

# Building AI Agents

Ben Clegg-Li, PhD

# Why AI?

- Significant impact on industry
- Paradigm shift in capabilities & user interaction
- Formerly “impossible” problems can now be solved...
- ...at the cost of new challenges

# Why me?

- Background in software engineering
- Built *Remy AI* w/ friends
  - Front-of-house agent for restaurants
  - Multi-channel (phone, SMS, etc.)

# What is an agent?

## Agents

- Able to:
  - Plan
  - Take actions
  - Reflect
- Autonomous & flexible...
- Challenge to regulate behaviour

## Workflows

- Like flowcharts
- Useful for well defined processes
- Easier to build & test
- Less flexible & adaptable
- Can combine with agents

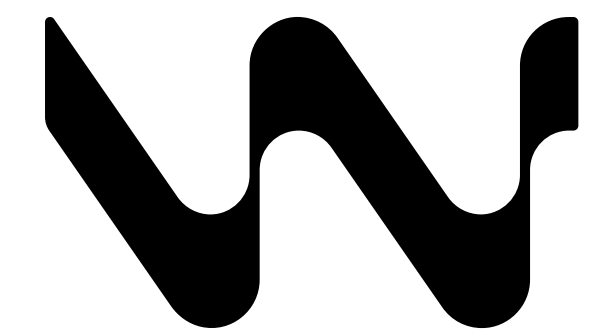
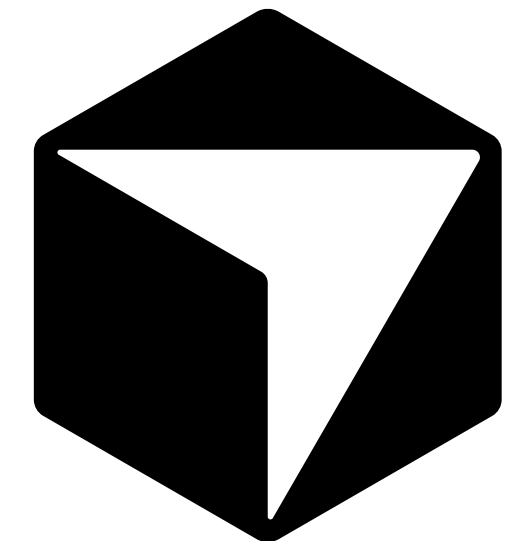
# Deciding on capabilities

- Prioritise common user pain points & tedious tasks
- Focus on a single “job”
- Identify actions required to achieve it

# Deciding on capabilities

*Example: coding agent*

- Understand requirements
- Read files
- Plan changes
- Edit files
- Run commands (e.g. build, test)
- Reflect on results — repeat if necessary



# The basics: LLMs

- Trained on lots of data
- Post-training for additional capabilities
- Generate single output tokens (words)
  - Based on probability
  - Repeated process
- Tradeoff between quality, cost, and speed
- Long, unfocused context degrades quality
- Non-deterministic

# Channels

- Web-chat & single prompt inputs
- SMS
- Voice (phone calls) — difficult!
- Existing messaging platforms





# Choosing a model

- Tradeoffs
- Benchmarks
- Experiment



# Context: Prompting

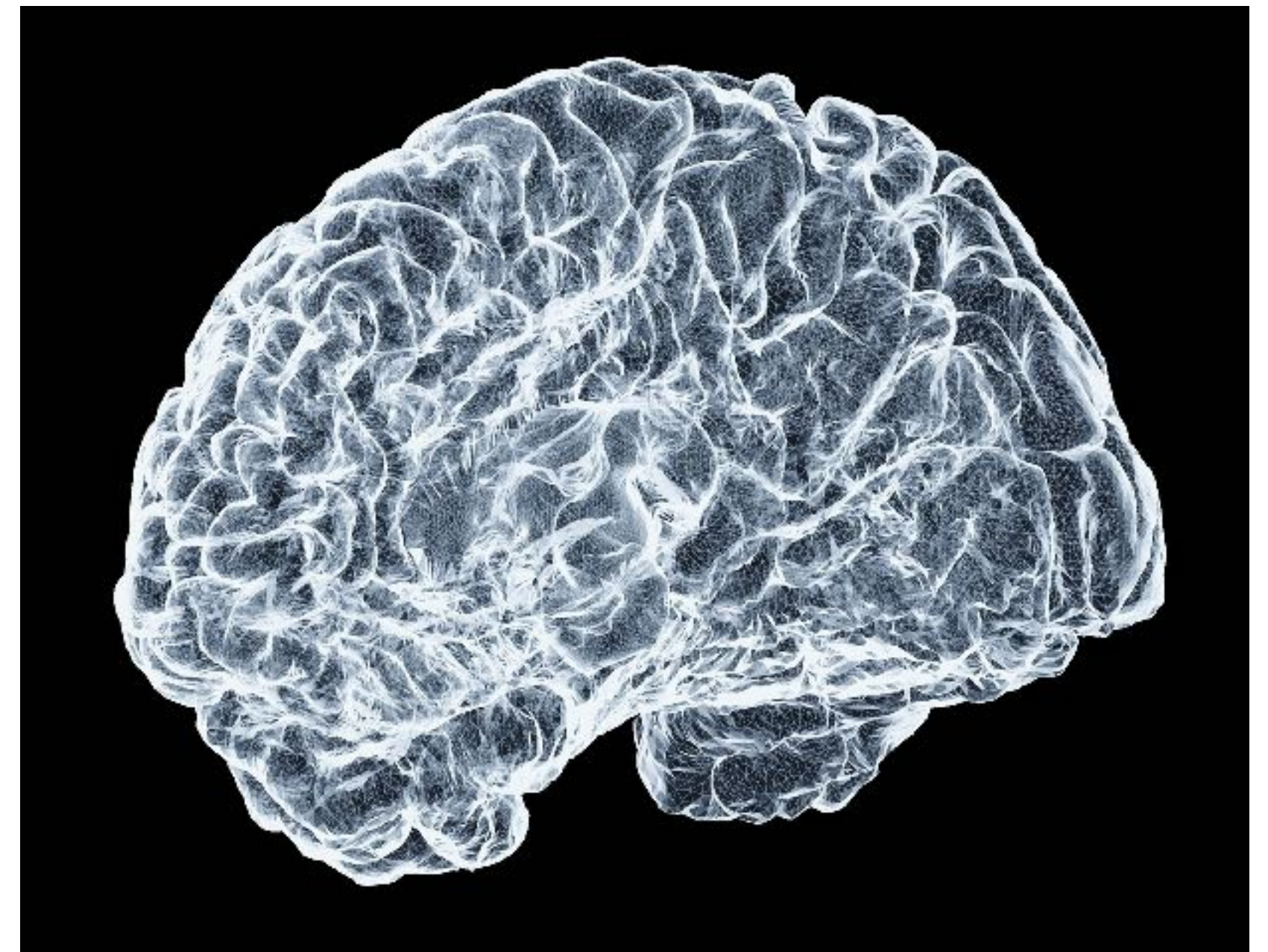
- Simple approach: string templates
- Keep it focused; avoid distractions & contradictions
- Shorter is better

The current time is  
`{current_time}`.

You are talking to  
`{user_name}`.

# Context: Memory

- Checkpointing
  - Store user messages within conversation
- Summarization
  - Retrieve summaries of past conversations
- Retrieval augmented generation (RAG)
  - Long term memory
  - Vector database & embedding model, or graph db



# Actions: Tool calling







- How can an LLM make an action?
- Tools = functions in code
  - Tell the LLM what tools are available
  - LLM will send a message like:

```
[TOOL_CALLS] [{ 'name': 'get_weather', 'arguments': { 'location': 'Stoke' } }]
```

- Execute the matching function
- Idiot-proofing

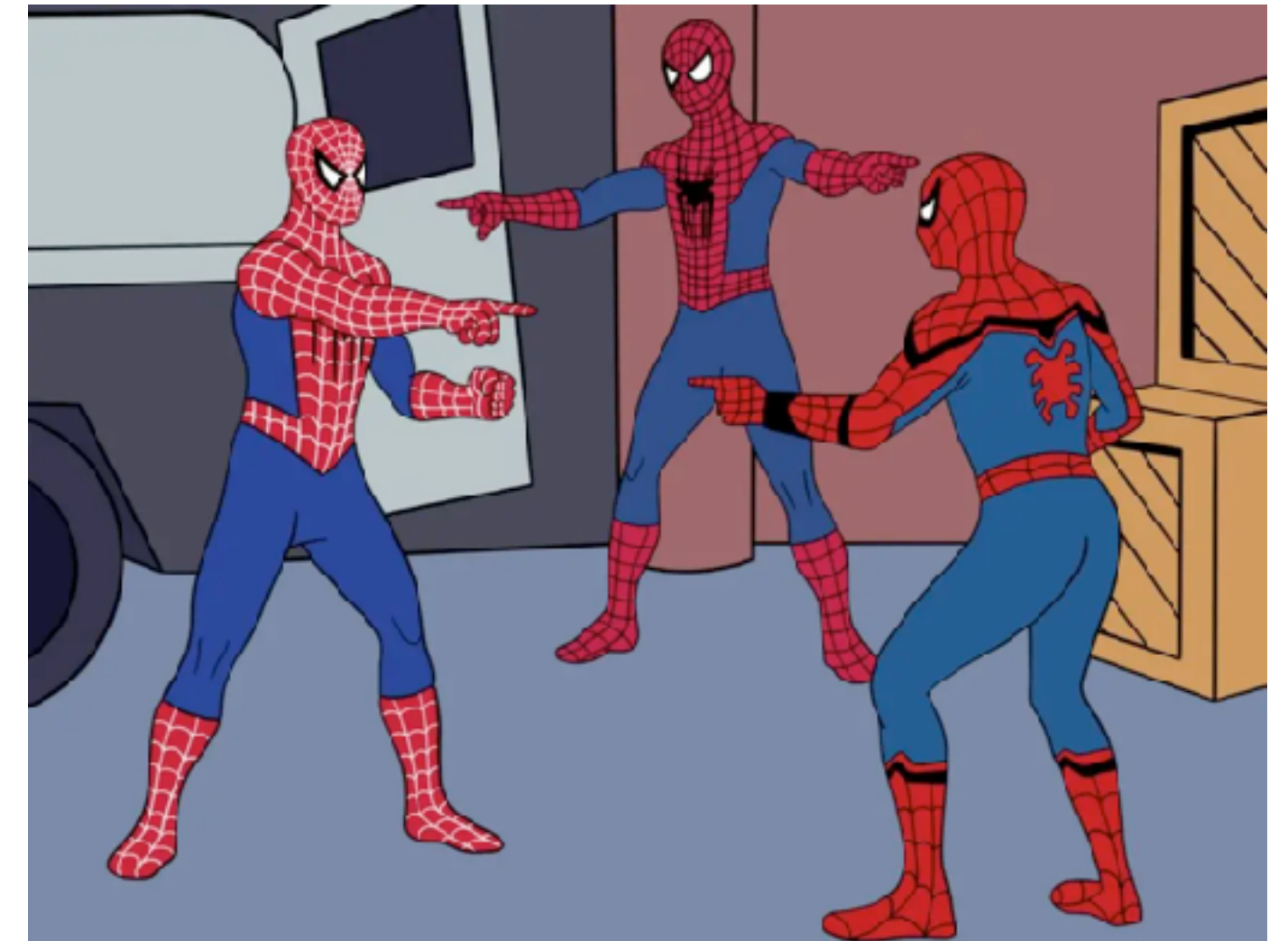
# Behavior: Reason and act

Make a loop: think → act (call tool, executed in app) → result → **repeat**

- User:  “If it’s raining in Stoke tomorrow, message Bob to bring an umbrella.”
- LLM:  ( I need to find the weather in Stoke tomorrow. )
- LLM:  *Action* — Get weather in Stoke on 21st November → *Result: Rain*
- LLM:  ( It’s raining, so I need to message Bob to bring an umbrella. )
- LLM:  *Action* — Send message to Bob (“Bring an umbrella”) → *Result: Sent*
- LLM:  “It’s going to rain in Stoke tomorrow, so I’ve messaged Bob.”

# QA: Evals

- Non-deterministic; how to test?
  - Use an LLM to evaluate outputs!
- How to generate test inputs?
  - Use another LLM to act as a user!
- End-to-end flow
- Use repetitions



# Cost-tracking & FinOps

- AI is expensive!
- Priced by input & output tokens consumed
- Can track by request or by time period
- Use evals to understand price:performance



# Resources to build your agent



# No-code platforms



- Great for prototyping & non-engineers
- Prefer open-source platforms to limit platform risk
- Can build workflows & simple agents

# Production-ready tech stack

Model Provider



G AI



Model Router / Proxy



LiteLLM



OpenRouter

AI Library / Framework



LangChain & LangGraph



Embabel



MS Agent Framework



Evals



OpenEvals & AgentEvals

...on top of a traditional stack!