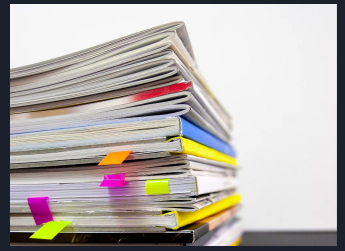
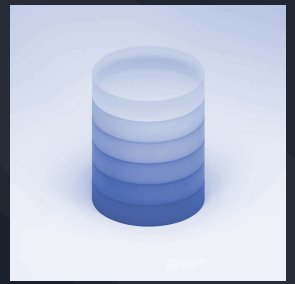
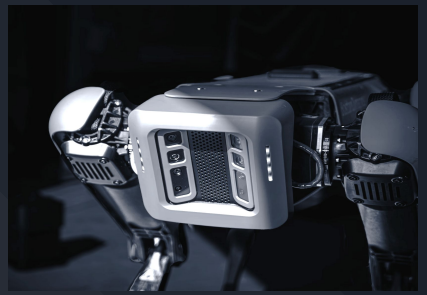


Ax

Build advanced LLM powered Agents





```
const ai = AI('openai', { apiKey: proc
ai.setOptions({ debug: true });

const researcher = new Agent(ai, {
  name: 'researcher',
  description: 'Researcher agent',
  signature: `physicsQuestion "physics
});

const summarizer = new Agent(ai, {
  name: 'summarizer',
  description: 'Summarizer agent',
  signature: `text "text so summarize"
});

```

Motivation & Goals

“A sophisticated production ready library that empowers everyone to harness the transformative power of LLMs and build the agentic applications of the future. “

1. Leverage In-context learnings
2. Build agentic workflows
3. Vertically integrated
4. Simple yet powerful API

Andrew Ng

Cofounder and head of Google Brain, former Chief Scientist at Baidu

"I expect that the set of tasks AI can do will expand dramatically this year because of agentic workflows…"

Features

Composible Prompts

Tunable Prompts

Typed Prompts

Streaming

Multi-modal

Agent Building

No dependencies

Full Observability

Stanford DSP Paper ❤️

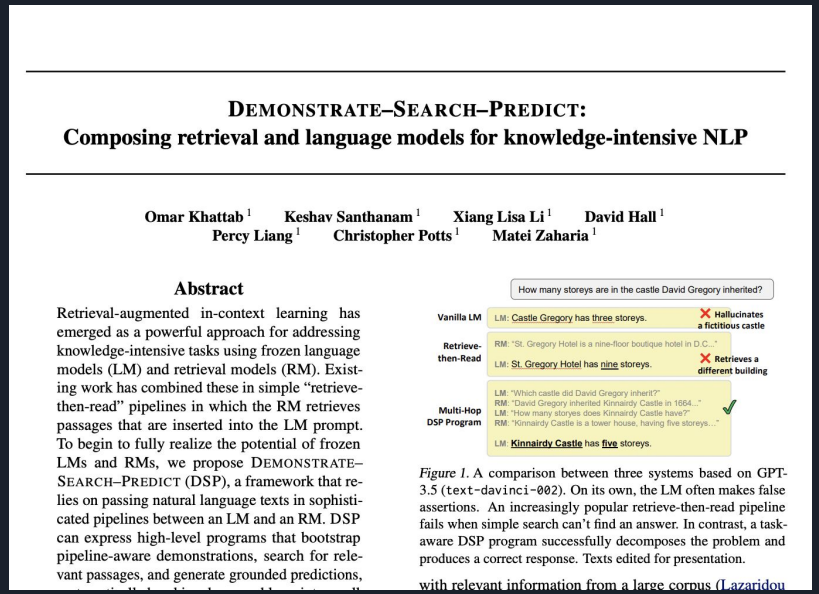
The Stanford DSP Paper

Author = Omar Khattab and Keshav Santhanam and Xiang Lisa Li and David Hall and Percy Liang and Christopher Potts and Matei Zaharia

Title = Demonstrate-Search-Predict: Composing retrieval and language models for knowledge-intensive NLP

Year = 2022

Eprint = arXiv:2212.14024



DEMONSTRATE–SEARCH–PREDICT:
Composing retrieval and language models for knowledge-intensive NLP

Omar Khattab [1] **Keshav Santhanam** [1] **Xiang Lisa Li** [1] **David Hall** [1]
Percy Liang [1] **Christopher Potts** [1] **Matei Zaharia** [1]

Abstract

Retrieval-augmented in-context learning has emerged as a powerful approach for addressing knowledge-intensive tasks using frozen language models (LM) and retrieval models (RM). Existing work has combined these in simple "retrieve-then-read" pipelines in which the RM retrieves passages that are inserted into the LM prompt. To begin to fully realize the potential of frozen LMs and RMs, we propose DEMONSTRATE–SEARCH–PREDICT (DSP), a framework that relies on passing natural language texts in sophisticated pipelines between an LM and an RM. DSP can express high-level programs that bootstrap pipeline-aware demonstrations, search for relevant passages, and generate grounded predictions,

How many storeys are in the castle David Gregory inherited?

Vanilla LM
LM: Castle Gregory has three storeys.
✗ Hallucinates a fictitious castle

Retrieve-then-Read
RM: "St. Gregory Hotel is a nine-floor boutique hotel in D.C..."
LM: St. Gregory Hotel has nine storeys.
✗ Retrieves a different building

Multi-Hop DSP Program
LM: "Which castle did David Gregory inherit?"
RM: "David Gregory inherited Kinnairdy Castle in 1664..."
LM: "How many storyes does Kinnairdy Castle have?"
RM: "Kinnairdy Castle is a tower house, having five storeys..."
LM: Kinnairdy Castle has five storeys.
✓

Figure 1. A comparison between three systems based on GPT-3.5 (`text-davinci-002`). On its own, the LM often makes false assertions. An increasingly popular retrieve-then-read pipeline fails when simple search can't find an answer. In contrast, a task-aware DSP program successfully decomposes the problem and produces a correct response. Texts edited for presentation.

with relevant information from a large corpus (Lazaridou

Prompt Signatures

Task instructions

"Help answer simple questions using the context if provided"

```
question:string,  context?:string[]  ->  answer:string
```

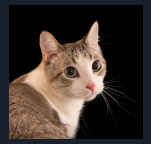
Input field with type

Optional Input field with type

Output field with type

Multi-modal Prompts

What family does this animal belong to?



-> Felidae

question:string, animal:image -> answer:string

Input field with type

Image input field

Output field with type

Prompt Signatures

Task instructions

"Help answer simple questions using the context if provided"

```
question:string, context?:string[] -> reason:string, answer:string
```

Input field with type

Optional Input field with type

Added output field

Output field with type

Why Prompt Signatures?

Types allow for automatic assertions

Field names provide context in the prompt

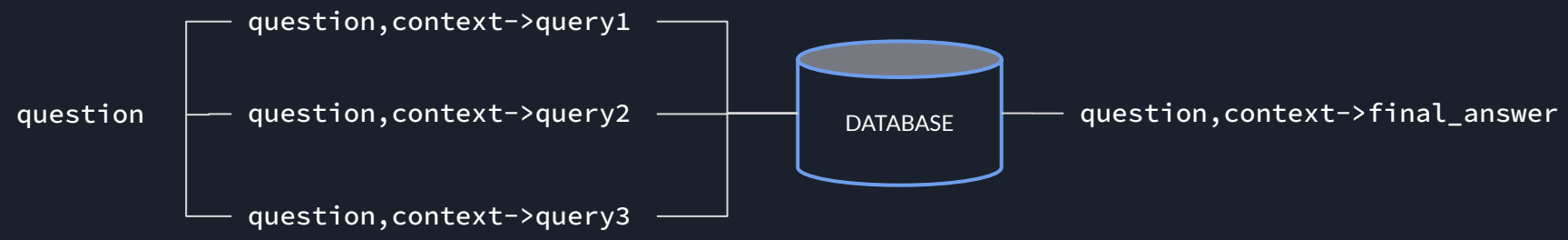
Error correction is simpler

Prompt programs are trees of fields

Easy to set examples and save traces

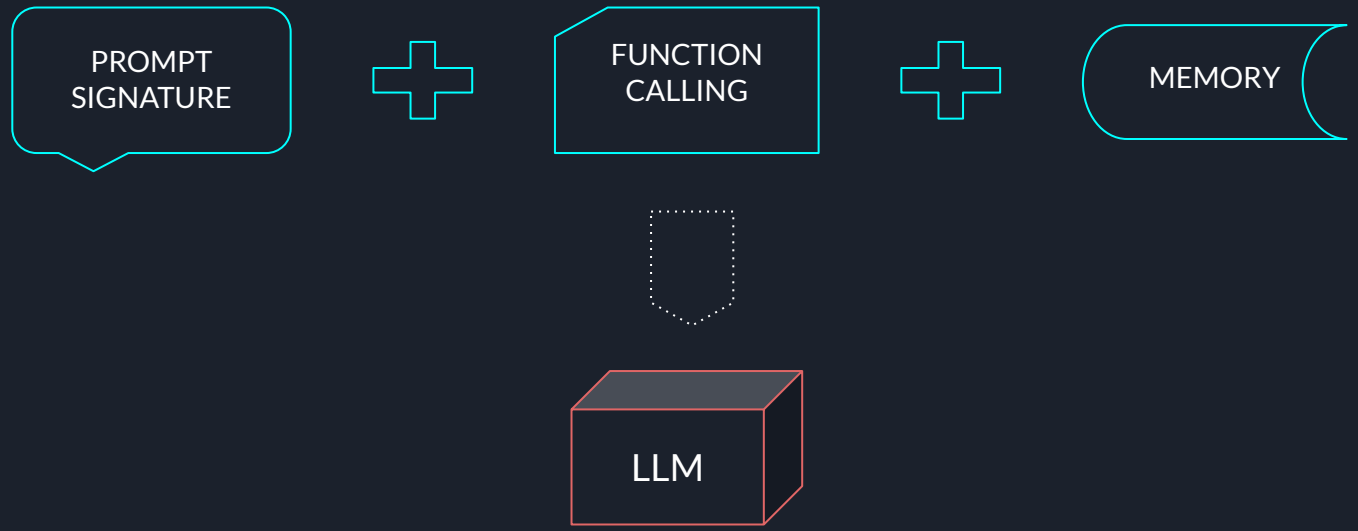
```
companyInfo:string,  stockDetails?:string[] ->  summary:string
```

Prompt Programs



"Programming over prompting philosophy of the Stanford DSP Paper."

An Agent

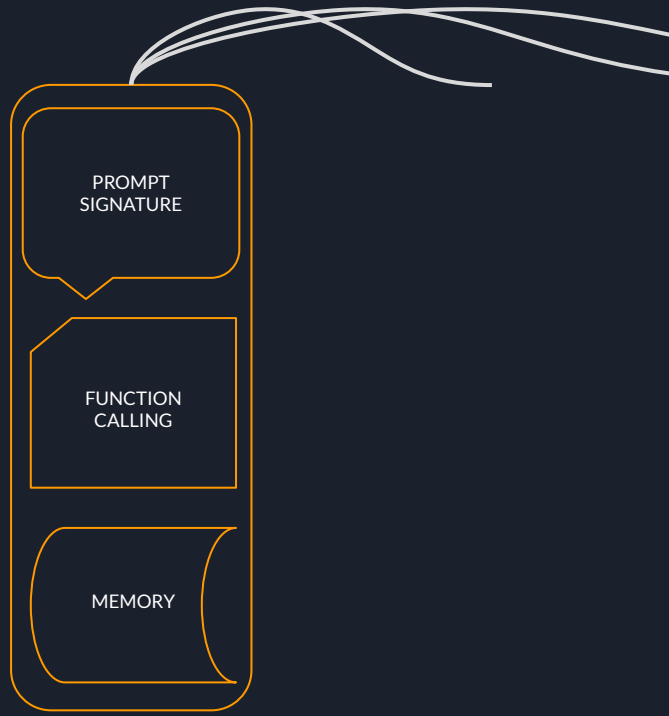


Long Horizon Tasks

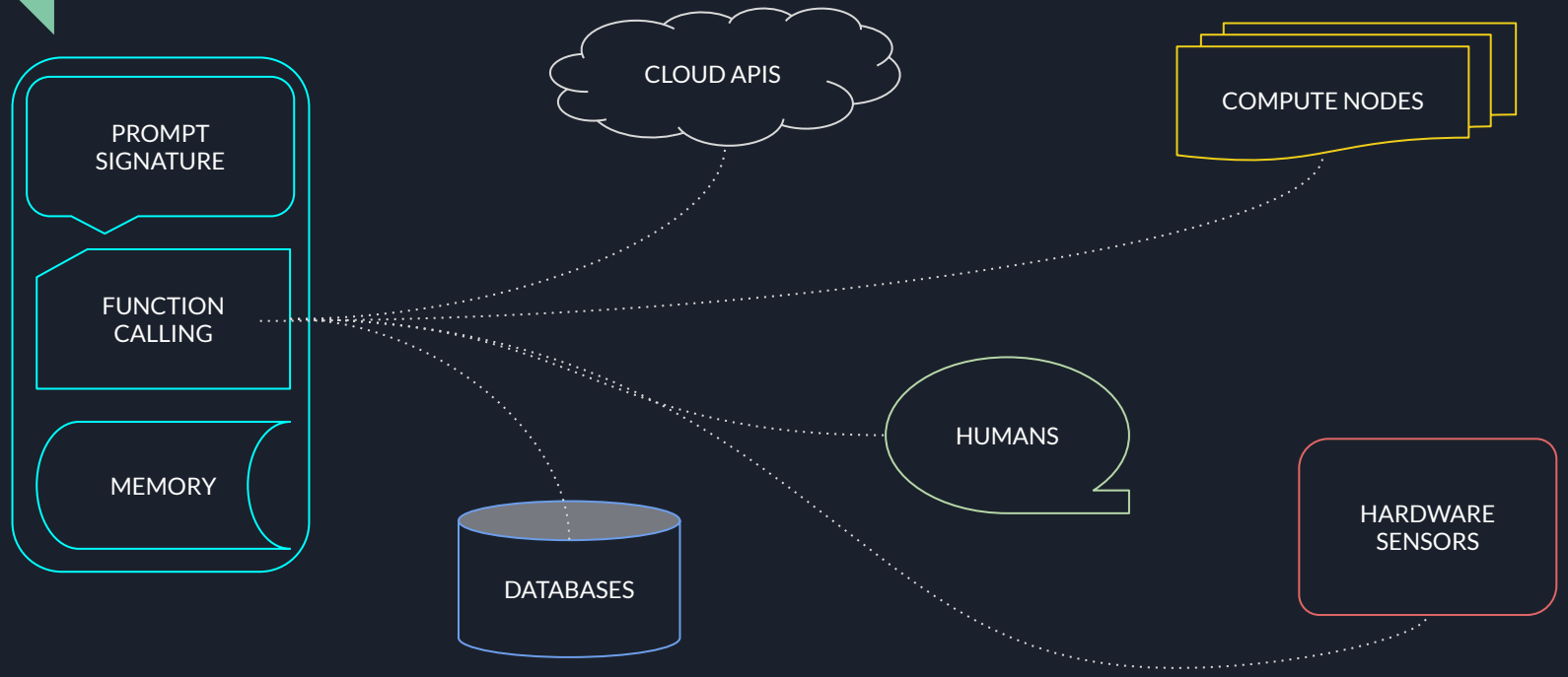
Problem: Our current long context models are not good at long horizon tasks.

Solution: Break the problem down into tasks and solve each in an independent context.

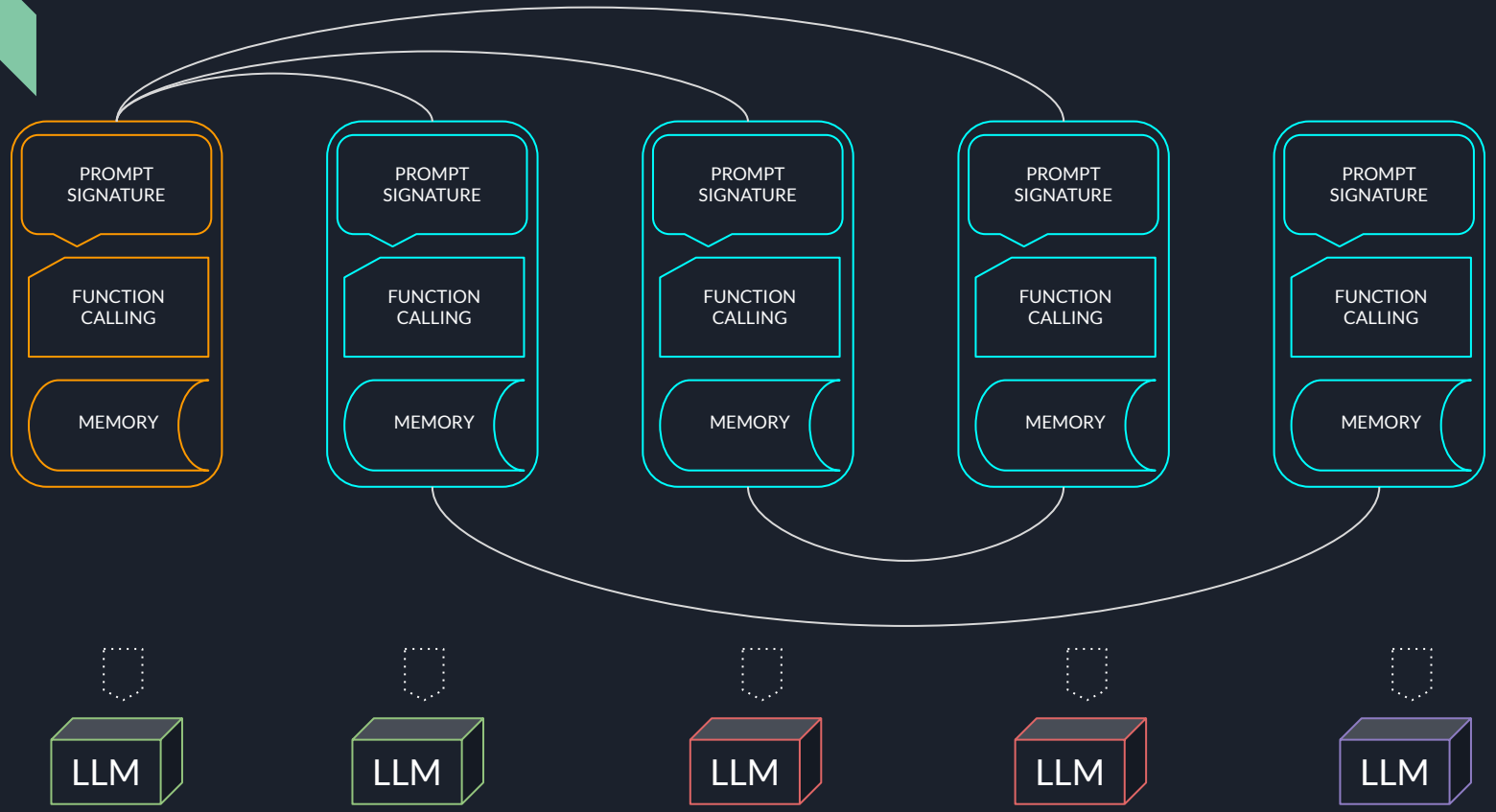
Advantage: Cheaper, Faster and more flexible



Functions / Tools



Teams of Agents



Streaming Error Correction

Summary: 1. Exponential Progress: The singularity concept hinges on the obser…\n2. Intelligence Explosion: A key prediction is that th…\n3. Unforeseeable Future: Beyond the singula…

Summary: 1. Exponential Progress: The singularity concept hinges on the obser…\nIntelligence Explosion…
ERROR: Missing numeric prefix eg. 2.

Create a Prompt Program

```
# Pick an LLM
const ai = new AxAI('openai', { apiKey: OPENAI_APIKEY });

# Create a Prompt Signature
const gen = new AxGenerate(ai, `text -> shortSummary`);

# Use it
const { shortSummary } = await gen.forward({ text });

# See summary
console.log(shortSummary);
```

Agents & Agents using agents

```
const researcher = new AxAgent(ai, {
  name: 'researcher',
  description: 'Researcher agent',
  signature: `question -> answer`,
  agents: [wolframAgent, arxivSearchAgent, pythonAgent]
});

researcher.forward({
   question: “How many atoms are in the universe"
})
```

Thanks, we have an interesting road ahead.

1. Remote agent collaboration
2. Human + LLM interfacing
3. Active learning with auto tuning
4. Prompt optimization strategies

And more…

twitter.com/dosco

Included images are collections on Unsplash some by Google Deepmind