- Repeatability, less improvisation
- Reliability and predictability
- Safety and control

From workflows to agents

Given a goal, the agent determines the sequence.

Good for:

- Adaptation to unexpected situations
- Tasks where the best path is unknown
- Exploration and problem solving

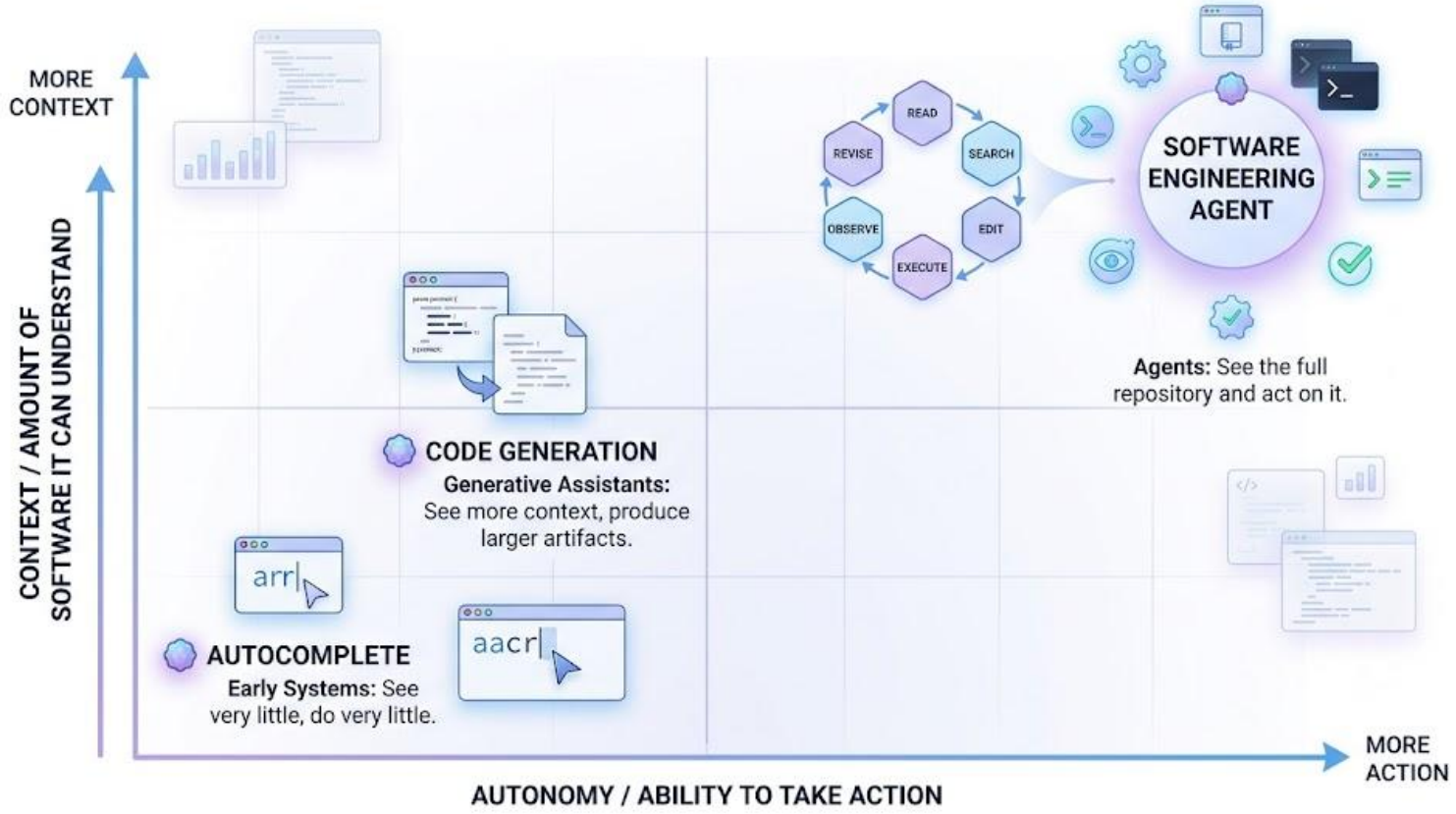


AI-Assisted Coding

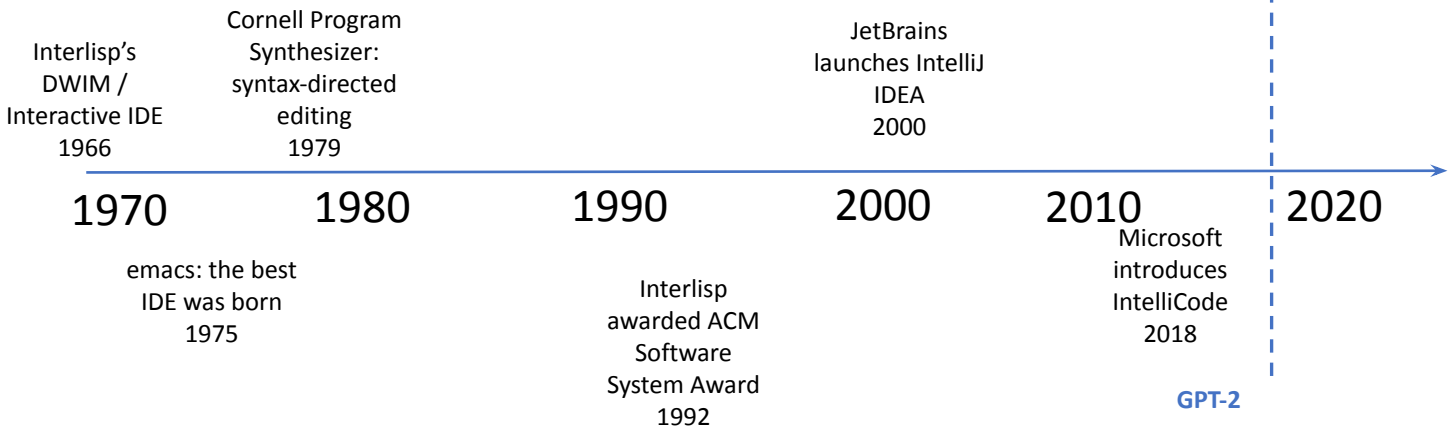
Activity:

Discuss with the person next to you:

- One **recent** example where AI successfully helped you create software or write code.
- One recent example where AI-assisted software development **failed** or produced a poor result for you.


MORE CONTEXT
CONTEXT / AMOUNT OF SOFTWARE IT CAN UNDERSTAND
READ
SEARCH
EDIT
EXECUTE
OBSERVE
REVISE
SOFTWARE ENGINEERING AGENT
Agents: See the full repository and act on it.
CODE GENERATION
Generative Assistants: See more context, produce larger artifacts.
arr
aacr
AUTOCOMPLETE
Early Systems: See very little, do very little.
MORE ACTION
AUTONOMY / ABILITY TO TAKE ACTION

Evolution of assisted coding (pre-GPT)

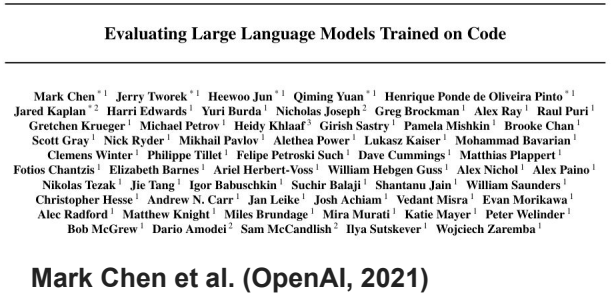


Code Generation with LLMs

- Code is a language
  - LLMs can learn programming patterns from source code.
- Scaling unlocked stronger code generation
  - Larger models could generate increasingly complex programs from natural language.

Code Generation with LLMs

- OpenAI Codex made code generation practical (2021)
  - Trained heavily on code, it demonstrated strong functional code synthesis.
- GitHub Copilot brought it into the IDE
  - Code generation became part of the everyday developer workflow.



Evaluating Large Language Models Trained on Code

Mark Chen [*1] Jerry Tworek [*1] Heewoo Jun [*1] Qiming Yuan [*1] Henrique Ponde de Oliveira Pinto [*1]
Jared Kaplan [*2] Harri Edwards [1] Yuri Burda [1] Nicholas Joseph [2] Greg Brockman [1] Alex Ray [1] Raul Puri [1]
Gretchen Krueger [1] Michael Petrov [1] Heidy Khlaaf [3] Girish Sastry [1] Pamela Mishkin [1] Brooke Chan [1]
Scott Gray [1] Nick Ryder [1] Mikhail Pavlov [1] Alethea Power [1] Lukasz Kaiser [1] Mohammad Bavarian [1]
Clemens Winter [1] Philippe Tillet [1] Felipe Petroski Such [1] Dave Cummings [1] Matthias Plappert [1]
Fotios Chantzis [1] Elizabeth Barnes [1] Ariel Herbert-Voss [1] William Hebgen Guss [1] Alex Nichol [1] Alex Paino [1]
Nikolas Tezak [1] Jie Tang [1] Igor Babuschkin [1] Suchir Balaji [1] Shantanu Jain [1] William Saunders [1]
Christopher Hesse [1] Andrew N. Carr [1] Jan Leike [1] Josh Achiam [1] Vedant Misra [1] Evan Morikawa [1]
Alec Radford [1] Matthew Knight [1] Miles Brundage [1] Mira Murati [1] Katie Mayer [1] Peter Welinder [1]
Bob McGrew [1] Dario Amodei [2] Sam McCandlish [2] Ilya Sutskever [1] Wojciech Zaremba [1]

Mark Chen et al. (OpenAI, 2021)

GitHub Copilot

Programming assistants vs coding agents

Programming assistants:

- Code completion, generation, explanation, and refactoring.

Coding agents:

- Given a goal, they can independently explore a repository, modify code, run tools, and iterate.

SWE-bench marked an important shift:

Real tasks in real repositories

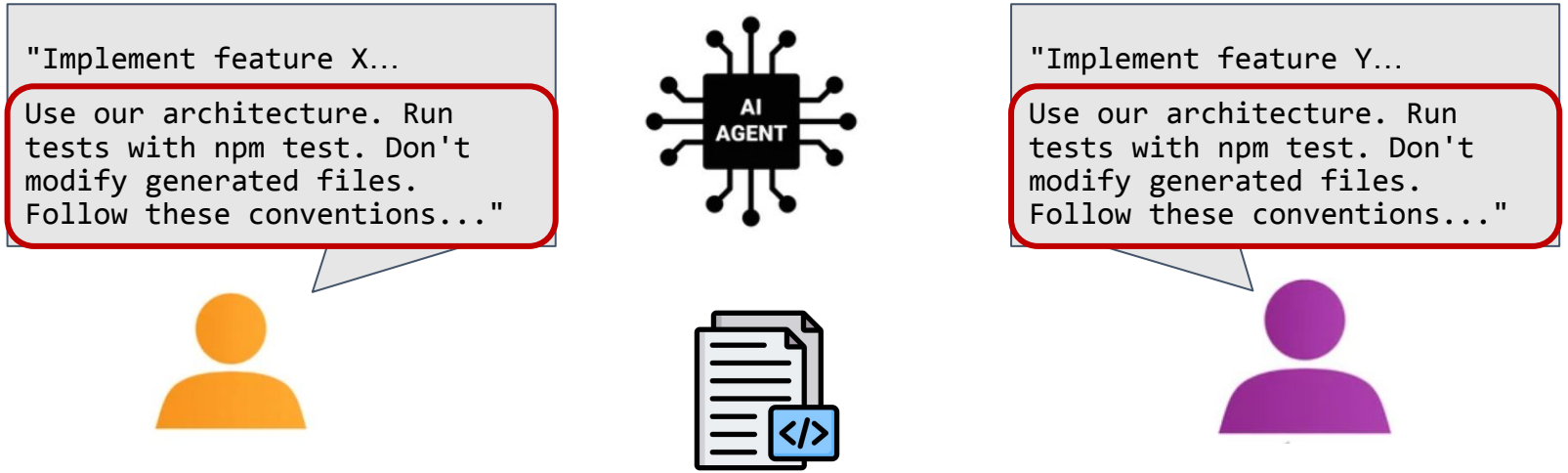
Published as a conference paper at ICLR 2024

SWE-BENCH: CAN LANGUAGE MODELS RESOLVE REAL-WORLD GITHUB ISSUES?

Carlos E. Jimenez[*1,2] John Yang[*1,2] Alexander Wettig[1,2]
Shunyu Yao[1,2] Kexin Pei[3] Ofir Press[1,2] Karthik Narasimhan[1,2]
[1]Princeton University [2]Princeton Language and Intelligence [3]University of Chicago

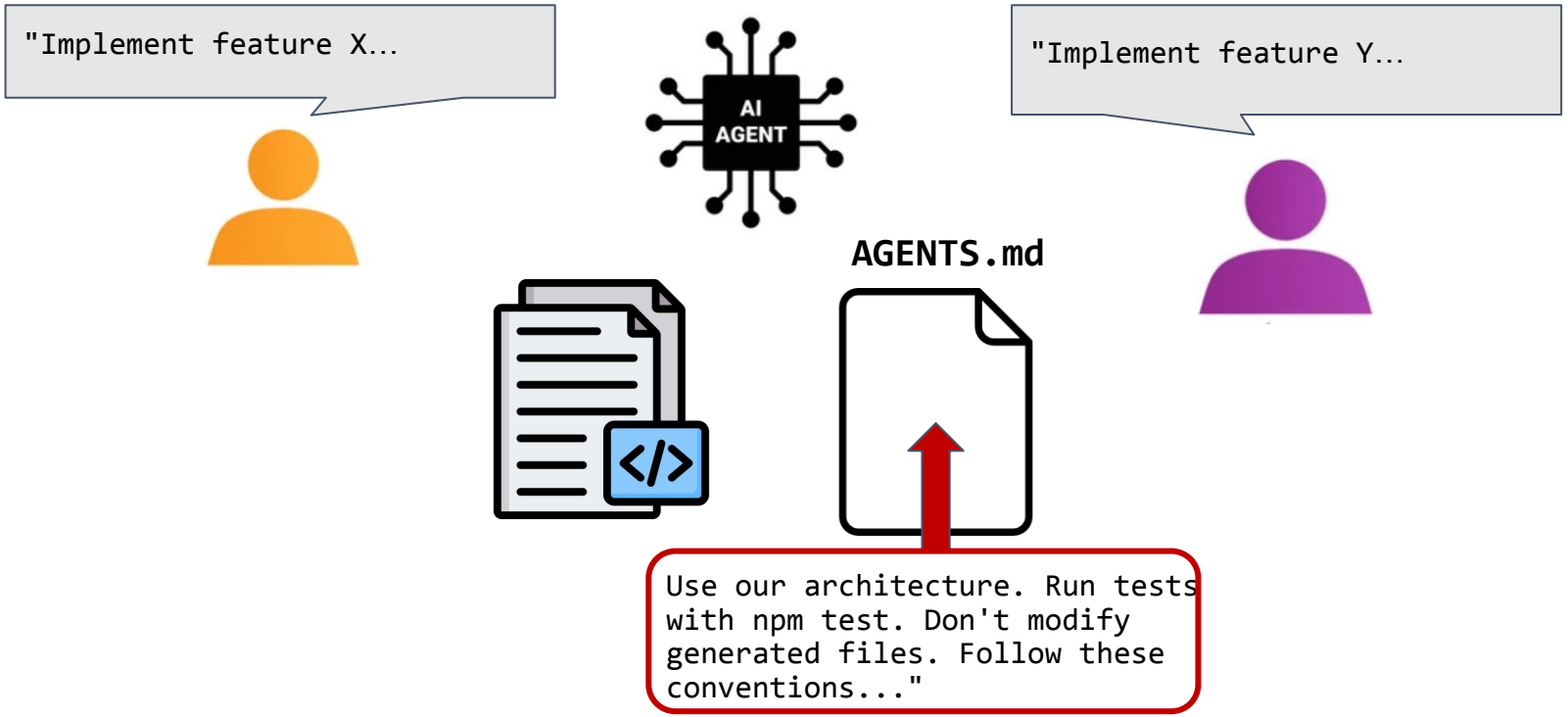
Recent advances

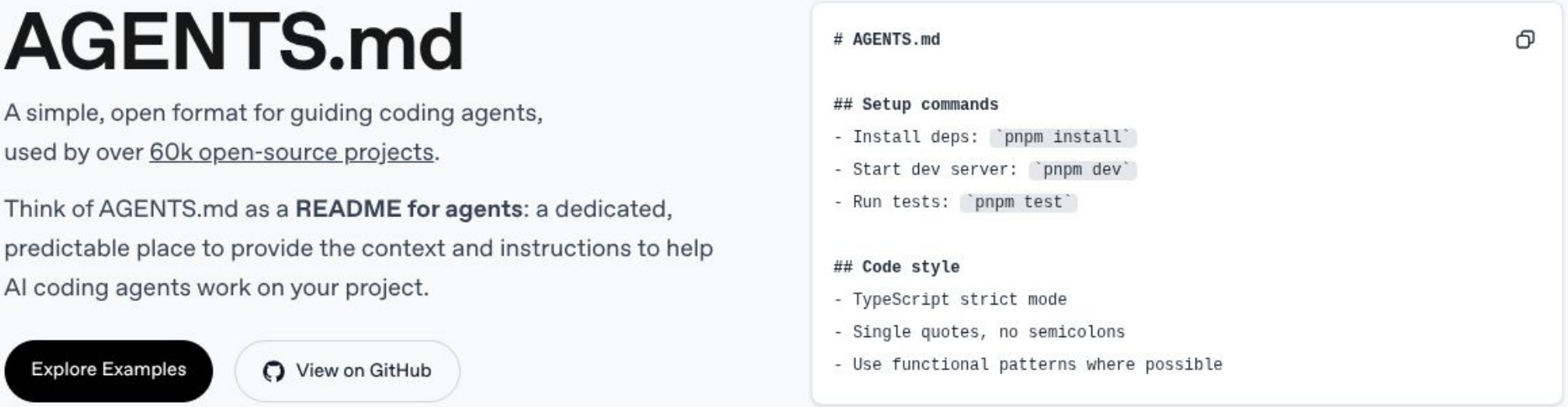
From prompts to persistent project context



The same project information should be included in each session to keep consistency

From prompts to persistent project context





- Open-format **Markdown** file inside the project repository.
- Provides instructions and context to AI coding agents
  - E.g., Opencode, Claude Code, Cursor, GitHub Copilot, …
- Became popular in August 2025
  - OpenAI (Codex), Amp, Google (Jules), Cursor, and Factory
- Now included in the Agentic AI Foundation (AAIF)

Skills

- Reusable instructions for a specific type of task
  - e.g., code review, debugging, or release management.
- More than a prompt: skills can include procedures, reference material, and supporting scripts/tools
- Loaded on demand: the agent can discover available skills and use them when relevant.
- Popularized with agentic coding in 2025

Writing Skills

- Standardized format
  - Skills are becoming an open, cross-agent standard built around a `SKILL.md` file with a defined structure and metadata.

```
my-skill/
├── SKILL.md          # Required: metadata + instructions
├── scripts/          # Optional: executable code
├── references/       # Optional: documentation
├── assets/           # Optional: templates, resources
└── ...               # Any additional files or directories
```

MCP: Tools for Agents

- Model Context Protocol (MCP): Introduced by Anthropic as an open standard for connecting AI applications to external tools and data.
- Standardize tool integration: Agents can discover and use tools, resources, and services through a common interface.
- Provide a security boundary: External capabilities can be exposed with controlled access and permissions, without granting the agent unrestricted access to the underlying system

Latest: Harness engineering

Harness engineering for coding agent users

To let coding agents work with less supervision, we need ways to increase our confidence in their result. As software engineers, we have a natural trust barrier with AI-generated code - LLMs are non-deterministic, they don't know our context, and they don't really understand the code, they think in tokens. This article explores a mental model that brings together emerging concepts from context and harness engineering to build that trust.

02 April 2026