More Context Engineering and Data Strategy

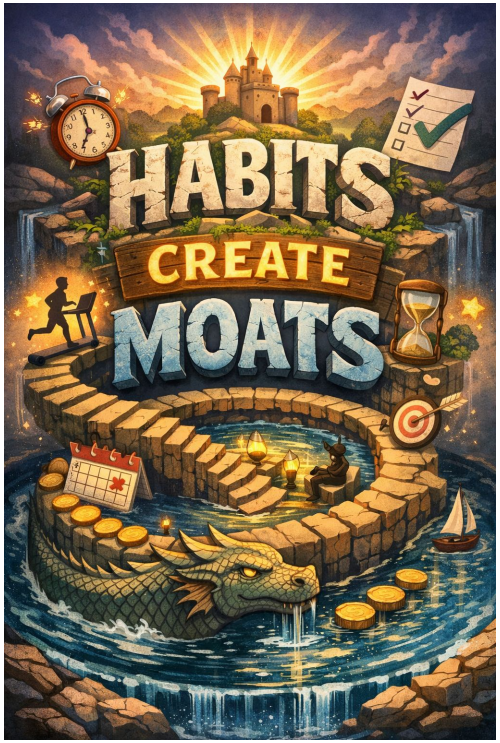
Beyond Prompt Engineering: Architecting the Full Context Stack for Reliable, Production-Grade AI Systems

John Whaley, Jan Jannink

CS 224G - 1/22/2026

Administrative Details

- Attendance secret code! Habits Create Moats
  - https://canvas.stanford.edu/courses/221239/quizzes/184860
- Today:
  - More on Context Engineering
  - Data Strategy



Learning Objectives

1:00–2:00

By the end of this lecture, you should be able to:

Distinguish Context vs. Prompt Engineering

Define context engineering as the broader architecture of information supply, distinct from query phrasing.

Enumerate LLM Call Components

Identify the full stack beyond the user message: system prompts, conversation history, tool definitions, and parameters.

Identify Application-Layer Levers

Determine which context elements (memory, RAG, uploads, tools) are controllable by the application developer.

Design Practical Context Plans

Construct a context strategy that balances information richness with token budget constraints.

Motivation: Prompting Is Not Enough

2:00–5:00

The quality of LLM output depends heavily on provided context. Consider this progression:

LEVEL 1: BAD PROMPT, BAD CONTEXT

"Is giving blood good for you?"

Generic query, zero personalization. Result: Generic WebMD-style advice.

LEVEL 2: BETTER PROMPT, BAD CONTEXT

"Does giving blood help with elevated SHBG levels?"

Specific scientific query, but stateless. Result: Theoretical answer, no specific advice.

LEVEL 3: BETTER PROMPT, BETTER CONTEXT

TARGET STATE

"Does giving blood help with elevated SHBG levels?
[Context Attached: Historical blood panels (2023-2025), donation dates, relevant gene testing results]"

Result: Highly personalized analysis comparing specific lab values against donation timing.

Key Takeaway: Better Prompt + Better Context = **Dramatically Better Results**

SECTION 02

What Goes Into an *LLM Call?*

From a simple chat box to a compiled request.

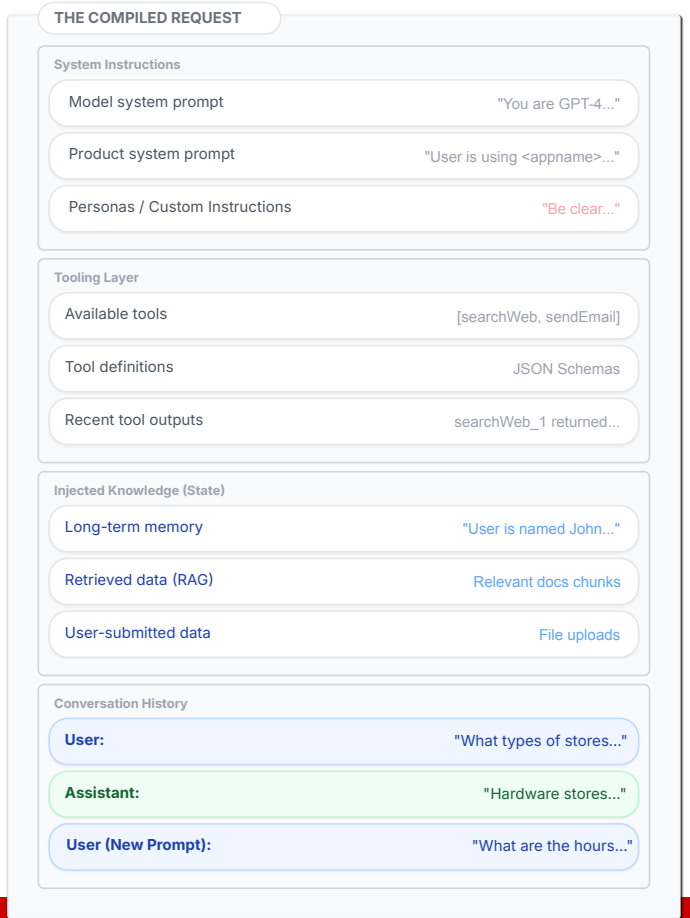
We'll walk through the layers of the context stack next.

John Whaley, Jan Jannink

CS 224G - 1/22/2026

The Context Engineering Model

5:30–9:00



The "Simple Chat" Illusion

What users see as a simple text box is actually the tip of an information iceberg. The application **compiles** this stack for every single turn.

1. **System & Persona Layer**
   Defines identity and behavior boundaries. Often composed of model defaults + app logic + user preferences.

2. **Injected State (The "Meat")**
   The stateless model knows nothing about you. Memories, RAG docs, and files must be injected afresh every time.

3. **Parameters (Invisible)**
   Temperature, Max Tokens, Logit Bias—these govern *how* the stack is processed.

CS224G Insight: Your application logic determines 90% of what the model "knows" before it generates a single token.

Call Parameters & The Context Window

9:00–14:00

Hidden Control Levers

Temperature

0.0 - 2.0

Controls randomness/creativity. Low temp (0.0) = deterministic; High temp (1.0+) = more creative but less stable.

Max Tokens

Integer

Hard limit on generation length. Does *not* limit input size, only output. Crucial for cost control and latency.

Logit Bias

Map<Token, Float>

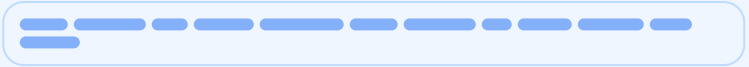
Surgical precision tool. Force the model to avoid or favor specific tokens (words). Often used for classification consistency.

The Token Budget Economy

The "Rule of Thumb"

1 Token ≈ 0.75 Words

(1,000 tokens ≈ 750 words)



Models see tokens, not words. Context limits are strict hard stops.

When the Context Window Fills Up:

The Compression Trade-off

Frontends must summarize history or truncate older messages. You gain space but lose **fidelity**.

The "Lost in the Middle" Phenomenon

Performance degrades as context fills up. Key instructions buried in the middle of a 100k+ token window are often ignored.

System Prompts & Instruction Layers

The Three Strata of Instructions

1. Model System Prompt

Immutable

The baseline identity baked in by the provider (OpenAI, Anthropic).

```
"You are GPT-4, a helpful assistant developed by OpenAI..."
```

2. Product System Prompt

Dev Control

Your application's specific framing. Defines the tool's role in your product.

```
"The user is using {AppName} to analyze financial data..."
```

3. Personas / Custom Instructions

User Control

User-defined preferences layered on top.

```
"Be concise. Answer in JSON. Prefer python for math."
```

Engineering Implications

The Underrated Lever: Personas

Most frontends bury this setting, but for power users and complex apps, allowing users to define **global invariants** is crucial.

Instead of repeating "format as markdown" in every prompt, bake it into the persona layer once. This reduces prompt noise and improves compliance.

Best Practices for System Prompts:

- **Sandwiching**
  Place critical instructions at both the start (System) and end (User) of the context window to combat "lost in the middle."
- **Separation of Concerns**
  Keep security guardrails in the Product Prompt (Layer 2) so users cannot easily override them in Layer 3.

SECTION 03

Injected *Knowledge*

Where most people think "context" lives:
Memories, Retrieval, and Uploads.

John Whaley, Jan Jannink

CS 224G - 1/22/2026

Injected Knowledge

> ***Crucial Mental Model:*** *The LLM is a stateless "brain-in-a-jar." It does not remember anything. Memory is a function of the application layer, not the model layer.*

1. Preserved Memories

Long-term state preserved about the user across sessions.
e.g., "User is named John, lives in CA, teaches at Stanford, prefers concise code."

2. Retrieved Data (RAG)

Data fetched from connected datastores before the call. The model doesn't "search"; the infrastructure searches and injects.

3. User-Submitted Data

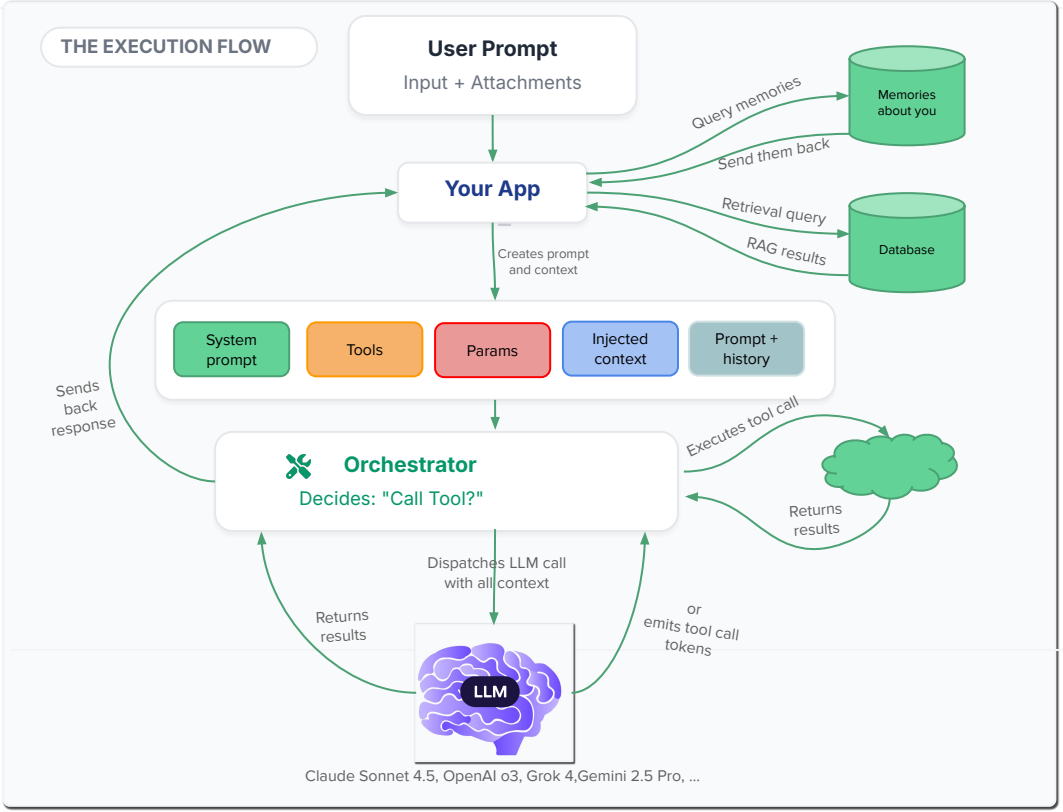
Ad-hoc files or "project" attachments.e.g., "Analyze this PDF."

Architecture Example

Customer Support Agent: App searches internal KB for query topic → Retrieves top 3 articles → Injects text into context → Model generates response.

Tools, Orchestrators & Agentic Loops

22:00–28:00



Tools are Programmable Context

Tools aren't magic. They are standard functions (API calls, scripts) wrapped in a schema the LLM can "read."

The "Brain-in-a-Jar" Reality

The LLM does not execute code or search the web. It emits **tokens** indicating intent. The **orchestrator** (your app) does the actual work.

AGENTIC WORKFLOW

The Loop

Orchestrator runs tool → Output becomes new context → Feed back to LLM → Repeat until done.

What You Can Control

28:00–33:00

The *Context Engineering* Checklist

While much of the LLM stack feels opaque (model weights, training data), these **five specific levers** are directly in your control as an application builder.

Builder's Tip

"Don't just tweak the prompt. Engineer the inputs that surround it."

1. Personas / Custom Instructions

Define global invariants here (e.g., "Always return JSON"). Don't waste tokens repeating this in every prompt.

2. Available Tools

Curate your toolkit. Giving a model 50 tools confuses it. Give it the 5 it actually needs for the task.

3. RAG / Datastore Connections

You control the search quality. Which databases are connected? How are chunks retrieved? This is a huge performance lever.

4. User-Submitted Data

Encourage file uploads. Explicit context ("Read this PDF") beats implicit retrieval every time.

5. Your Prompt

Still matters! But treat it as the final instruction, not the entire context payload.

In-Class Discussion

33:00–37:00

EXERCISE

The Token Budget Crisis

Your long-running agent chat is hitting the **128k context limit**. Performance is degrading.
What is your eviction policy?

1. **Compression vs. Eviction**

   Do you summarize history (lossy compression) or drop oldest turns (eviction)? Why?

2. **The "Important" Bits**

   What specific pieces of context must be **pinned** and never dropped? (e.g. system instructions, tool definitions)

3. **Re-retrieval Strategy**

   If you drop a file from context, how does the model get it back if needed later?

FORMAT

- Discuss
- 4 Minutes
- Share with class

Context Engineering Recap

38:00–42:00

1

Context Engineering > Prompt Engineering

The prompt is just the tip of the spear. The surrounding context (injected knowledge, history, system instructions) often determines performance more than the phrasing of the question.

2

Context is a Compiled Artifact

Don't think of it as "chatting." Think of it as your application **compiling** a massive text file (history + RAG + tools + system prompts) on every single turn.

3

State Lives in the App, Not the Model

The LLM is a stateless brain-in-a-jar. All "memory" (user preferences, project files, past turns) is retrieved and injected by your infrastructure.

4

Master the Five Levers

You have control over: `Personas`, `Tools`, `RAG`, `Uploads`, and the `Prompt`. Use all five, not just the last one.

> *"Think of the LLMs as a new, genius Ph.D.-level team member... but one for whom it is **always** their first day at work."*
>
> — **Andy Bromberg**

Building an Effective Data Strategy via UX

Agenda

- Introduction to Data Strategy and UX
- Data as a Strategic Asset
- Product Engagement: The Core of Success
- Enhancing Feedback Loops in Product Usage
- Leveraging LLMs for Better Feedback
- The Role of Internal Tools
- The Importance of Small Details
- Data Quality: Not All Users Are Equal
- Addressing Bias and Diversity in Data
- Challenges and Call to Action
- Acknowledgements and Closing

Introduction

Data strategy is the true competitive edge in the AI era.

The winners in this era of AI will be the organizations with ongoing access to highly-relevant, differentiated, quality data. In a world where all public data will be ingested and used, unique data is the most important strategic asset.

Data

46:00–48:00

Data's Strategic Value

AI Strategy

The winners in AI will prioritize building effective data feedback loops. A static data set is not nearly as useful as ongoing access to data.

Data Is Not Oil: Differentiation Is Important

Data's worth lies in differentiation and targeting, not as a commodity. Effective and sustainable AI requires unique, high-quality data.

Understand Where To Use Data

Good data is not only for machine learning training. It enables you to offer personalized experiences and make data-driven decisions.

Strategy

48:00–51:00

A strong data strategy starts with product and user engagement.

Product Engagement

Implement strategies to capture high-quality data from engaged users. Great engagement drives retention and data insights.

User Engagement

Emphasize the importance of user engagement in building a strong data strategy. Longitudinal data reflects sustained user engagement.

Data Insights

Highlight the correlation between user engagement, retention, and effective data strategies. Encourage actions that promote user engagement.

Feedback

Making feedback effortless is essential.

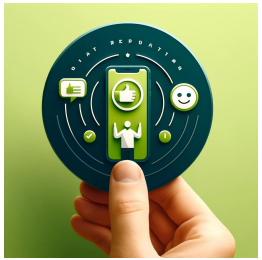
51:00–54:00

Make It Inherent

Capture feedback seamlessly during product usage to enhance data collection without disrupting user experience.

Integration of Tools

Integrate editing and collaboration tools directly into the product for both user convenience as well as data feedback.



Instant User Gratification

Provide instant gratification for user corrections to enhance engagement.

User Feedback Techniques

Label Feedback

The user provides labels (A is good, B is bad, C is neither) to assess products or features.

Qualitative Feedback

Encourage users to provide detailed qualitative feedback to understand their experiences and pain points.

Discreet Feedback

Ask for discreet feedback to simplify data collection and gauge overall satisfaction (good/bad).

Passive Learning

Observe user behavior passively to gather data on how they interact with the product or platform.

Feedback Explanations

Prompt users to explain the reasons behind their ratings or labels to uncover insights.

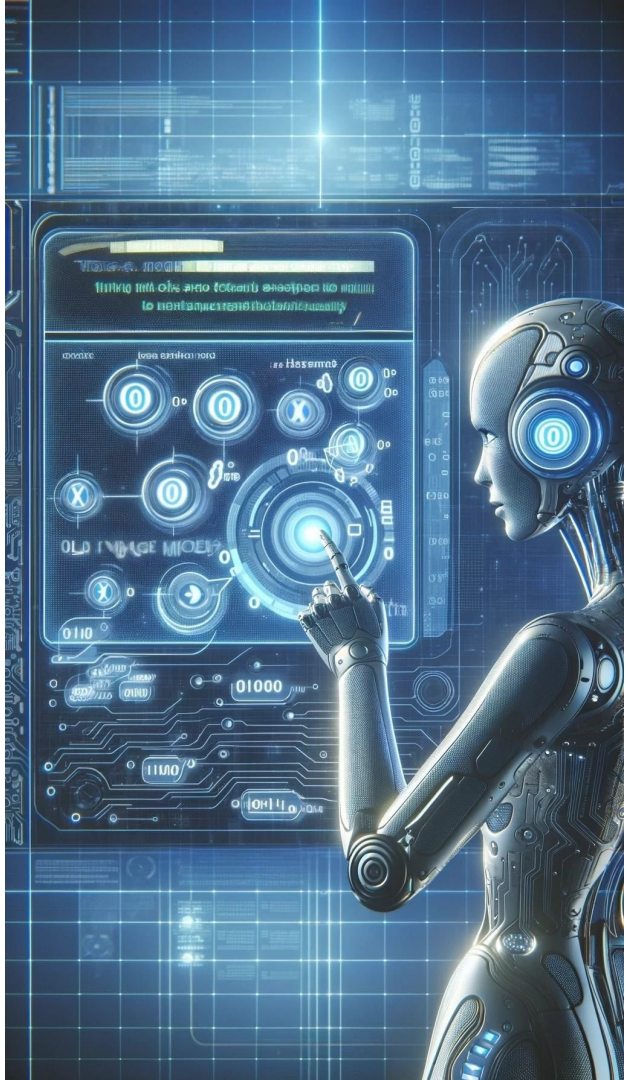
AI Learning Overrides

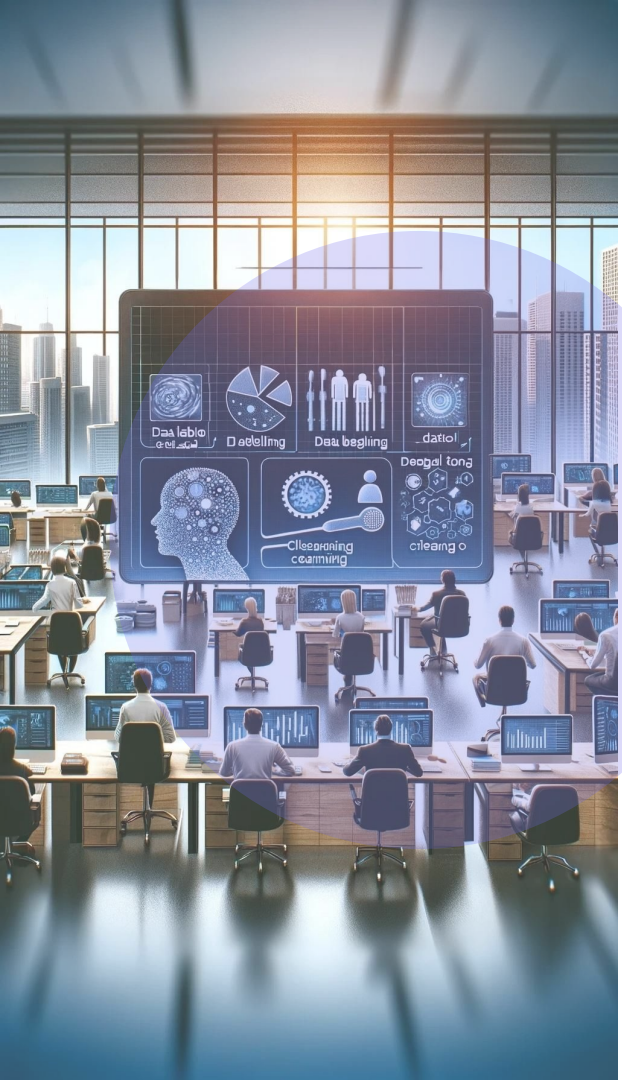
Allow users to override AI-generated feedback, and learn from these changes to improve personalization.

LLMs

Using LLMs to Enhance Feedback Quality

- LLMs allow for natural follow-up questions and to capture feedback subtleties.
- Improve understanding of not just the what, but the why.
- Integrate natural language for an enhanced user feedback loop.
- Efficiently summarize and extract key points from large unstructured textual data.





Tools

“Product” Includes Internal Tools

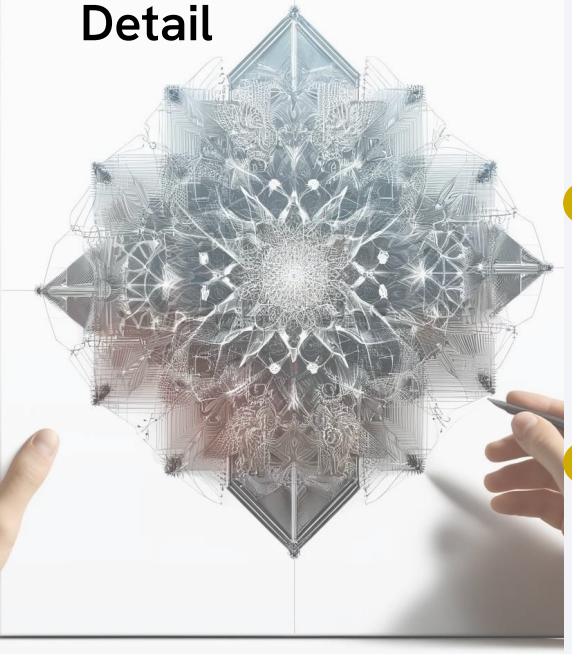
Internal tools, like those for data labeling and cleaning, are essential for maintaining high data quality in AI.

It is usually worth the investment, as empowering your employees, data labelers, and data reviewers has a multiplying effect.

For instance, at Redcoat AI, we devoted equal effort to our internal data labeling tool as our external product.

Details

The Impact of Attention to Detail



63:00–65:00

Small Details Matter

Examples like spell-check and Grammarly demonstrate how small details can make a big difference in user experience.

Gamification Elements

Incorporating gamification elements can make the feedback process more enjoyable for users and encourage their participation.

Immediate Feedback

Offering immediate feedback satisfaction, like allowing users to correct data in real-time, enhances engagement and data accuracy.

Data

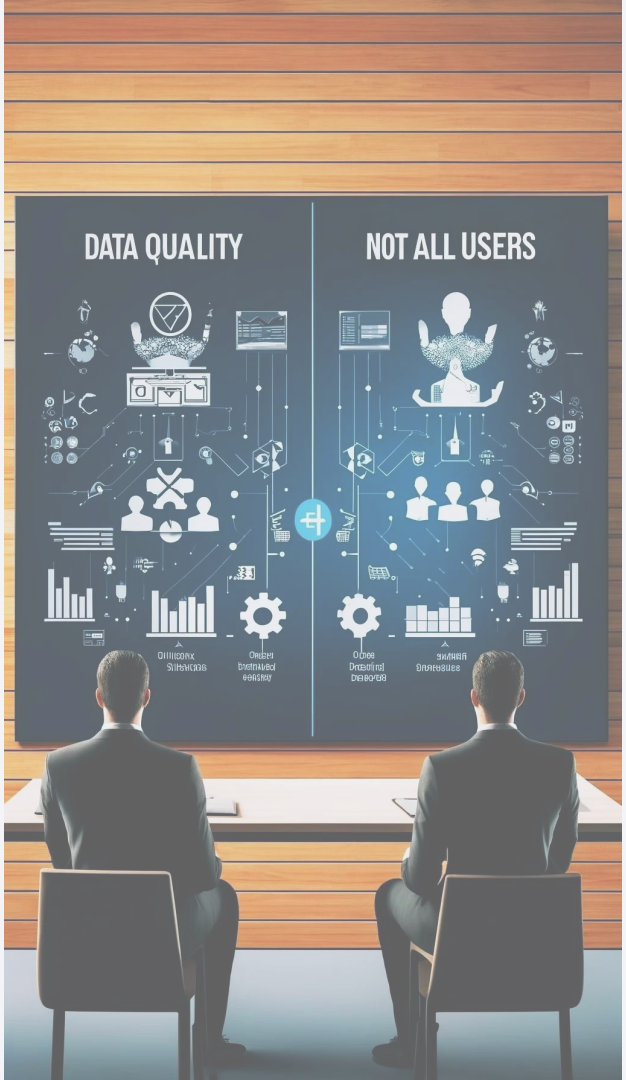
Data Quality: Not All Users Are Equal

Factors Affecting Data Quality

- Data from experts is more valuable than data from novices.
- Data from consistent users is more valuable than data from inconsistent ones.
- Longitudinal data is valuable, as it is hard to get and essential to understand long-term value and retention.

Prioritize Data Value Through Validation And Diversification

- Use metadata and cross-validation to identify the most valuable users.
- Prioritize contributions from experts and consistent users.
- Prioritize data quality and diversity over sheer data quantity.



Managing Bias and Diversity in Data

Dangers from Lack of Diversity and Representation in Data

- Example at UnifyID: 95% of beta users were male techies from the Bay Area. So it didn't work for middle-aged women in Korea.
- Know your data and users to detect and address potential biases.
- Understand the inherent bias in generative AI tools and models.

Importance of Diverse Data Sets

- Creating diverse data sets is vital for quality and representation.
  - Diversity in data boosts innovation, minimizes biases, and fosters inclusivity.
  - Be creative about how to get data from a more diverse, representative set of users.
  - Building a diverse data set is hard! But doing hard things creates differentiation and value.

Audience Challenge

72:00–75:00

Calls To Action

We can do better than conversational chatbots.

Encourage user engagement with proactive interfaces. Foster a sense of interactivity to enhance the user experience.

Let’s not recreate Clippy. Embrace creativity!

Embrace creativity and a user-centric design approach to deliver unique and impactful user experiences.

AI agents can be more interactive.

The best humans know when and what questions to ask. AI agents should do the same.

Consider data strategy as you design and build your products.

Think about how you can incorporate data strategy into products. Evaluate UX ideas based on the value of the data they will generate.