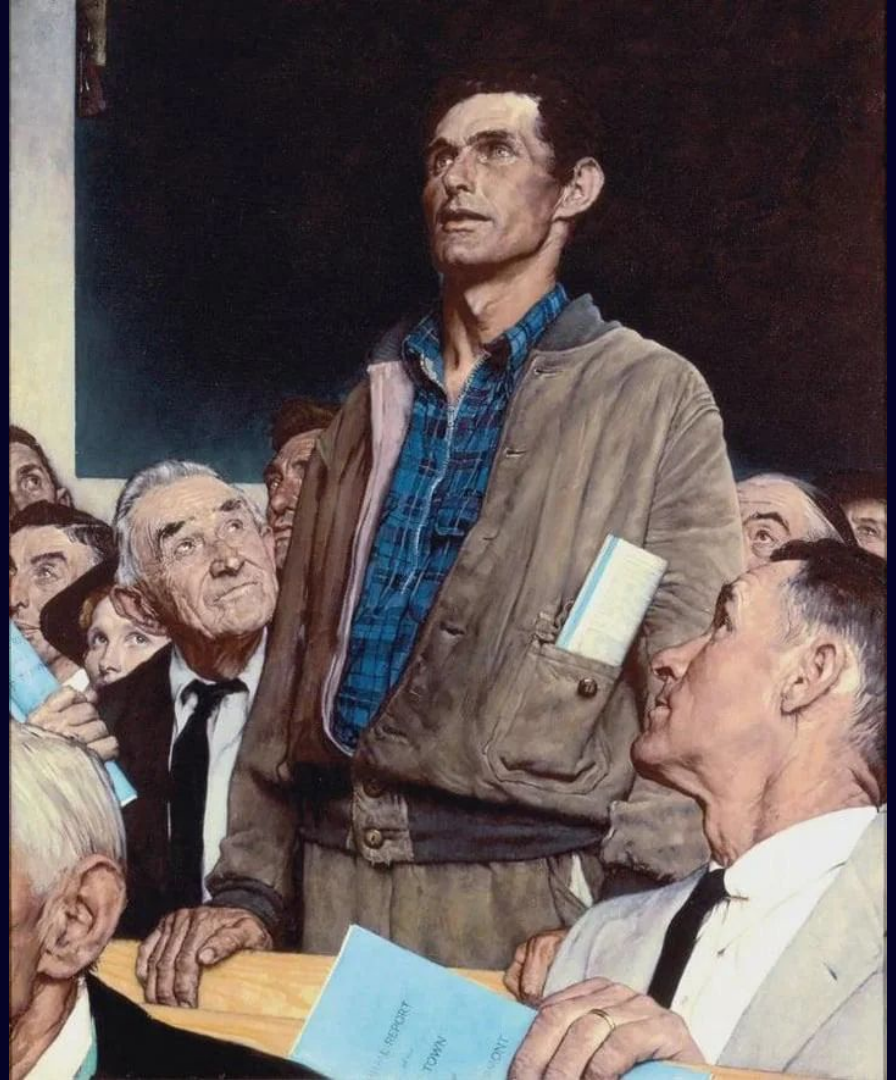
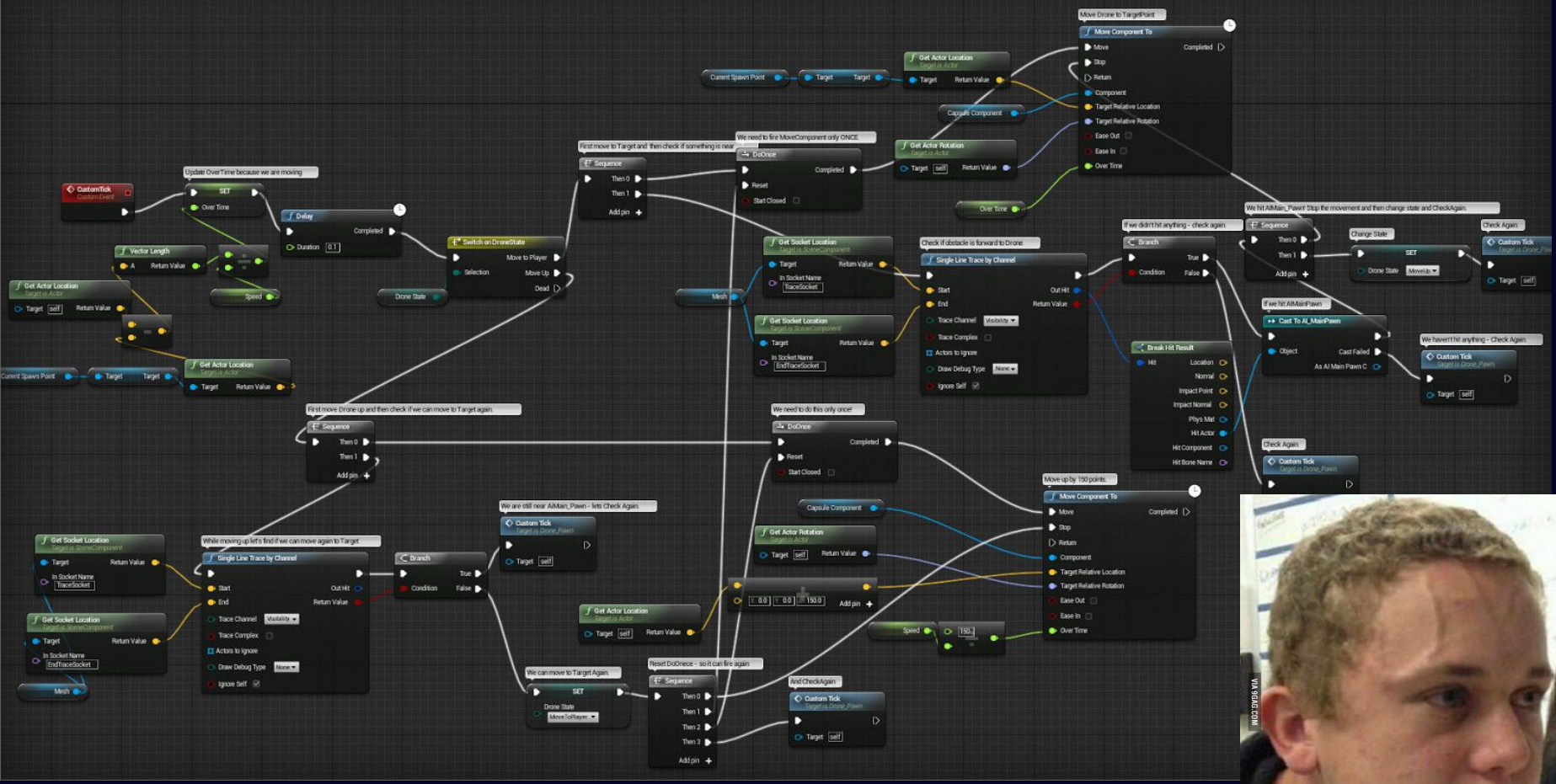

REPORT
TOWN

What you think you need:


Move Drone to TargetPoint
Move Component To
Get Actor Location
Current Spawn Point
Target
Capsule Component
Get Actor Rotation
Over Time
We need to fire MoveComponent only ONCE
DoOnce
First move to Target and then check if something is near
Sequence
Update OverTime because we are moving
CustomTick
SET
Over Time
Delay
Duration
Completed
Vector Length
Speed
Drone State
Switch on DroneState
Move to Player
Move Up
Dead
Selection
Get Socket Location
In Socket Name
TraceSocket
EndTraceSocket
Mesh
Check if obstacle is forward to Drone
Single Line Trace by Channel
Start
End
Out Hit
Return Value
Trace Channel
Visibility
Trace Complex
Actors to Ignore
Draw Debug Type
None
Ignore Self
If we didn't hit anything - check again
Branch
True
False
Condition
We hit AIMain_Pawn! Stop the movement and then change state and CheckAgain.
Change State
Check Again
Custom Tick
If we hit AIMainPawn
Cast To AI_MainPawn
Object
Cast Failed
As AI Main Pawn C
We haven't hit anything - Check Again
Break Hit Result
Hit
Location
Normal
Impact Point
Impact Normal
Phys Mat
Hit Actor
Hit Component
Hit Bone Name
First move Drone up and then check if we can move to Target again
We need to do this only once!
Move up by 150 points
While moving up let's find if we can move again to Target
We are still near AIMain_Pawn - lets Check Again
We can move to Target Again
Reset DoOnece - so it can fire again
And CheckAgain
Add pin
Then 0
Then 1
Then 2
Then 3
Reset
Start Closed

VIA 9GAG.COM


Royal
Masquerade


CURSOR



GREP

Is Grep All You Need?
How Agent Harnesses Reshape Agentic Search

Sahil Sen
PricewaterhouseCoopers, U.S.
sahil.s.sen@pwc.com

Akhil Kasturi
PricewaterhouseCoopers, U.S.
akhil.kasturi@pwc.com

Elias Lumer
PricewaterhouseCoopers, U.S.
elias.lumer@pwc.com

Anmol Gulati
PricewaterhouseCoopers, U.S.

Vamse Kumar Subbiah
PricewaterhouseCoopers, U.S.

CL] 14 May 2026

Abstract

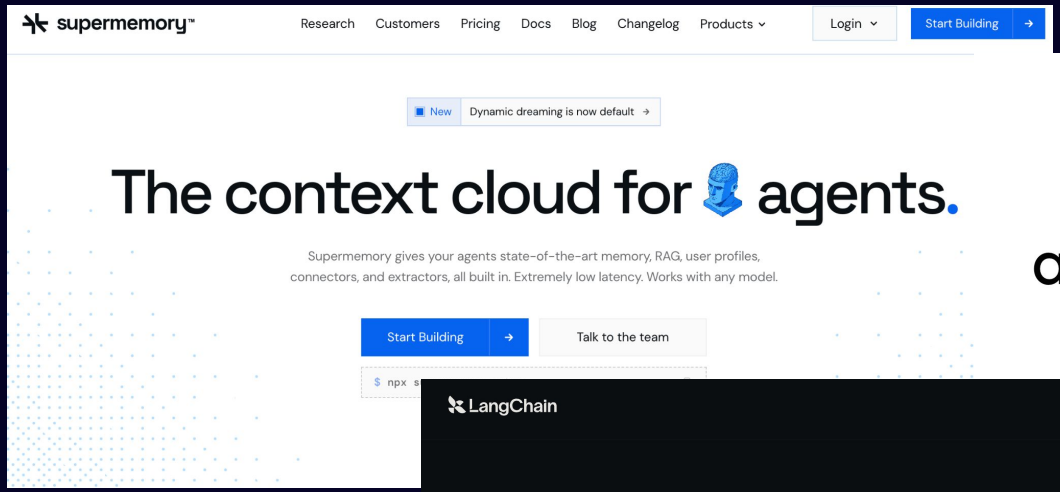
Recent advances in Large Language Model (LLM) agents have enabled complex agentic workflows where models autonomously retrieve information, call tools, and reason over large corpora to complete tasks on behalf of users. Despite the growing adoption of retrieval-augmented generation (RAG) in agentic systems, existing literature lacks a systematic comparison of how retrieval strategy choice interacts with agent architecture and tool-calling paradigm. Important practical dimensions, including how tool outputs are presented to the model and how performance changes when searches must cope with more irrelevant surrounding text, remain under-explored in agent loops. This paper reports an empirical study organized into two experiments. Experiment 1 compares grep and vector retrieval on a 116-question sample from LongMemEval,

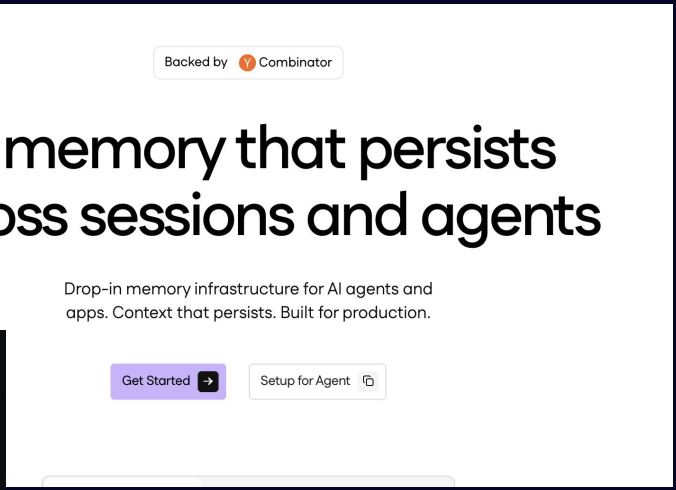
with agent architecture and tool-calling paradigm in end-to-end agentic workflows remains poorly understood. Despite growing adoption of agentic search [1, 7, 24], existing research evaluates retrieval strategies largely in isolation from agent architecture. The information retrieval community has extensively benchmarked lexical and dense retrieval methods [4, 13, 23], and has studied retrieval quality, chunking, and reranking in standalone pipelines [5, 26]. Yet, these evaluations typically assume a fixed pipeline where retrieved documents are concatenated into a prompt, ignoring the iterative, tool-mediated retrieval loops that characterize modern agentic systems [17, 19]. In practice, agents receive ranked lists but do not treat them as terminal: they decide what to search, how many queries to issue, and whether the retrieved results are sufficient or require further refinement, all mediated by the agent harness and

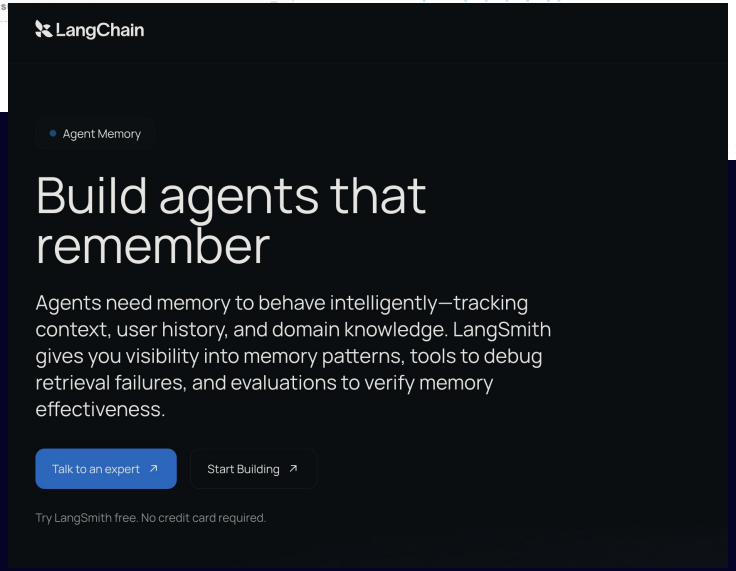
1) Keep it simple stupid



What you think you need:


supermemory™
Research
Customers
Pricing
Docs
Blog
Changelog
Products
Login
Start Building
New
Dynamic dreaming is now default
The context cloud for agents.
Supermemory gives your agents state-of-the-art memory, RAG, user profiles, connectors, and extractors, all built in. Extremely low latency. Works with any model.
Start Building
Talk to the team


Backed by Combinator
AI memory that persists across sessions and agents
Drop-in memory infrastructure for AI agents and apps. Context that persists. Built for production.
Get Started
Setup for Agent


LangChain
Agent Memory
Build agents that remember
Agents need memory to behave intelligently—tracking context, user history, and domain knowledge. LangSmith gives you visibility into memory patterns, tools to debug retrieval failures, and evaluations to verify memory effectiveness.
Talk to an expert
Start Building
Try LangSmith free. No credit card required.

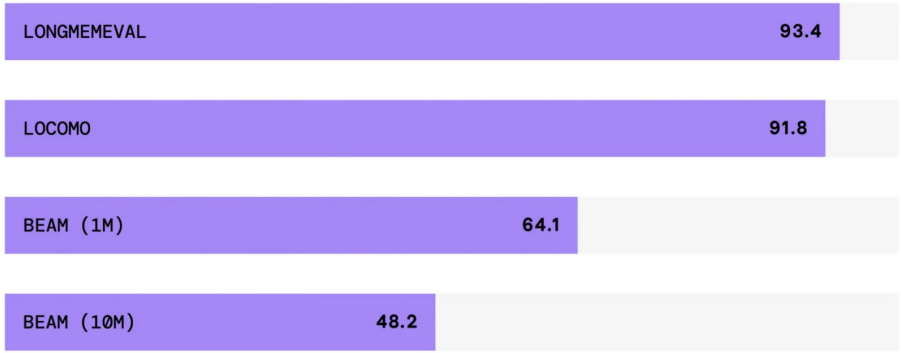


New Algorithm

Benchmarking Mem0

Single-pass hierarchical distillation. Multi-signal retrieval.
Benchmarked across LoCoMo, LongMemEval, and BEAM.

View Research





Persistent Memory Of Your Life

Your values
Your preferences
Your expertise
People
Projects
Places
…

How does my agent do across success criteria I’ve set.

2) Always Start From The User Experience And Work Backwards

What do you actually need!



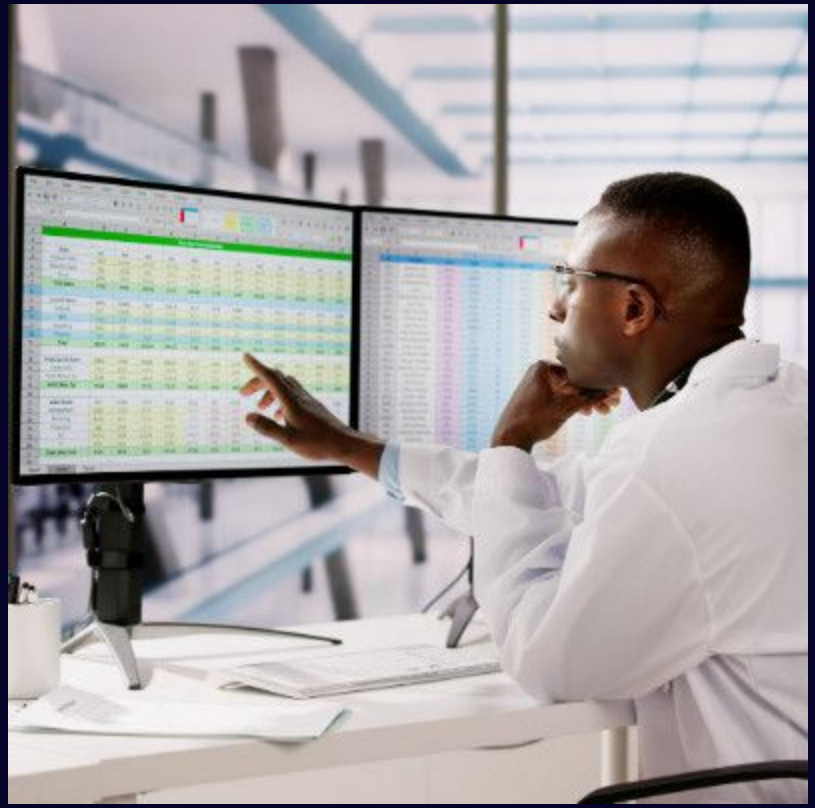
“If you are going to make a lot of state changes, having them all happen inline does have advantages; you should be made constantly aware of the full horror of what you are doing.” - John Carmack





3) Make sure you see the full state of the context window.

What do you actually need!




YOUR
SCORE
HIGH
SCORE

You think you solved the problem but you didn’t

4) Run evals first

1) Keep It Simple Stupid
2) Always Start From The User Experience
3) Always look at the final state of the prompt
4) Run Evals First