Using LLMs in Data Science

Scott A. Handley, PhD

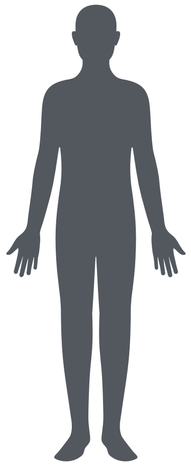
Professor , Department of Pathology and Immunology
Washington University School of Medicine
Member, The Edison Family Center of Genome Sciences & Systems Biology

Director of Agentic AI for Therapeutic Discovery
The McDonnell Genome Institute (MGI)

1-June, 2026 | Data Science with R and Agentic Coding | Durban, South Africa

What is our Job?

Defining identity in the world of LLMs

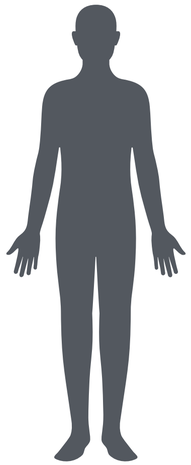


you

Read Papers

= AI/LLM

January 2026 European Workshop



you

Workshops

Outreach

Data Analysis

Teach

Form Hypothesis

Read Papers

Emails

Study Design

Meetings

Writing

Collect Samples

Applications

Wet lab work

Coding

Papers

Troubeshoot

Coffee

Stats

Snacks

= AI/LLM

Goal of this presentation

- Demystification of what an LLM actually is
- Human LLM interactions - Focus on the terminal
- Using LLMs in bioinformatics
- Using LLMs to **G**et **T**hings **D**one
  - To assist in discovery not to discover itself
- Also, I have no idea what I am doing ...
  - Most of this is all new (really activity started in 2025)
    - Rapidly changing and evolving
- I am an AI/LLM optimist, but have peaks and valleys of skepticism

Talk Structure

- **Basics of LLMs text based models**
  - Not going to discuss image generation which are primarily diffusion models
    - We are going to focus on text prediction and generation
- **How to interact with LLMs**
  - Chatbots, desktop and terminal apps
- **Using LLMs for bioinformatics**
  - This will be difficult to time and get right!
    - Prompt speed, analysis speed, my speed

Can we control and fix this?

How do LLMs work

Stage 1: Pre-training

The foundation: Learning language from the internet

Goal: Predict the next word, billions of times

- **INPUTS:**
- ~10 trillion tokens of text
  - Books, websites, code, paper
  - Wikipedia, stack overflow, etc.

- **OUTPUTS:**
- Model that “understands” language
- Encodes all the worlds knowledge
  - At that point in time
    - Requires retraining
- Cost: ~$100M+ for frontier models

Additional Environmental Costs

Just for pertaining, not maintenance

- **ENERGY:** A single pertaining can consume 10-50+ GWh of electricity
  - Equivalent to powering 1,000 - 5,000 homes for a year
  - The energy released by burning roughly 1–5 million kilograms of coal
- **CARBON:** Pretraining can emit hundreds of thousands of metric tons of CO_2
  - Training in coal-powered regions can produce 10-20X more emissions than training than training with renewables
- **WATER:** Data center cooling may use 3-5 million liters of water per pre-training run
- **HARDWARE:** Manufacturing of specialized chips requires rare earth mining, energy intensive fabrication and generates e-waste as hardware is rapidly replaced

Job Costs

Projected Labor Market Costs

- **ENTRY LEVEL IMPACT:**
  - Entry-level employment in software engineering and customer service declined ~20% between late 2022 and July 2025 (Stanford/ADP study)
  - 66% of global enterprises plan to cut entry-level hiring due to AI adoption (IDC/Deel 2025 survey)
  - U.S. programmer employment dropped 27.5% between 2023 and 2025
- **BROADER PROJECTION:**
  - Goldman Sachs estimates 6–7% of U.S. jobs (roughly 10 million) could be displaced
- **COUNTERPOINT:**
  - World Economic Forum projects 170 million new jobs created vs. 92 million displaced by 2030 (net +78 million)
  - AI specialist roles (ML engineers, AI researchers) continue growing even as general software hiring declines

What does this mean for the academic research?

Now back to how LLMs work

Step 1: Tokenization

Before training, text must be converted to numbers

- Why common & rare? If every word needed its own token, the vocabulary would be enormous
- Spaces typically count as their own tokens
- Numbers are sometimes broken down individually or can be treated like rare
- Capitalization matters (Hello and hello are different)
- Mostly trained on English, but there are efforts for multi-lingual training models and cross-lingual transfer
- Spelling errors inflate tokens

https://platform.openai.com/tokenizer

Step 2: Embeddings

Tokens are just numbers. They do not capture meaning

- **SOLUTION:** Score each word on qualities

"Cat": furry=0.9, pet=0.8
"Dog": furry=0.9, pet=0.9
"Pizza": furry=0.0, pet=0.0

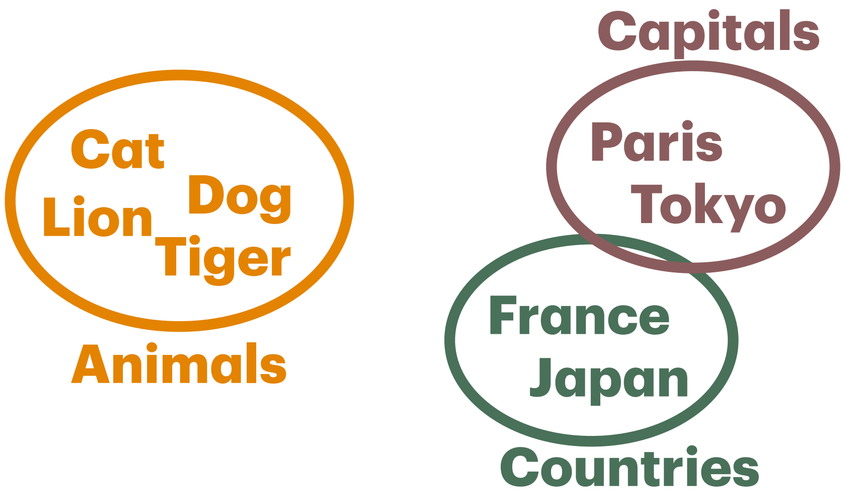
Joburg & Durban: Similar coords -> nearby
Cat & dog: similar scores -> similar meaning

INSIGHT: Words with similar meanings end up with similar vectors, they are closer together in "embedding space"

- Embedding values aren't picked by hand, they are actually back propagated starting with random word associations and optimizes based on common word patterns
- Real embeddings use hundreds of thousands of dimensions, not just two

Embeddings: Words in Space

Similar words cluster together in “meaning space”



What the model learns:

- Words used in similar contexts end up near each other
- Word arithmetic works
  - Paris - France + Japan = Tokyo
- The “capital of” relationship is a direction in space
- Real embeddings: ~4,096 dimensions (not just 2)

https://huggingface.co/spaces/hesamation/primer-llm-embedding?section=what_are_embeddings?

Step 3: Training the neural network

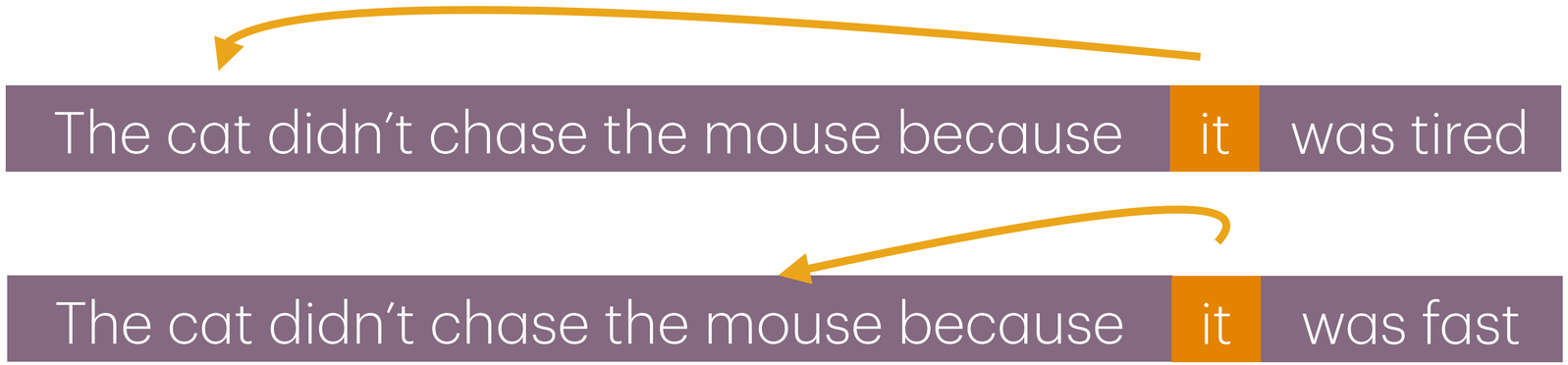
The core idea: Predict the next token

- Like studying flashcards: See the front, guess the answer, flip to check, learn from mistakes
  - **Front of card:** “The capital of France is _____”
  - **Model guesses:** “Paris”
  - **Flip the card:** Training text says “Paris” (correct!)
  - **Learn:** Adjust to be more confident next time
- Repeat billions of times across all of the text on the internet

Transformer Architecture

The breakthrough that made LLMs possible

- Key Innovation: Attention
  - Each word “looks at” all other words
  - Learns which relationships matter
  - Parallelizable (fast training)
- Scale Matters
  - GPT-3: 175 billion parameters
  - GPT-4: Rumored ~1.7 trillion
  - Llama 3: 70-400 billion
- Captures long-range dependencies. “it” can attend to a noun from 50 words ago



Before transformers: models read left-to-right, forgetting earlier words

After transformers: every word sees the full context at once

https://poloclub.github.io/transformer-explainer/

What do we have at this point?

A powerful but raw text predictor

- **It CAN:**
  - Complete sentences naturally
  - Generate coherent text
  - Answer questions (sometimes)
  - Write code, poetry, essays
- **It CAN’T (yet):**
  - Follow instructions reliably
  - Have conversations
  - Refuse harmful requests
  - Admit when it doesn’t know

The base model is like a knowledgable but untrained assistant

Step 4: Post-training

Supervised Fine Tuning (SFT)

Goal: Transform a text predictor into an assistant

- **Problem:**
- The base model just predicts "what comes next on the internet"
- If you ask a questions, it might continue with more questions!

- **Solution:**
- Show "it" hundreds of thousands of examples of ideal conversations
- Human labelers create high-quality Q&A pairs
- Teaches format, tone, and behavior. Not new knowledge

Supervised Fine Tuning (SFT)

Human labelling to craft ideal response pairs:

- Human labelers write both prompt and the ideal response
- ~100K conversation per SFT (varies wildly)
  - Compared to token - embedding - transformer this favors quality over quantity

Example prompt:
Write a haiku about programming

Assistant:
Lines of code unfold,
Logic dances with the bugs,
Coffee grows cold

Reinforcement Learning

Optimizing for human preferences

Goal: Make model responses match what humans prefer

- **Reinforcement Learning from Human Feedback (RLHF)**
  - Human rate response pairs
  - Trains reward model
  - Optimize LLM for rewards
  - No objectively correct answer -> needs human
  - “Write a friendly email” (subjective. Humans decide
- **Verifiable RL:**
  - Tasks with correct answers
  - Math code, puzzles
  - Correct answer exists
  - Automatic verification
  - What is 347 + 347 = 694

System Prompts

The Conversation Format

Special tokens structure the dialogue (system prompt)

<|system token|>
You are a helpful AI assistant.

<|user|>
What is the capital of France?

<|assistant|>
The capital of France is Paris.

<|system token|>
You are a pirate.

<|user|>
What is the capital of France?

<|assistant|>
Yar! The capital of France be Paris!

- Why this matters?
  - The model learns the “rhythm” of conversation
  - Knows when to respond and continue
  - These are “installed” into LLM interfaces like chatbots, but can be modified by a user
  - https://github.com/Piebald-AI/claude-code-system-prompts

Soul documents

System prompts at the point of training

“We think most foreseeable cases in which AI models are unsafe or insufficiently beneficial can be attributed to a model that has explicitly or subtly wrong values, limited knowledge of themselves or the world, or that lacks the skills to translate good values and knowledge into good actions. For this reason, we want Claude to have the good values, comprehensive knowledge, and wisdom necessary to behave in ways that are safe and beneficial across all circumstances”

- Amanda Askell, Anthropic

- Claude 4.5 Opus Soul Document
  - https://gist.github.com/Richard-Weiss/efe157692991535403bd7e7fb20b6695#file-opus_4_5_soul_document_cleaned_up-md
- Submitted during training, not conversations
- 14,000 tokens
  - Be helpful
  - Be honest
  - Avoid harm

The Hallucination Problem

LLMs confidently state the things that aren't true

Why? The model is optimizing for "sounds right" not "is right"

- **Examples:**
  - Made-up citations
  - Fake historical events
  - Invented statistics
  - Non-existent people
- **Root causes:**
  - Trained on the internet (includes lies)
  - Optimized to be confident
  - No "I don't know" in training
    - Newer models are better
  - Pattern matching, not reasoning

Mitigating Hallucinations

Give the model access to real information instead of memories

- **Training**
  - Teach to say “I don’t know”
  - Penalize false confidence
  - Include uncertainty in examples
- **Tools**
  - Web search for facts
  - Code execution
  - Database lookups
- **Retrieval (RAG)**
  - Fetch relevant docs
  - Include in context
  - **https://context7.com/**

Tool Use: Extending LLM Capabilities

LLMs can use external tools

- **Web Search**
  - Get current information

[SEARCH]What is the weather in Tokyo?**[/SEARCH]**

- **Code Interperter**
  - Run python, do math

[TOOL]Blast this sequence against SwissProt @seq.fasta**[/TOOL]**

- **APIs**
  - Send emails, query DBs

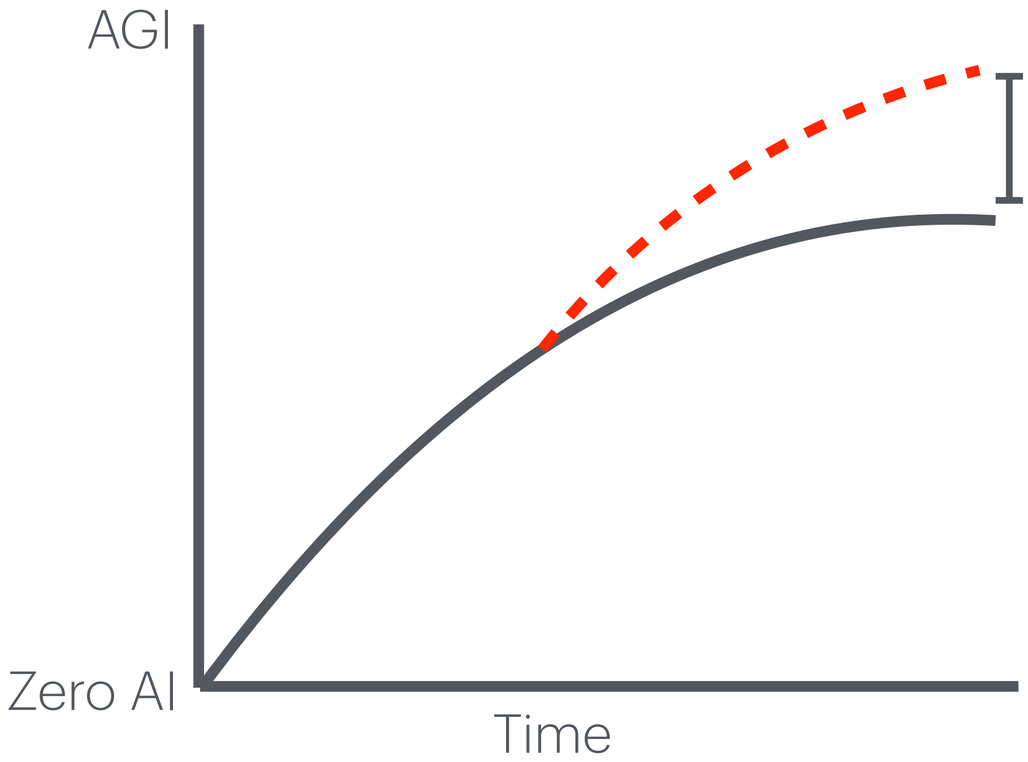
LLM Capabilities & Limitations

- **LLMs are good at:**
  - Pattern matching
  - Interpolating between examples
  - Following demonstrated patterns
  - Broad knowledge recall
  - ***Tool calling***
- **LLMs struggle with:**
  - Novel reasoning
  - Counting and precise math (better now)
  - Long-term planning (better now)
  - Knowing what they don’t know

- **Think of LLMs as “super powered interns” with vast knowledge, but need guidance and verification**

Approaching AGI

Artificial General Intelligence (AGI): AI that can do any intellectual task a human can do



Tool use | YOLO mode

You **O**nly **L**ive **O**nce

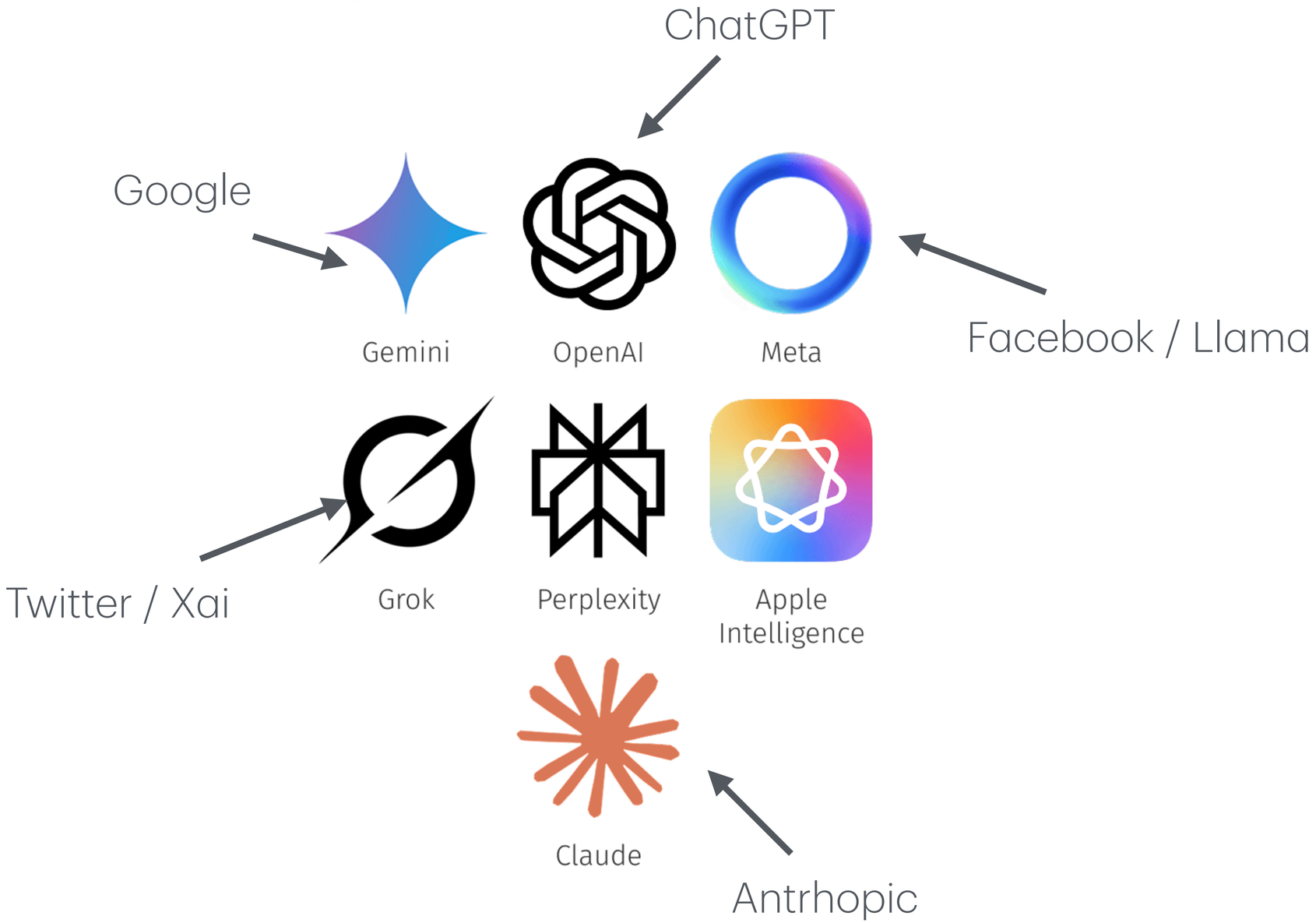
- Letting the LLM work without human intervention

- “Feels” close for certain tasks that are fully defined by text
  - Coding, summarizing, reviewing

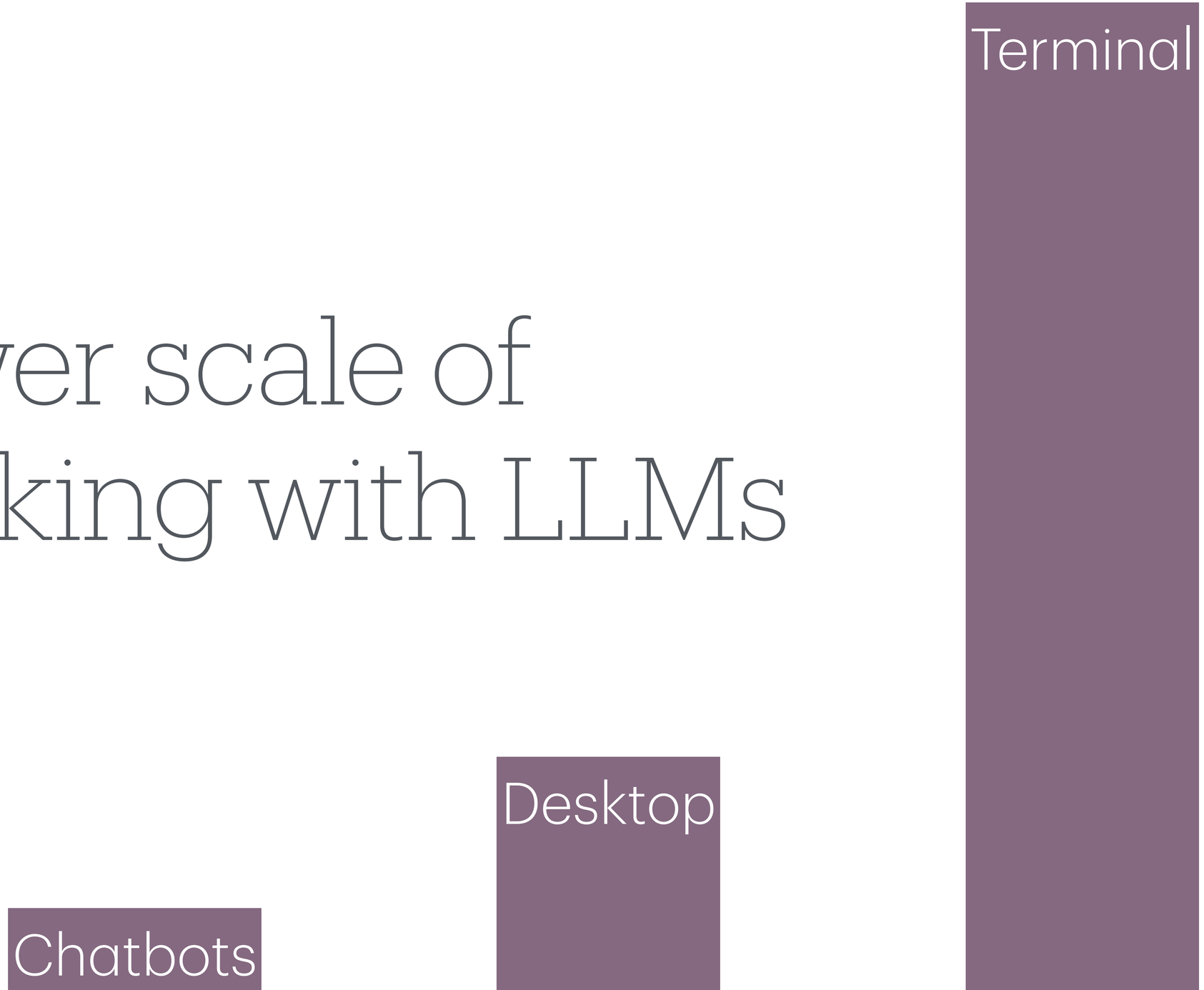
Ways to interact with LLMs

How do you interact with LLMs?

LLM Providers



Power scale of working with LLMs

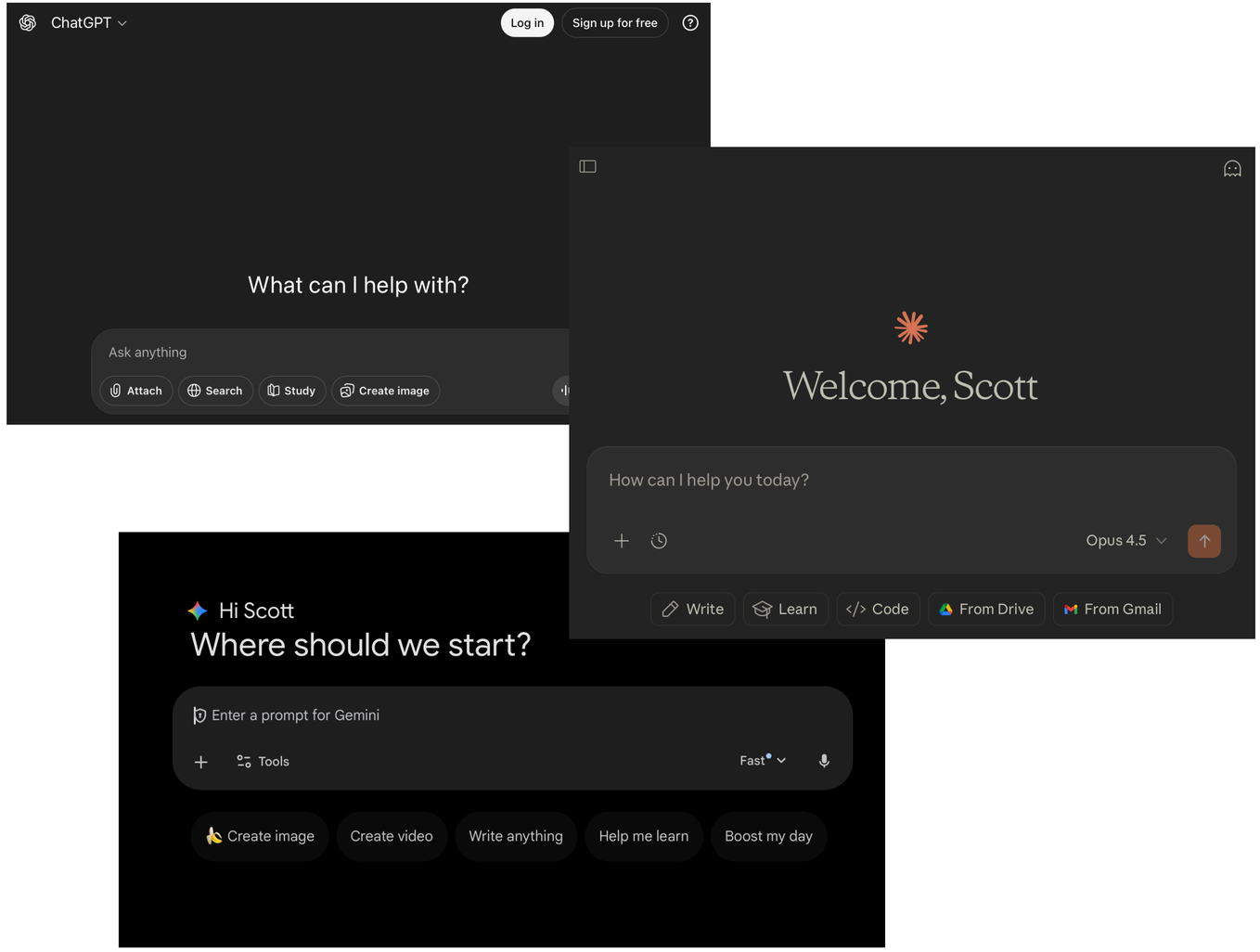


Chatbots

Chatbots

The way most people have interacted with LLMs

- **Chatbots**
  - Websites or phone apps
  - Conversational, with limited optional tools
  - May or may not have “memory”
  - May have image generation
  - Manual copy-paste to get outputs into other apps
  - Good for conversations, limited for more complex workflows

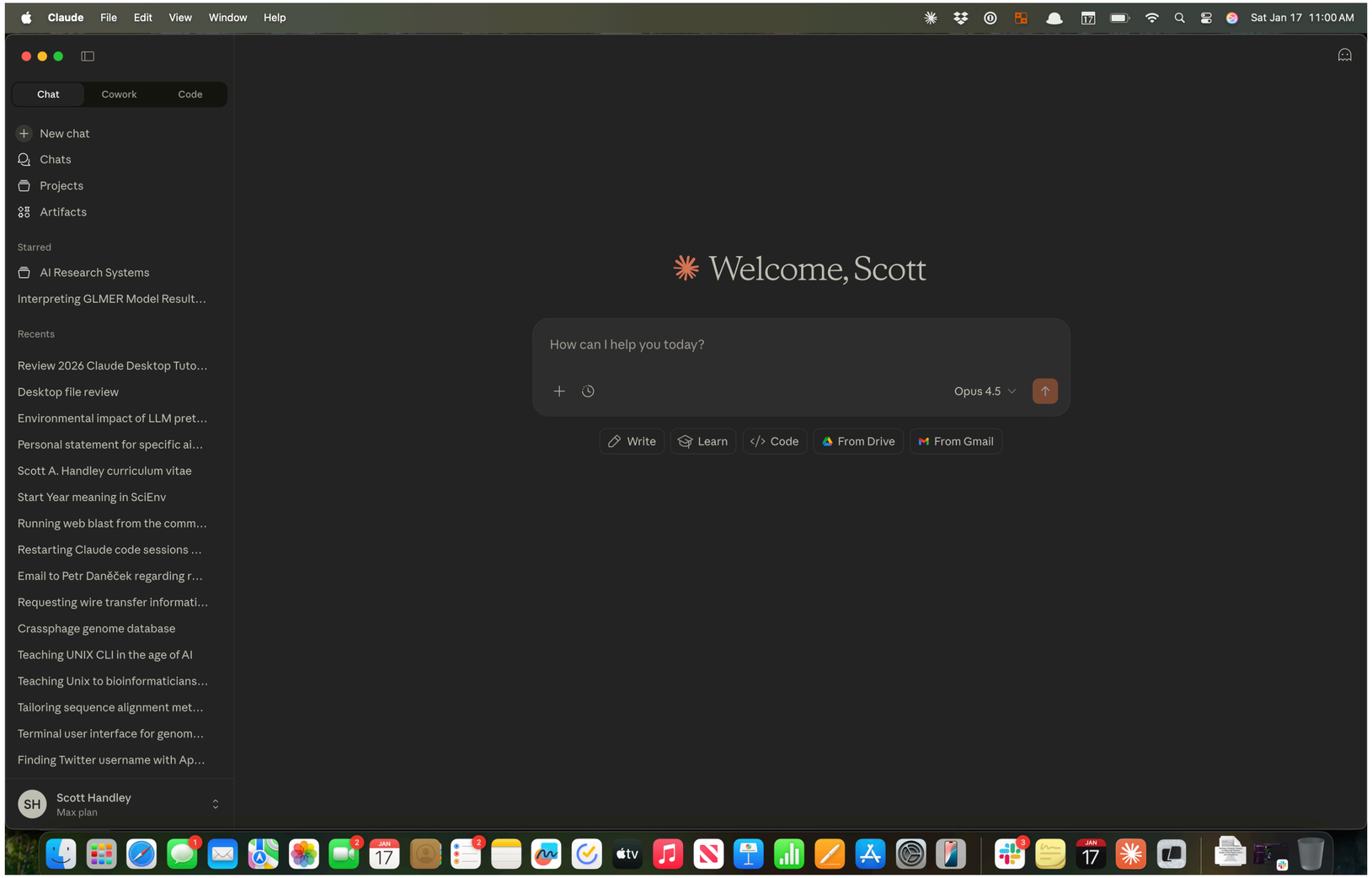


Chatbots

Desktop Apps

Desktop Apps Provide LLM Access to Local Machine

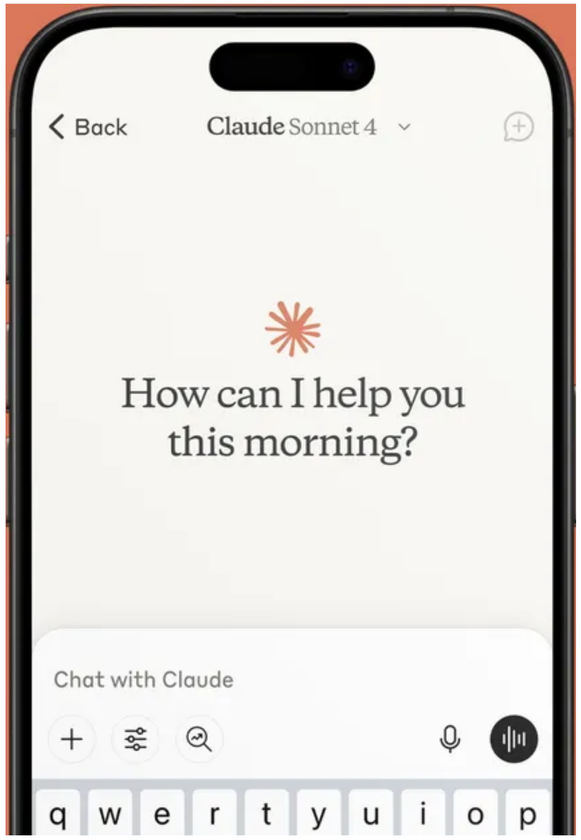
- Enables LLM access to read, write and edit local files
- Tools allow access to external sources
  - Gmail, Chrome
- Nice features for keeping track of chats and projects

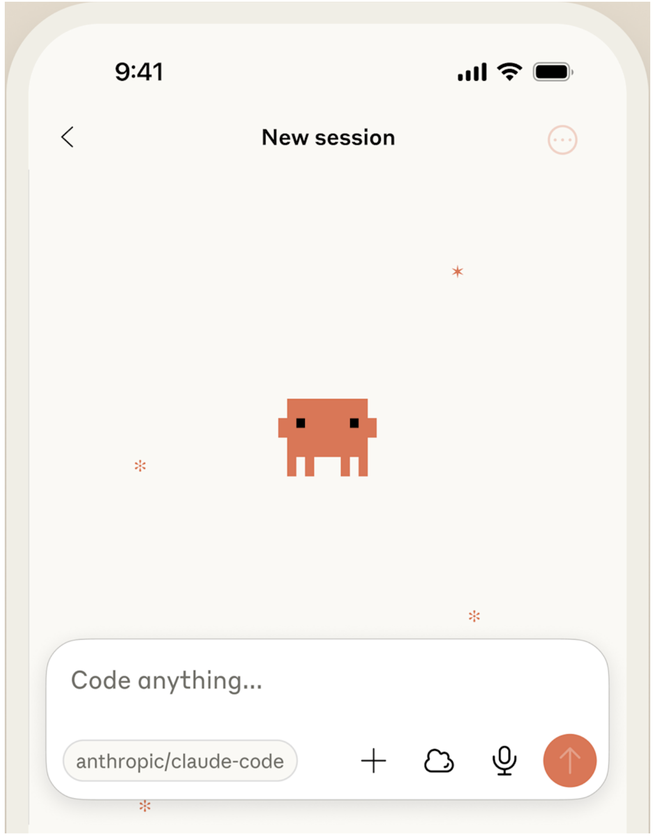


Claude Desktop Demo

1. Basic chat
2. Desktop files
3. Dashboards

Phone Apps





“Doom coding”

Hacker News new | past | comments | ask | show | jobs | submit

▲ Stop Doom Scrolling, Start Doom Coding: Build via the terminal from your phone (github.com/rberg27)

576 points by rbergamini27 10 days ago | hide | past | favorite | 404 comments

▲ magospietato 10 days ago | next [–]

I'm using the Android terminal and Claude Code to vibecode on the go. Or rather, as a fairly boring father of two, when I'm tied up in the necessary chores of family life - cooking and cleaning. Nothing as complicated as this - just Claude Code and a fairly standard Linux dev term, but it's remarkable.

Over the recent break, across four or five sessions, I wrote a set of prompts around ~500 words in total.

The result was Claude scanning my network for active ports using nmap, fuzzing those ports with cURL, documenting its findings, self-directing web searches for API/SDK docs for my Hue bridge and ancient Samsung TV, then building a set of scripts to control my lighting system and a fully functional HTML+JS remote for my TV.

The most entertaining part was Claude prompting _me_ to pop into the living room and press the button on the Hue bridge so it could fetch an API key.

The most valuable part? The understanding I gained secondary to generative act. I now understand the button on a Hue bridge literally just tells the device to issue a new API key at the next request. I understand how Entertainment mode works, and why. I understand how Samsung SmartThings is mediated via websockets - and just how insecure decade old Samsung TVs are.

Around 500 words to gain all this? I hate to buy into the hype, but it feels inflectional.

https://news.ycombinator.com/item?id=46517458

Terminal Apps

Break?

Agentic Coding

LLMs that can read, write and run code autonomously

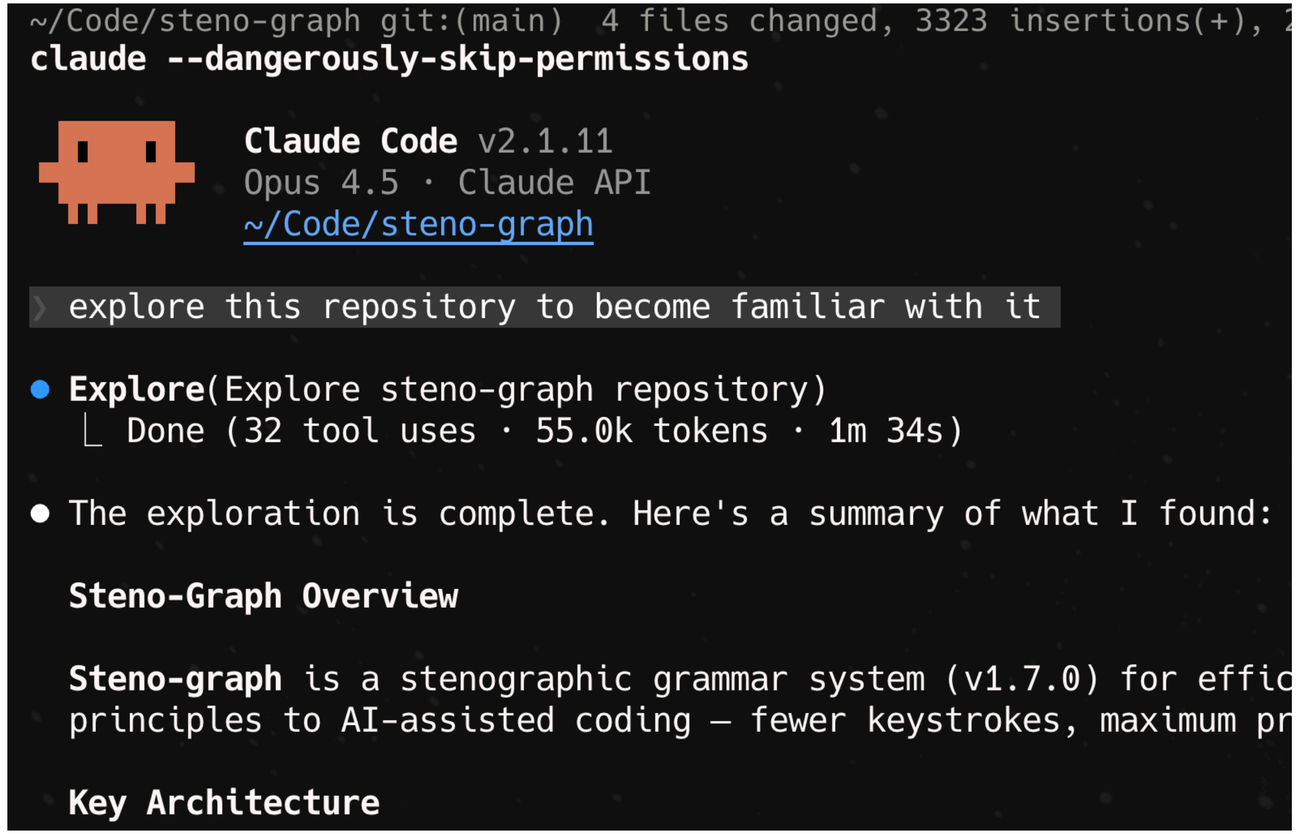
- Runs in the terminal
- Full access to your project files and folders
- Iterates on errors automatically (write -> run -> fix -> repeat)
  - You approve and reject or run in YOLO mode
- It's not just chat. It actually does things on your computer
- Example: Asking an AI assistant to debug your snakelike pipeline and it actually opens the files, finds the errors and fixes it

Agent: An AI that can take actions, observe results, and decide what to do next on its own

How AI agent coding works

The mechanics of AI-Human interaction

- Tool Calling
  - Agent decided what tools to use (read, run, search)
  - Agent outputs structured requests, gets results, continues reasoning
- The Loop
  - Think -> Act -> Observer -> Repeat
- Human in the Loop
  - You approve risky actions (file edits, running commands)
  - Can intervene, redirect, or stop at any point



Extending Agent Capabilities

Emerging AI harnessing ecosystem

- **Model Context Protocol (MCP)**
  - Standard way to connect LLMs to external tools and data sources
  - They are like “plugins” and add capabilities without retraining
  - Can connect to resources like Slack, Gmail, AWS or databases like UniProt
- **Hooks**
  - Custom scripts that run before/after agent actions
  - Example: Auto-format code after every edit. Auto commit, etc.
- **Skills**
  - Teach the agent project-specific knowledge

https://platform.claude.com/docs/en/home

Paying for AI

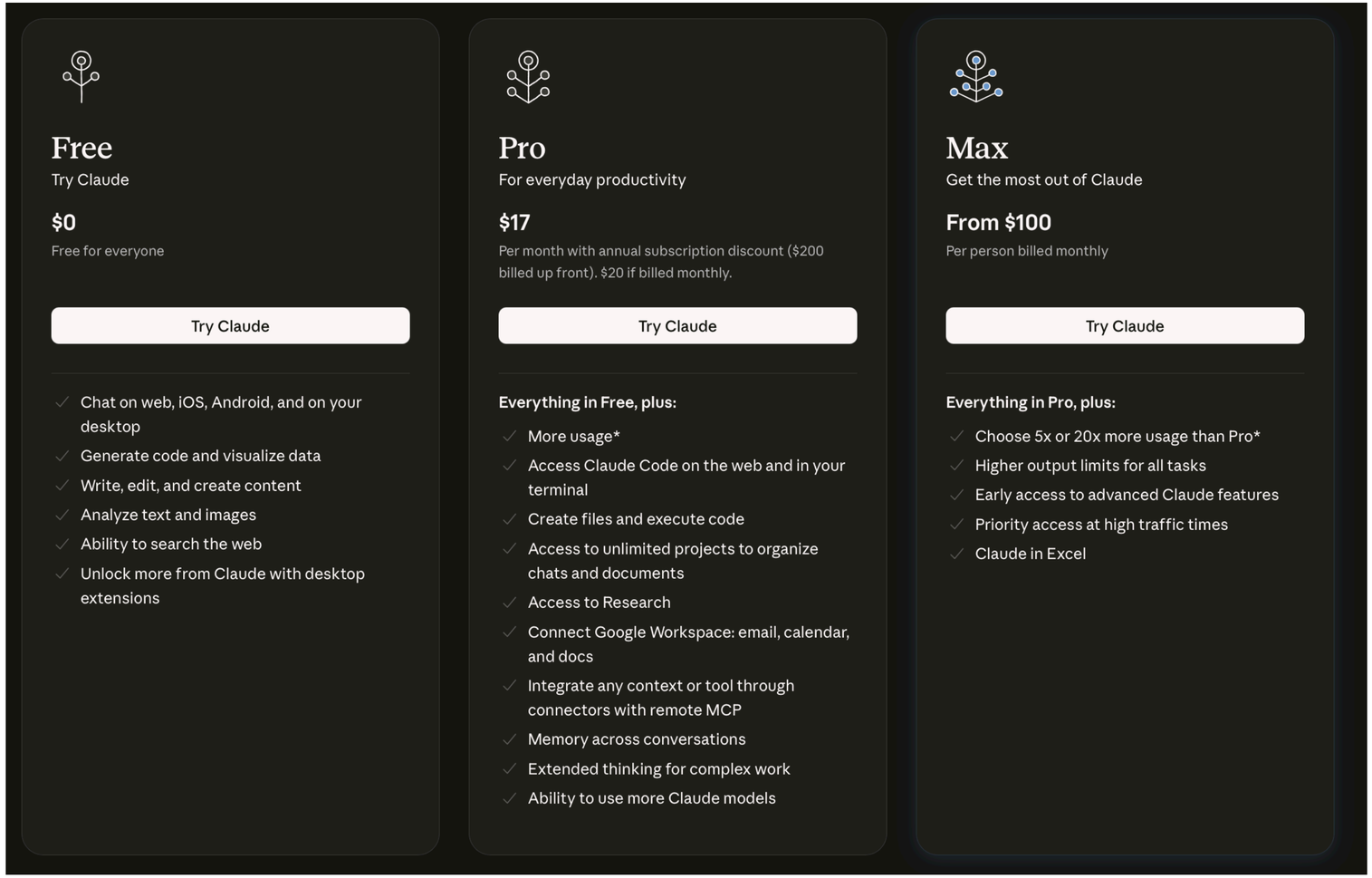
Per token

- Application Programming Interface (API) is a connection between computers or between computer programs
    - Pay per token input/ouput: https://yourgpt.ai/tools/openai-and-other-llm-api-pricing-calculator

Paying for LLMs

Plans

- Constantly changing
- Major caveats
  - Claude code only included in Pro and Max plan
- Can be challenging to get reimbursed by departments/ institutions



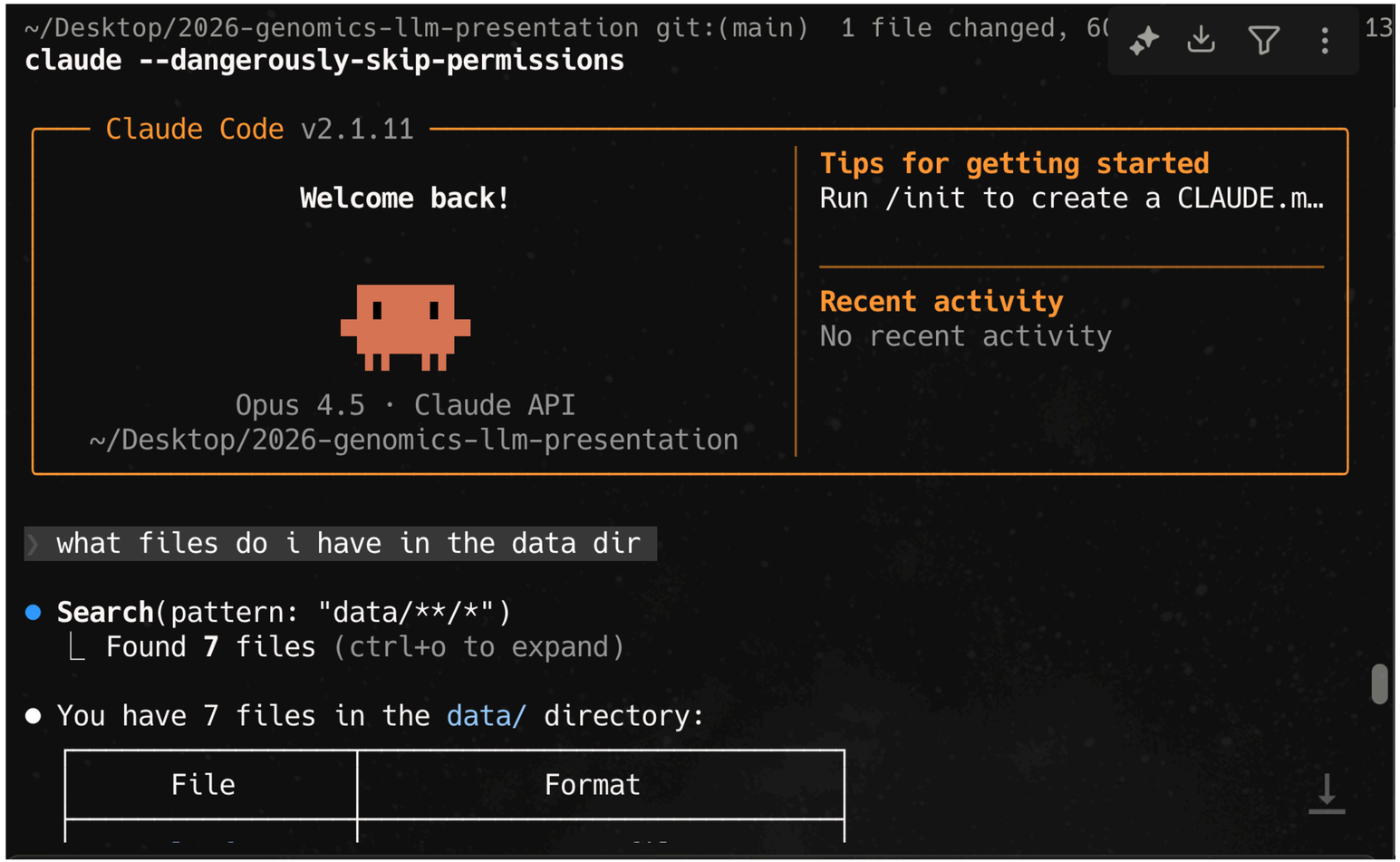
Example 1: Simple LLM usage in the terminal

1. Evaluate the 2026 Workshop Schedule (https://elsherbini.github.io/data-science-for-biology/)
2. Write a document
3. Create a new GitHub repo
4. Analyze some files in /data
    1. Tools
    2. Web
    3. Nextflow
5. Image ingestion (https://www.cell.com/immunity/fulltext/S1074-7613(16)30519-2#fig1

Agentic Research

Using AI agents and LLM knowledge to guide and complete analysis

- Using AI agents and LLM knowledge to process data
  - Structured:
    - **Inputs**
    - **Commands**
      - Deterministic vs. Nondeterministic?
        - Ex: sequence test
    - **Outputs**
    - **Interpretation** with LLM knowledge

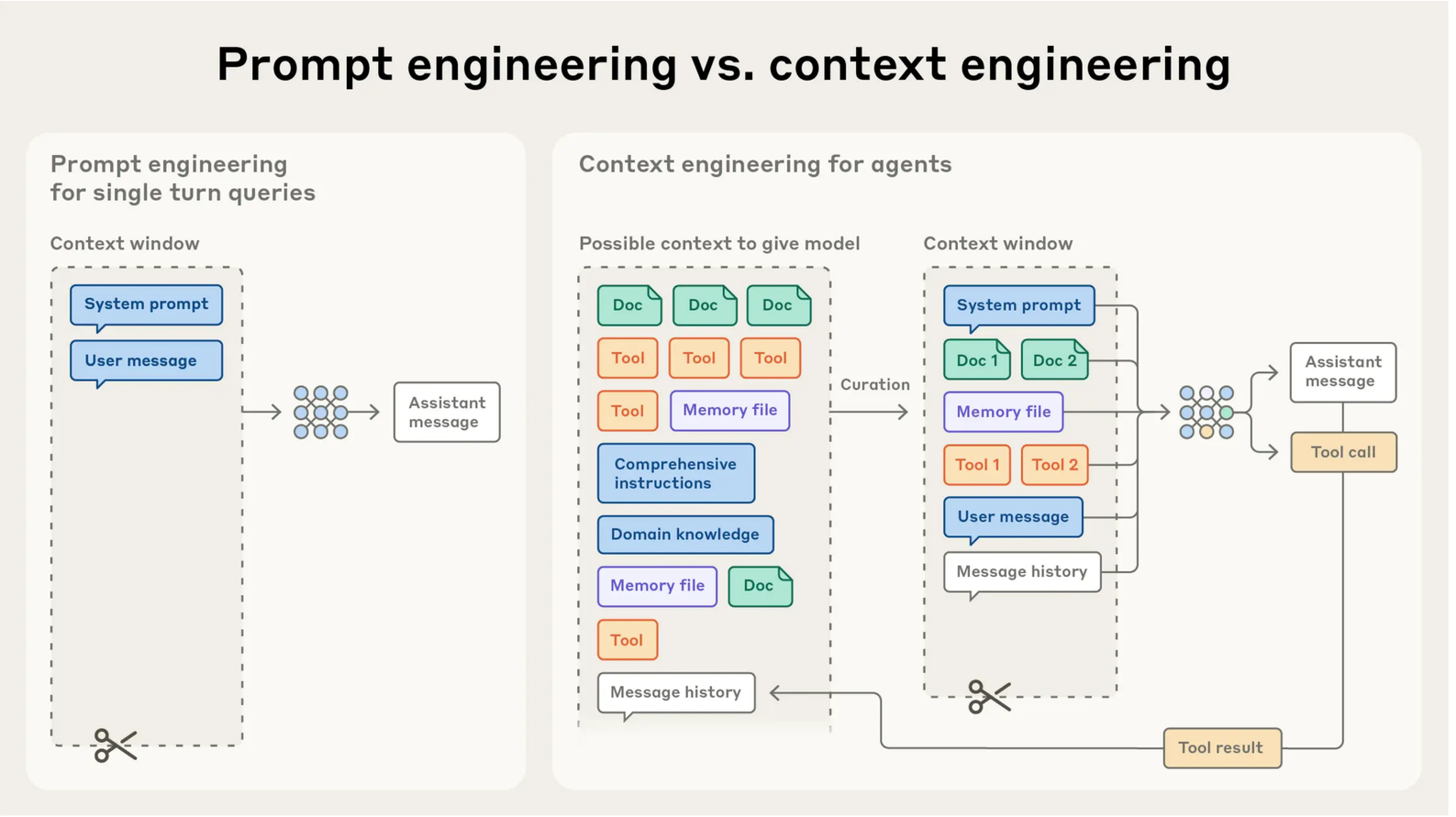


Context Engineering

Context Engineering

Context is a critical finite resource for AI agents

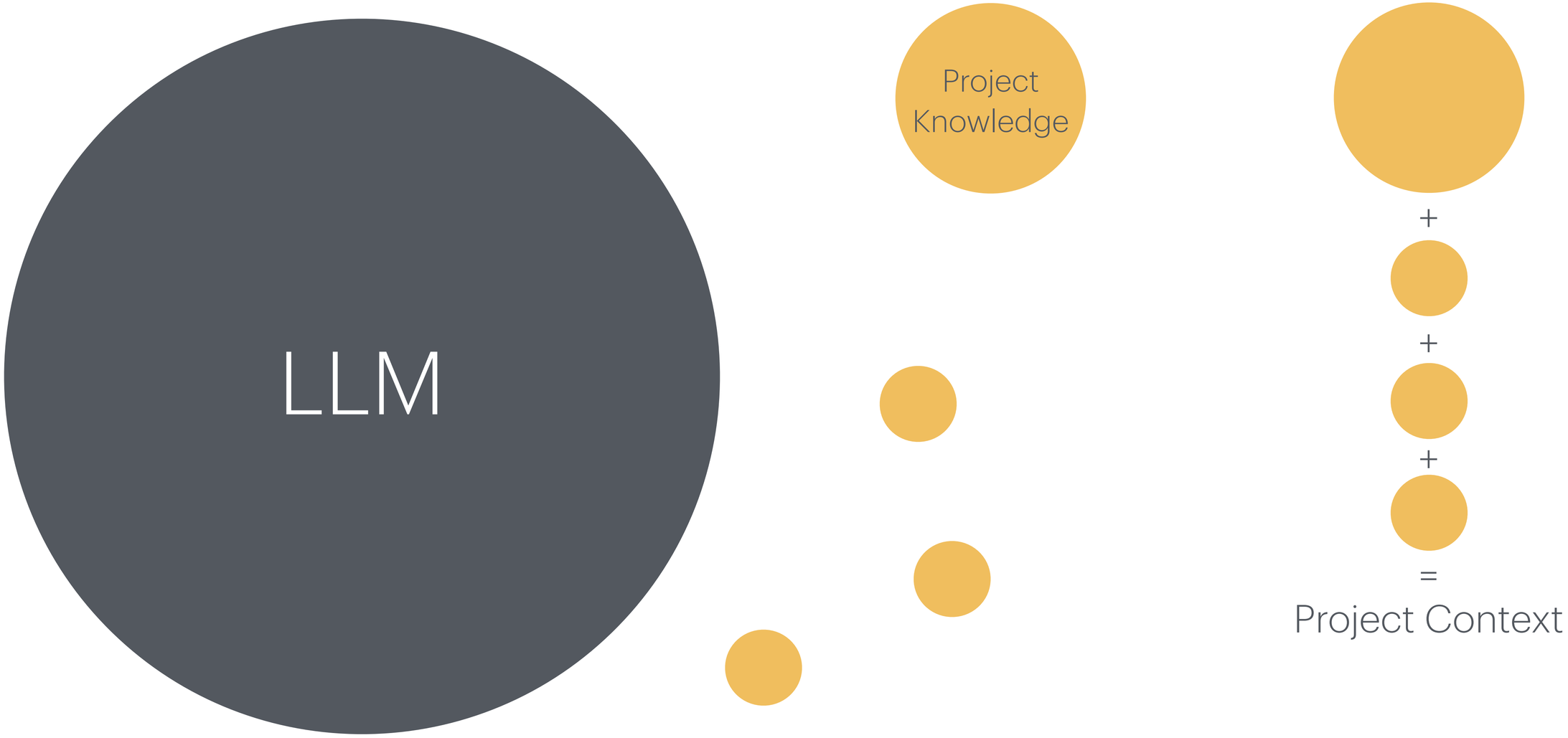
- Context refers to the set of tokens when sampling from a large language model
- Engineering challenge is to optimize the utility of context tokens at any given time to respond with the desired state
- In other words, getting the LLM to understand your context and report back effectively



https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents

LLM and Context knowledge

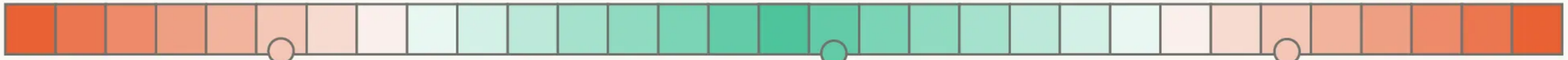
Context provides conversation specific knowledge



Why is context engineering important?

- Like humans, LLMs have limited memory capacity
- LLMs will lose focus or experience confusion at a certain point (context rot or the “dumb zone”)
- This attention scarcity comes from the transformer architecture which enables every token to see every other token across all context resulting in an n^2 pairwise relationship
  - As context length increases, the models ability to capture these pairwise relationships degrades
- There is a natural tension between context size and attention focus
- Effective prompts can help “guide” context engineering (next slide)
- ***This is the currently #1 issue when working with LLM agents!***

Calibrating the system prompt



Too specific

You are a helpful assistant for Claude's Bakery.
You must respond to the name Claude.
For every user request you MUST FOLLOW THESE STEPS:

1. Identify the user intent as one of the following: ["incident_resolution", "general_inquiry", "order_resubmission", "account_maintenance", "requires_escalation"]
2.
    - if user intent is "incident_resolution", ask 3 followup questions to gather information, then always call the resolve tool
    - if user intent is "general_inquiry", do not ask followup questions and answer in one shot
    - if user intent ...
    - ...
3. Here is an exhaustive list of cases that should be tagged as "requires_escalation":
    - If the intent is incident_resolution but the user is in a different country
    - If the user left a physical belonging in the store
    - ...
4. Once you've ruled out escalation scenarios you should consider all the tools at your disposal.
5. If the user_request contains an order_id you should tag the user intent as "order_resubmission", unless the user meets 5/7 of the following requirements:
    - User is asking for time update
    - User is asking for location update
    - ...
6. If the user wants to request a new order, but they already have another order in flight, you should follow these 5 steps of the resolution procedure:
    - (1) Call check_order tool to see where the current order is
    - ...
...

Just right

You are a customer support agent for Claude's Bakery.
You specialize in assisting customers with their orders and basic questions about the bakery. Use the tools available to you to resolve the issue efficiently and professionally.

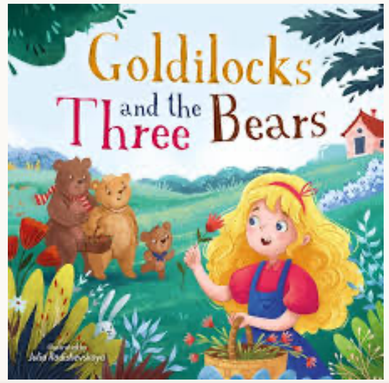
You have access to order management systems, product catalogs, and store policies. Your goal is to resolve issues quickly when possible. Start by understanding the complete situation before proposing solutions, ask follow-up questions if you do not understand.

Response Framework:
1. Identify the core issue - Look beyond surface complaints to understand what the customer actually needs
2. Gather necessary context - Use available tools to verify order details, check inventory, or review policies before responding
3. Provide clear resolution - Offer concrete next steps with realistic timelines
4. Confirm satisfaction - Ensure the customer understands the resolution and knows how to follow up if needed

Guidelines:
- When multiple solutions exist, choose the simplest one that fully addresses the issue
- If a user mentions an order, check its status before suggesting next steps
- When uncertain, call the human_assistance tool
- For legal issues, health/allergy emergencies, or situations requiring financial adjustments beyond standard policies, call the human_assistance tool
- Acknowledge frustration or urgency in the user's tone and respond with appropriate empathy

Too vague

You are a bakery assistant, you should attempt to solve customers issues in a manner consistent with the principles and essence of the company brand. Escalate to a human if needed.



https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents

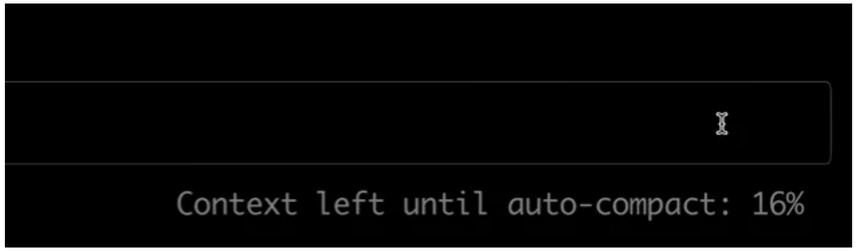
Navigating the Goldilocks Paradigm

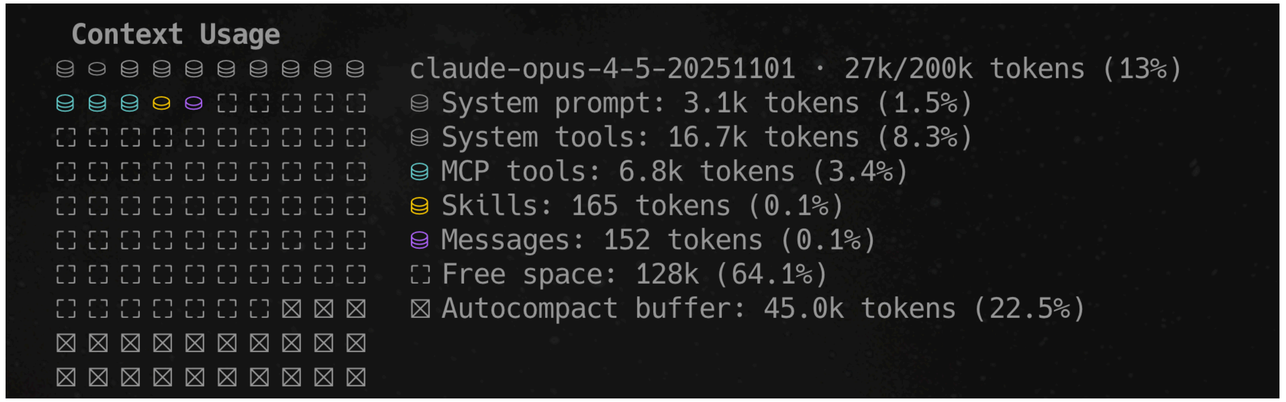
Effective rules for context optic

- Prompts should be extremely clear and use simple, direct language that presents at the right *altitude* for the agent
- Two extremes
  - Engineered hardcoded and designed to elicit exact agents behavior
  - Vague high-level guidance that does not provide concrete signals
- The optimal value lies somewhere in between

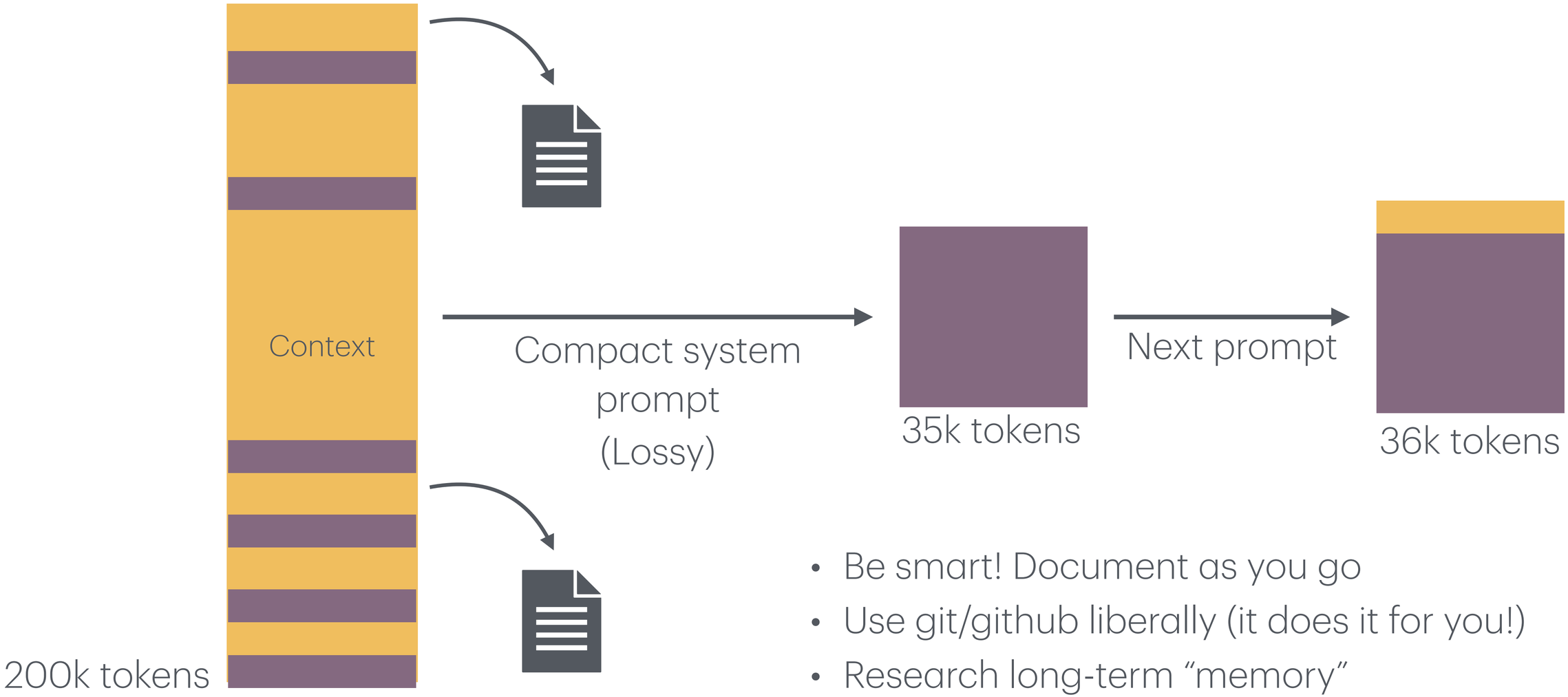
Context Compaction

- All agentic coding tools will enforce compaction at a set number of tokens
  - Claude code = 200k or 1m tokens
  - Gemini = 1m tokens
- You can manually compact (/compact)





Context Compaction



- Be smart! Document as you go
- Use git/github liberally (it does it for you!)
- Research long-term “memory”
  - Several well developed packages
  - https://github.com/steveyegge/beads

Agentic Research Summary

- Agents are just tools
  - Excel at text based tasks
  - Access to the world knowledge
  - You are the manager, the agent is an employee
    - Requires guidance, evaluation and validation
  - Agents are tools that use tools. Use tools for deterministic evaluation
- Context is queen
  - LLMs have attention spans
  - Effective context management keep your agents smart
    - Most “bad experiences” with LLMs come from ineffective context management

https://www.anthropic.com/learn

Links

- Tokenizer: https://platform.openai.com/tokenizer
- Embeddings: https://huggingface.co/spaces/hesamation/primer-llm-embedding?section=what_are_embeddings?
- Hugging Face: https://huggingface.co/
- Transformers: https://poloclub.github.io/transformer-explainer/
- Claude system prompts: https://github.com/Piebald-AI/claude-code-system-prompts
- Claude soul document: https://gist.github.com/Richard-Weiss/efe157692991535403bd7e7fb20b6695#file-opus_4_5_soul_document_cleaned_up-md
- Claude developer docs: https://platform.claude.com/docs/en/home
- LLM friendly context documentation: https://context7.com/
- Pricer per token per model: https://yourgpt.ai/tools/openai-and-other-llm-api-pricing-calculator
- Context engineering: https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents
- Ahthropic Academy: https://www.anthropic.com/learn
- Beads (memory layer): https://github.com/steveyegge/beads

Questions?