Building Effective Unstructured Data Systems

Shreya Shankar

~~UC Berkeley EECS~~ **Soon...Carnegie Mellon!**

June 2026

sh-reya.com

Data Systems

You've heard of them: e.g., Postgres, MySQL, Spark, Power BI, Tableau

*Given a user's question, get the **right answer** fast*

- Query languages
- Query optimization and execution
- Storing and organizing data

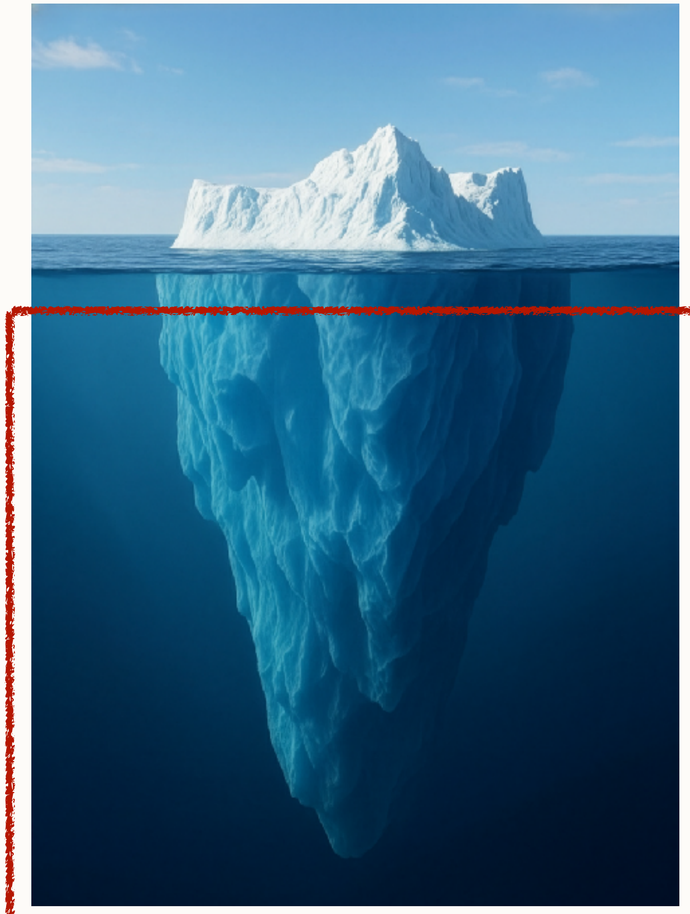
Inner Loop

*Help the user ask the **right questions***

- Interaction design
- Visualization and communication
- Sensemaking theory

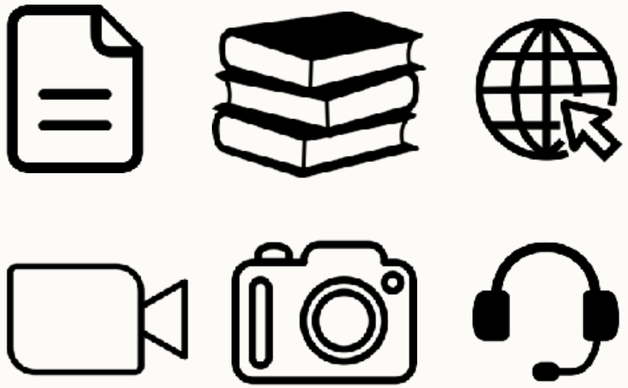
Outer Loop

What's Next?



Structured data

Unstructured data



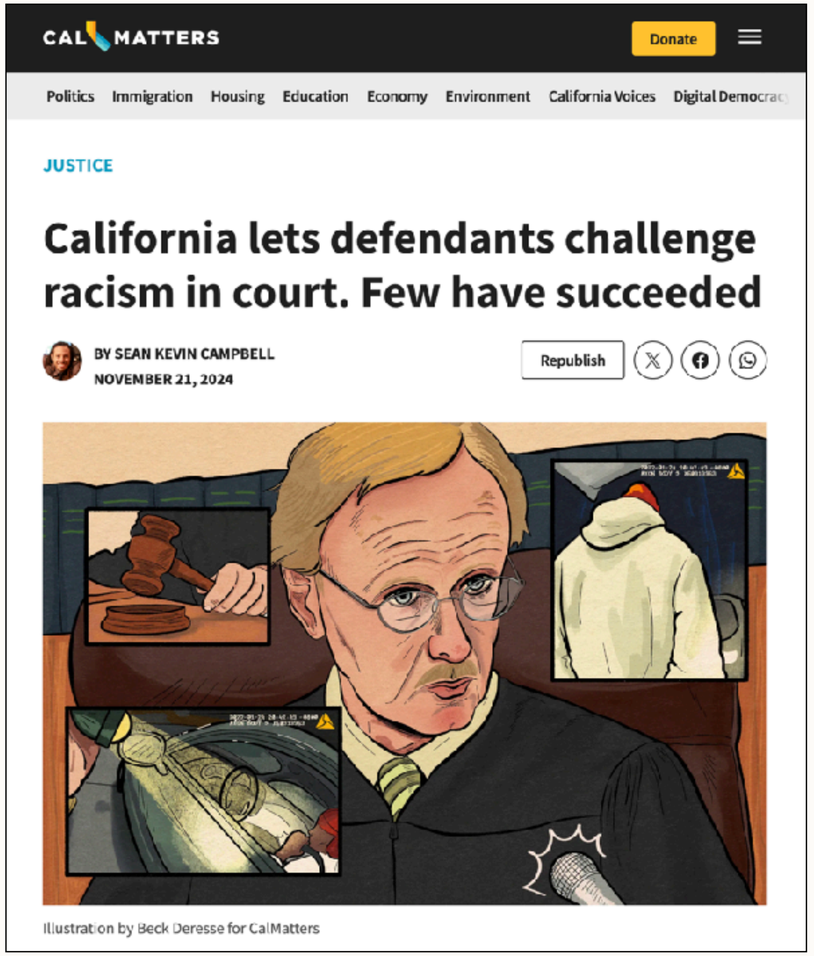
← Can't use SQL to query this type of data

Use Case: Public Defenders

California's Racial Justice Act (2020): Defendants can challenge convictions for racial bias

Analysis requires: extract evidence from court documents; compare across cases. Millions of documents a year.

Only ~12 successful challenges in 4 years (out of ~10k eligible) because the task is so hard!



Use Cases are Everywhere

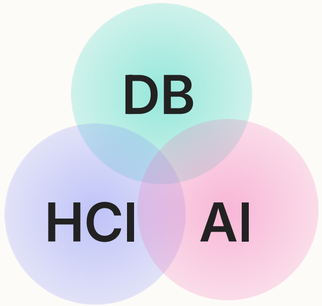
What side effects do patients report for Ozempic?

Which contracts have unfavorable arbitration clauses?

What are common complaints in customer reviews?

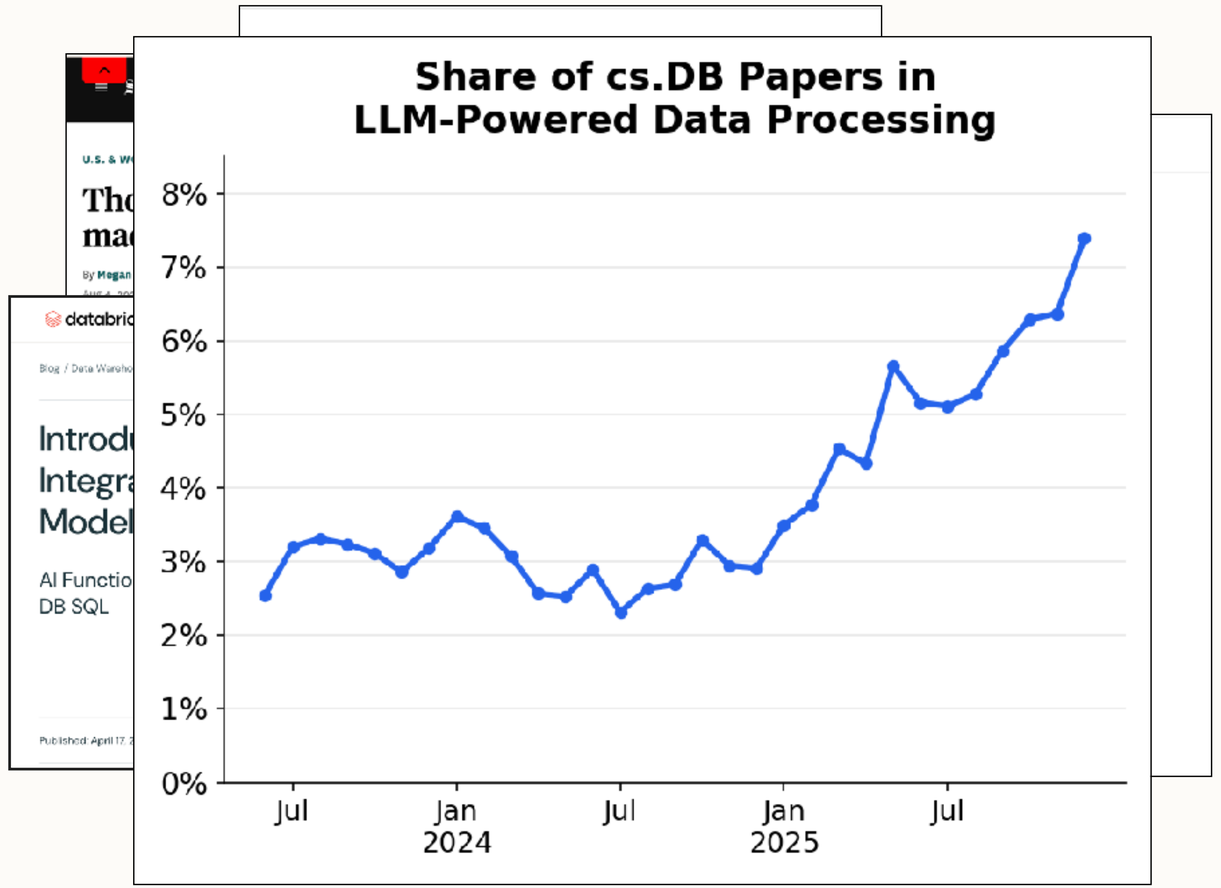
Interdisciplinary challenges:

- ✦ Multi-operation, need to run at scale
- ✦ Ambiguous operation definitions
- ✦ Human-defined; context-dependent



Why Care About LLM-Powered Data Processing?

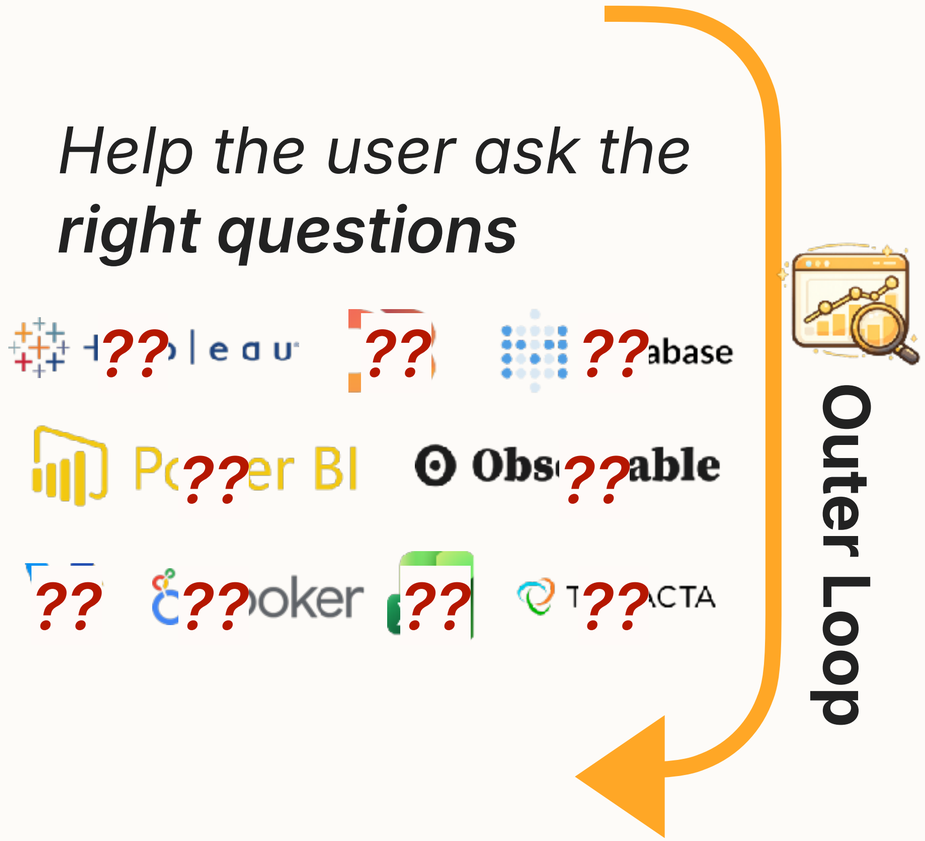
- ✦ **Real and high-stakes deployments:** e.g., journalists, public defenders, climate scientists
- ✦ **Industry adoption:** Snowflake, BigQuery, Databricks
- ✦ **Growing research community**



LLM-Powered Unstructured Data Systems

*Given a user's question, get the **right answer** fast*

*Help the user ask the **right questions***



Today's Talk

1. **DocETL: an unstructured data system that I built**
2. **Inner loop (query optimization)**
   - ✦ *What is the space of query plans?*
   - ✦ *How to search the plan space?*
3. **Outer loop (iterating on queries)**
   - ✦ *How should users author queries?*
   - ✦ *How can users validate results at scale?*
4. **Where do we go from here?**

Today's Talk

1. **DocETL: an unstructured data system that I built**

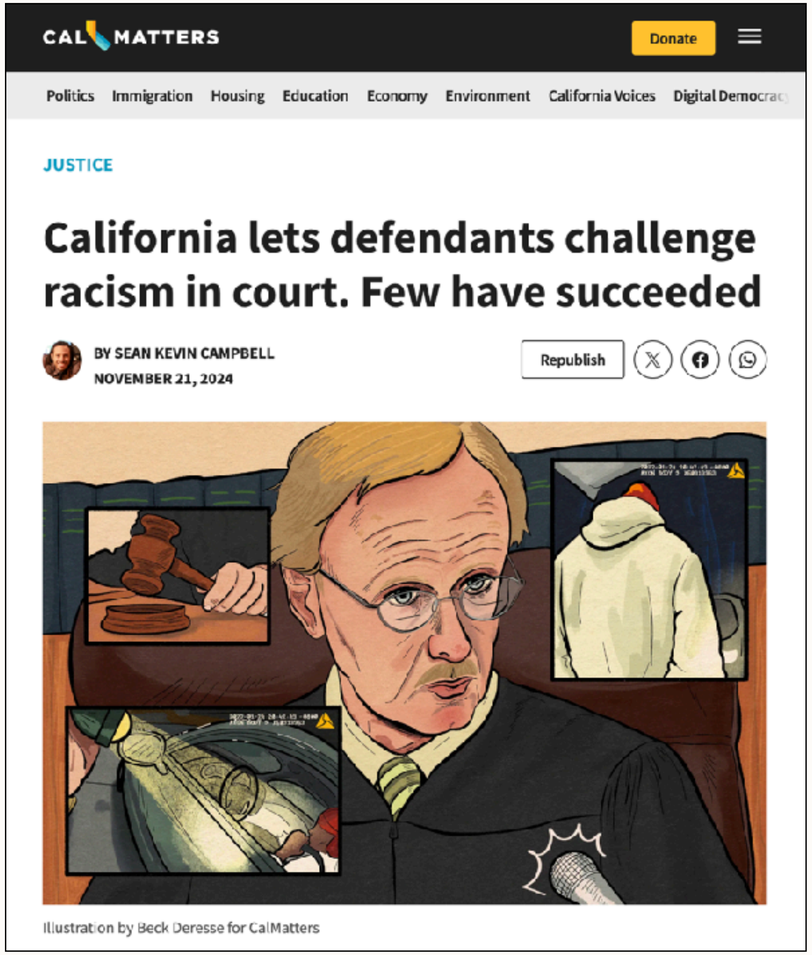
Grounding Example: Public Defenders

Data: bulk discovery dumps, spanning multiple defendants and cases
Queries: hundreds per defendant

Types of operations in queries:

- ✦ *Extract* bias, language, and facts from each case
- ✦ *Synthesize* summaries and reports
- ✦ *Compare* across similar cases

Naive cost: >$1,000/defendant with GPT-5.2
Goal: high accuracy, low cost



Example: DocETL Pipeline/Query

```
type: map
prompt: “Determine the name of
the judge in this trial”
output:
  schema:
    name: str
```

```
type: reduce
reduce_key: name
prompt: “Summarize the common implicit
biases shown by this judge across all
their trial transcripts”
output:
  schema:
    common_biases: str
```

id	transcript	name
…	…	…

name	common_biases
…	…

DocETL Operators

4 common operators: `map`, `reduce`, `filter`, and `join`.

- ✦ Only a subset of DocETL operators.
- ✦ Declarative & natural language; inspired by relational operators

New challenges:

- ✦ Users may not be able to say precisely what they want
- ✦ Correctness must be evaluated empirically (using samples)!

```
type: reduce
reduce_key: name
prompt: “Summarize the common implicit biases shown by this judge across all their trial transcripts”
output:
  schema:
    common_biases: str
```

Today's Talk

2. **Inner loop (query optimization)**
   - ✦ *What is the space of query plans?*
   - ✦ *How to search the plan space?*

Roadmap: Query Optimization

Required for declarative data systems!

Given a query ("what"), find a good execution plan ("how").

1. Plan Space

What are the different ways to execute this query?

- ✦ Background: rewrite rules
- ✦ Our rewrite directives
- ✦ Our new operators

2. Search

How do we estimate plan costs and find the best plan?

- ✦ Background: DP/memo-based search
- ✦ Simpler setting: Task Cascades
- ✦ A more general setting: MCTS-based search

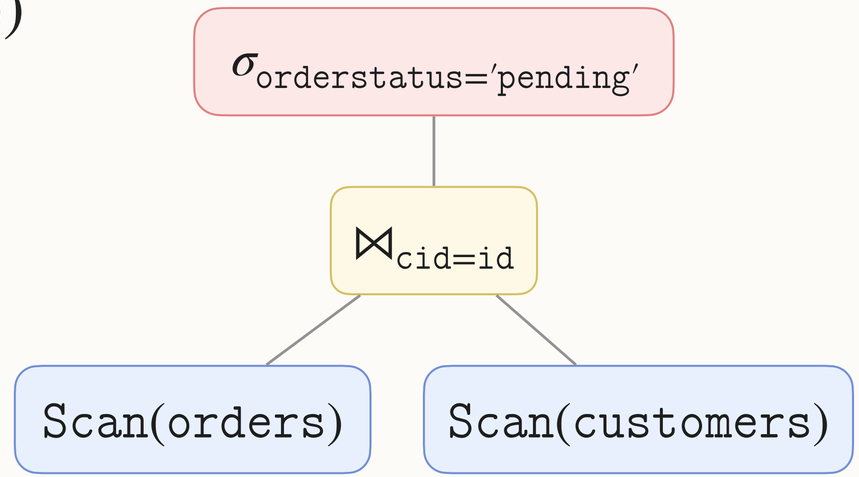
Background: Rewrite Rules

Rewrite rules enumerate equivalent query plans. Some are algebraic; some implementation-based:

- Projection or filter pushdown; e.g.,

$$\pi_x(A \bowtie B) \rightarrow \pi_x(\pi_y(A) \bowtie \pi_z(B)) \text{ or } \sigma_p(f(A)) \rightarrow f(\sigma_p(A))$$

```
SELECT * FROM orders JOIN customers
ON orders.cid = customers.id
WHERE orderstatus = 'pending'
```

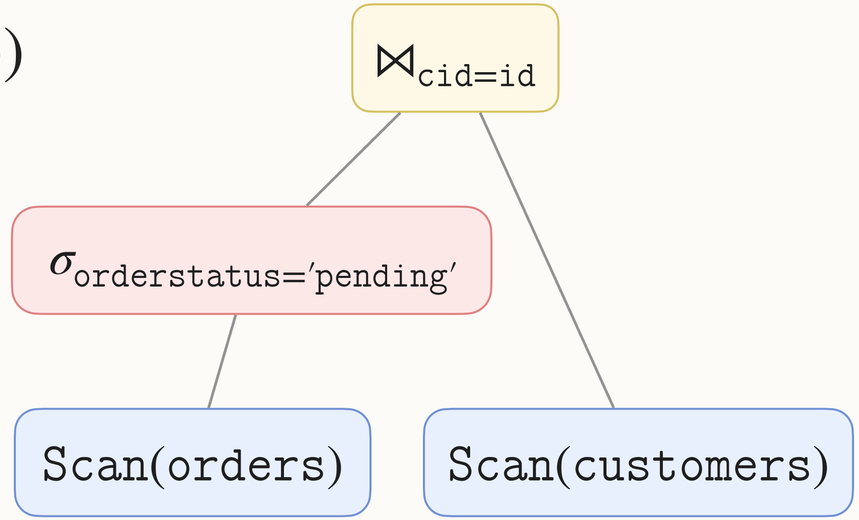


Background: Rewrite Rules

Rewrite rules enumerate equivalent query plans. Some are algebraic; some implementation-based:

- Projection or filter pushdown; e.g.,
  $\pi_x(A \bowtie B) \rightarrow \pi_x(\pi_y(A) \bowtie \pi_z(B))$ or $\sigma_p(f(A)) \rightarrow f(\sigma_p(A))$
- Choosing algorithms; e.g.,
  $A \bowtie B \rightarrow \text{HashJoin}(A, B)$

```
SELECT * FROM orders JOIN customers
ON orders.cid = customers.id
WHERE orderstatus = 'pending'
```

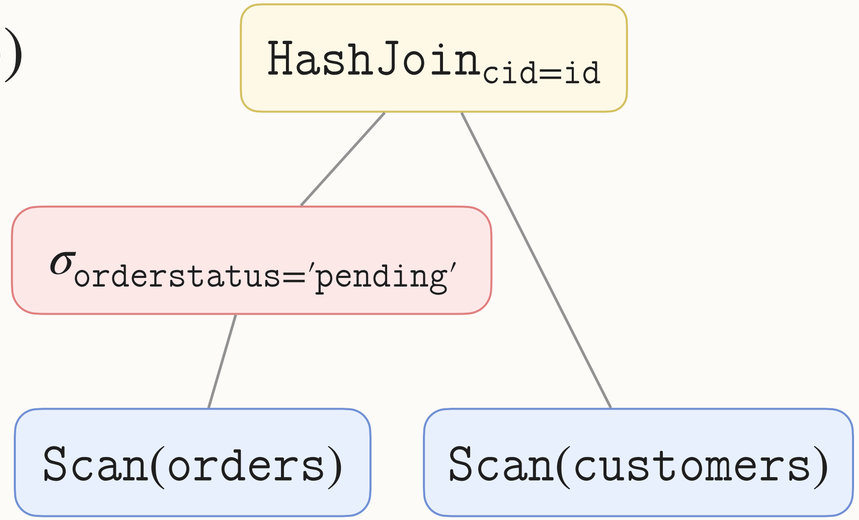


Background: Rewrite Rules

Rewrite rules enumerate equivalent query plans. Some are algebraic; some implementation-based:

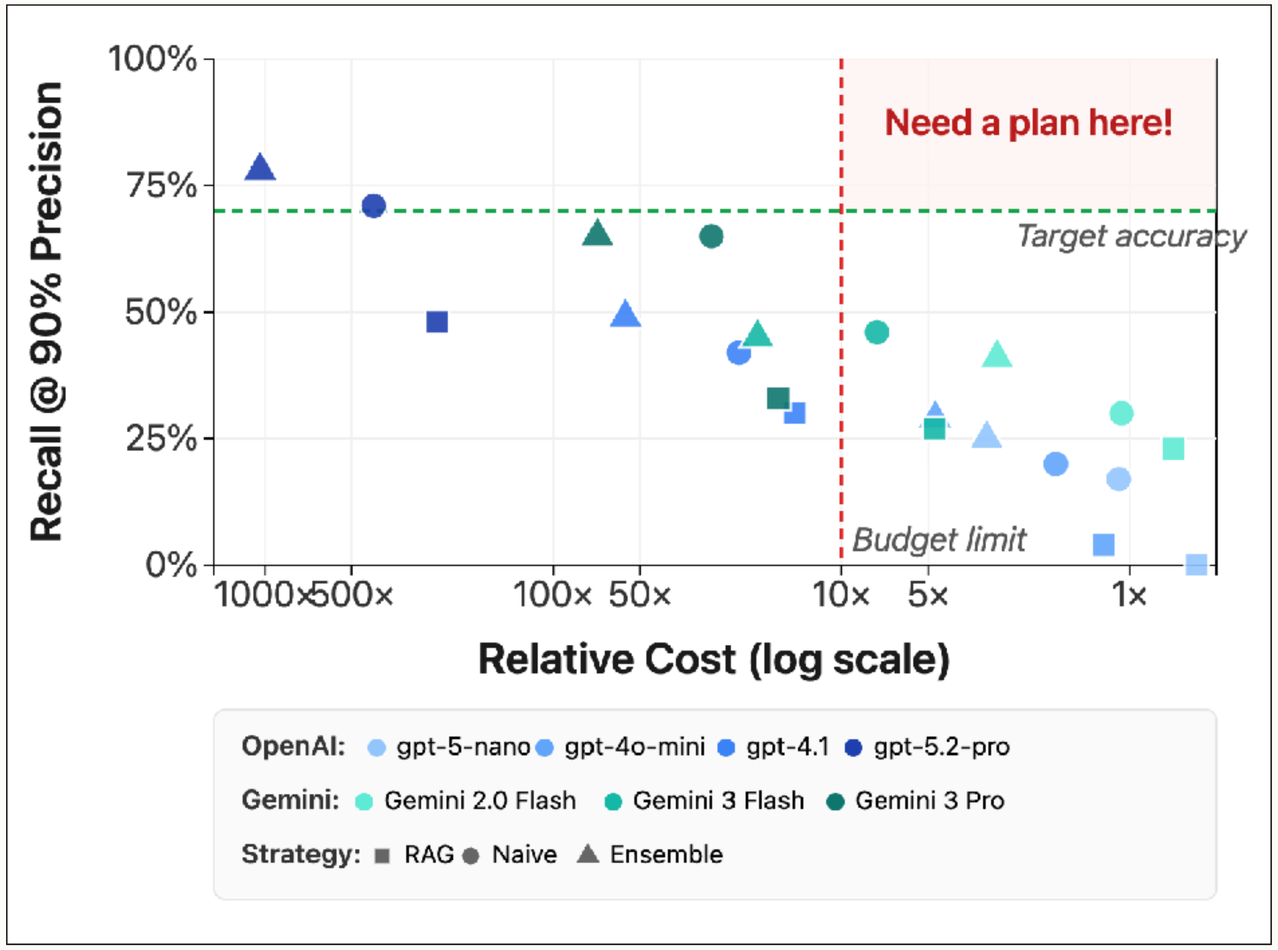
- Projection or filter pushdown; e.g.,
  $\pi_x(A \bowtie B) \rightarrow \pi_x(\pi_y(A) \bowtie \pi_z(B))$ or $\sigma_p(f(A)) \rightarrow f(\sigma_p(A))$
- Choosing algorithms; e.g.,
  $A \bowtie B \rightarrow \text{HashJoin}(A, B)$

```
SELECT * FROM orders JOIN customers
ON orders.cid = customers.id
WHERE orderstatus = 'pending'
```



Applying to DocETL: same algebraic rules; new NL operator implementations (e.g., vary LLMs, prompting strategies, context reduction).

Rewrite Rules Don’t Work Well



Building Effective Unstructured Data Systems. Shreya Shankar.

Can We Solve the Problem?

LLM accuracy depends on task & data:

✦ Document length

✦ # facts relevant for the answer in each document

1000s of docs

Each doc is 1000s of pages long

```
type: map
prompt*: “Find all
statements made by the
judge indicating implicit
bias. Explain why.”
```

**Prompt very simplified, for space constraints.*

Can We Solve the Problem?

LLM accuracy depends on task & data:

✦ Document length

✦ # facts relevant for the answer in each document

Insight: Decomposition could improve cost & accuracy.

```
type: map
prompt*: “Find all
statements made by the
judge indicating implicit
bias. Explain why.”
```

**Prompt very simplified, for space constraints.*

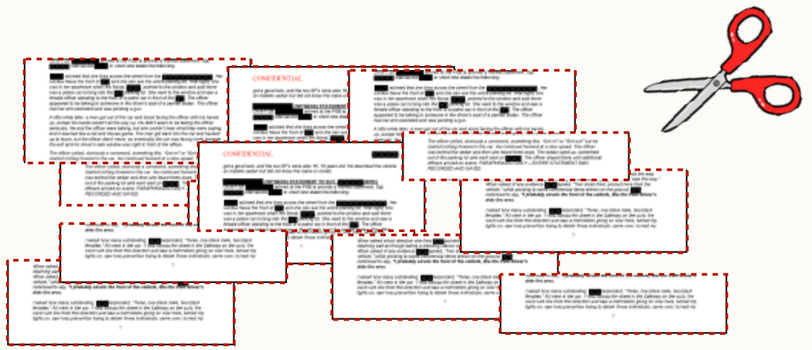
Can We Solve the Problem?

LLM accuracy depends on task & data:

✦ Document length

✦ # facts relevant for the answer in each document

Insight: Decomposition could improve cost & accuracy.



```
type: map
prompt*: “Find all
statements made by the
judge indicating implicit
bias. Explain why.”
```

∗Prompt very simplified, for space constraints.

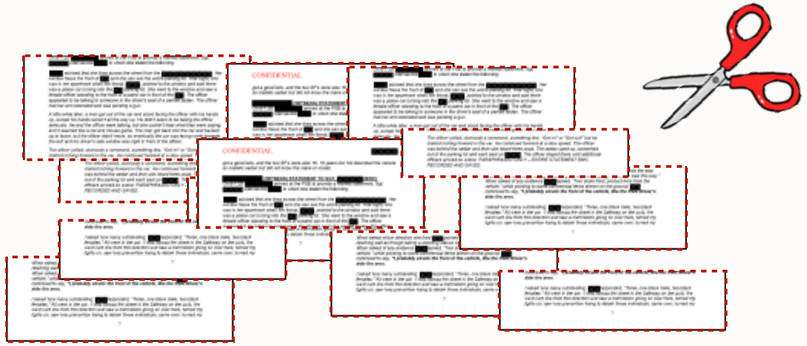
Can We Solve the Problem?

LLM accuracy depends on task & data:

✦ Document length

✦ # facts relevant for the answer in each document

Insight: Decomposition could improve cost & accuracy.



```
type: map
prompt*: “find racial slurs”
```

```
type: map
prompt*: “find stereotyping”
```

∗Prompt very simplified, for space constraints.

Can We Solve the Problem?

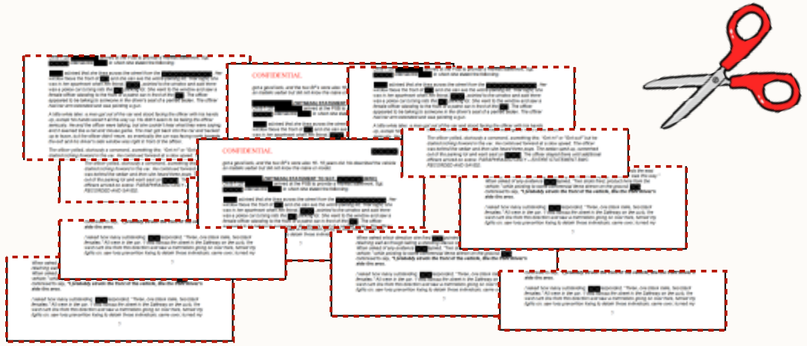
LLM accuracy depends on task & data:

✦ Document length

✦ # facts relevant for the answer in each document

Insight: Decomposition could improve cost & accuracy.

Can we apply decomposition via rewrite rules during query optimization?



```
type: map
prompt*: “find racial slurs”
```

```
type: map
prompt*: “find stereotyping”
```

∗Prompt very simplified, for space constraints.

Our Solution: Rewrite *Directives*, not Rules

DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. **Shankar** et al. *VLDB '25.*

Rewrite directives formalize semantic rewrites in the query optimizer.

✦ They generate new operators to better match LLM capabilities

✦ Instantiation requires reasoning about the data and task

Example: $\pi_p(D) \rightarrow \gamma_{id,p'}(\pi_{p''}(\text{Split}_k(\text{AddId}(D))))$ for data decomposition

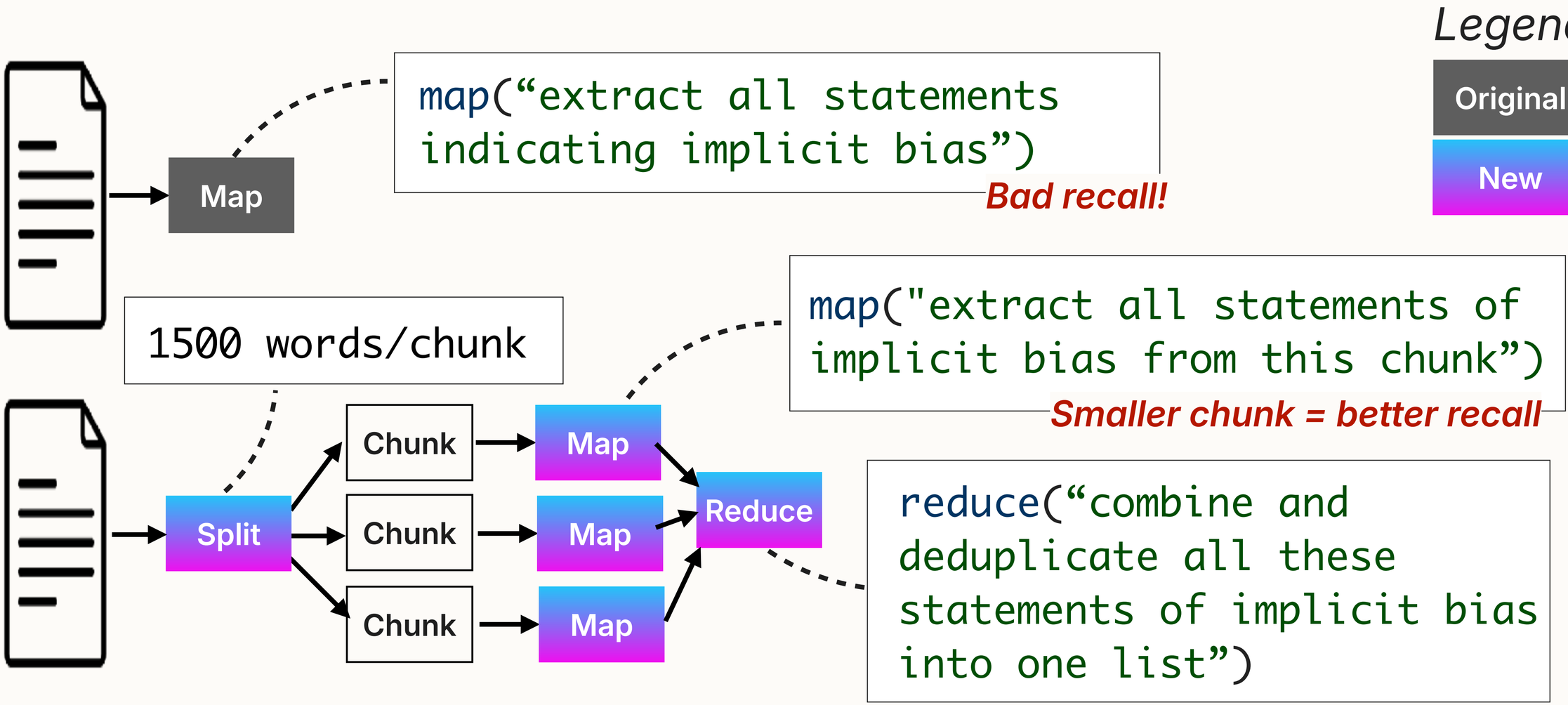
Some challenges:

✦ Need other (non-relational) operators

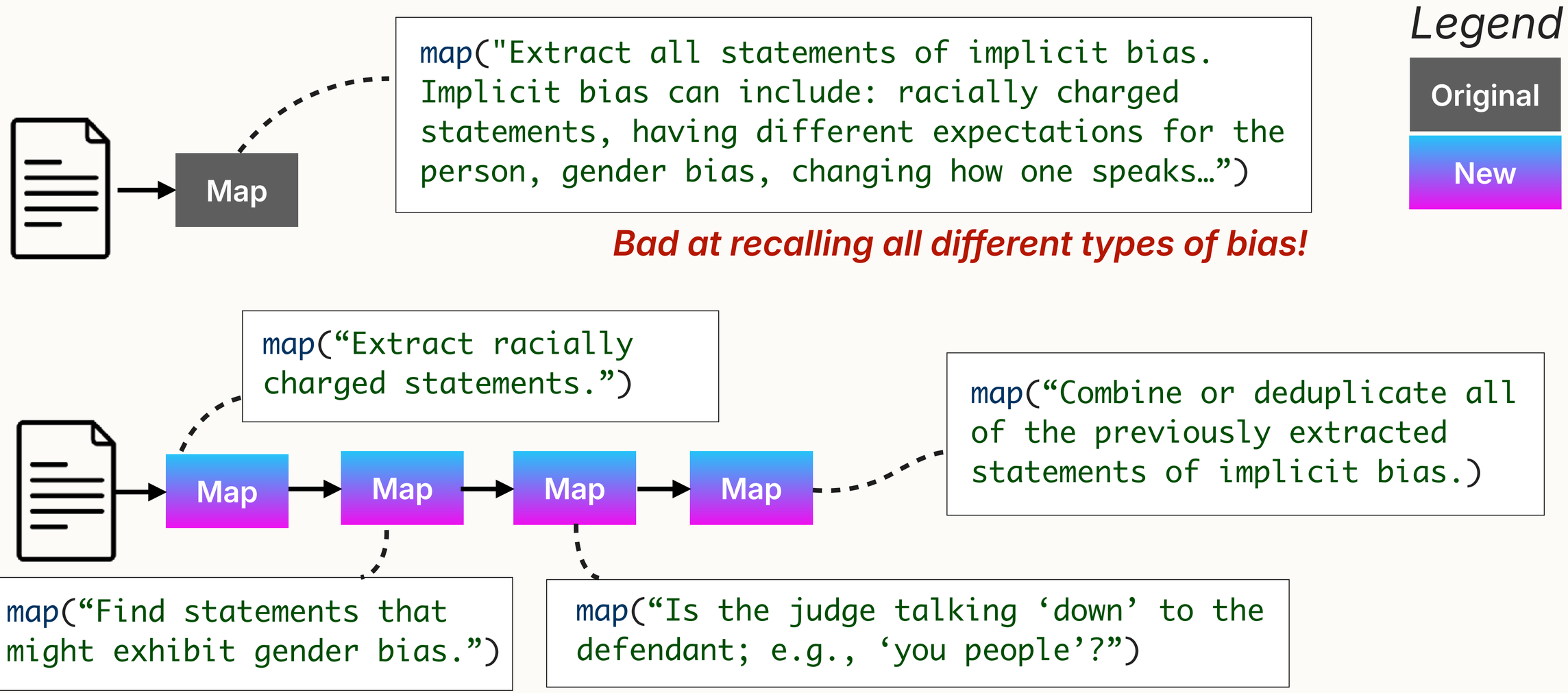
✦ Open-ended instantiation space; instantiated by LLM agents

✦ Many instantiations might be bad!

Data Decomposition as a Rewrite Directive



Prompt Decomposition as a Rewrite Directive



New Operators for DocETL

Rewrite directives require new operators:

- ✦ Split
- ✦ Gather
- ✦ Resolve
- ✦ CodeOp
- ✦ Sample

New Operators for DocETL

Rewrite directives require new operators:

- ✦ Split
- ✦ **Gather**
- ✦ Resolve
- ✦ CodeOp
- ✦ Sample

After `split`, `gather` ***augments each chunk with relevant context.***

Challenge

Chunk 1
👮 Officer J. Smith...

Chunk 2
He then proceeded to...

Who is “he”?

What happened before?

Example Gather Configs

⬇️ Previous/Next Chunks

See Figure 2 on the next page for...

[Fig 2] Architecture diagram...

Need next chunk for referenced context

📝 Transformed Content

Previous 200 pages: “Suspect was last seen in LA...”

“He boarded a train to...”

Summary of a long prefix

🗂️ Document Metadata

1. Contract Terms
1.1 Licensing
1.1.2 Usage Rights

The licensee shall...

Section hierarchy context

New Operators for DocETL

Rewrite directives require new operators:

- ✦ Split
- ✦ Gather
- ✦ **Resolve**
- ✦ CodeOp
- ✦ Sample

```
type: map
prompt: “Determine the name of the judge in this trial”
output:
  schema:
    judge: str
```

→

```
type: reduce
reduce_key: “judge”
prompt: “Summarize the common implicit biases shown by this judge across all their trial transcripts”
output:
  schema:
    common_biases: str
```

Challenge

Document 1
Judge X. Quinnsworth…

Document 2
Judge Xander Quinnsworth was…

judge: “X. Quinnsworth”

judge: “Xander Quinnsworth”

The judges won’t be in the same group!

Resolve performs entity resolution.

Rewrite Directives in DocETL

Extensible library of 15 operators and 31 directives across 5 categories

- ✦ Fusion & Reordering
- ✦ Code Synthesis
- ✦ Data Decomposition
- ✦ Projection Synthesis
- ✦ From LLM research
  - ✺ e.g., prompt optimization; self-refinement

Roadmap: Query Optimization

Required for declarative data systems!

Given a query ("what"), find a good execution plan ("how").

1. Plan Space

What are the different ways to execute this query?

- ✦ Background: rewrite rules
- ✦ Our rewrite directives
- ✦ Our new operators

2. Search

How do we estimate plan costs and find the best plan?

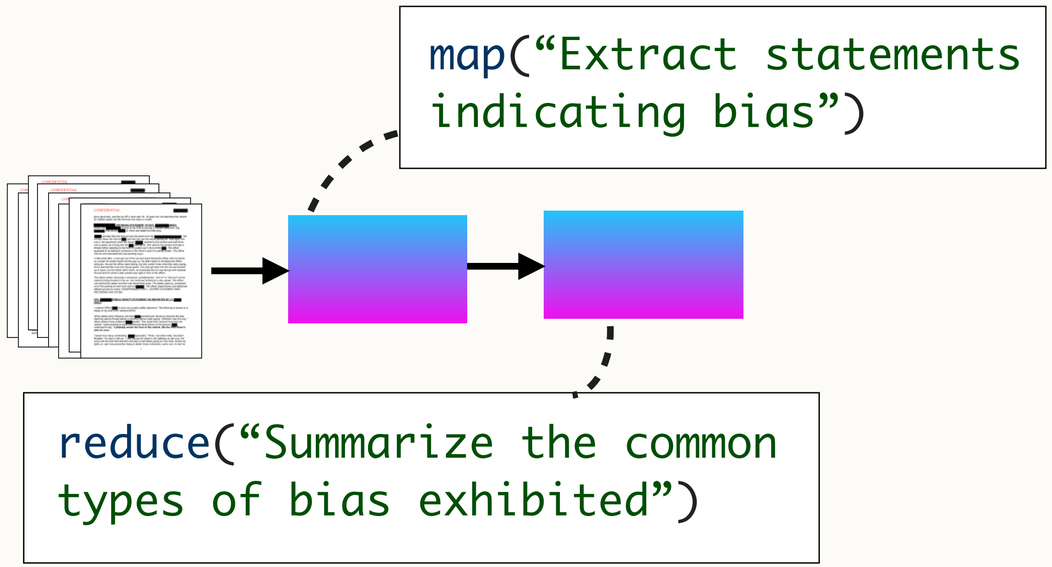
- ✦ Background: DP/memo-based search
- ✦ Simpler setting: Task Cascades
- ✦ A more general setting: MCTS-based search

Background: Search

Traditional (e.g., Cascades)	Unstructured (what's new?)
✦ Single objective (latency) + find best plan 😊	✦ Multi-objective (price, accuracy) + find Pareto frontier 😢
✦ Assumes optimal substructure, so accelerated search via memoization 😊	✦ No optimal substructure 😢

Key Insight: No Optimal Substructure

Even in the simplest case of only varying LLMs...



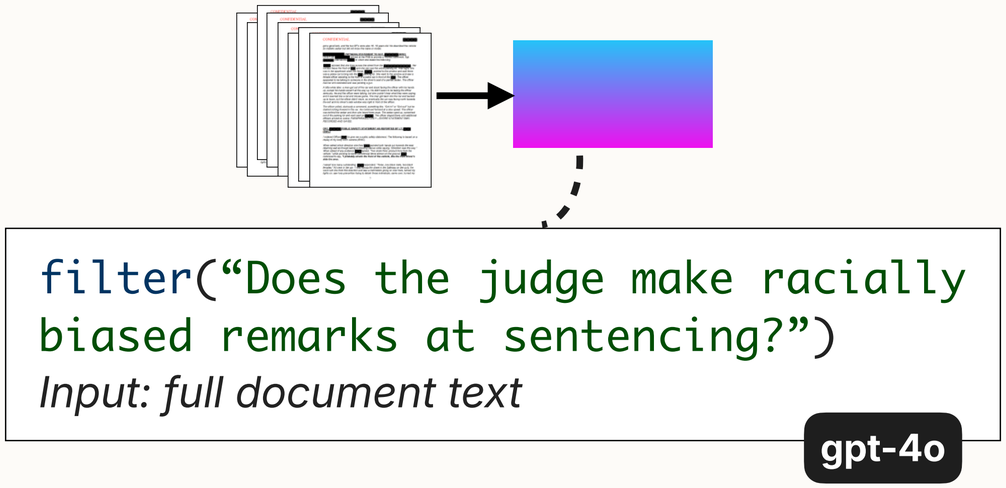
LLM behavior doesn't cleanly compose across operators!

Revisiting Search

Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26.*

Simpler problem: reduce cost for a single map or filter operator

Approach: search over cost-reducing rewrites, maintaining target accuracy (e.g., 90%) w.r.t. original



Revisiting Search

Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26.*

Simpler problem: reduce cost for a single map or filter operator

Approach: search over cost-reducing rewrites, maintaining target accuracy (e.g., 90%) w.r.t. original

Cost-Reducing Rewrites

- ✦ Smaller document portion

```
filter("Does the judge make racially biased remarks at sentencing?")
```

Input: top-k chunks in document text

gpt-4o

Revisiting Search

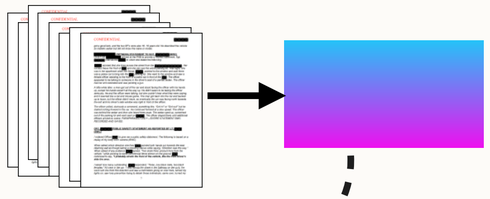
Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26.*

Simpler problem: reduce cost for a single map or filter operator

Approach: search over cost-reducing rewrites, maintaining target accuracy (e.g., 90%) w.r.t. original

Cost-Reducing Rewrites

- ✦ Smaller document portion
- ✦ Cheaper model



```
filter("Does the judge make racially biased remarks at sentencing?")
```

Input: top-k chunks in document text

gpt-4o-mini

Revisiting Search

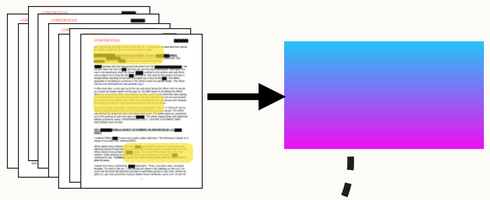
Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26.*

Simpler problem: reduce cost for a single map or filter operator

Approach: search over cost-reducing rewrites, maintaining target accuracy (e.g., 90%) w.r.t. original

Cost-Reducing Rewrites

- ✦ Smaller document portion
- ✦ Cheaper model
- ✦ Simpler operation



`filter("Does the transcript contain a sentencing hearing?")`

Input: top-k chunks in document text

gpt-4o-mini

Revisiting Search

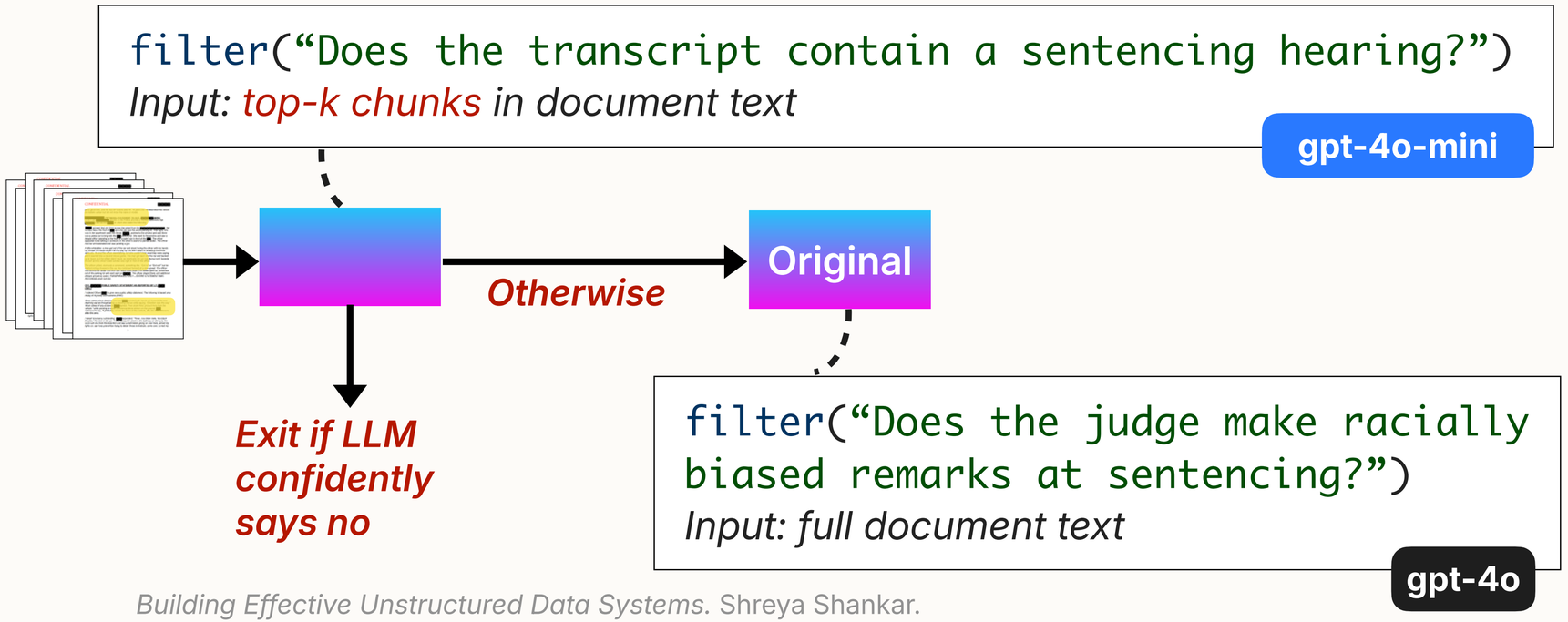
Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26.*

Simpler problem: reduce cost for a single map or filter operator

Approach: search over cost-reducing rewrites, ***maintaining target accuracy*** (e.g., 90%) w.r.t. original

Cost-Reducing Rewrites

- ✦ Smaller document portion
- ✦ Cheaper model
- ✦ Simpler operation



Revisiting Search: A *Task Cascade*

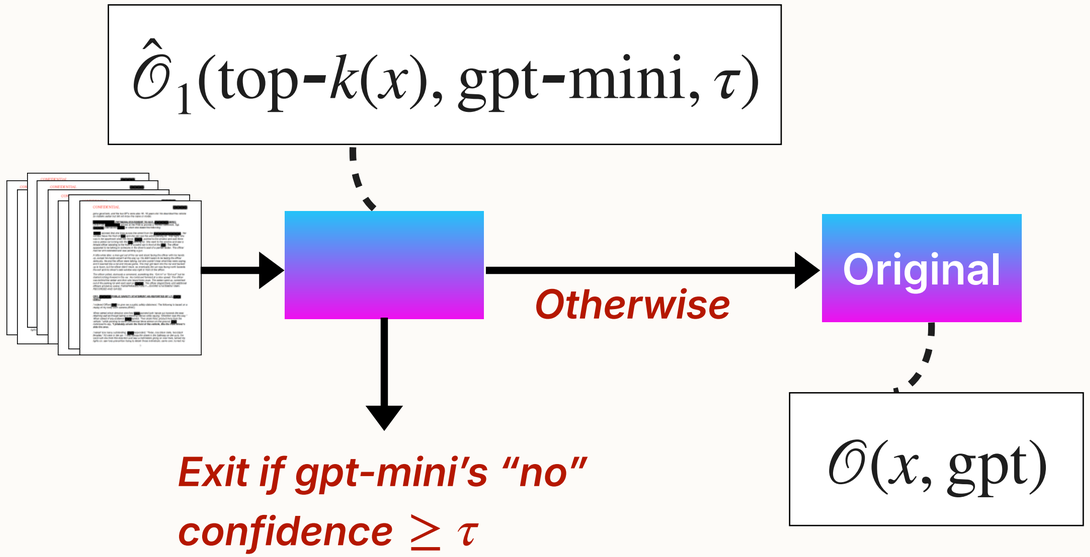
Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26.*

Simpler problem: reduce cost for a single map or filter operator

Approach: search over cost-reducing rewrites, ***maintaining target accuracy*** (e.g., 90%) w.r.t. original

Cost-Reducing Rewrites

- ✦ Smaller document portion
- ✦ Cheaper model
- ✦ Simpler operation



Revisiting Search: A *Task Cascade*

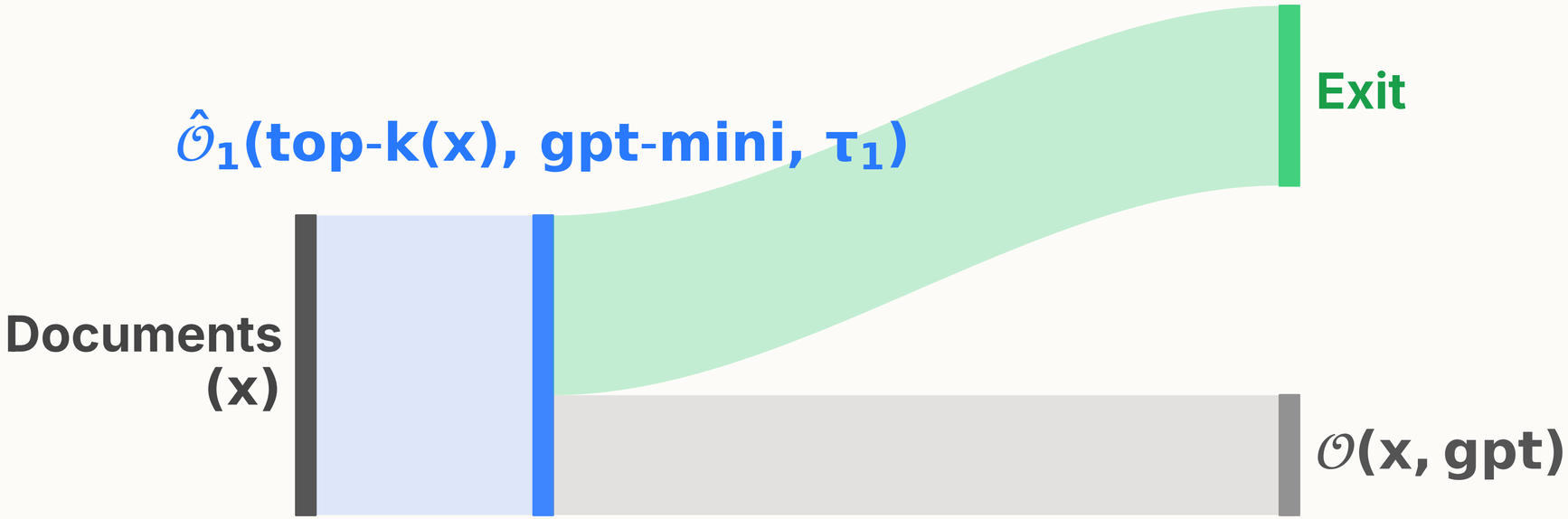
Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26.*

Simpler problem: reduce cost for a single map or filter operator

Approach: search over cost-reducing rewrites, ***maintaining target accuracy*** (e.g., 90%) w.r.t. original

Cost-Reducing Rewrites

- ✦ Smaller document portion
- ✦ Cheaper model
- ✦ Simpler operation



Revisiting Search: A *Task Cascade*

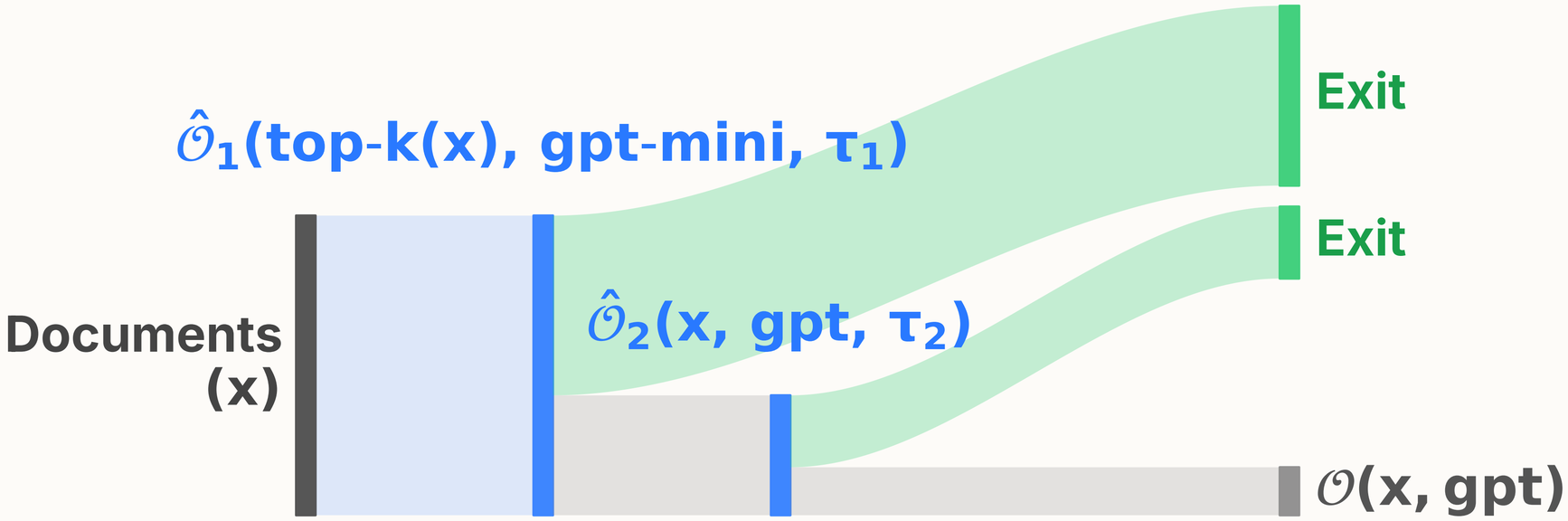
Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26.*

Simpler problem: reduce cost for a single map or filter operator

Approach: search over cost-reducing rewrites, ***maintaining target accuracy*** (e.g., 90%) w.r.t. original

Cost-Reducing Rewrites

- ✦ Smaller document portion
- ✦ Cheaper model
- ✦ Simpler operation



Revisiting Search: A *Task Cascade*

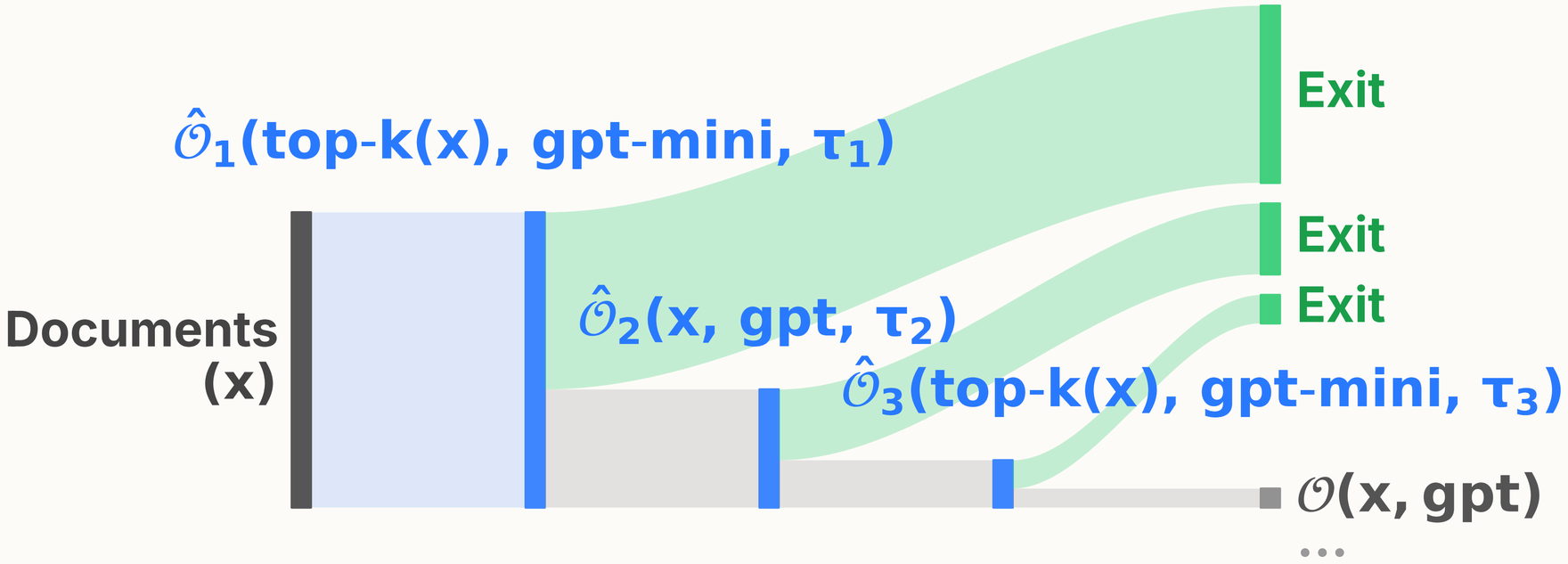
Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26.*

Simpler problem: reduce cost for a single map or filter operator

Approach: search over cost-reducing rewrites, ***maintaining target accuracy*** (e.g., 90%) w.r.t. original

Cost-Reducing Rewrites

- ✦ Smaller document portion
- ✦ Cheaper model
- ✦ Simpler operation

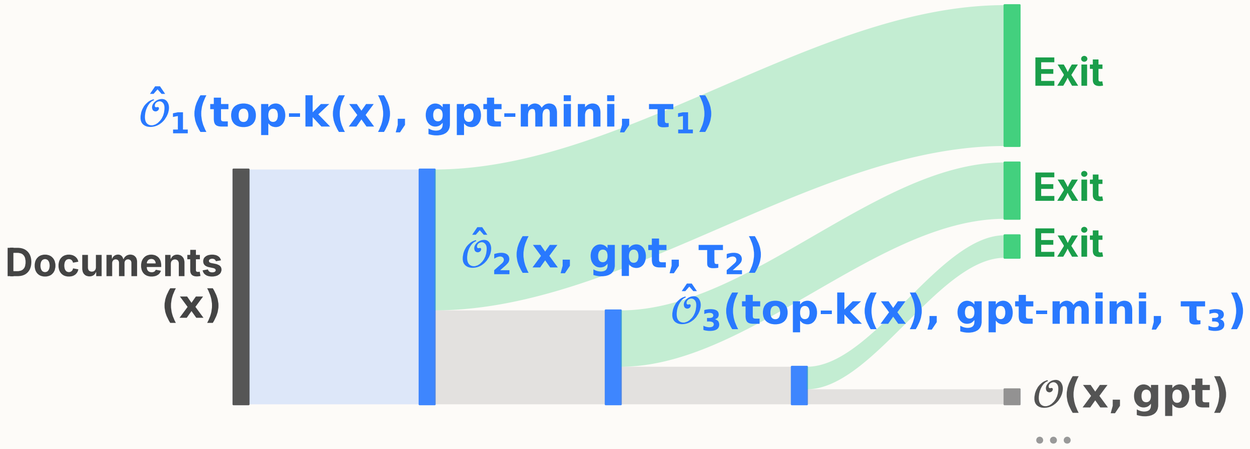


Revisiting Search: A *Task Cascade*

Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26.*

Components of a *Task*

- ✦ Operation (i.e., prompt)
- ✦ Document portion
- ✦ Model
- ✦ Confidence threshold(s) τ



Cost model: pay per token; 50—90% less for a cached token

Multiple challenges in constructing a min-cost cascade:

- ✦ How to generate the tasks
- ✦ How to select/order tasks for a cascade
- ✦ How to set τ to guarantee accuracy by construction

THEOREM 3.1. *Constructing an optimal task cascade is* NP-HARD.

How do we Set Confidence Thresholds?

Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26.*

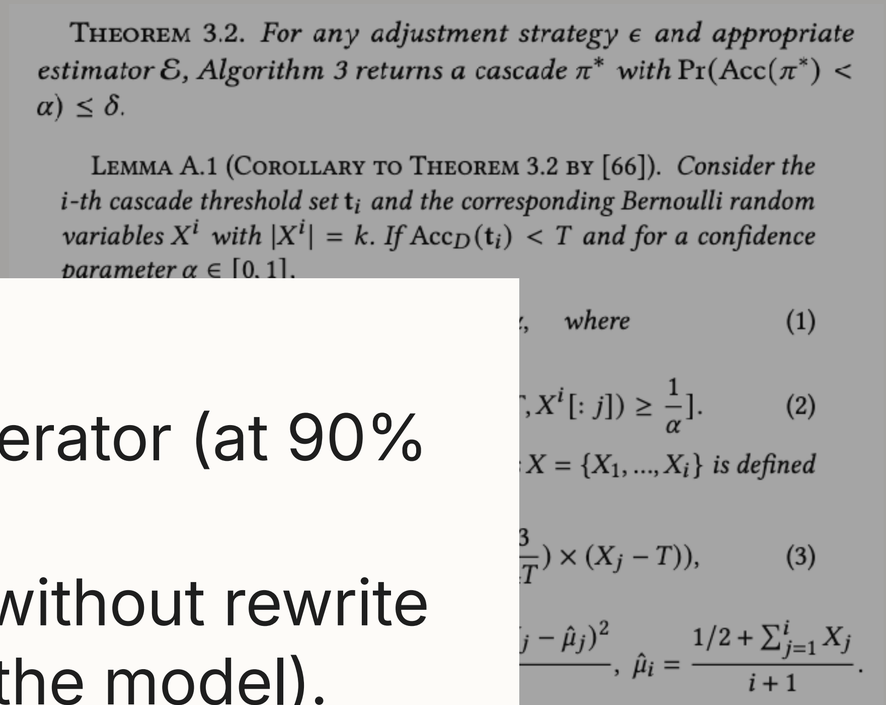
Goal: cheapest cascade with accuracy $\geq \alpha$ w.r.t. original

Approach:

1. Label a small sample with the original plan
2. Search over
   cheapest tha

Problem: Need
*Guaranteeing t
samples!*

Our solution: te
directly, with tighter bounds that exploit low variance.



THEOREM 3.2. *For any adjustment strategy ϵ and appropriate estimator $\mathcal{E}$, Algorithm 3 returns a cascade π^* with* $\Pr(\text{Acc}(\pi^*) < \alpha) \leq \delta$.

LEMMA A.1 (COROLLARY TO THEOREM 3.2 BY [66]). *Consider the i-th cascade threshold set $\mathbf{t}_i$ and the corresponding Bernoulli random variables X^i with $|X^i| = k$. If $\text{Acc}_D(\mathbf{t}_i) < T$ and for a confidence parameter $\alpha \in [0, 1]$,*

where (1)

$[: j]) \geq \frac{1}{\alpha}]$. (2)

$X = \{X_1, \ldots, X_i\}$ *is defined*

$\frac{3}{T}) \times (X_j - T))$, (3)

$\hat{\mu}_j)^2$, $\hat{\mu}_i = \frac{1/2 + \sum_{j=1}^{i} X_j}{i+1}$.

Results:

- ***86% cheaper*** than original operator (at 90% target acc).
- ***49% cheaper*** than baselines without rewrite directives (i.e., only changing the model).

Revisiting Search

Multi-Objective Agentic Rewrites for Unstructured Data Processing. Wei*, **Shankar*** et al. *VLDB '26*.

What about searching for low-cost *and* high-accuracy plans?

Our approach (won't go in detail):

- ✦ MCTS-style search (node = plan; edge = rewrite)
- ✦ UCB selects which nodes to rewrite, based on sample accuracies & costs
- ✦ LLM agents instantiate rewrite directives

Ground truth: user-labeled samples and user-defined accuracy function

DocETL Outperforms Baselines

Baselines

- LOTUS[1] and Palimpzest (PZ)[2]: Open-source systems. No rewrite directives.
- Simple Agent (SA): GPT-5 agent with the DocETL docs + engine as "tools"; tries pipelines until context exhausted. No search algorithm.

Can use 11 models (8 GPT; 3 Gemini).

	LOTUS	PZ	SA
Accuracy Gain	3.1×	1.27×	1.94×
Cost Savings	-51%	-46%	-56%

Across 6 workloads:

- **We get 1.27–3.1× higher accuracy!**
- **We get 46–56% lower cost!**

[1]*Patel et al. Semantic Operators and Their Optimization: Enabling LLM-Based Data Processing with Accuracy Guarantees in LOTUS. VLDB 2025.*

[2]*Liu et al. Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing. CIDR 2025. and Russo et al. ABACUS: A Cost-Based Optimizer for Semantic Operator Systems. VLDB 2026.*

DocETL Finds Nontrivial Plans

2.3x more operators than baseline plans

9 different rewrite directives across top-accuracy plans

55% of top-5 plans include agent-synthesized code

✦ Medical example: ~75% less context, same accuracy

```
error_indicators = [
    r'(?i)diagnos[ie]d?',
    r'(?i)treated|treatment',
    r'(?i)prescribed|prescription',
    r'(?i)cause[ds]?|caused by|due to',
    r'(?i)laboratory|labs?|test|exam|finding'
]

for idx, sentence in enumerate(sentences):
    for pattern in error_indicators:
        if re.search(pattern, sentence):
            relevant_indices.add(idx)
            # Include ±1 sentence for context
            extend_context(idx)
```

Today's Talk

2. **Inner loop (query optimization)**
   - ✦ *What is the space of query plans?*
   - ✦ *How to search the plan space?*

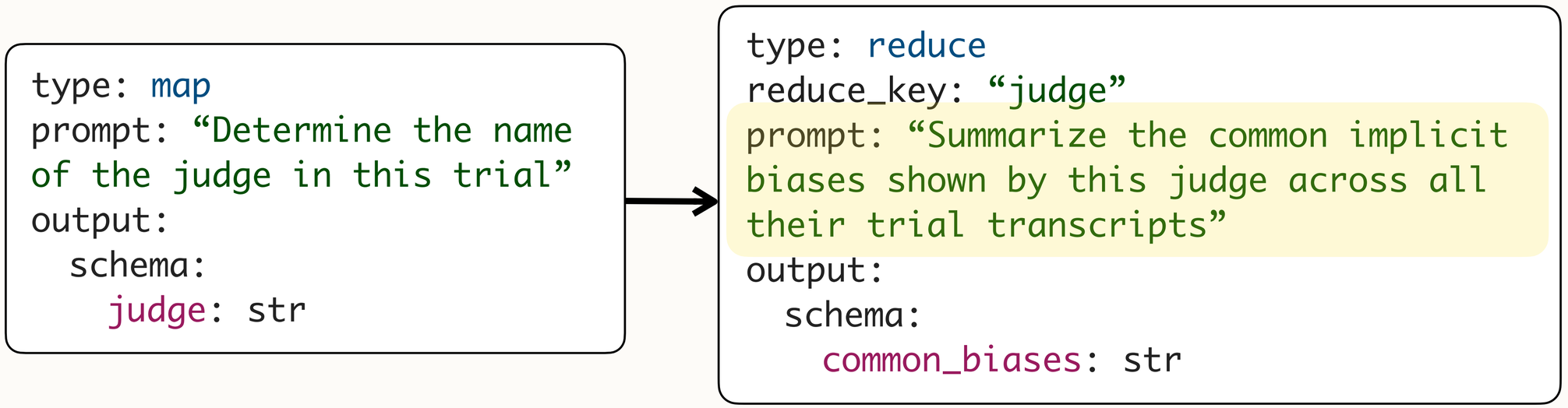
*Insight: must explore semantically equivalent plans (**rewrite directives**)*

Today's Talk

3. **Outer loop (iterating on queries)**
   - ✦ *How should users author queries?*
   - ✦ *How can users validate results at scale?*

Motivation: Outer Loop

Users struggle to write good queries!



Research question: What tools do users need to effectively steer LLM-powered unstructured data analysis?

DocWrangler

Steering Semantic Data Processing with DocWrangler. **Shankar***, Chopra* et al. *UIST '25.* 🏆

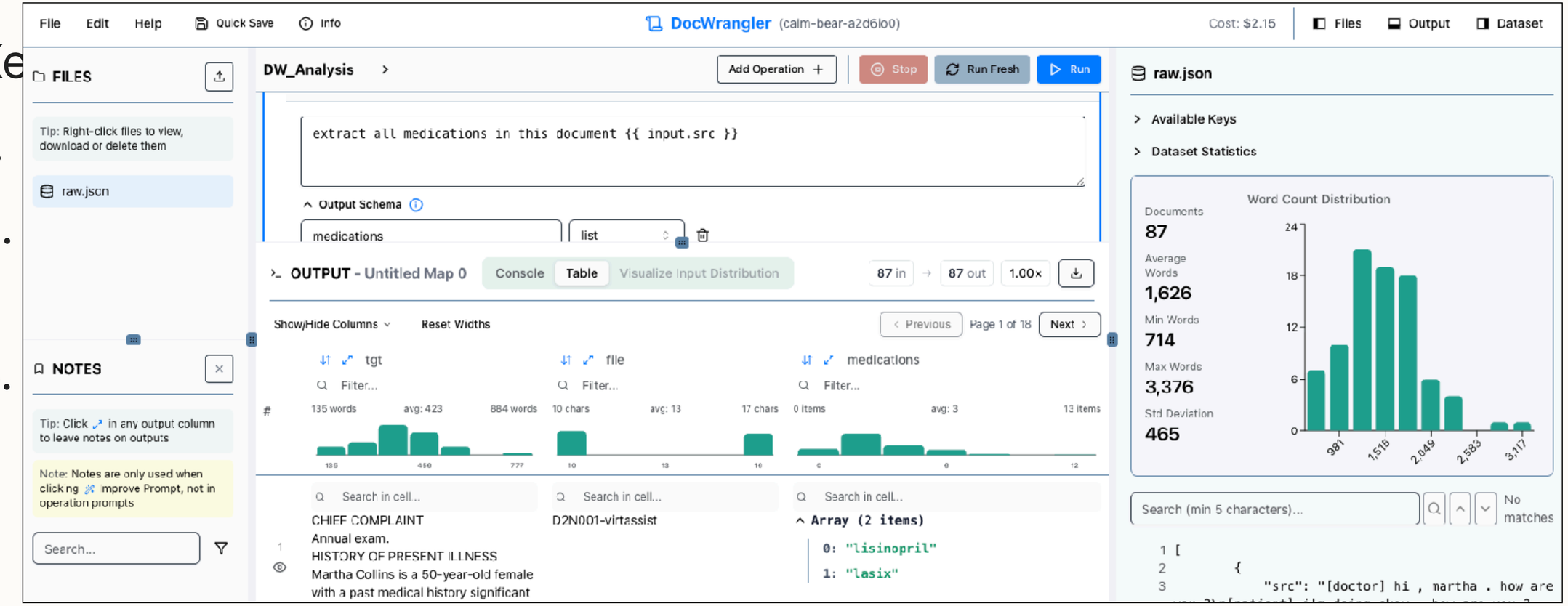
A mixed-initiative IDE for DocETL.

Ke

1.

2.

3.



File Edit Help Quick Save Info DocWrangler (calm-bear-a2d6io0s) Cost: $0.01 Files Output Dataset

FILES

Tip: Right-click files to view, download or delete them

raw.json

NOTES

Tip: Click in any output column to leave notes on outputs

Note: Notes are only used when clicking Improve Prompt, not in operation prompts

Search...

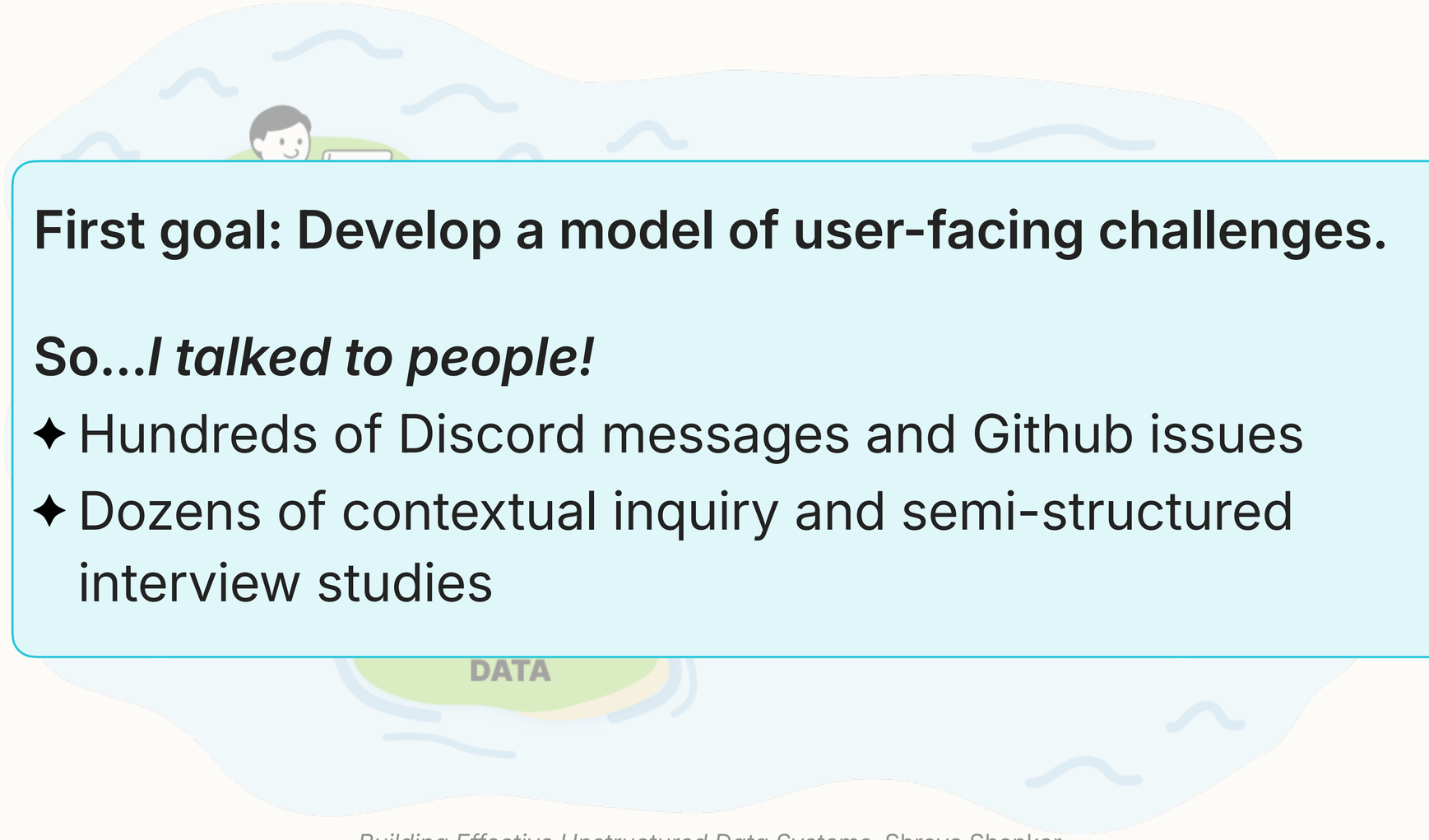
Untitled_Analysis Overview System Prompts 15 Add Operation Stop Run Fresh Run

+ Add Operation

Speed > 20x

Key Idea: Three Gulfs Framework

Steering Semantic Data Processing with DocWrangler. **Shankar***, Chopra* et al. *UIST '25.* 🏆



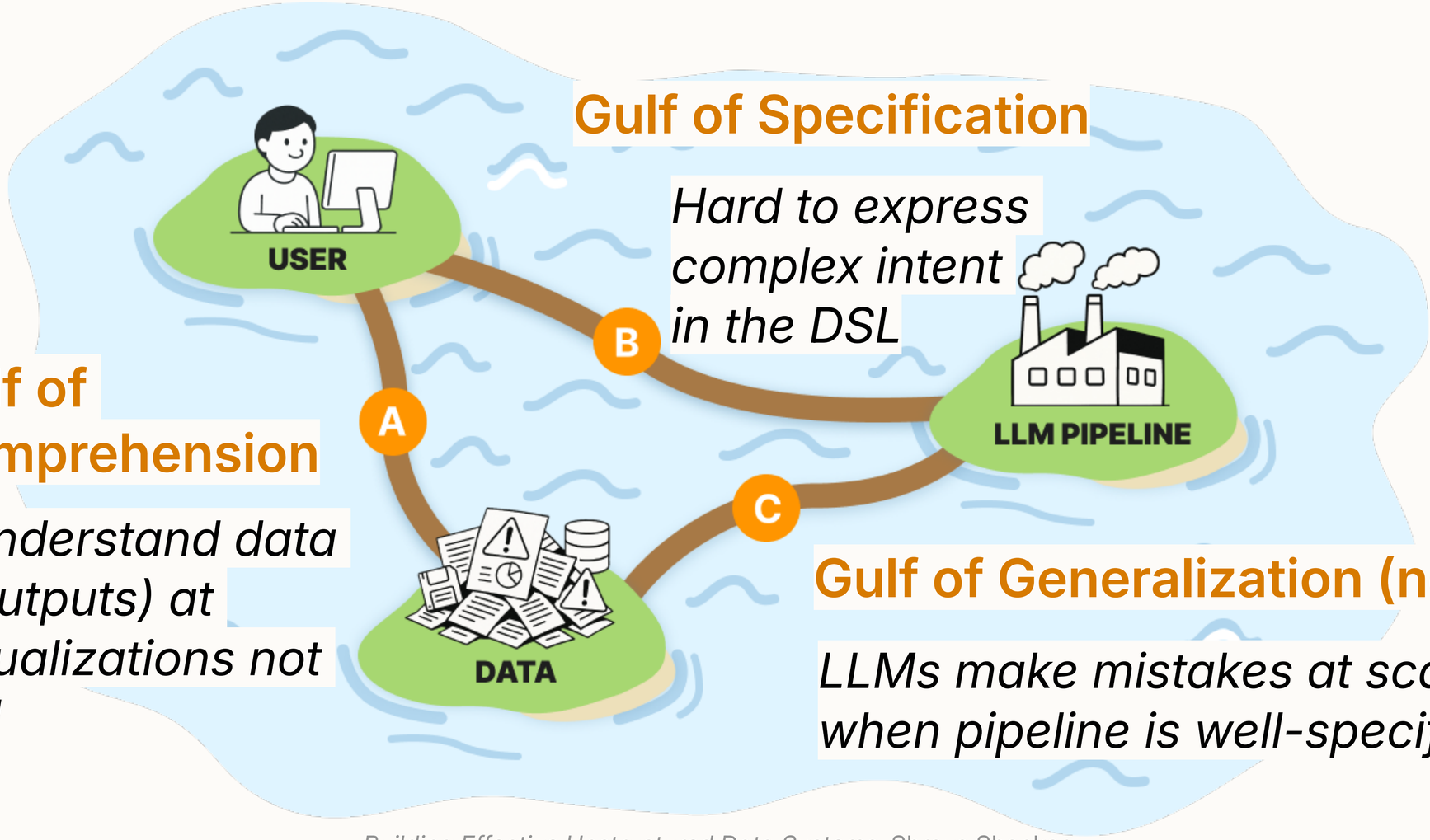
First goal: Develop a model of user-facing challenges.

So...*I talked to people!*

- ✦ Hundreds of Discord messages and Github issues
- ✦ Dozens of contextual inquiry and semi-structured interview studies

Key Idea: Three Gulfs Framework

Steering Semantic Data Processing with DocWrangler. **Shankar***, Chopra* et al. *UIST '25.* 🏆



Gulf of Specification

Hard to express complex intent in the DSL

Gulf of Comprehension

Hard to understand data (or LLM outputs) at scale; visualizations not as helpful

Gulf of Generalization (new!)

LLMs make mistakes at scale, even when pipeline is well-specified

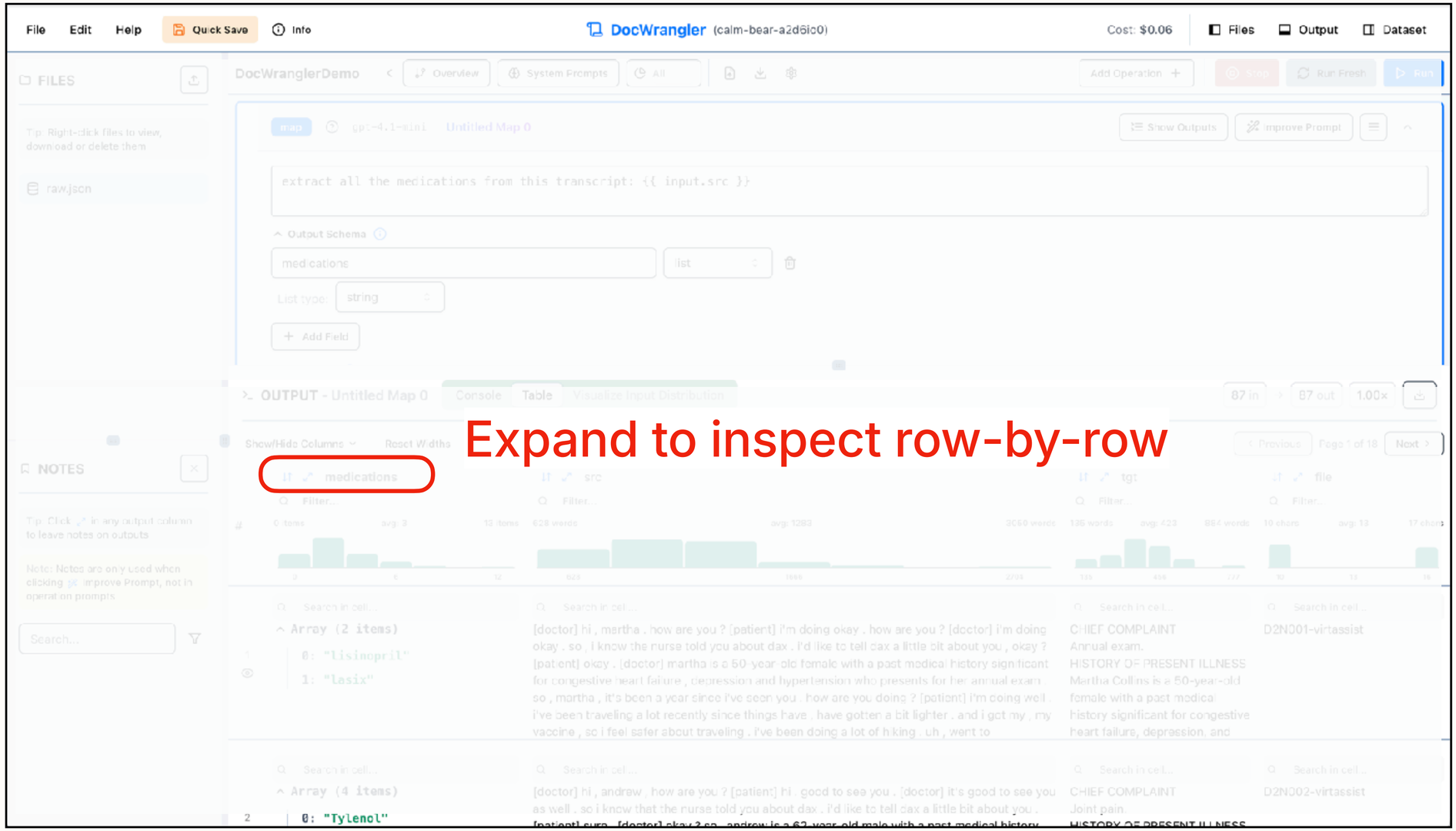
Illustrating the Gulf of Comprehension

Task

"Extract all 💊 from each transcript"

Challenge

Discovering what you really want & the long tail of LLM behaviors



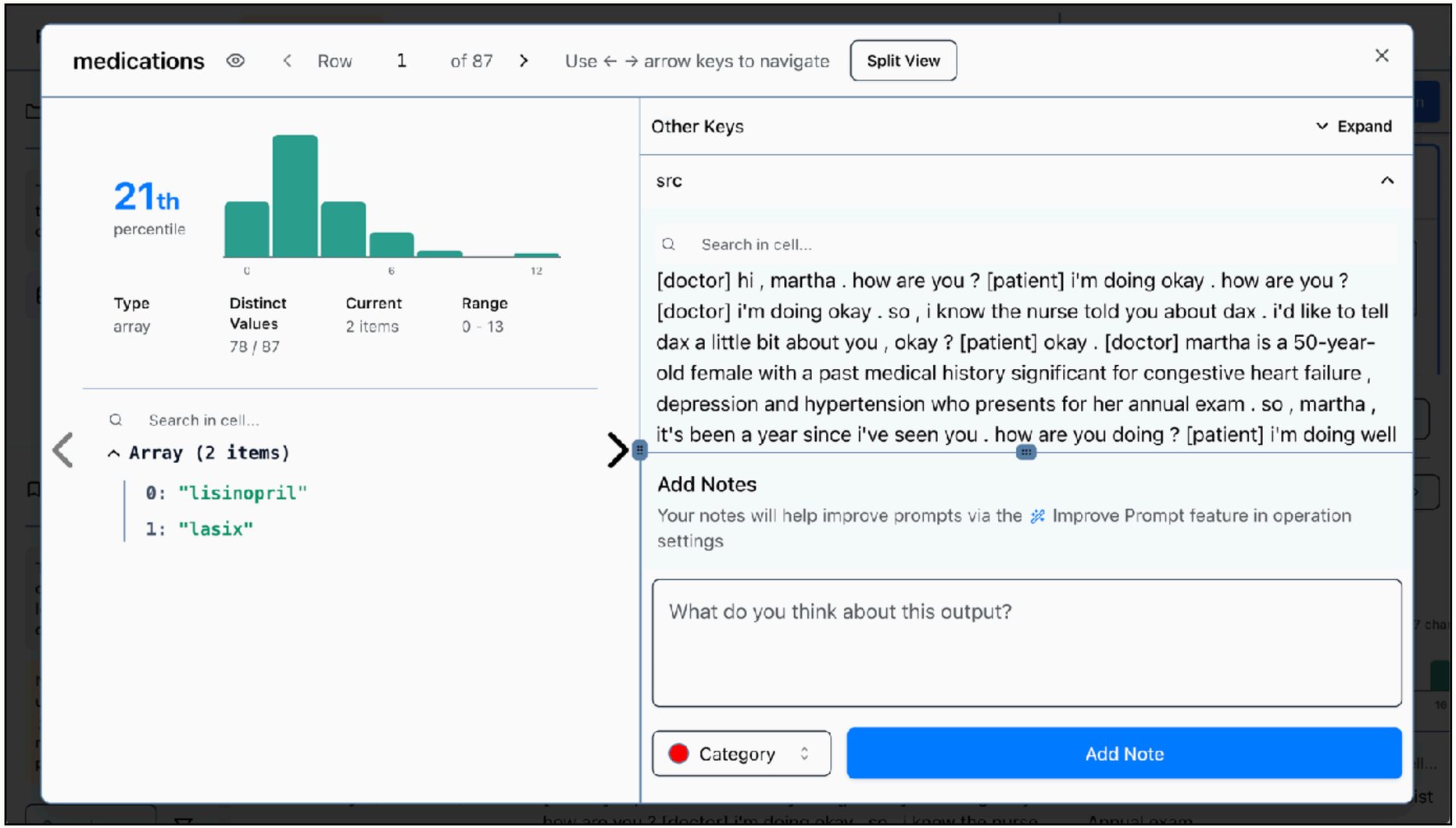
In-Situ Notetaking to *Comprehend* Data Better

Task

"Extract all 💊 from each transcript"

Challenge

Discovering what you really want & the long tail of LLM behaviors



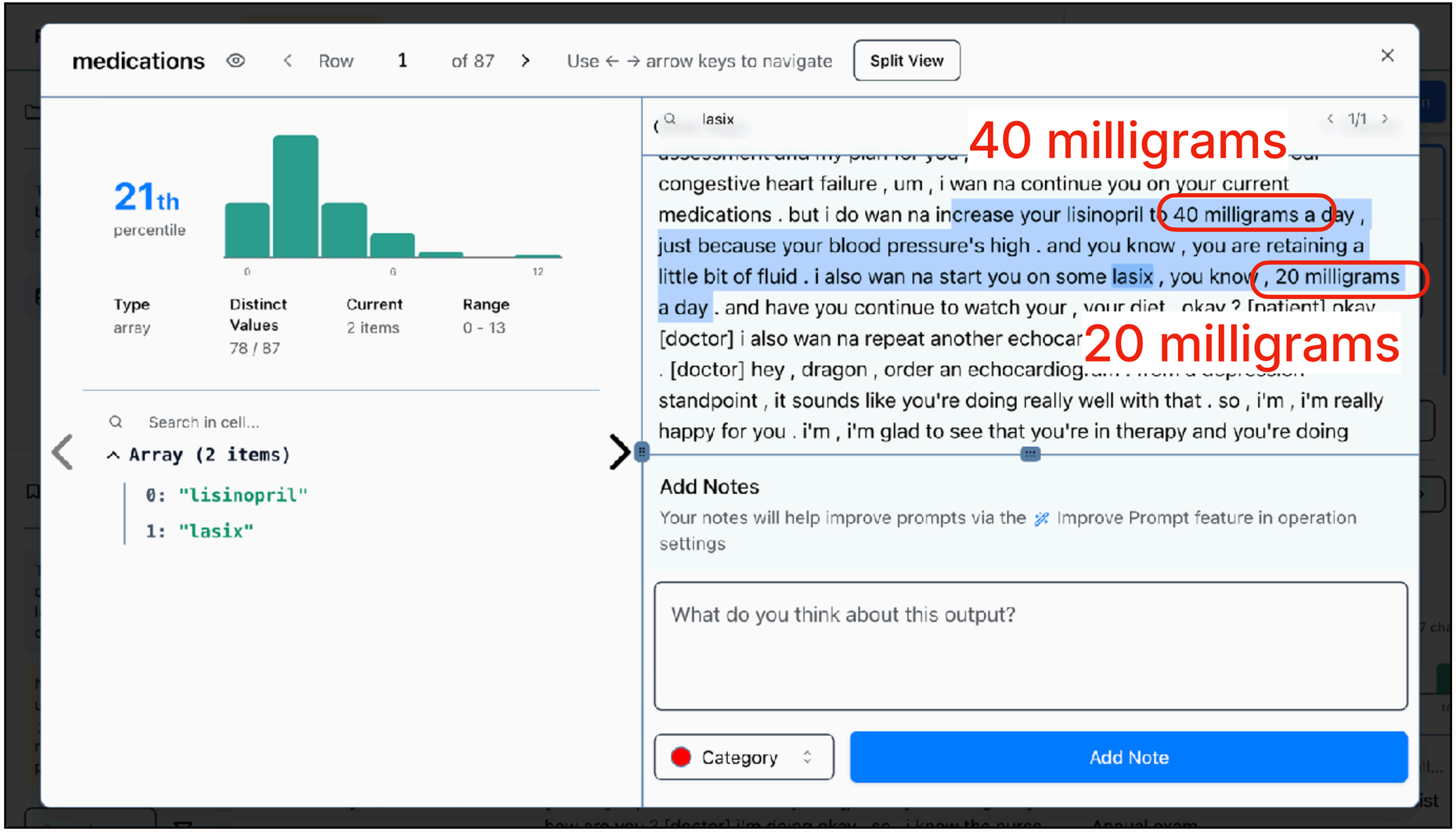
In-Situ Notetaking to *Comprehend* Data Better

Task

"Extract all 💊 from each transcript"

Challenge

Discovering what you really want & the long tail of LLM behaviors



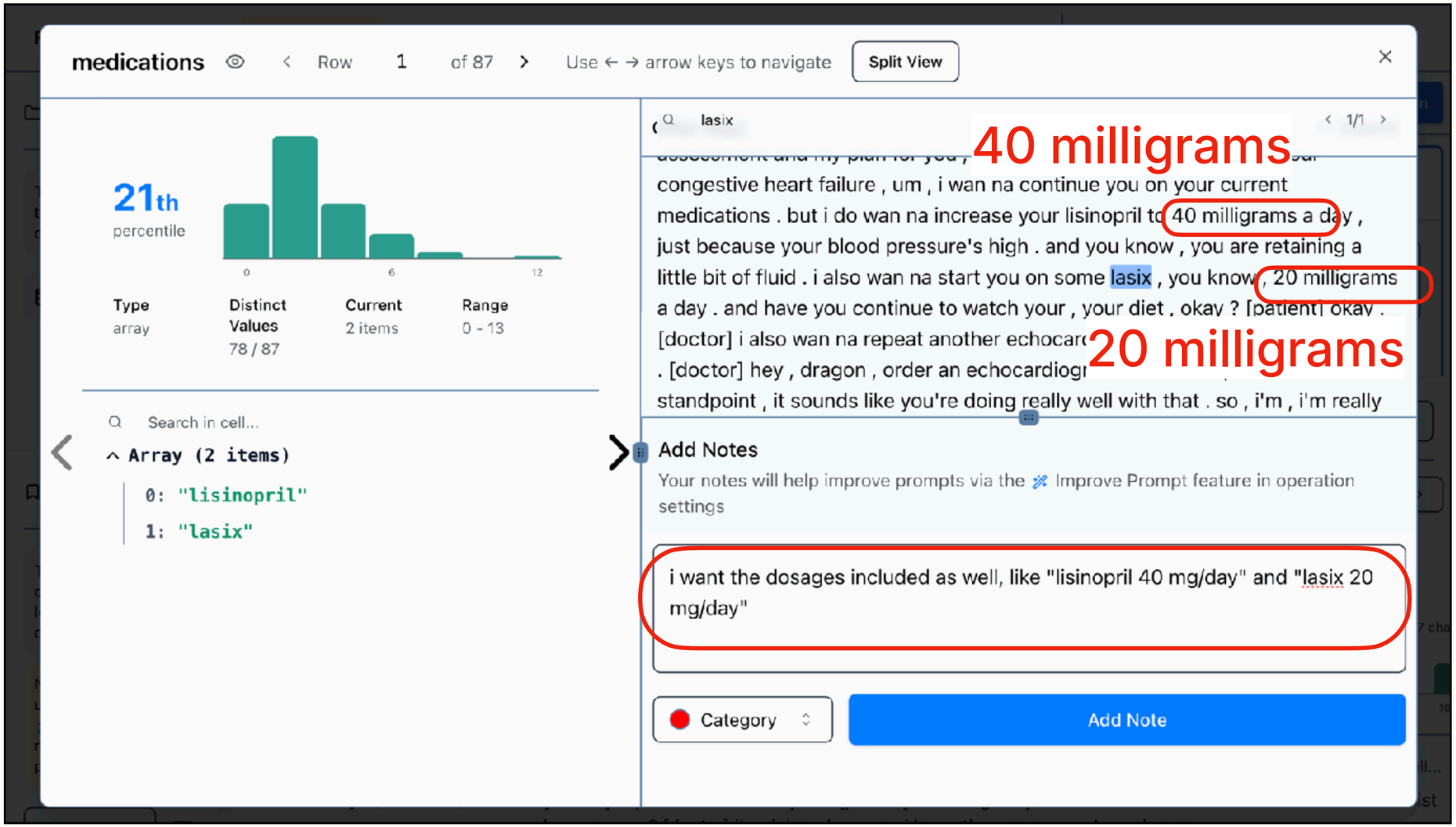
In-Situ Notetaking to *Comprehend* Data Better

Task

"Extract all 💊 from each transcript"

Challenge

Discovering what you really want & the long tail of LLM behaviors



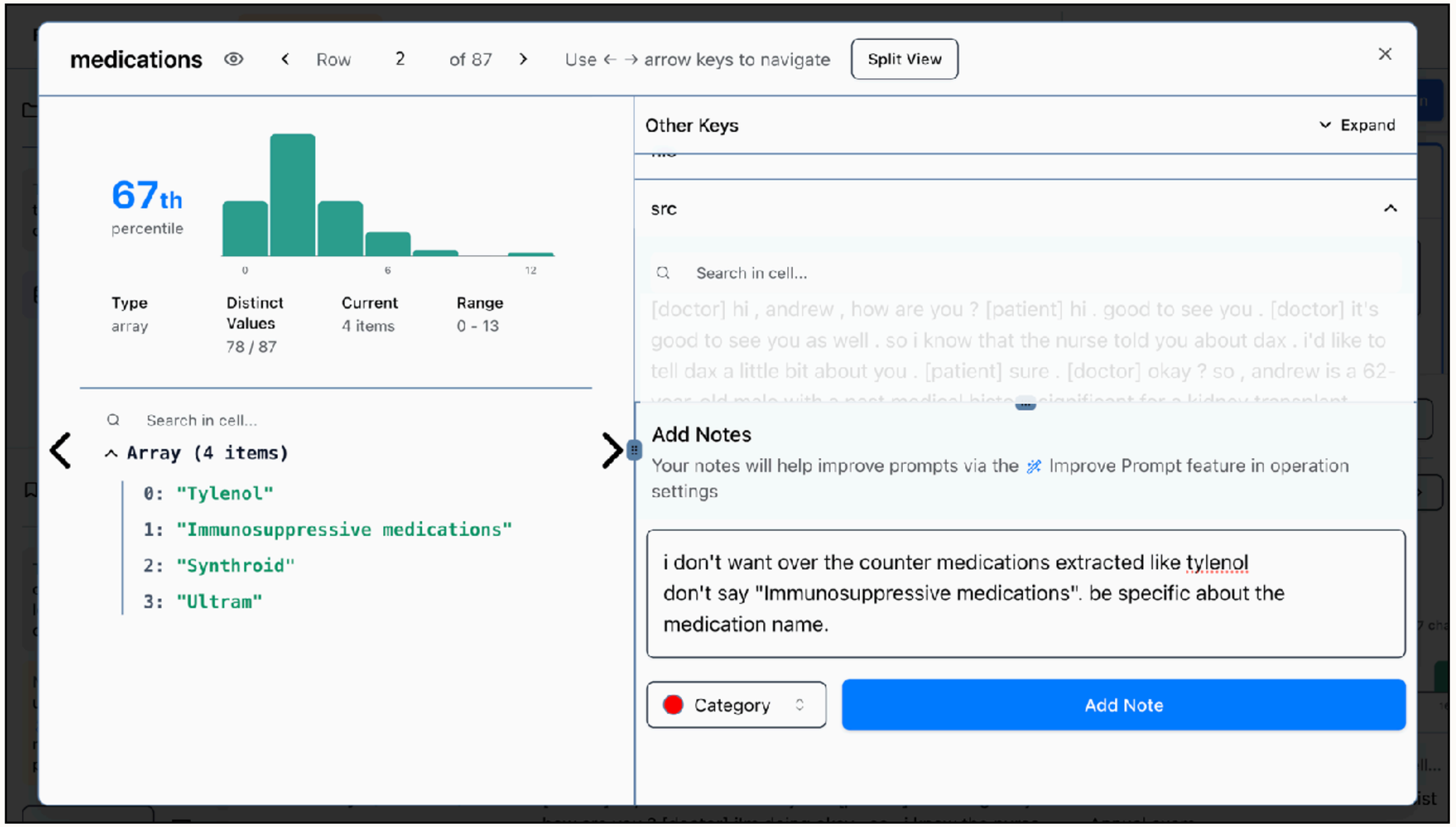
In-Situ Notetaking to *Comprehend* Data Better

Task

"Extract all 💊 from each transcript"

Challenge

Discovering what you really want & the long tail of LLM behaviors



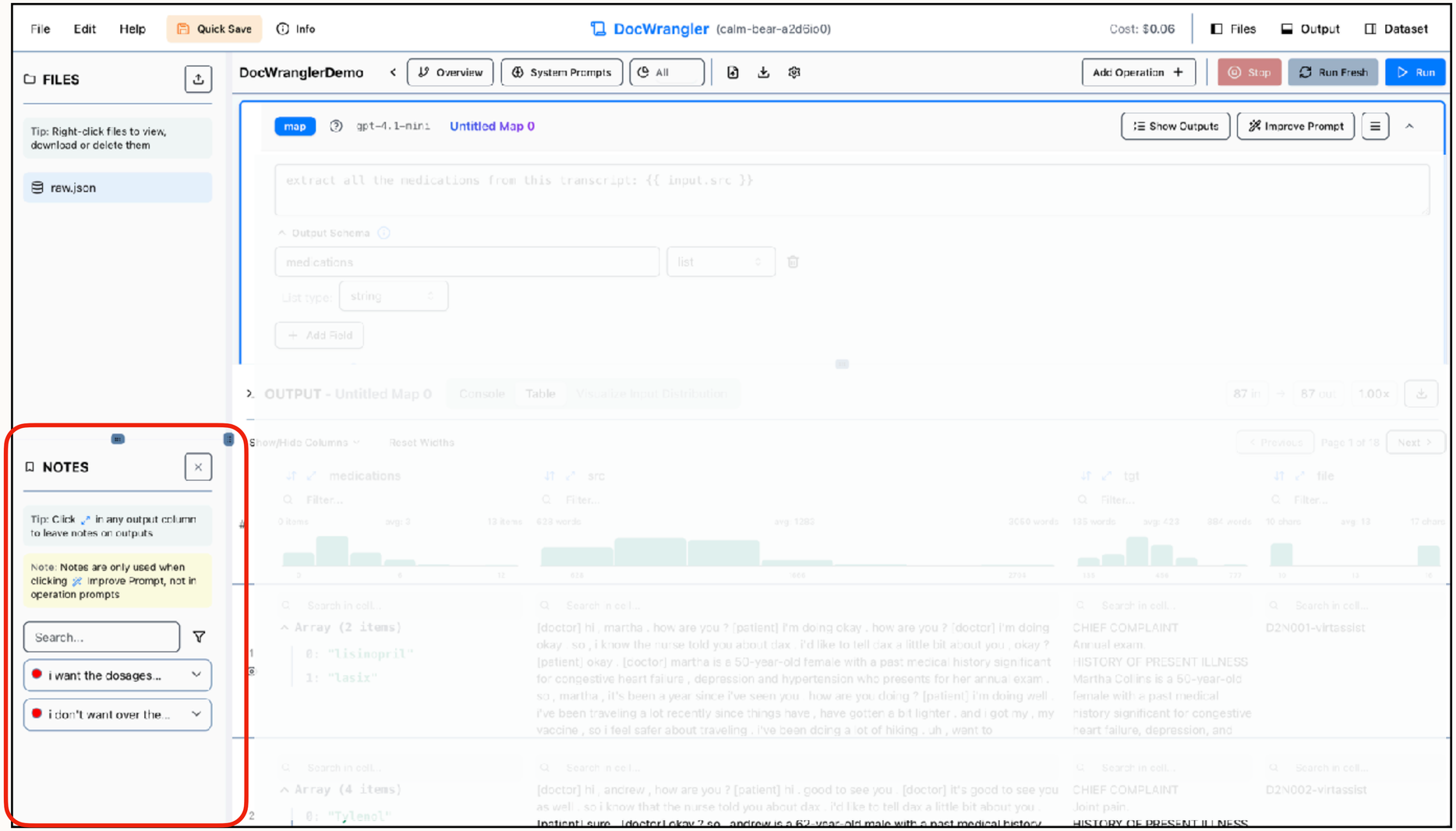
In-Situ Notetaking to *Comprehend* Data Better

Task

"Extract all 💊 from each transcript"

Challenge

Discovering what you really want & the long tail of LLM behaviors



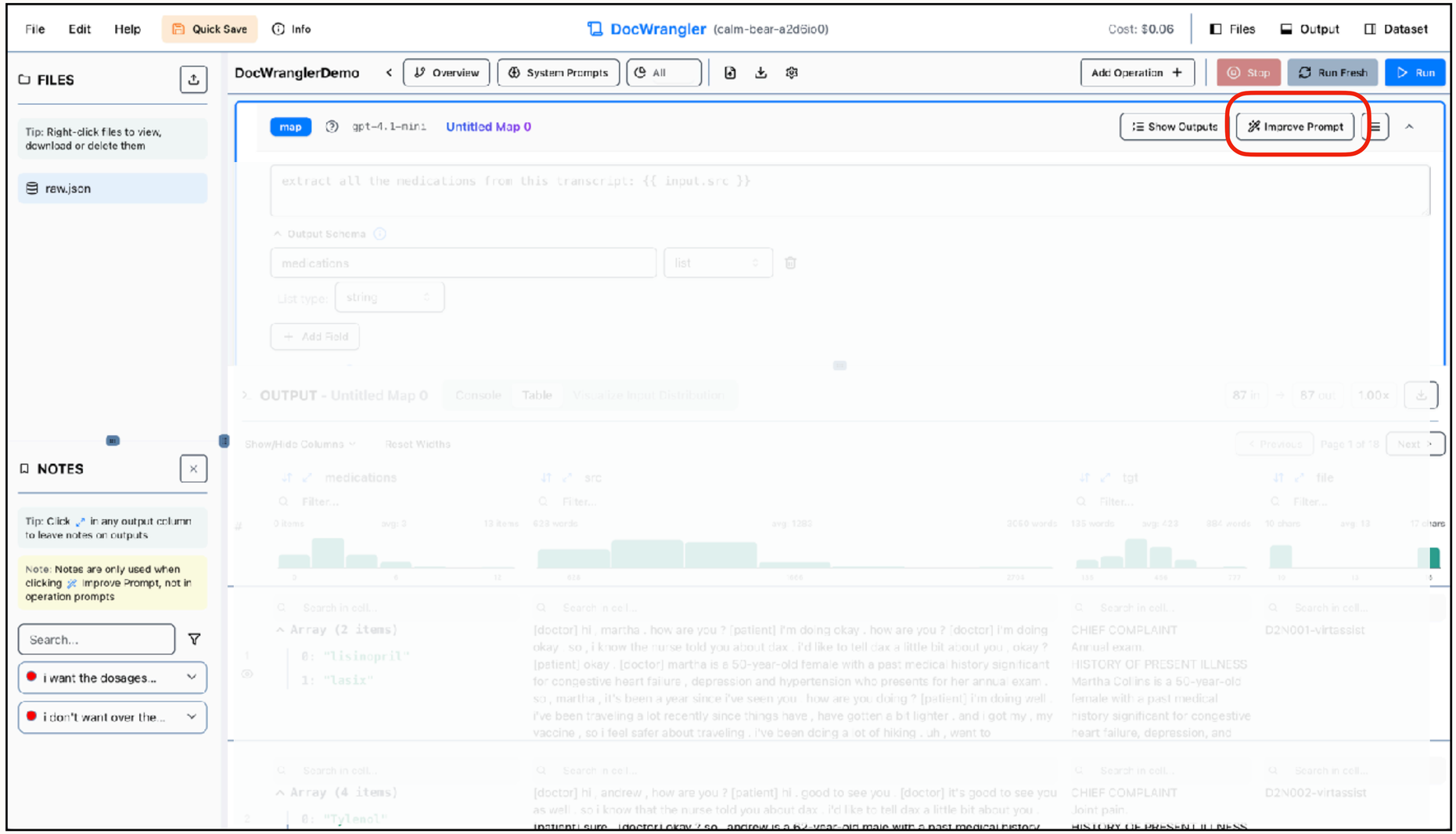
Illustrating the Gulf of Specification

Task

"Extract all 💊 from each transcript"

Challenge

Turning your notes and requirements into better prompts



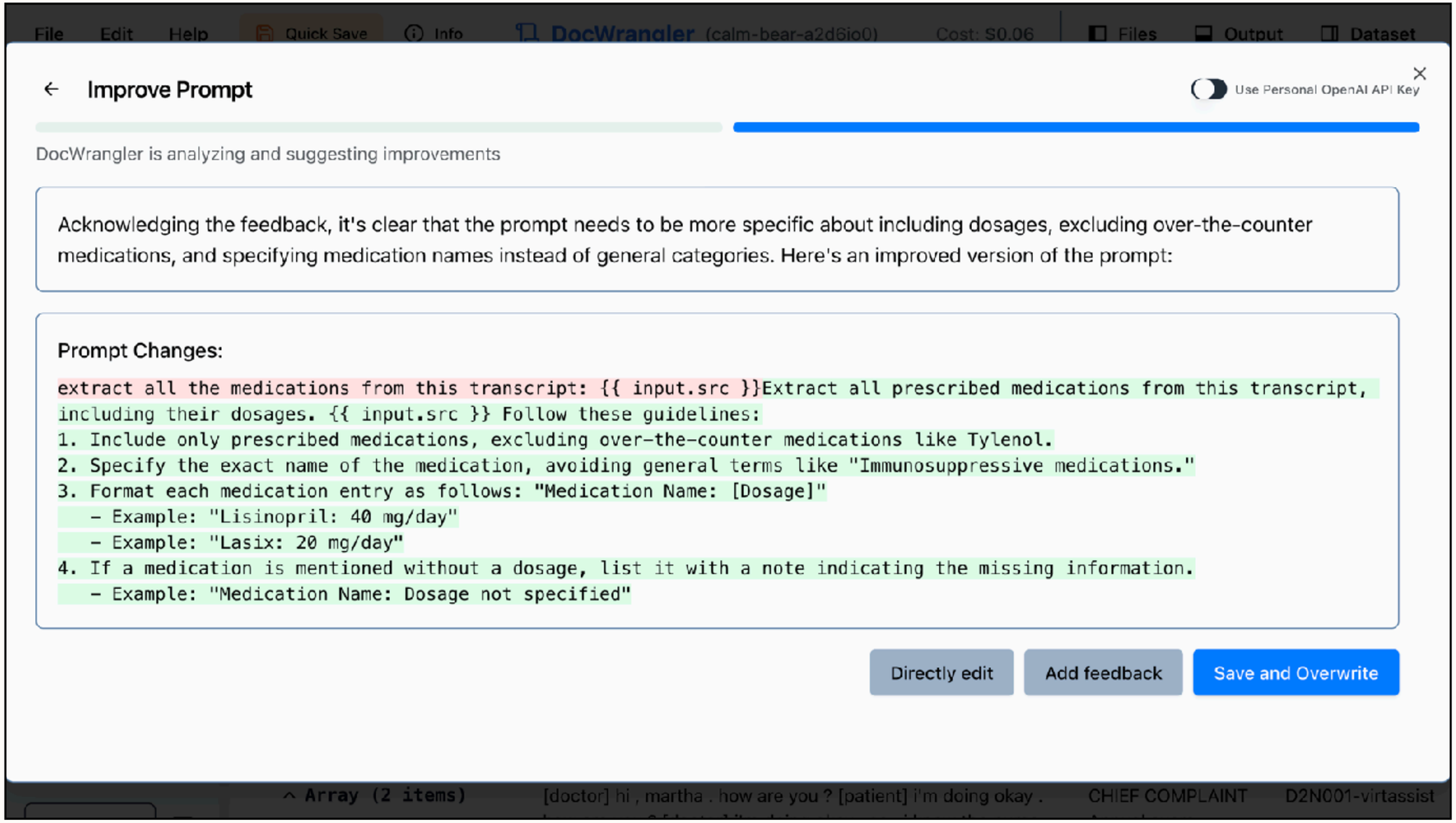
Interactive Assistance in *Specifying* Operations

Task

"Extract all 💊 from each transcript"

Challenge

Turning your notes and requirements into better prompts



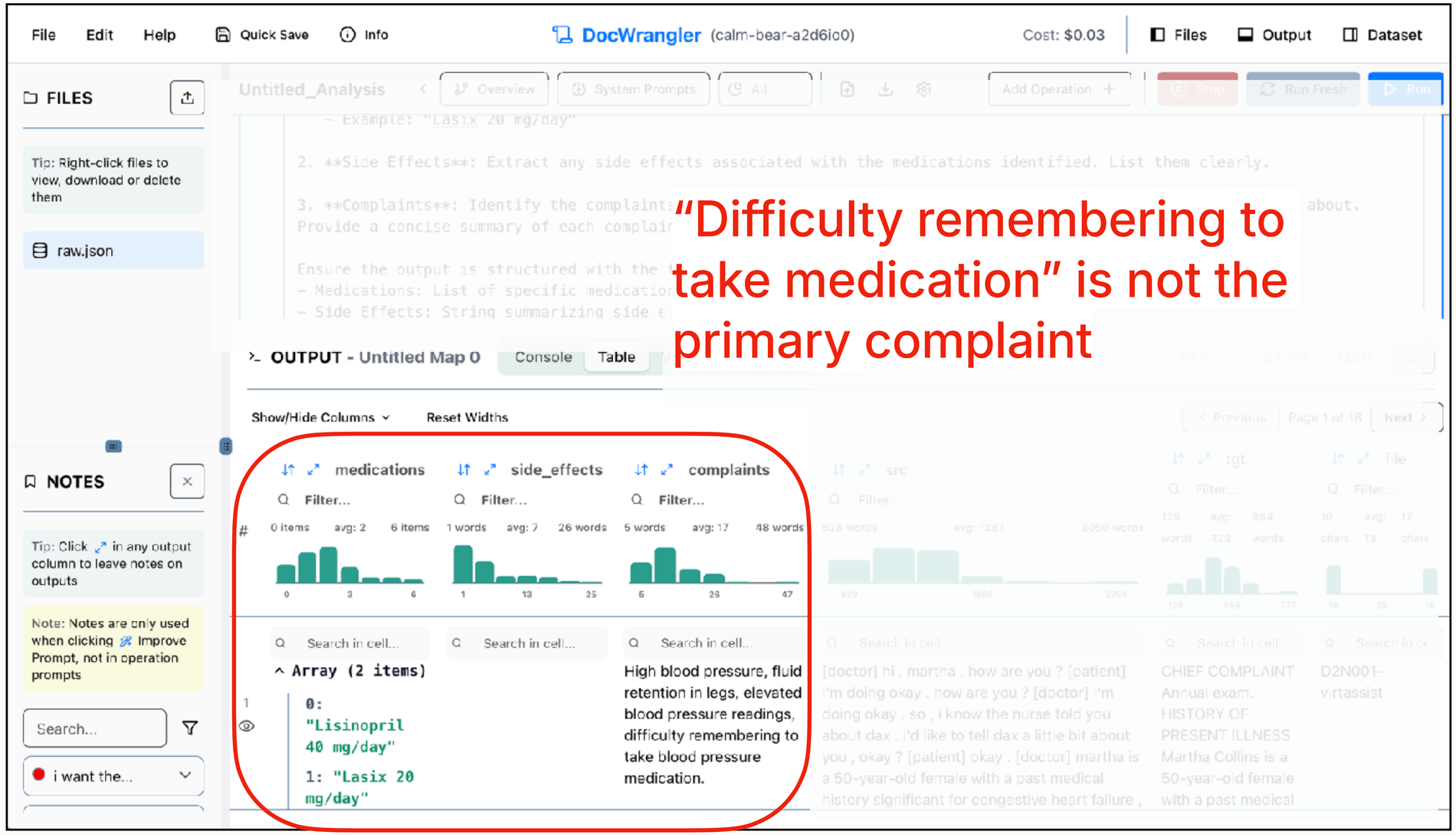
Illustrating the Gulf of Generalization

Task

"Extract all 💊, side effects, and patient complaints from each transcript"

Challenge

Too complex for a single LLM call



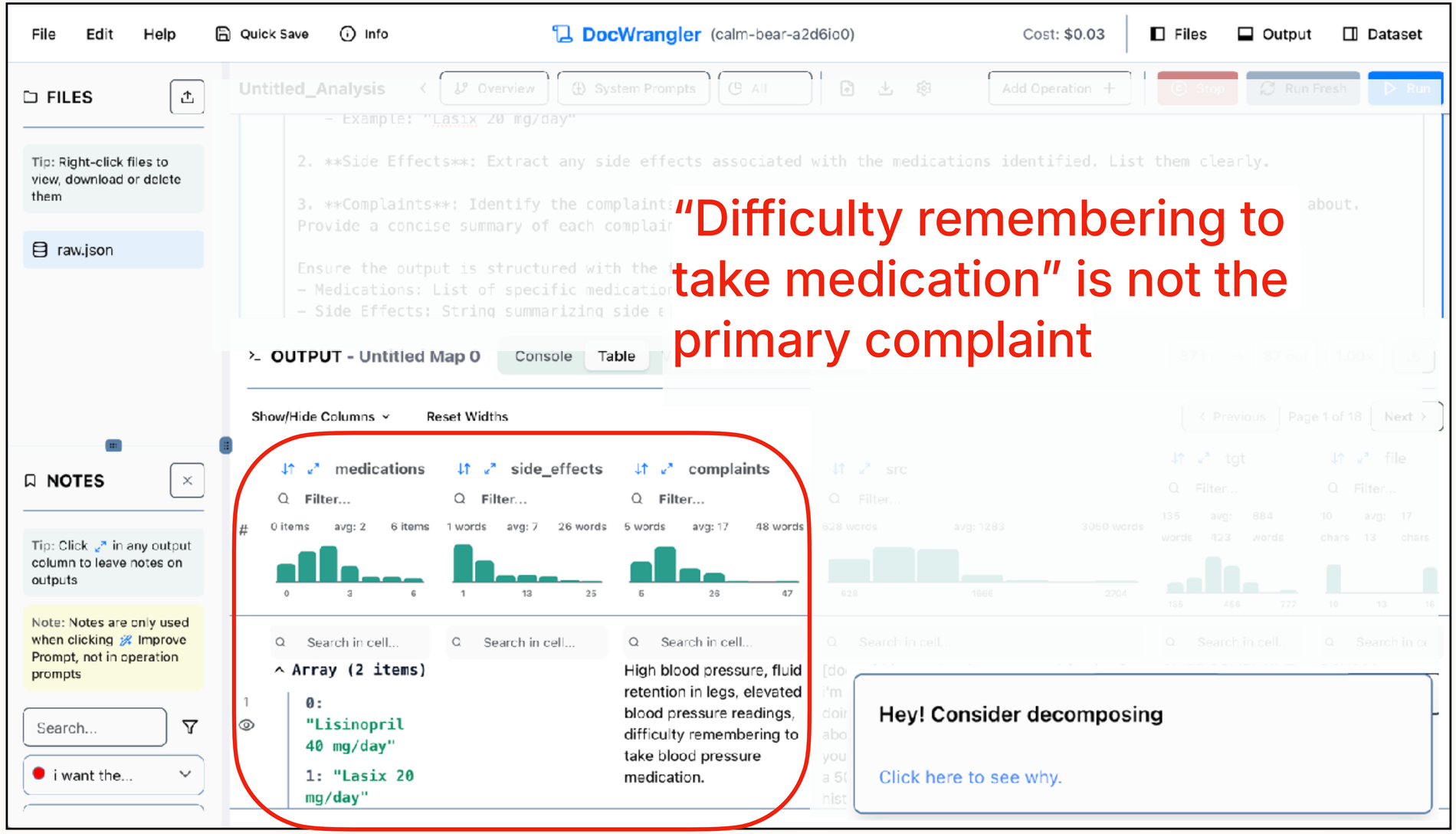
Automatic Decomposition to Help *Generalize*

Task

"Extract all 💊, side effects, and patient complaints from each transcript"

Challenge

Too complex for a single LLM call



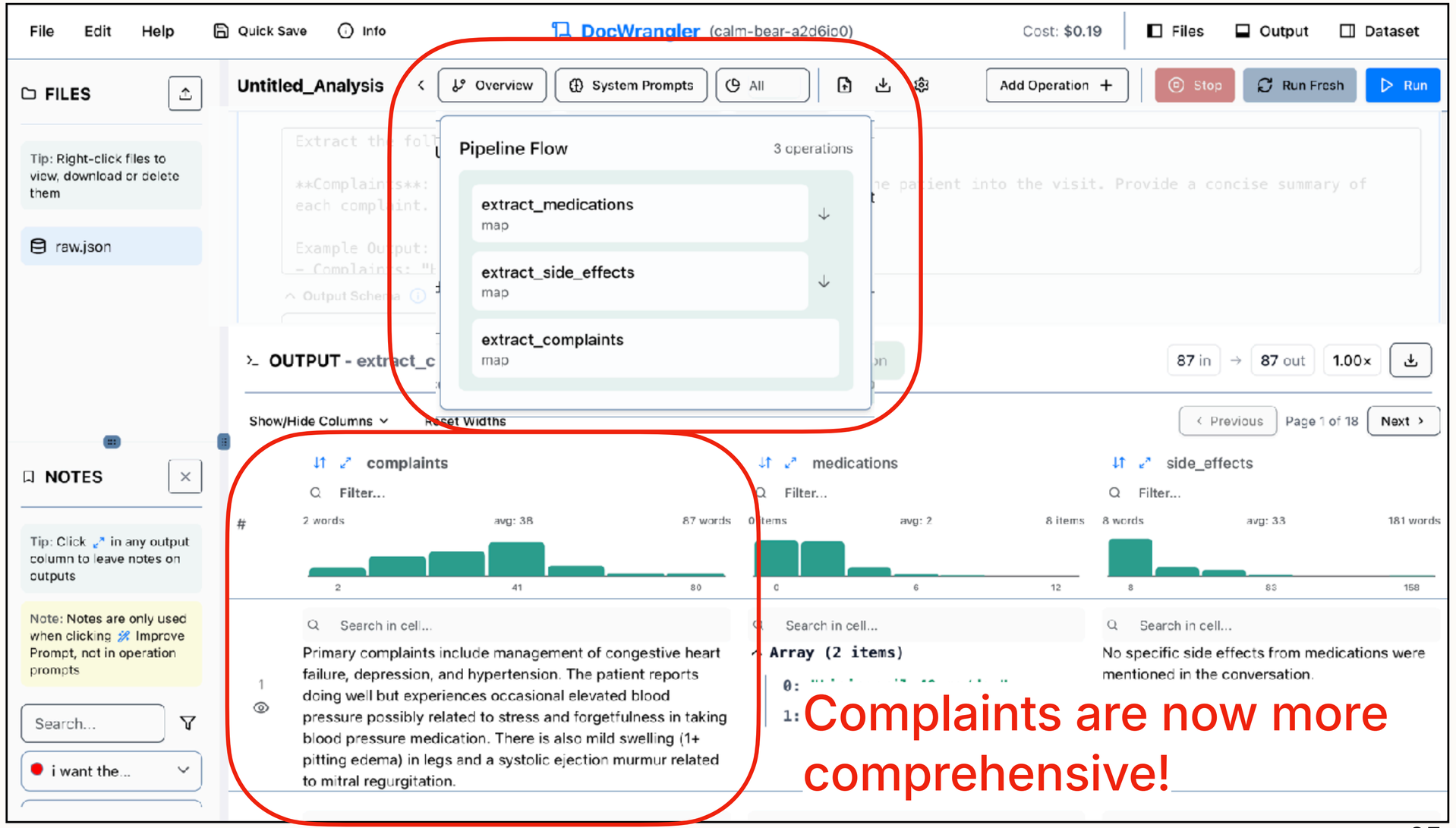
Automatic Decomposition to Help *Generalize*

Task

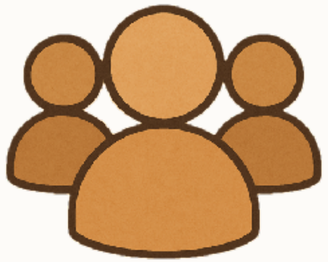
"Extract all 💊, side effects, and patient complaints from each transcript"

Challenge

Too complex for a single LLM call



DocWrangler User Study: Design



10 participant think-aloud lab study

- ✦ 60-90 minute sessions
- ✦ 3-5 analysis tasks per session

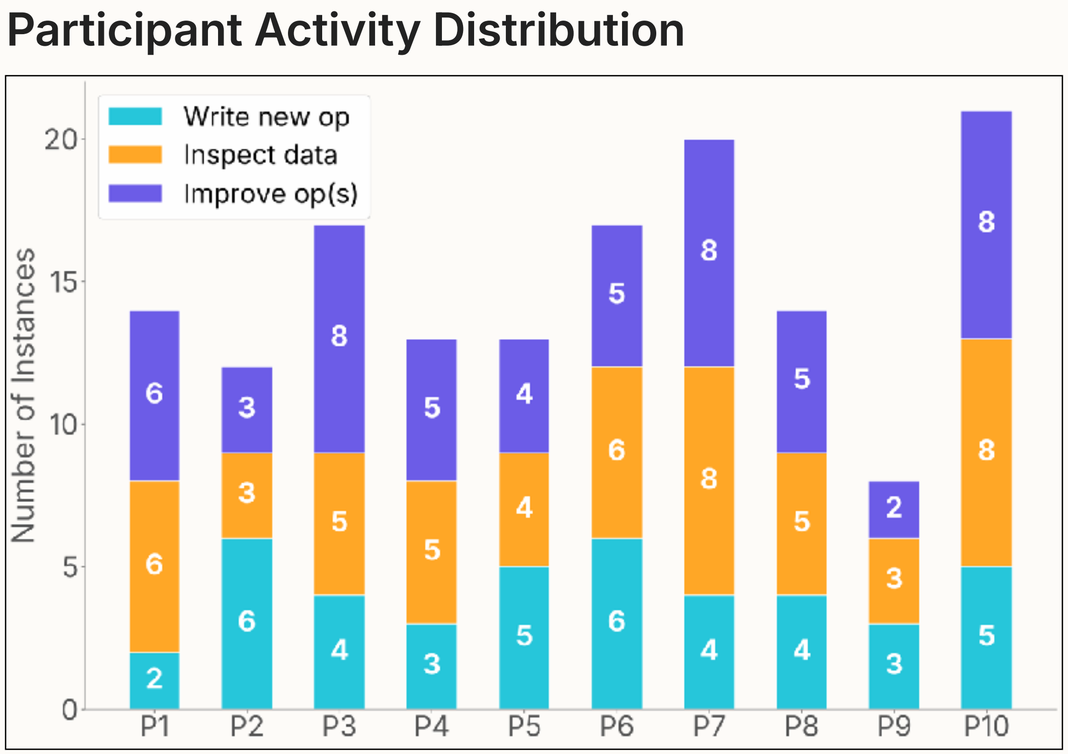
Online deployment

- ✦ BYO API keys
- ✦ ~4,500 pipelines; ~30GB of data (~5 billion words!)
- ✦ 30+ S&P 500 industries represented

Quantitative and qualitative analysis

What Did We Learn?

1. Users iterate *a lot* to get one query



What Did We Learn?

1. Users iterate *a lot* to get one query
2. Users don’t read raw documents, even when we make it easy

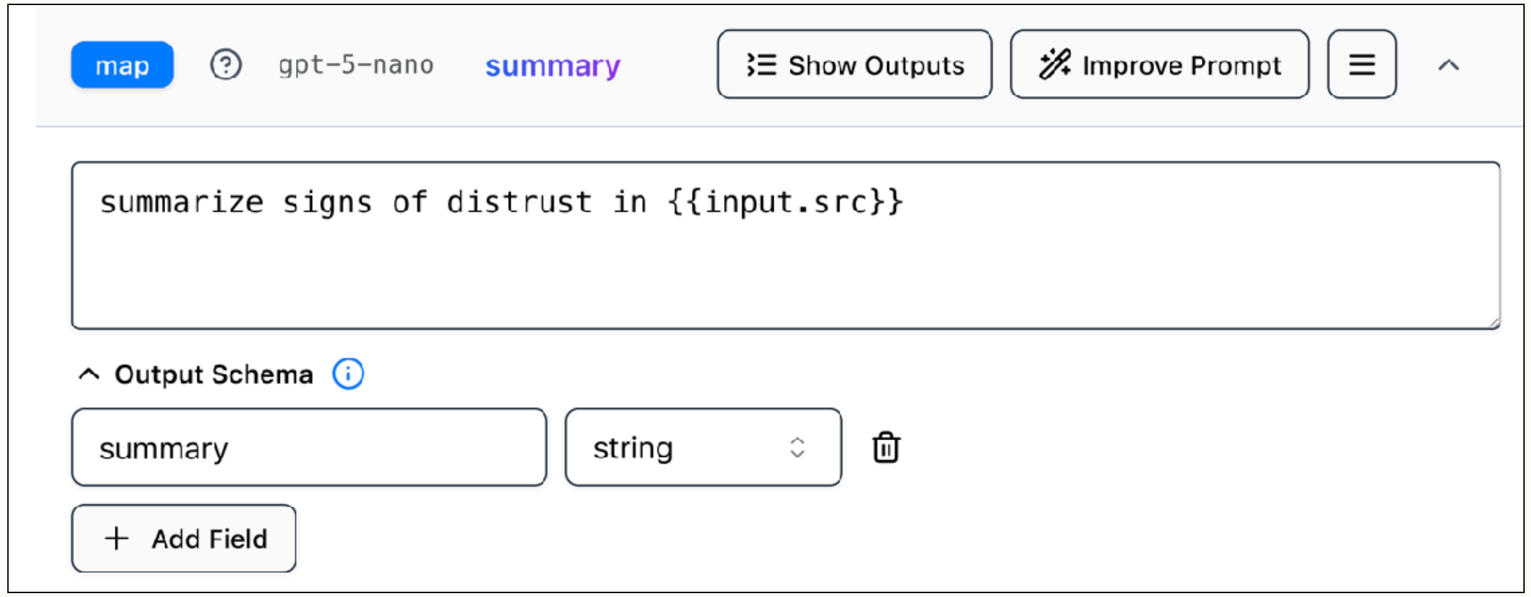
```
filter(“Does the patient trust the doctor?”)
```

??

What Did We Learn?

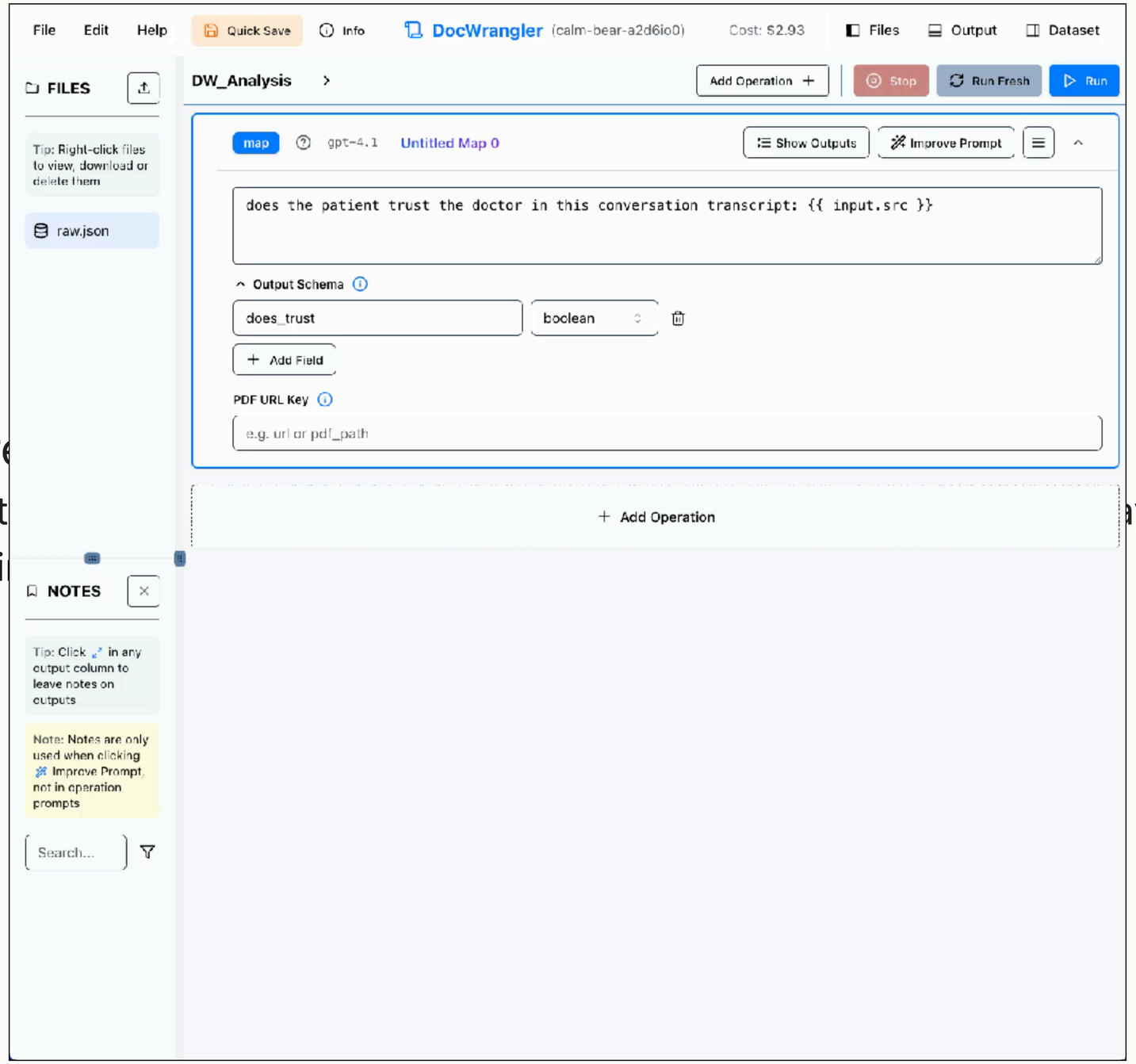
1. Users iterate *a lot* to get one query

2. Users don't read raw documents, even when we make it easy
   - E.g., they create exploratory ops to *summarize* documents in task-specific ways



1. Users iterate

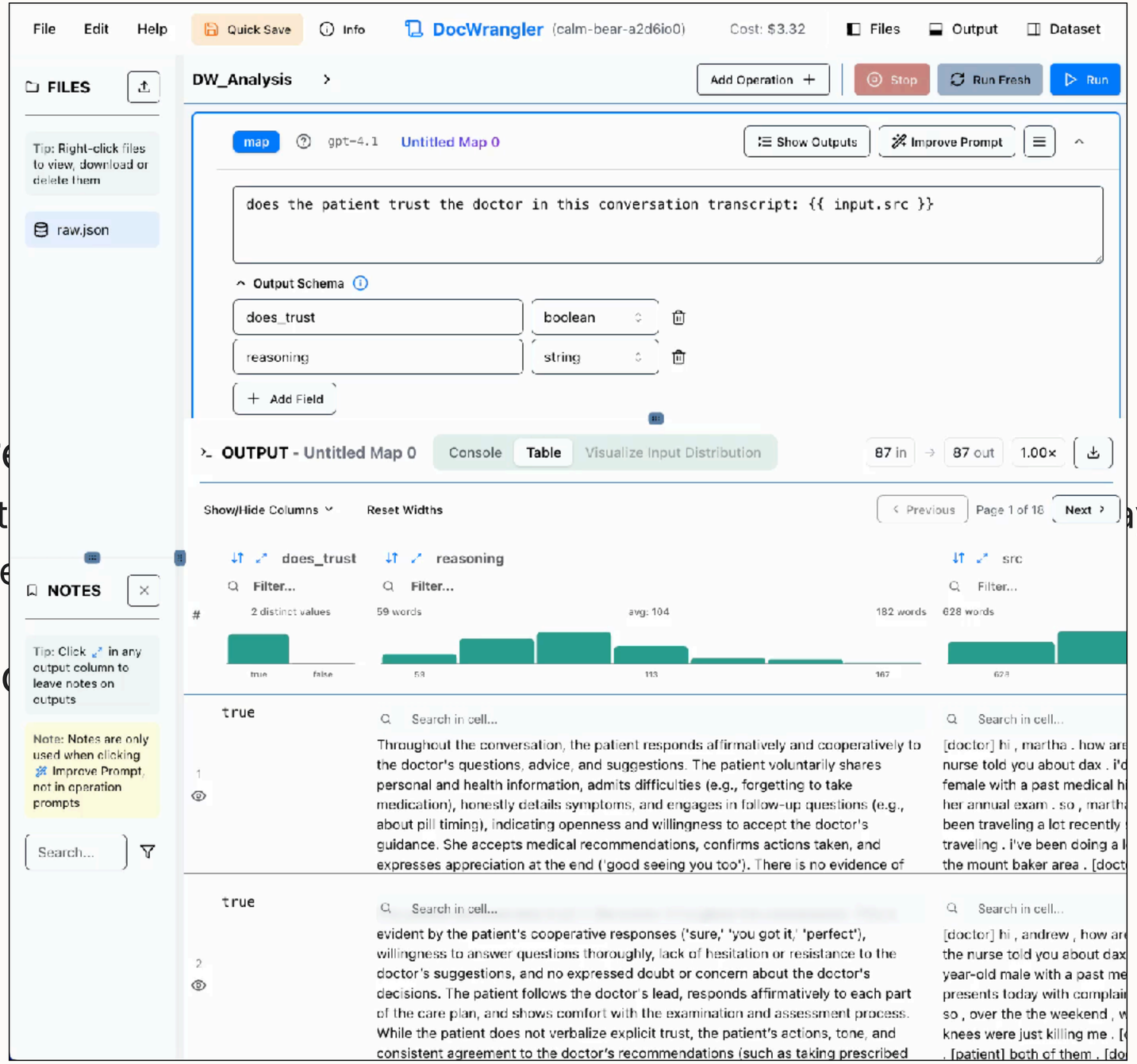
2. Users don’t re
   - E.g., they creat
   - Even when tryi



1. Users iterate

2. Users don’t re
   - E.g., they creat
   - And open-ende

3. Underspecifi



What Did We Learn?

1. Users iterate *a lot* to get one query

2. Users don't read raw documents, even when we make it easy
   - E.g., they create exploratory ops to *summarize* documents in task-specific ways
   - And open-ended exploratory *attributes* to validate LLM behavior

3. Underspecification drives discovery
   - E.g., Users tune attribute granularity & use visualizations to refine
   - "Prompt rubber ducking" (P9) — outputs help users discover more things to ask for

What Did We Learn?

1. Users iterate *a lot* to get one query

2. Users don't read raw documents, even when we make it easy
   - E.g., they create exploratory ops to *summarize* documents in task-specific ways
   - And open-ended exploratory *attributes* to validate LLM behavior

3. Underspecification drives discovery
   - E.g., Users tune attribute granularity & use visualizations to refine
   - "Prompt rubber ducking" (P9) — outputs help users discover more things to ask for

4. Users need to slow down!

What Did We Learn?

1. Users iterate *a lot* to get one query

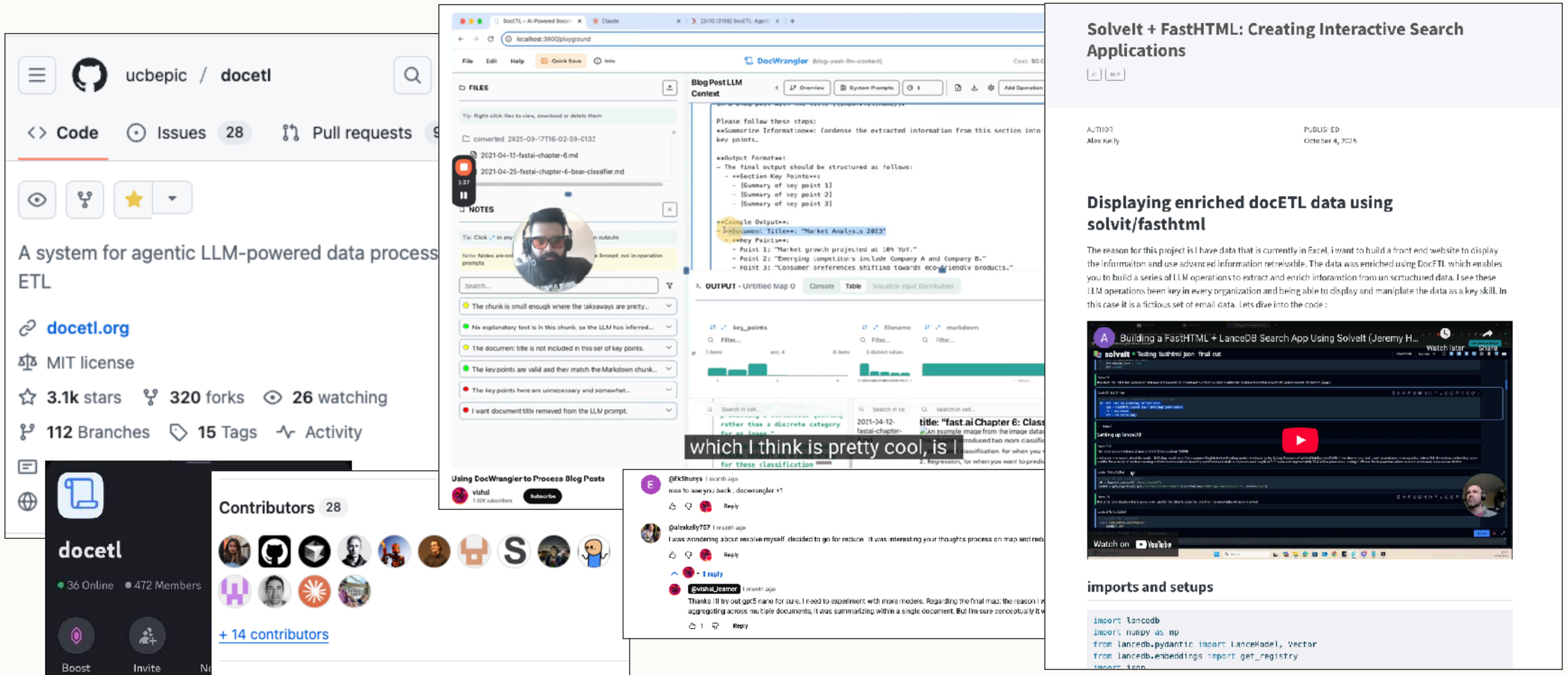
2. Users don't read raw documents, even when we make it easy
   - E.g., they create exploratory ops to *summarize* documents in task-specific ways
   - And open-ended exploratory *attributes* to validate LLM behavior

3. Underspecification drives discovery
   - E.g., Users tune attribute granularity & use visualizations to refine
   - "Prompt rubber ducking" (P9) — outputs help users discover more things to ask for

4. Users need to slow down!
   - >50% of prompts auto-refined with *zero notes* 😬
   - Output schemas as "speed brakes" (P8) — forces users to think about what they want

Impact: DocWrangler Users

Impact: DocETL Community



Building Effective Unstructured Data Systems. Shreya Shankar.

Today's Talk

3. **Outer loop (iterating on queries)**
   - ✦ *How should users author queries?*
   - ✦ *How can users validate results at scale?*

*Insight: build tools around the **three gulfs** of user-facing challenges*

Today's Talk

4. **Where do we go from here?**

Today: *Cusp* of Unstructured Data Analysis

*Given a user's question, get the **right answer** fast*

DocETL

??

??DB

??

??das

??

Microsoft SQL Server

Google Big Query

snowflake

Inner Loop

*Help the user ask the **right questions***

DocWrangler

??abase

Power BI

Chroma

LangChain

OpenAI

Outer Loop

Tomorrow: *Ubiquitous* Unstructured Data Analysis

Inner Loop

*Given a question, get the **right answer** fast*

"Full-stack" in the database

Outer Loop

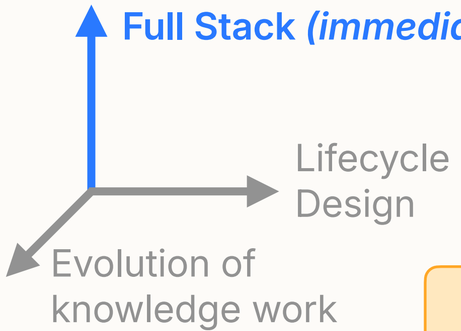
*Figure out the **right question** to ask*

"Full-lifecycle" for data-oriented tasks

How is AI changing knowledge work?

Field studies with domain experts, new research methodologies

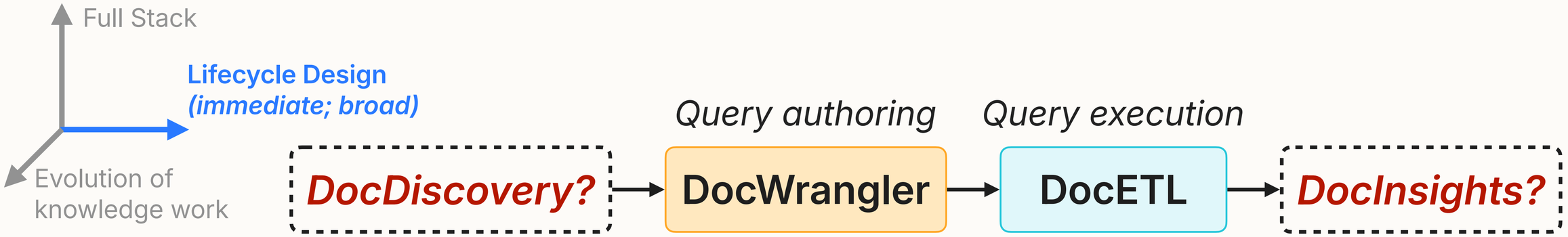
Tomorrow: “Full-Stack” in Inner Loop



DocETL

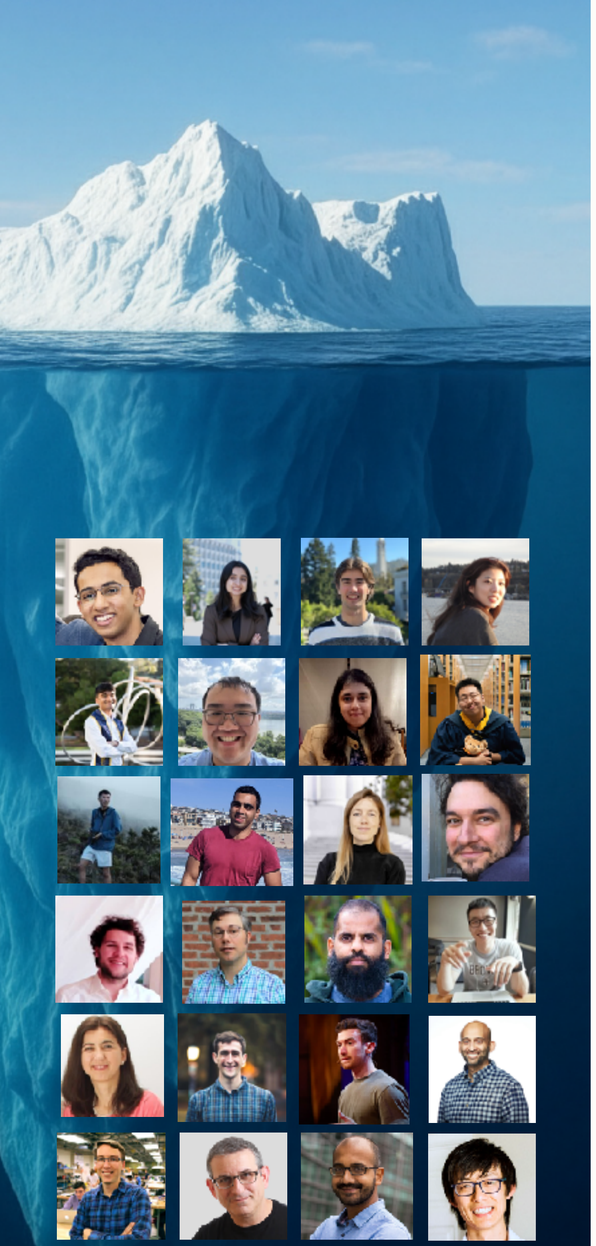
Operators	*Natural language is ambiguous*	*NL + formal guardrails*
Optimizer	*Takes a long time*	*RL agents to learn query rewrites*
Execution Engine	*Treats LLM as a black box*	*Open-source LLMs; custom KV caching*
Storage	*Documents are large and redundant*	*New indexes, new compression*

Tomorrow: “Full-Lifecycle” in Outer Loop



Lifecycle gaps:

- ✦ Discovery: exploratory tools to help decide *what* to analyze
- ✦ Insights: BI (e.g., Polaris/Tableau) & collaboration over unstructured data



Summary

Unstructured & LLM-powered data systems: a new subfield.

*Given a question, get the **right answer** fast*

*Insight: must explore semantically equivalent plans (**rewrite directives**)*

Inner Loop

*Figure out the **right question** to ask*

*Insight: build tools around the **three gulfs** of user-facing challenges*

Outer Loop

What's next? Make unstructured data analysis ubiquitous—by going deeper, broader, and beyond CS.