

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

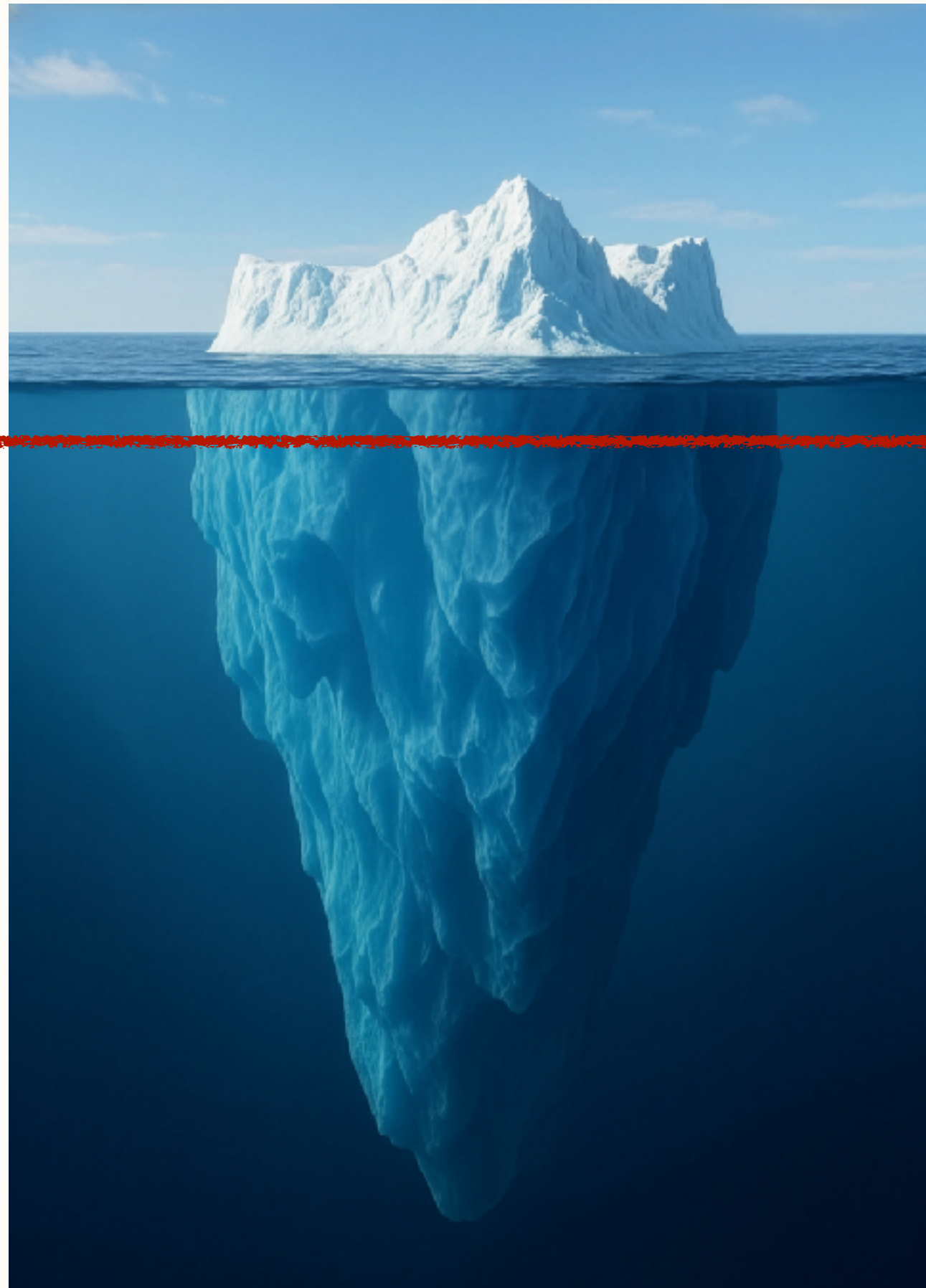
*Help the user ask the
right questions*

- ◆ Interaction design
- ◆ Visualization and communication
- ◆ Sensemaking theory



Outer Loop

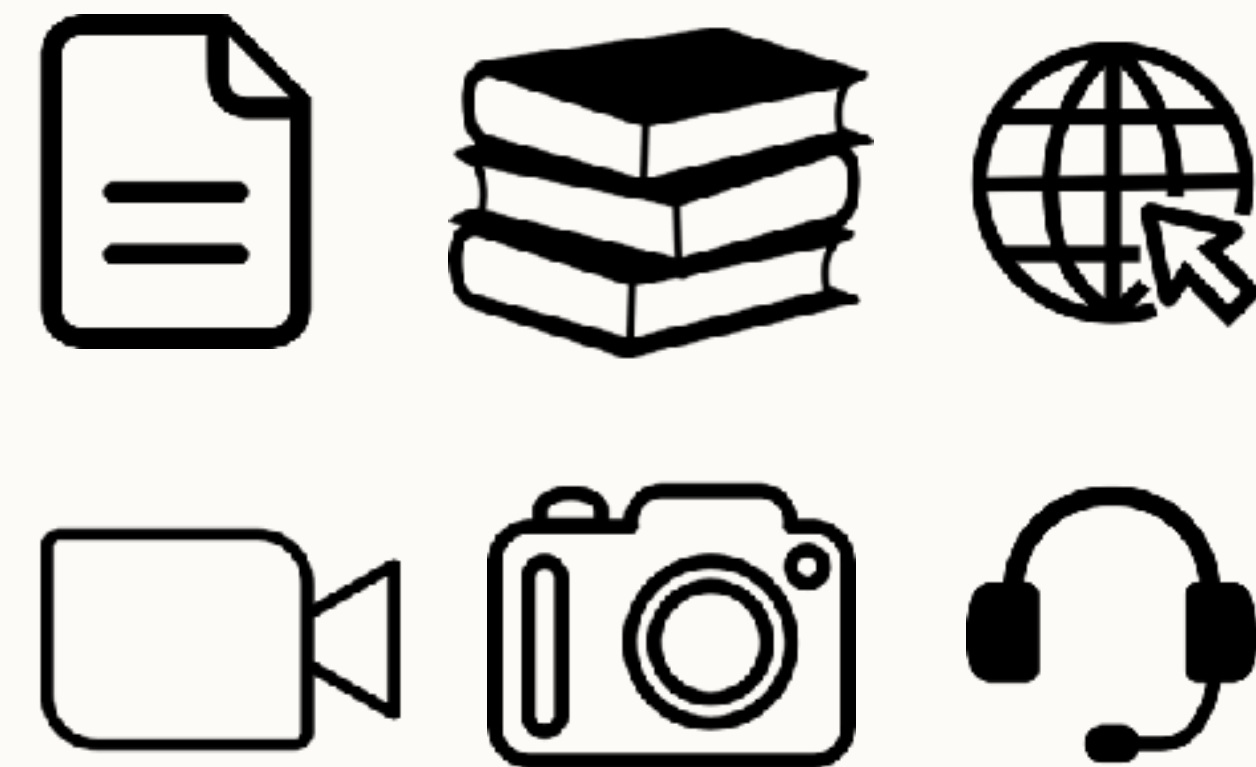
What's Next?



Structured data



Unstructured data



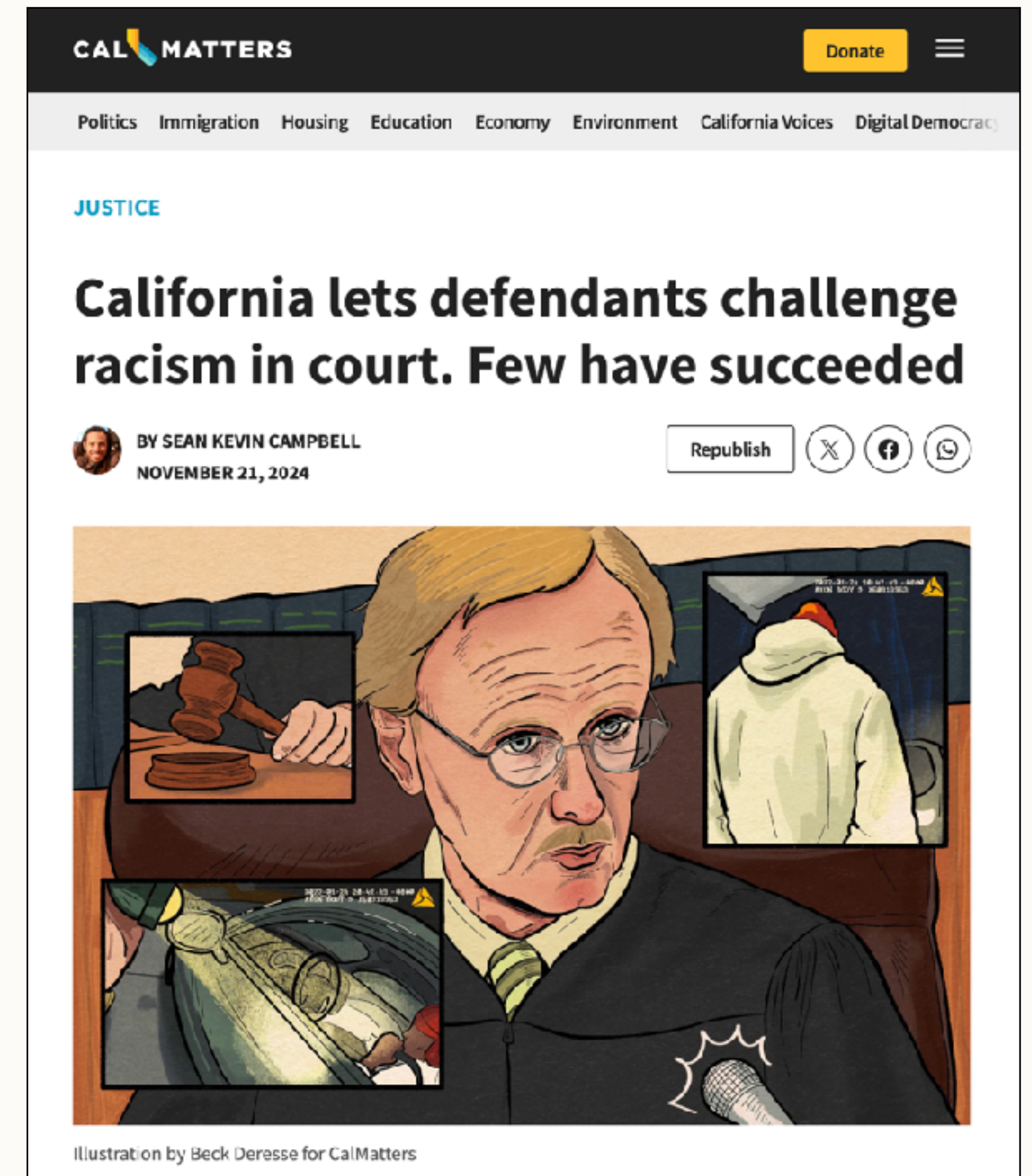
← Can't use SQL to query this type of data

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?

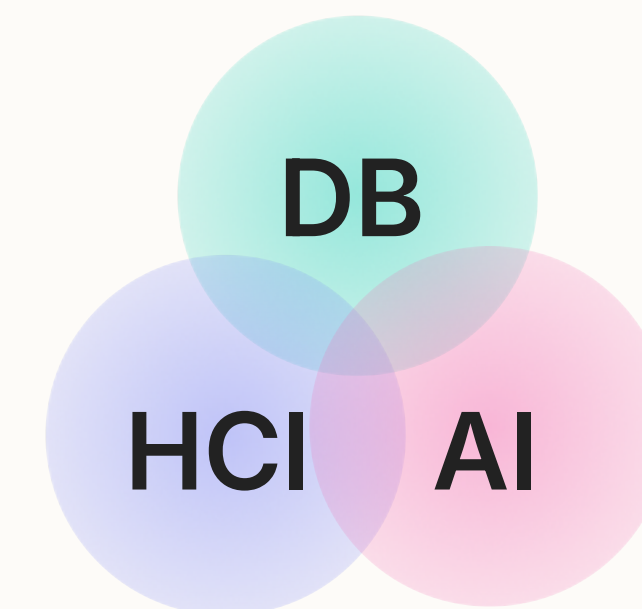
Which contracts have unfavorable arbitration clauses?



What are common complaints in customer reviews?

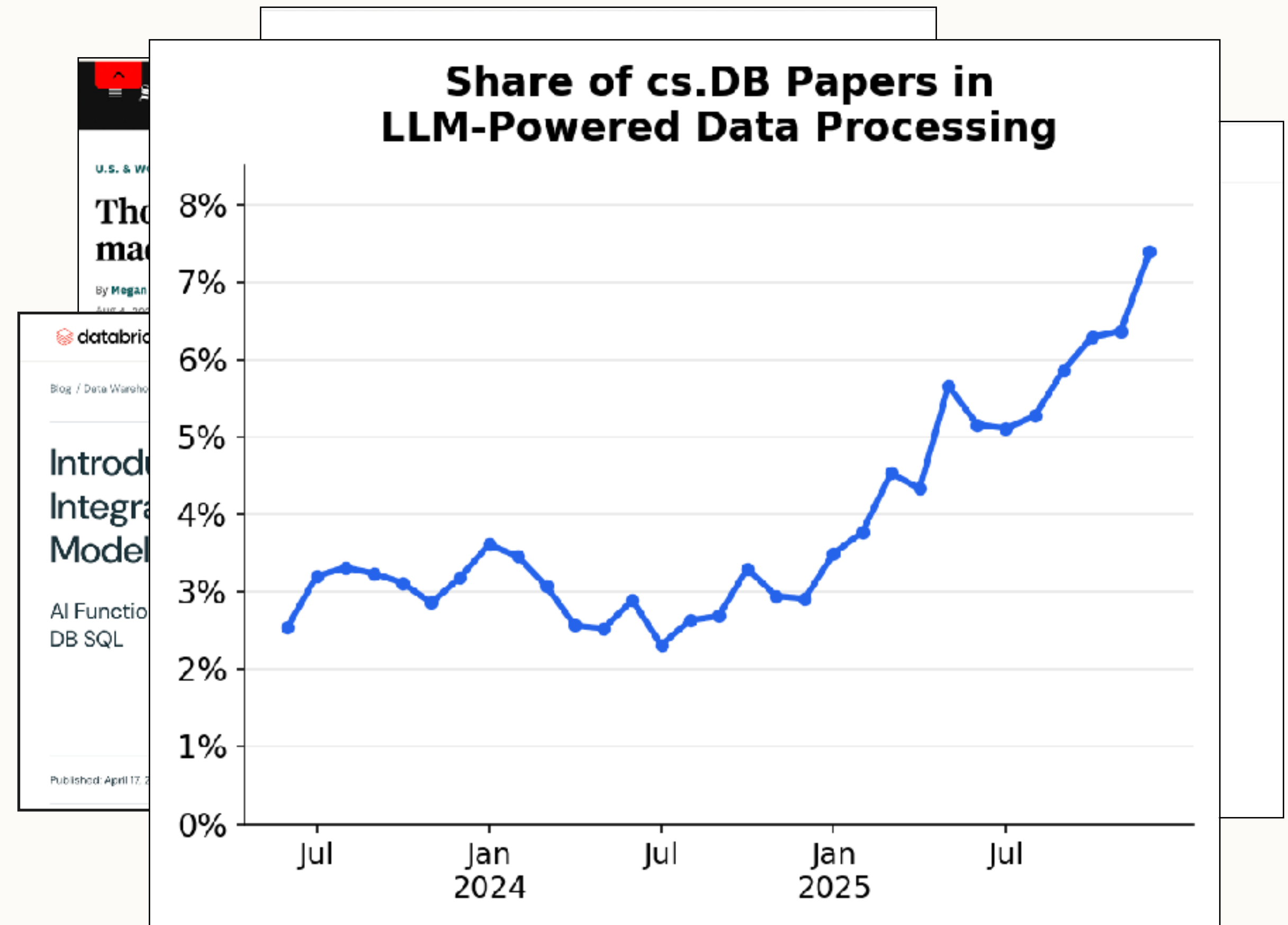
Interdisciplinary challenges:

- ◆ 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



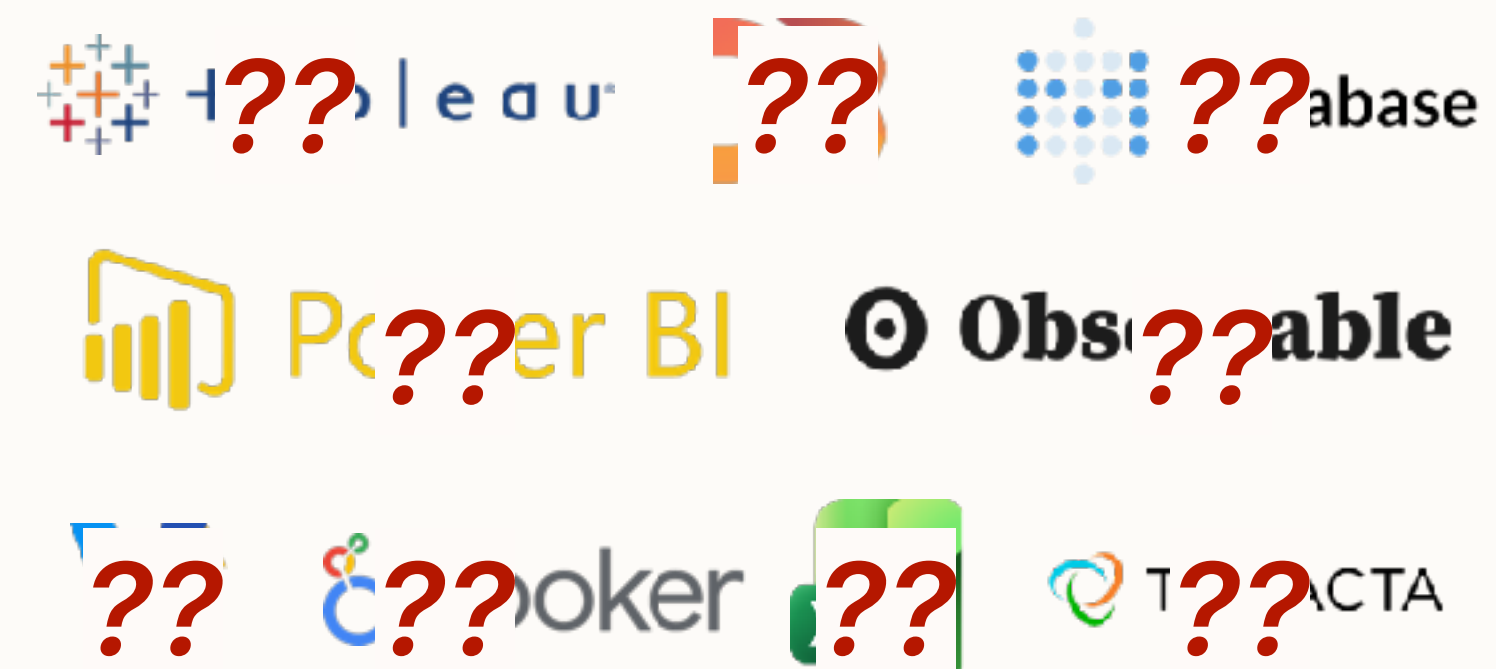
LLM-Powered Unstructured Data Systems

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



Inner Loop

Help the user ask the right questions



Outer Loop

An iceberg floating in the ocean. The small tip above the water is labeled 'Structured data (Tables & SQL)'. The much larger, submerged part of the iceberg is labeled 'Unstructured data'.

Structured data
(Tables & SQL)

Unstructured data

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?

An iceberg floating in the ocean. The small tip above the water is labeled 'Structured data (Tables & SQL)'. The much larger, submerged part of the iceberg is labeled 'Unstructured data'.

*Structured data
(Tables & SQL)*

Unstructured data

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

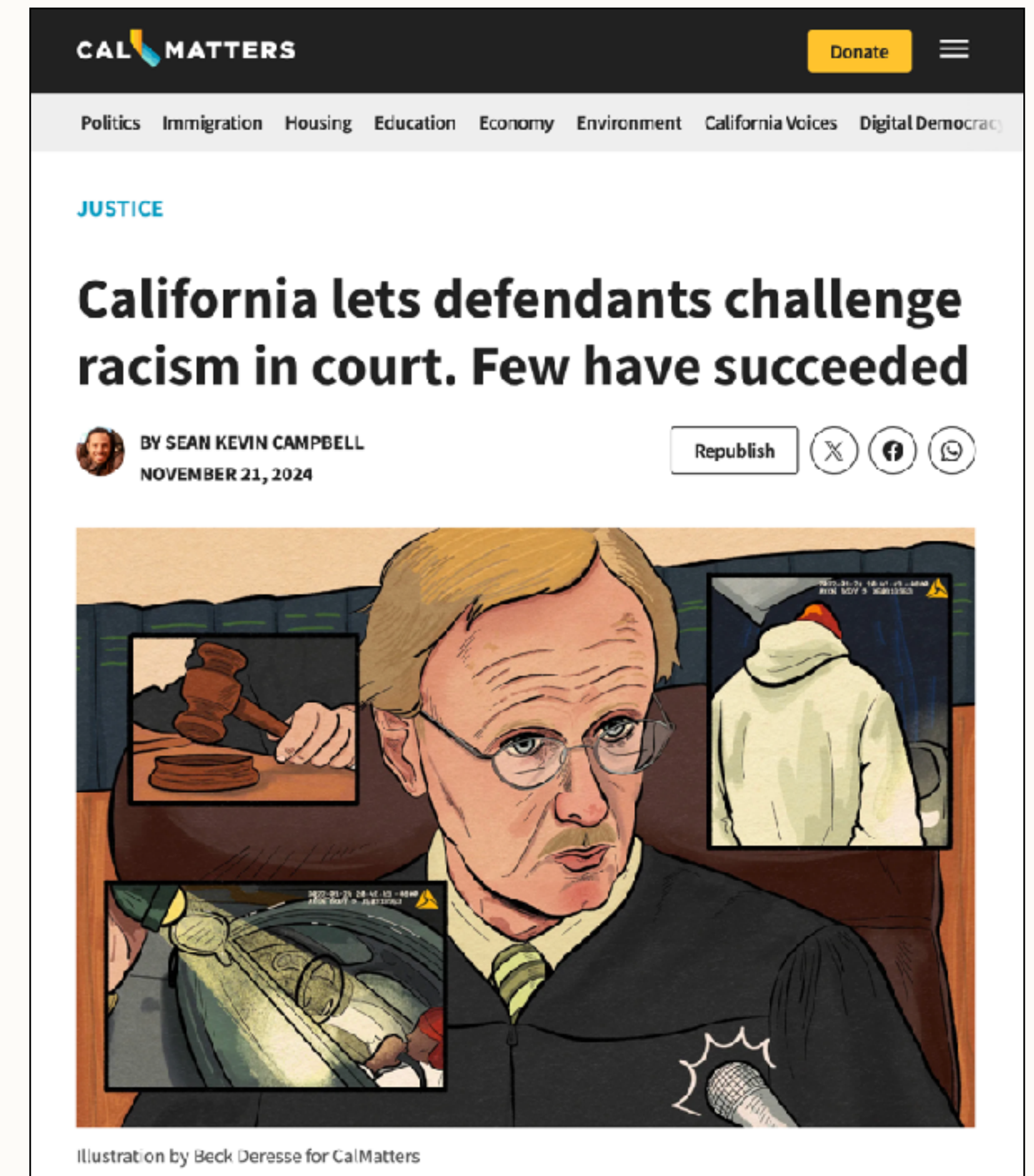
Queries: hundreds per defendant

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	<code>name</code>
...	...	...

<code>name</code>	<code>common_biases</code>
...	...

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
```

An iceberg floating in the ocean. The small tip above the water is labeled 'Structured data (Tables & SQL)'. The much larger, submerged part of the iceberg is labeled 'Unstructured data'.

Structured data
(Tables & SQL)

Unstructured data

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

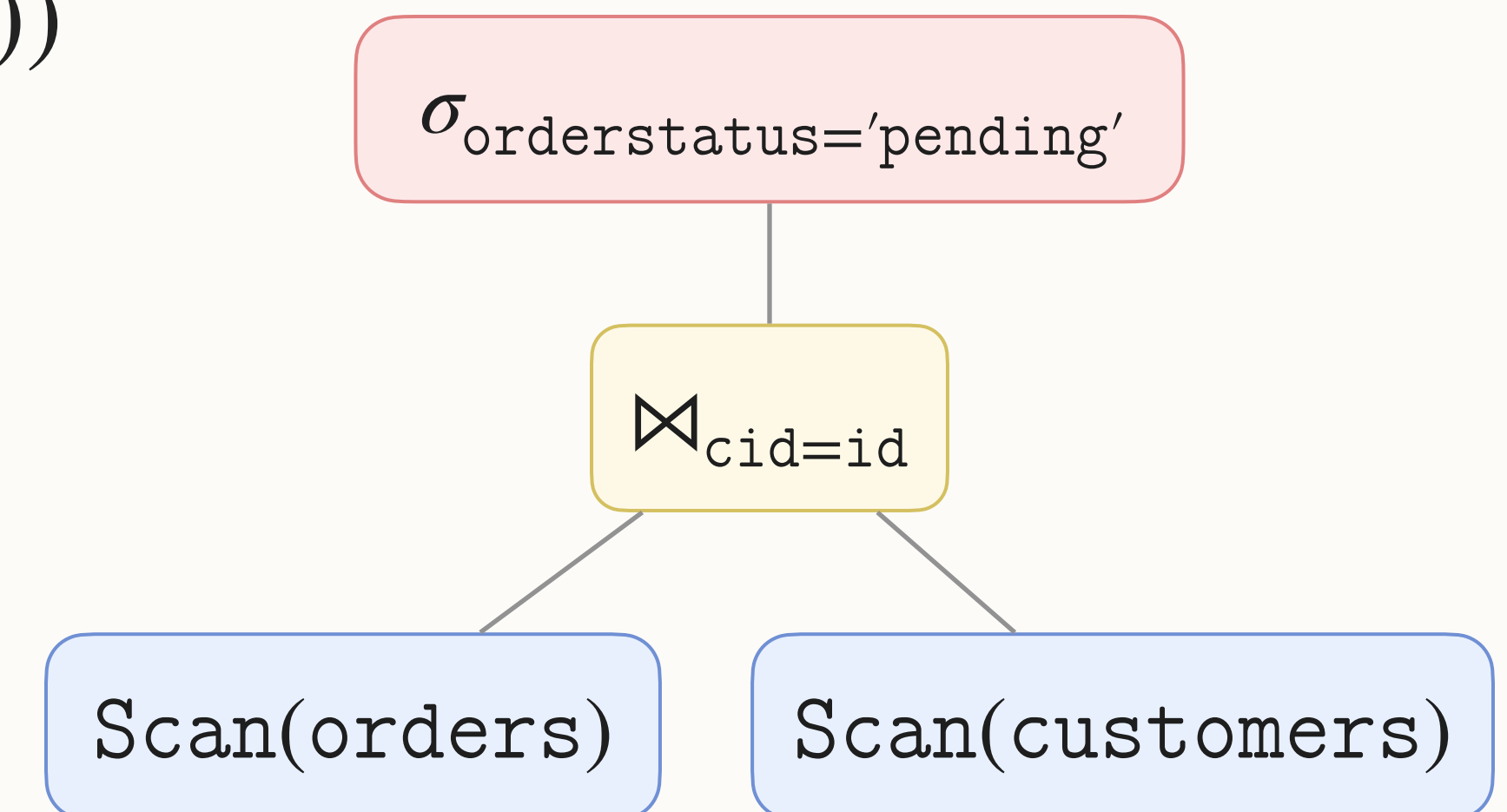
Background: Rewrite Rules

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:

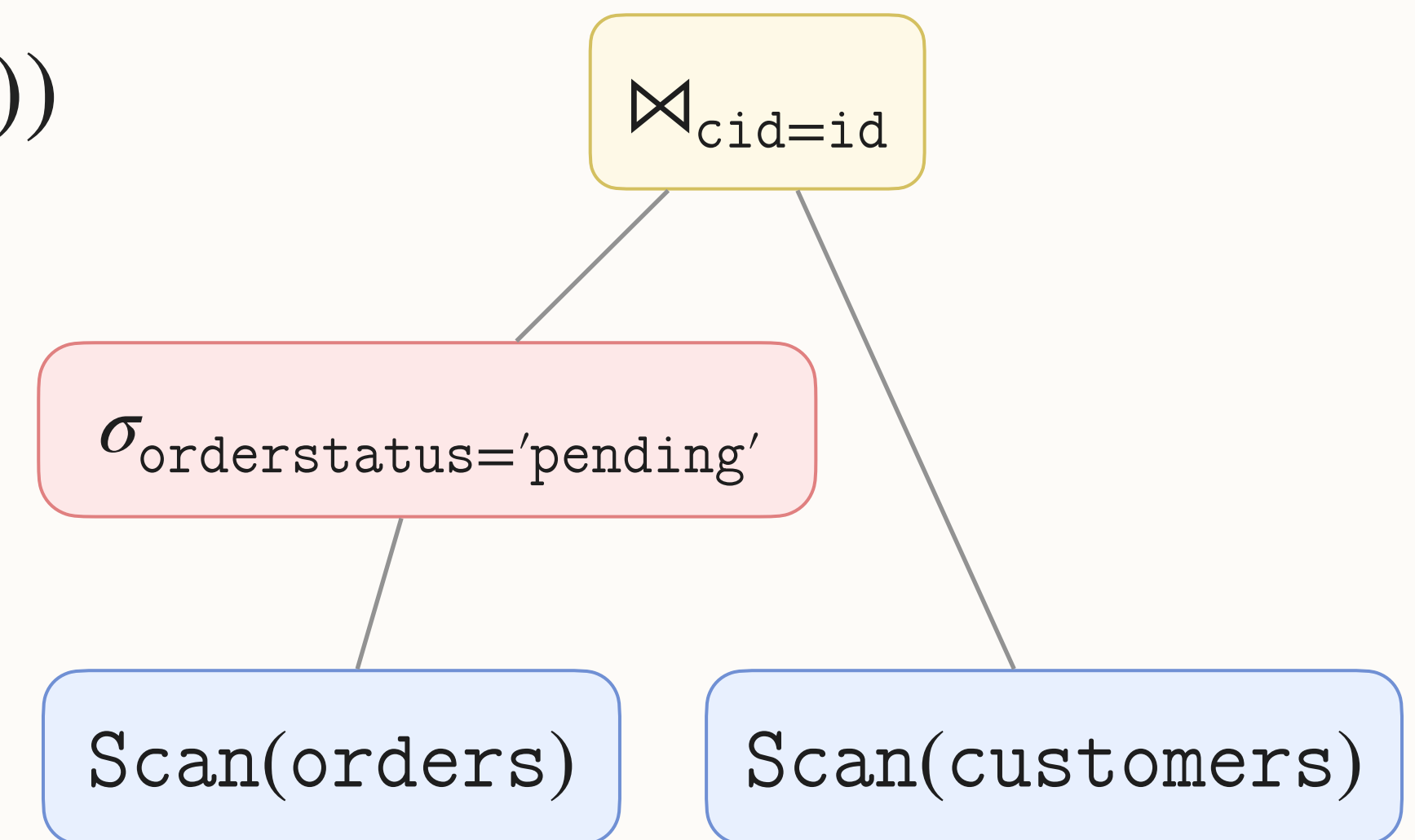
♦ Projection or **filter pushdown**; e.g.,

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

♦ 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.,

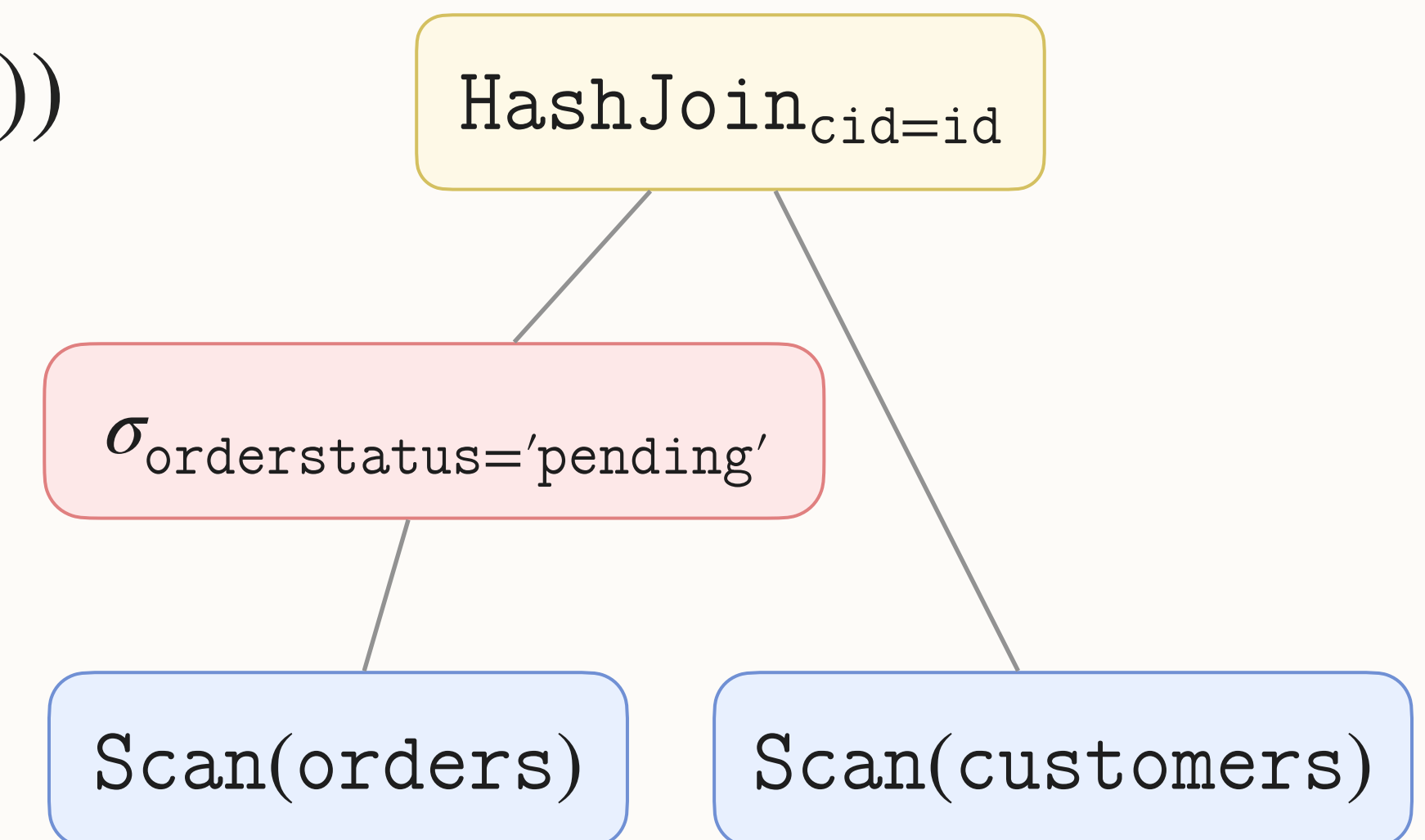
$$\pi_x(A \bowtie B) \rightarrow \pi_x(\pi_y(A) \bowtie \pi_z(B)) \text{ or } \sigma_p(f(A)) \rightarrow f(\sigma_p(A))$$

♦ Choosing algorithms; e.g.,

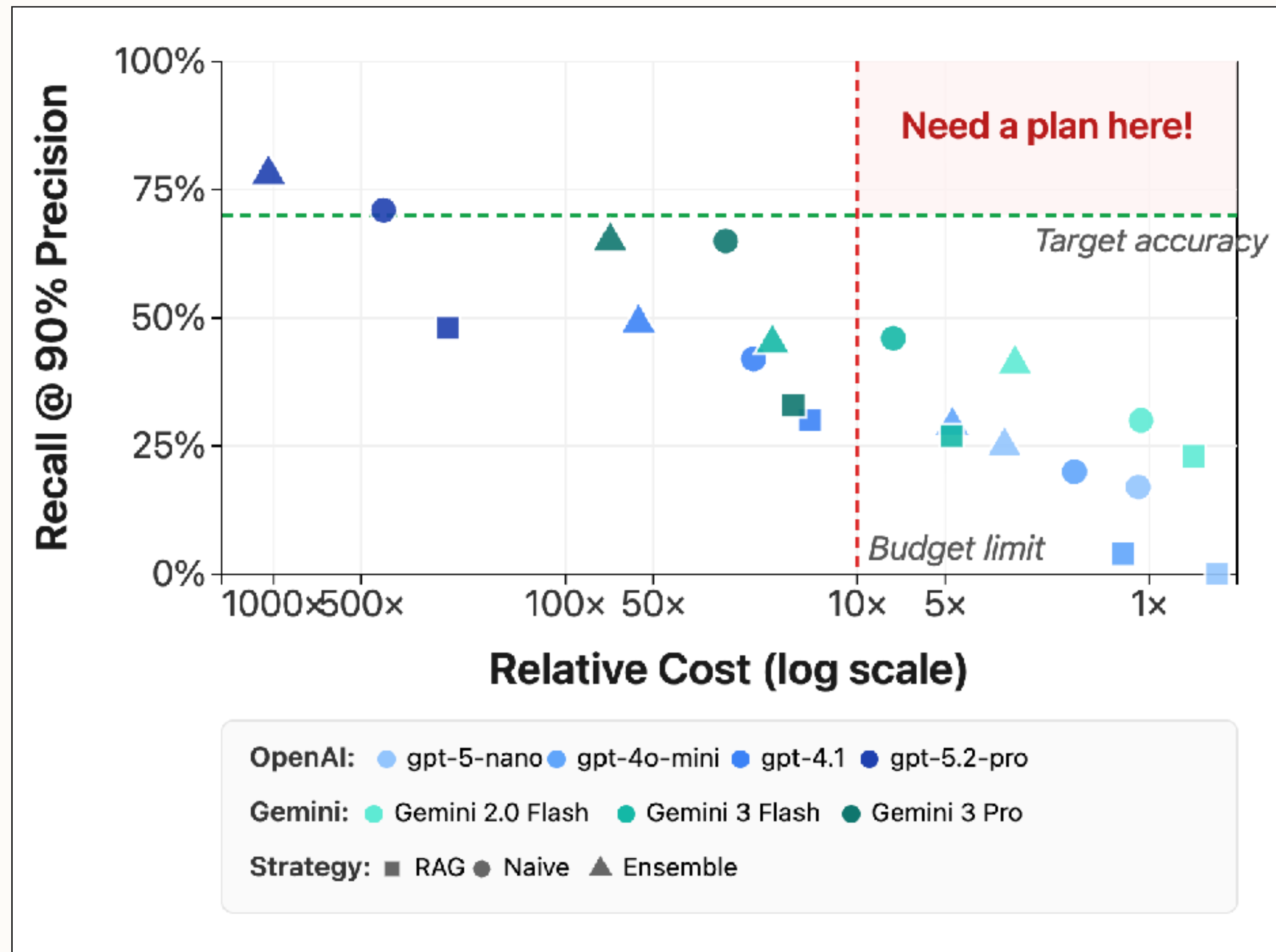
$$A \bowtie B \rightarrow \text{HashJoin}(A, B)$$

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

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



Rewrite Rules Don't Work Well



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.”
```

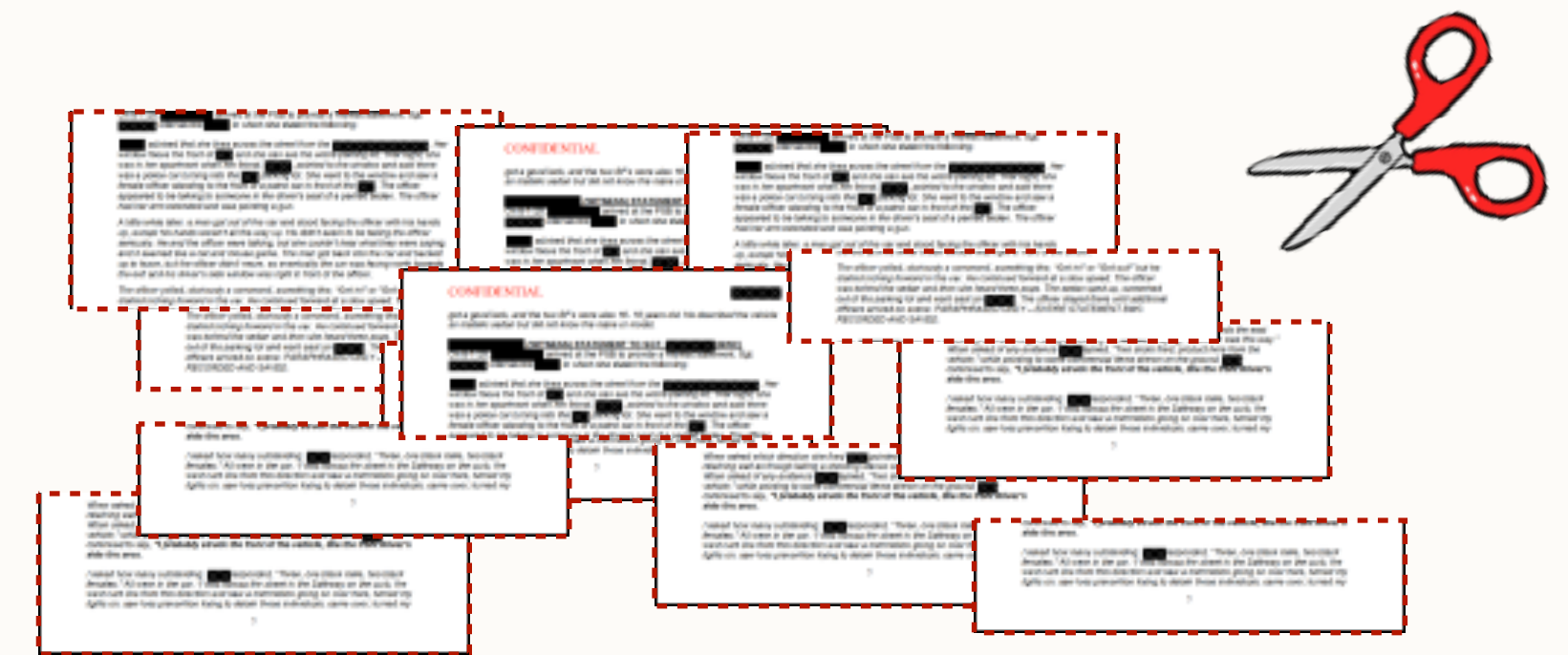
**Prompt very simplified, for space constraints.*

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.”
```

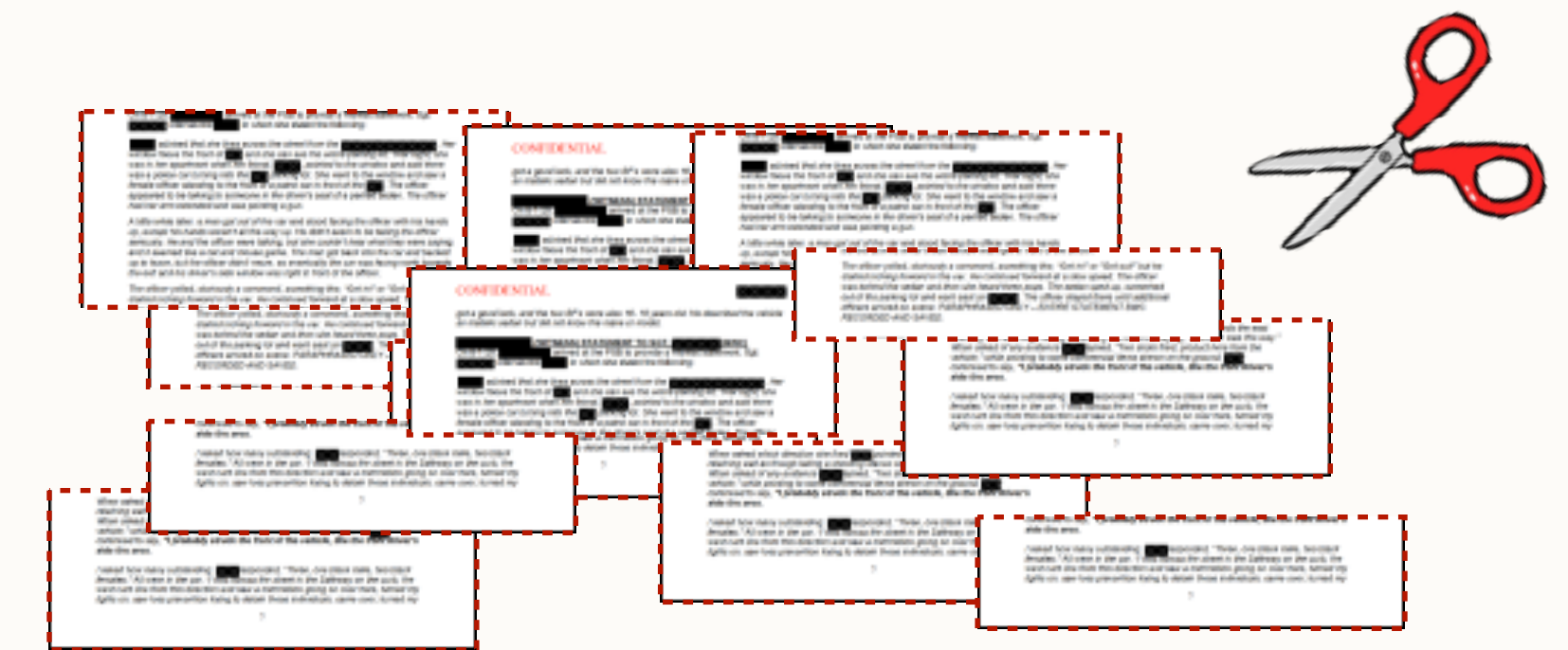
**Prompt very simplified, for space constraints.*

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.*

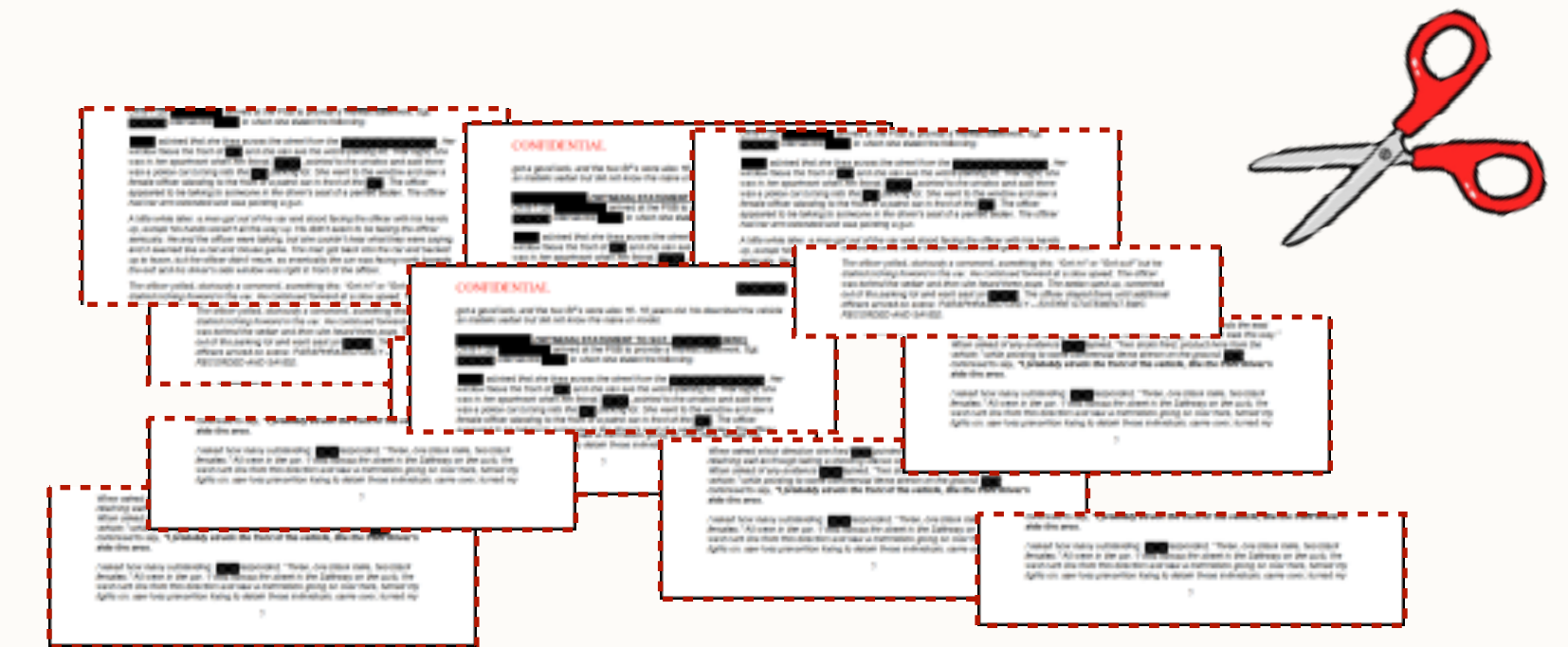
Can We Solve the Problem?

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.

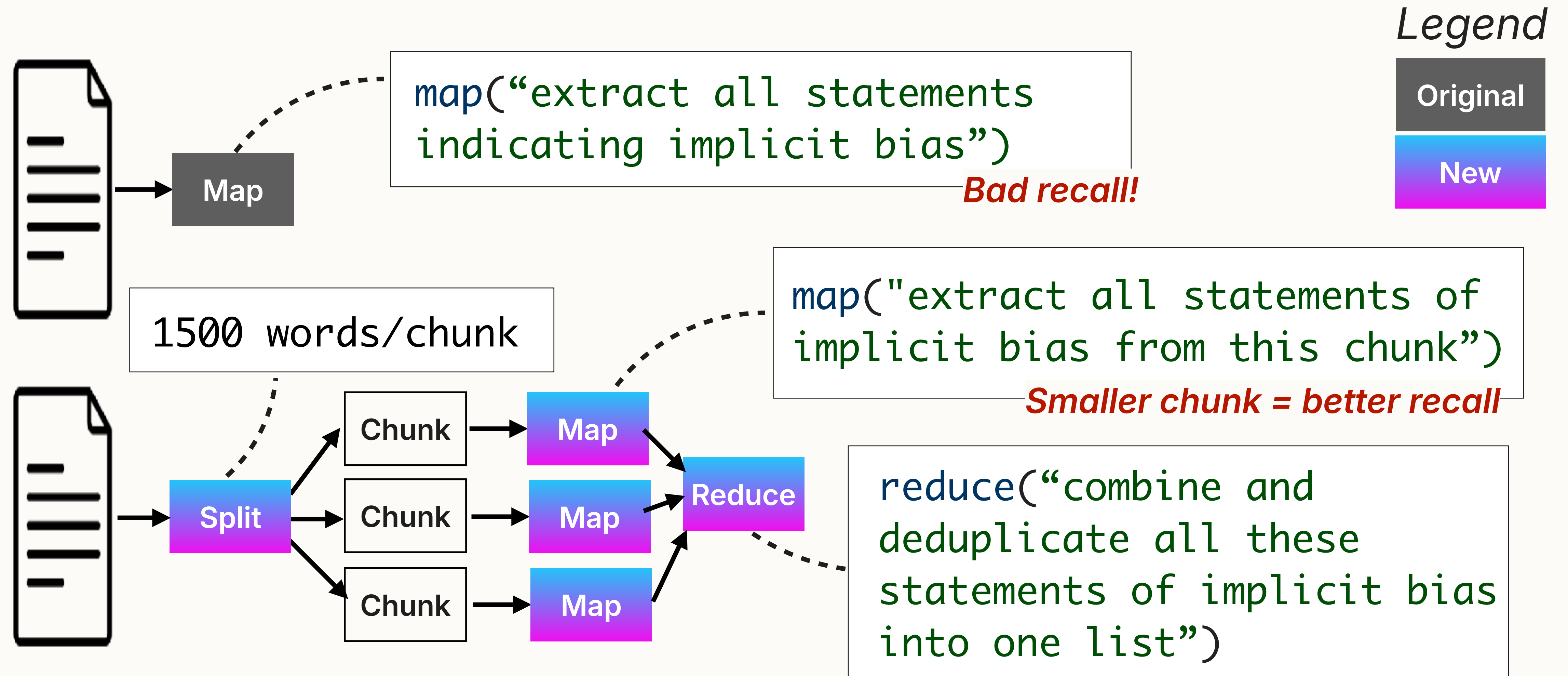
- ◆ They generate new operators to better match LLM capabilities
- ◆ Instantiation requires reasoning about the data and task

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

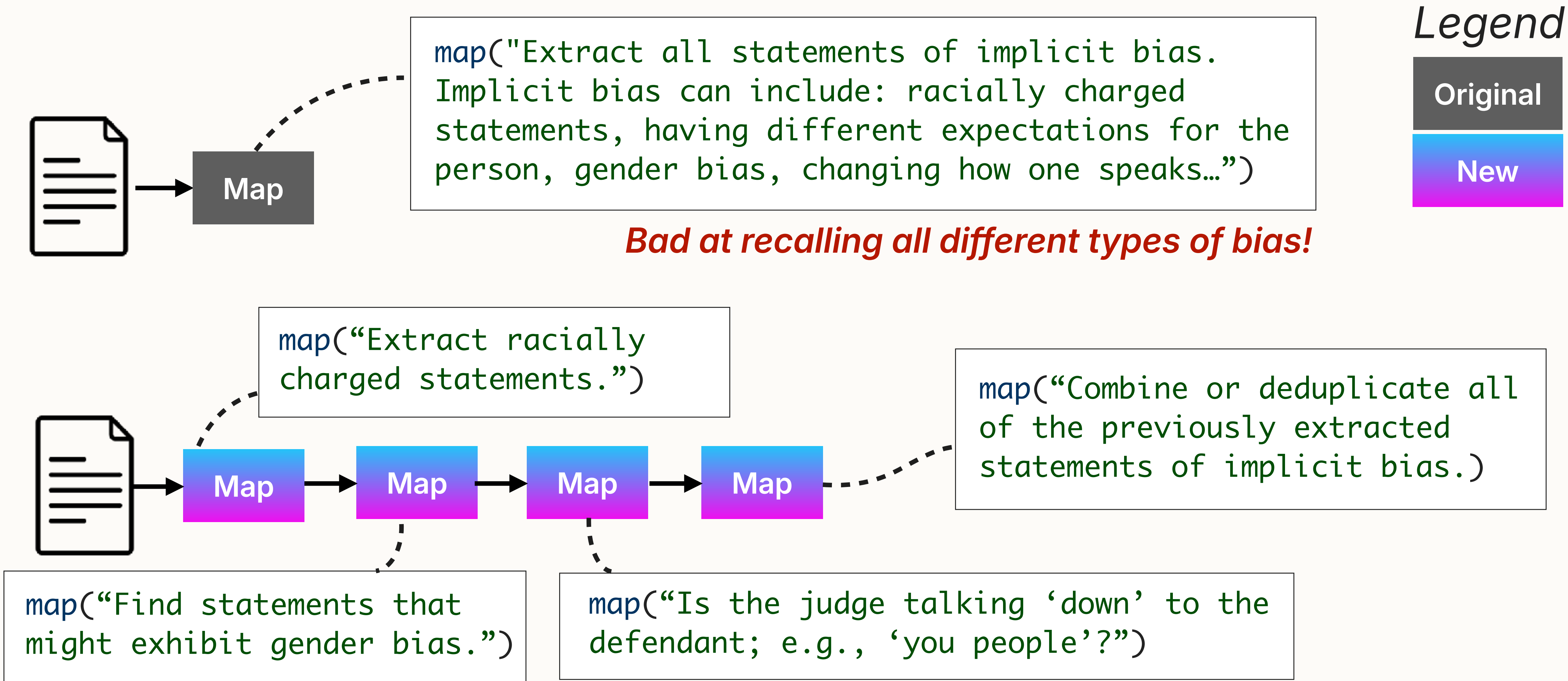
Some challenges:

- ◆ 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...

judge: "X.
Quinnsworth"

Document 2
Judge Xander Quinnsworth was...

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?

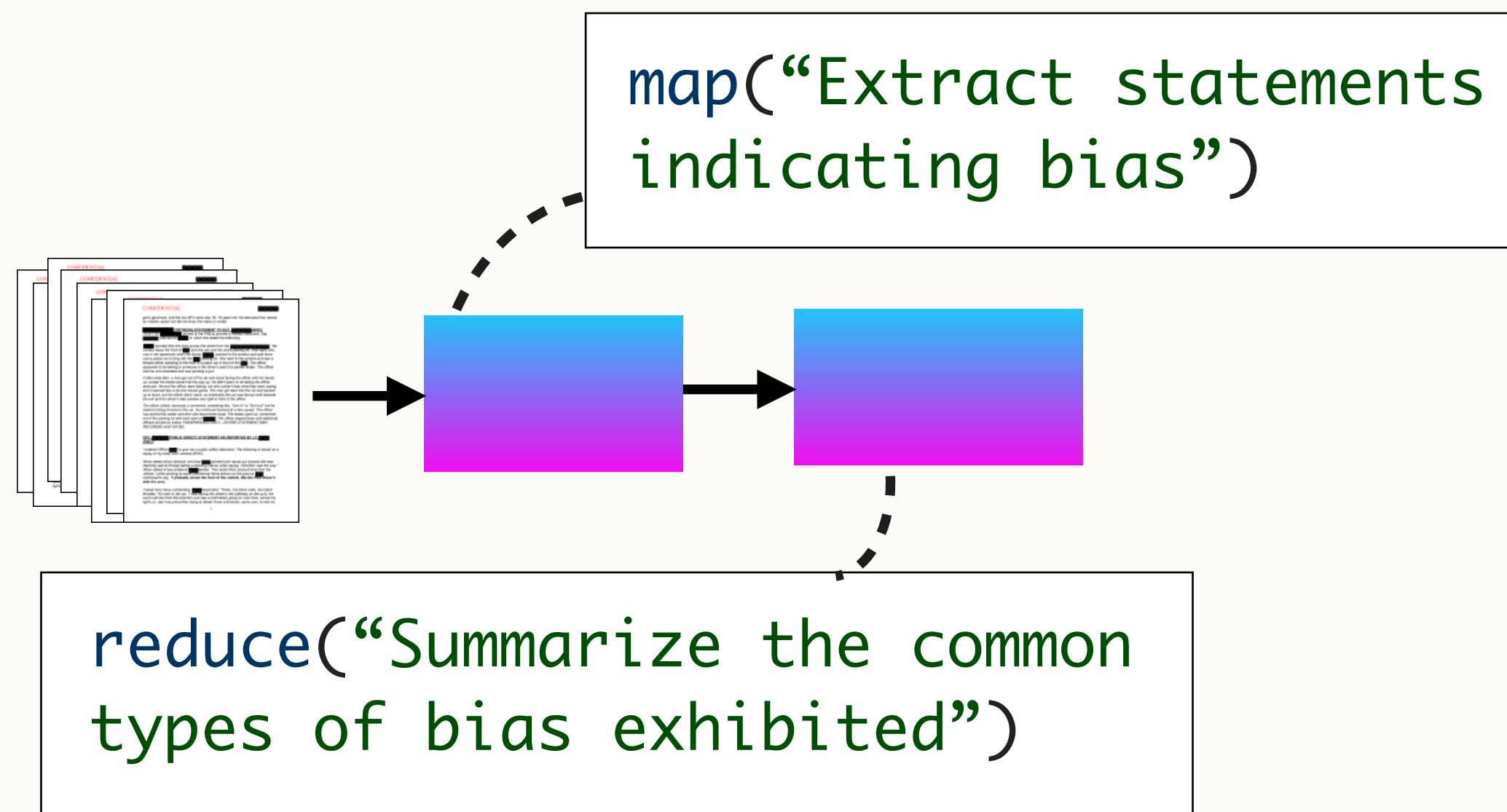
- ◆ Background: DP/memo-based search
- ◆ Simpler setting: Task Cascades
- ◆ A more general setting: MCTS-based search

Background: Search

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

Key Insight: No Optimal Substructure

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



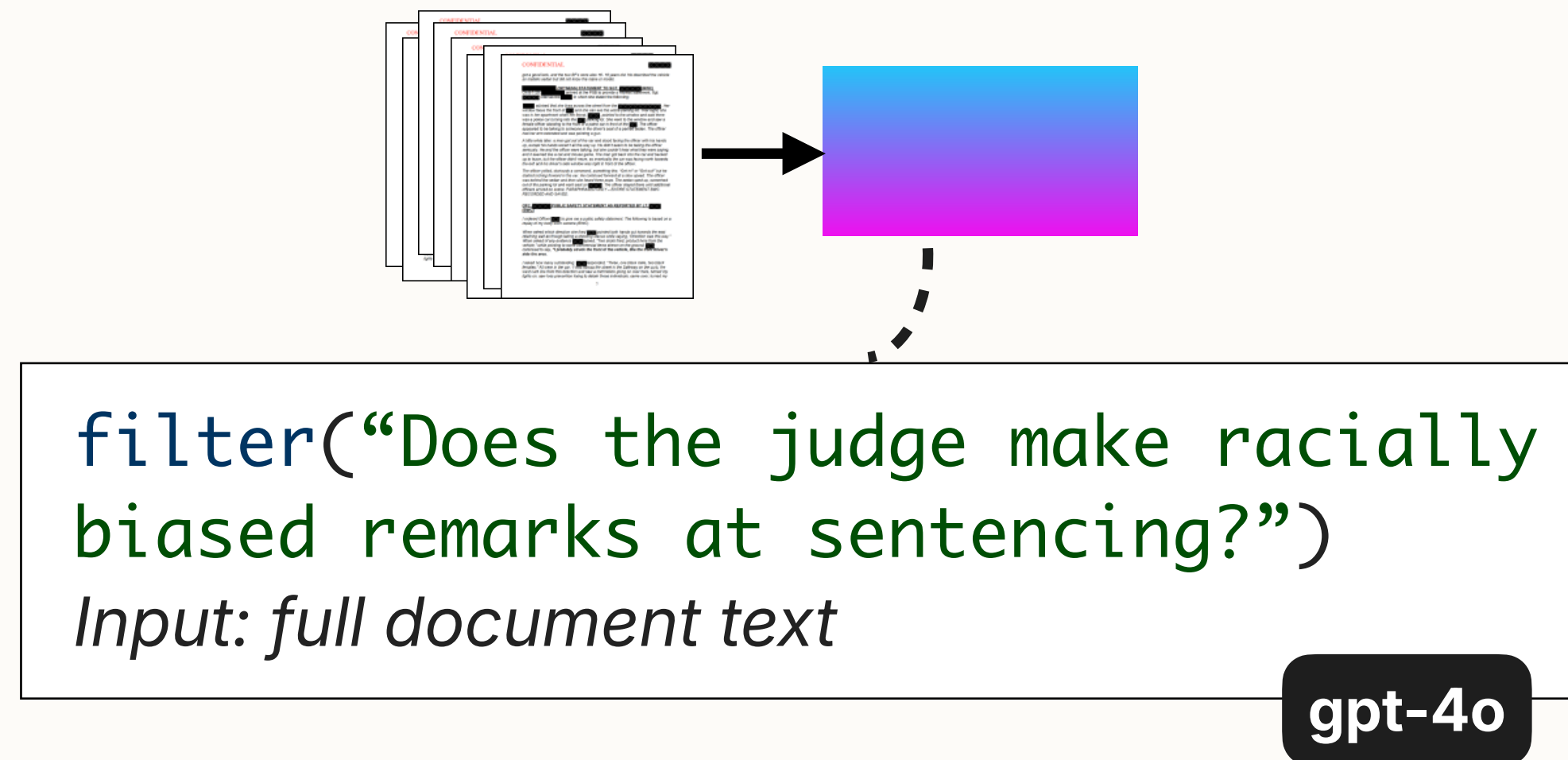
LLM behavior doesn't cleanly compose across operators!

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

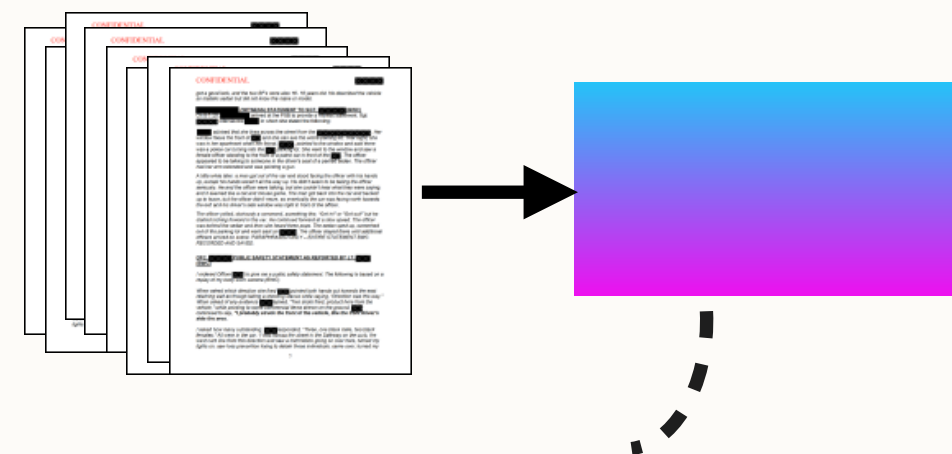
Task Cascades for Efficient Unstructured Data Processing. Shankar, Zeighami, Parameswaran. SIGMOD '26.

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

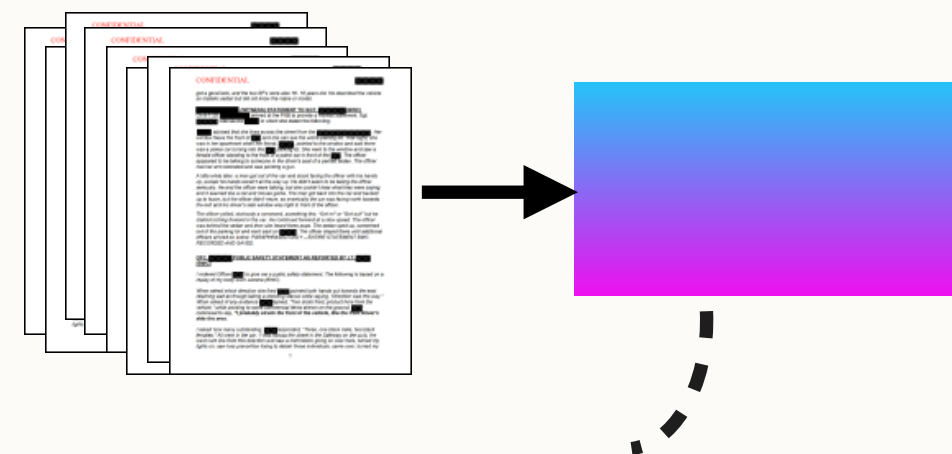
Task Cascades for Efficient Unstructured Data Processing. Shankar, Zeighami, Parameswaran. SIGMOD '26.

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

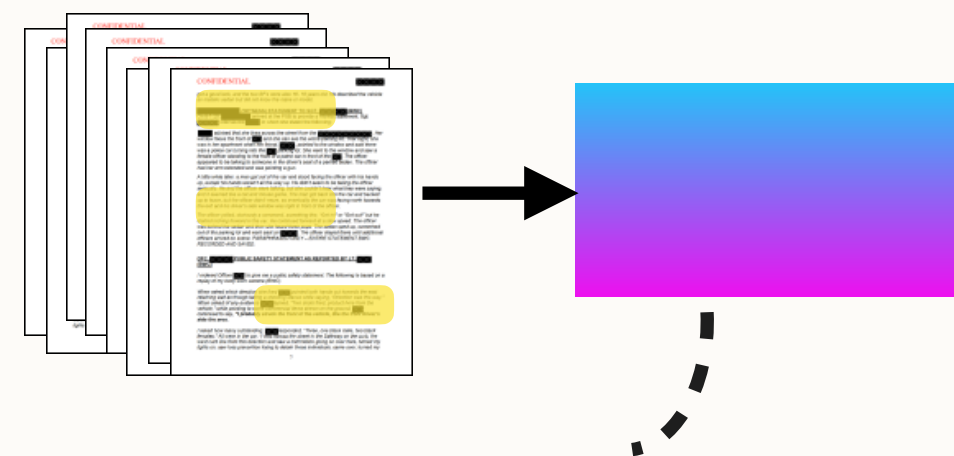
Task Cascades for Efficient Unstructured Data Processing. Shankar, Zeighami, Parameswaran. SIGMOD '26.

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

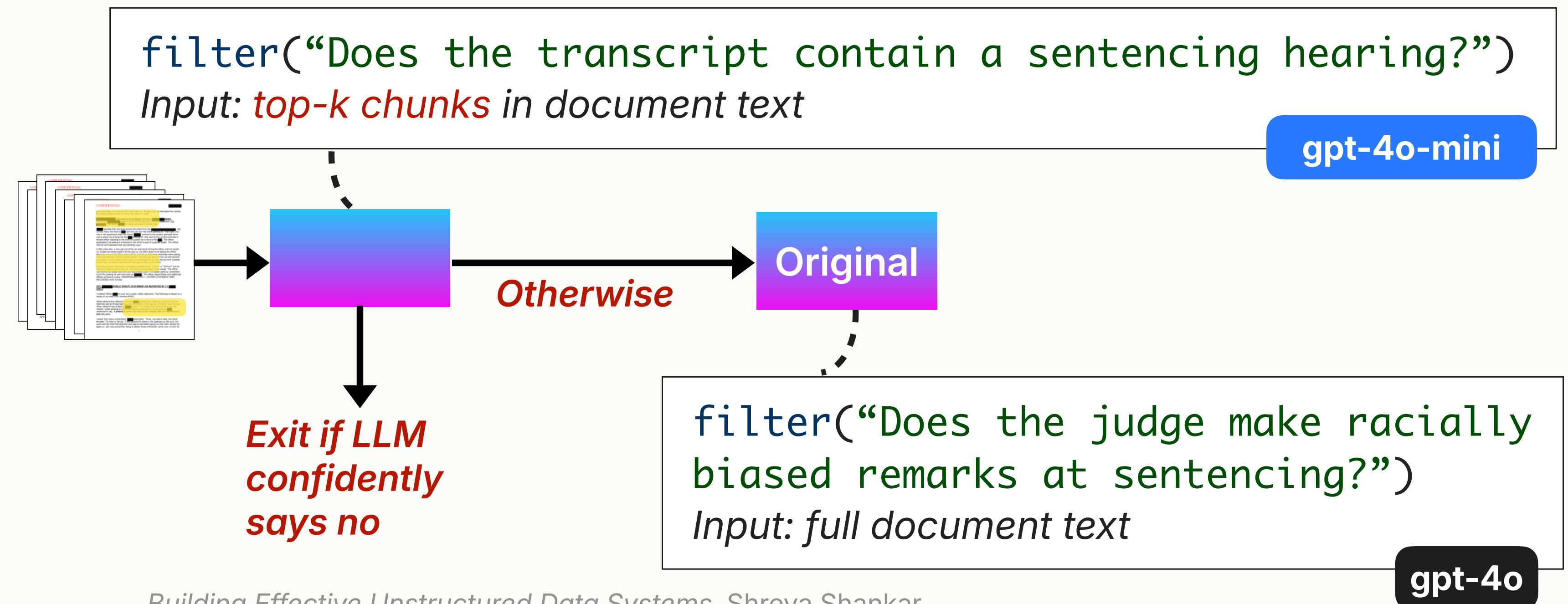
Task Cascades for Efficient Unstructured Data Processing. Shankar, Zeighami, Parameswaran. SIGMOD '26.

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*

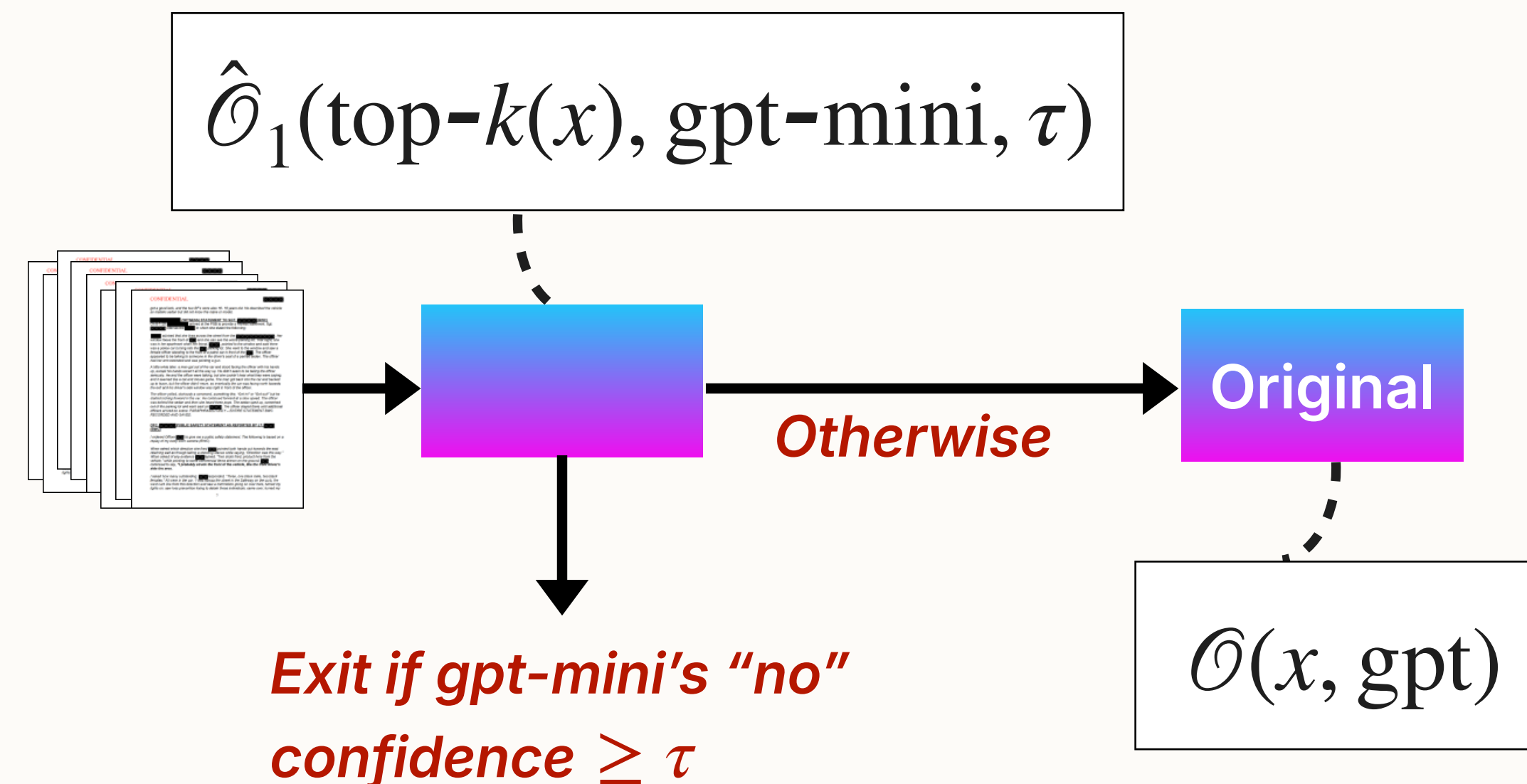
Task Cascades for Efficient Unstructured Data Processing. Shankar, Zeighami, Parameswaran. *SIGMOD '26*.

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*

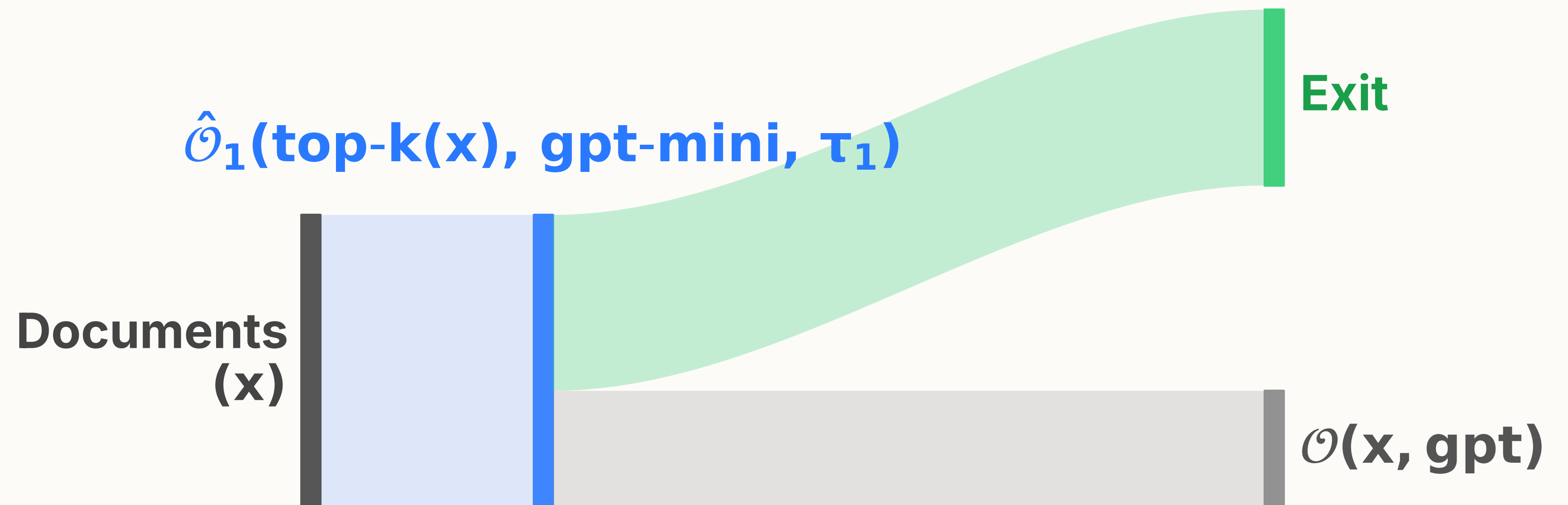
Task Cascades for Efficient Unstructured Data Processing. **Shankar**, Zeighami, Parameswaran. *SIGMOD '26*.

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*

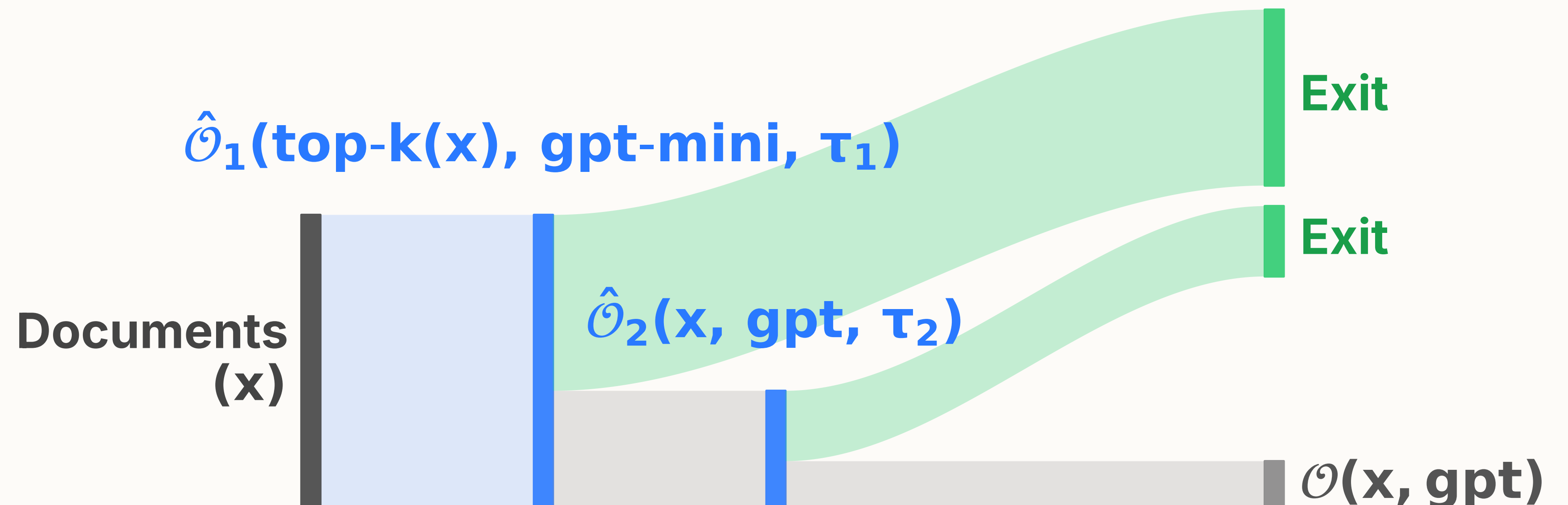
Task Cascades for Efficient Unstructured Data Processing. Shankar, Zeighami, Parameswaran. *SIGMOD '26*.

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*

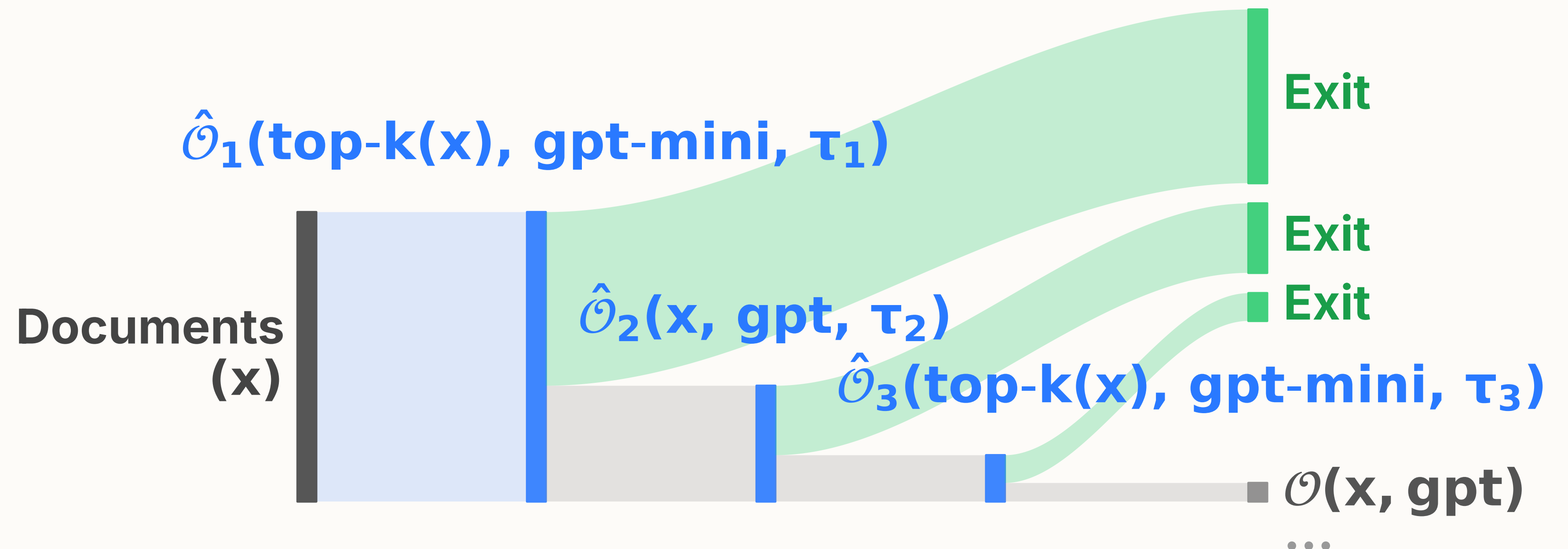
Task Cascades for Efficient Unstructured Data Processing. Shankar, Zeighami, Parameswaran. *SIGMOD '26*.

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

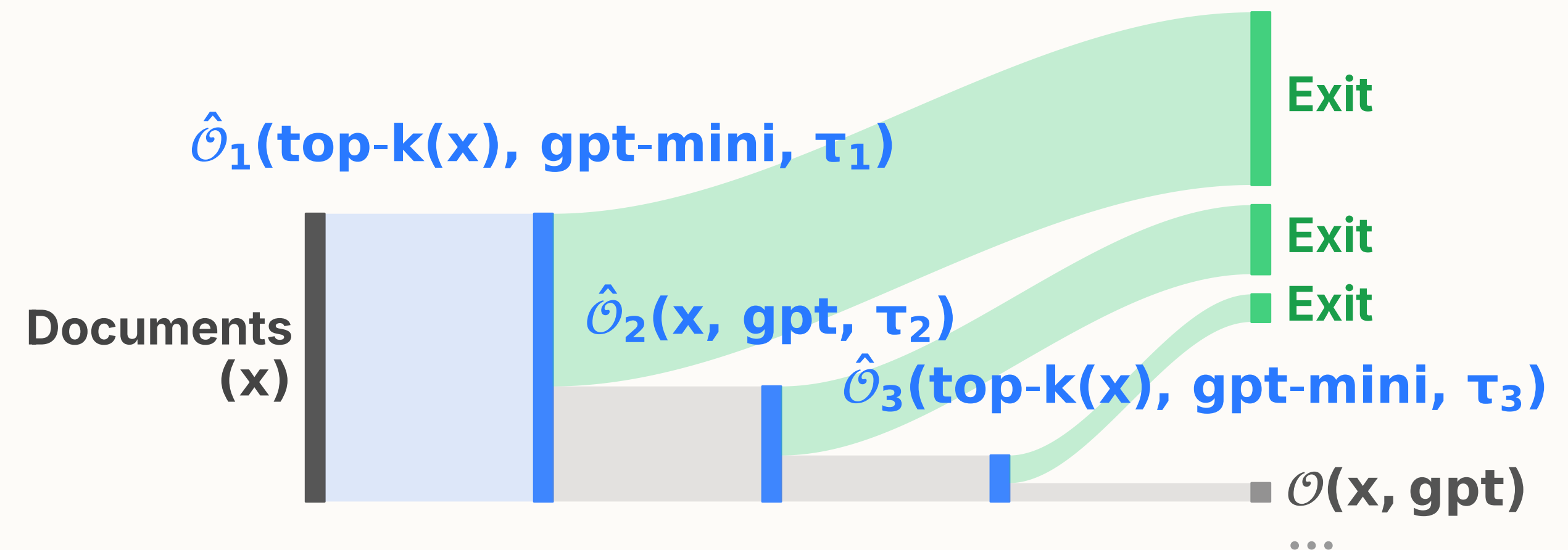


Revisiting Search: A *Task Cascade*

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 all possible cascades to find the cheapest that meets the target accuracy

Problem: Need to guarantee target accuracy on new samples!

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

Results:

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

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)

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

$X = \{X_1, \dots, X_i\}$ is defined

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

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

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

- ♦ LOTUS¹ 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).

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

²Liu 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.

	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!

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)
```

An iceberg floating in the ocean. The small tip above the water is labeled 'Structured data (Tables & SQL)'. The much larger, submerged part of the iceberg is labeled 'Unstructured data'.

Structured data
(Tables & SQL)

Unstructured data

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)*

An iceberg floating in the ocean. The small tip above the water is labeled 'Structured data (Tables & SQL)'. The much larger, submerged part of the iceberg is labeled 'Unstructured data'.

Structured data
(Tables & SQL)

Unstructured data

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!

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

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.

Key

1.

2.

3.

The screenshot displays the DocWrangler IDE interface. The top menu bar includes File, Edit, Help, Quick Save, and Info. The main workspace is titled 'DW_Analysis' and contains a prompt: 'extract all medications in this document {{ input.src }}'. Below the prompt, the 'Output Schema' shows a 'list' of 'medications'. The 'OUTPUT - Untitled Map 0' section displays a table view with columns for 'tgt', 'file', and 'medications'. The 'tgt' column shows word counts and averages, the 'file' column shows character counts and averages, and the 'medications' column shows an array of medication names. The right sidebar shows 'raw.json' with 'Available Keys' and 'Dataset Statistics'. The 'Dataset Statistics' section includes a 'Word Count Distribution' bar chart and a table of statistics: Documents (87), Average Words (1,626), Min Words (714), Max Words (3,376), and Std Deviation (465). The bottom of the interface shows a search bar and a list of search results.

File Edit Help Quick Save Info DocWrangler (calm-bear-a2d6lo0) Cost: \$2.15 Files Output Dataset

FILES

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

raw.json

DW_Analysis

Add Operation + Stop Run Fresh Run

extract all medications in this document {{ input.src }}

Output Schema

medications list

OUTPUT - Untitled Map 0 Console Table Visualize Input Distribution 87 in → 87 out 1.00x

Show/Hide Columns Reset Widths

Search in cell...

CHIEF COMPLAINT

Annual exam.

HISTORY OF PRESENT ILLNESS

Martha Collins is a 50-year-old female with a past medical history significant

medications

Array (2 items)

0: "lisinopril"

1: "lasix"

raw.json

Available Keys

Dataset Statistics

Documents: 87

Average Words: 1,626

Min Words: 714

Max Words: 3,376

Std Deviation: 465

Word Count Distribution

Search (min 5 characters)...

No matches

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

<

OverviewSystem Prompts15

Add Operation +

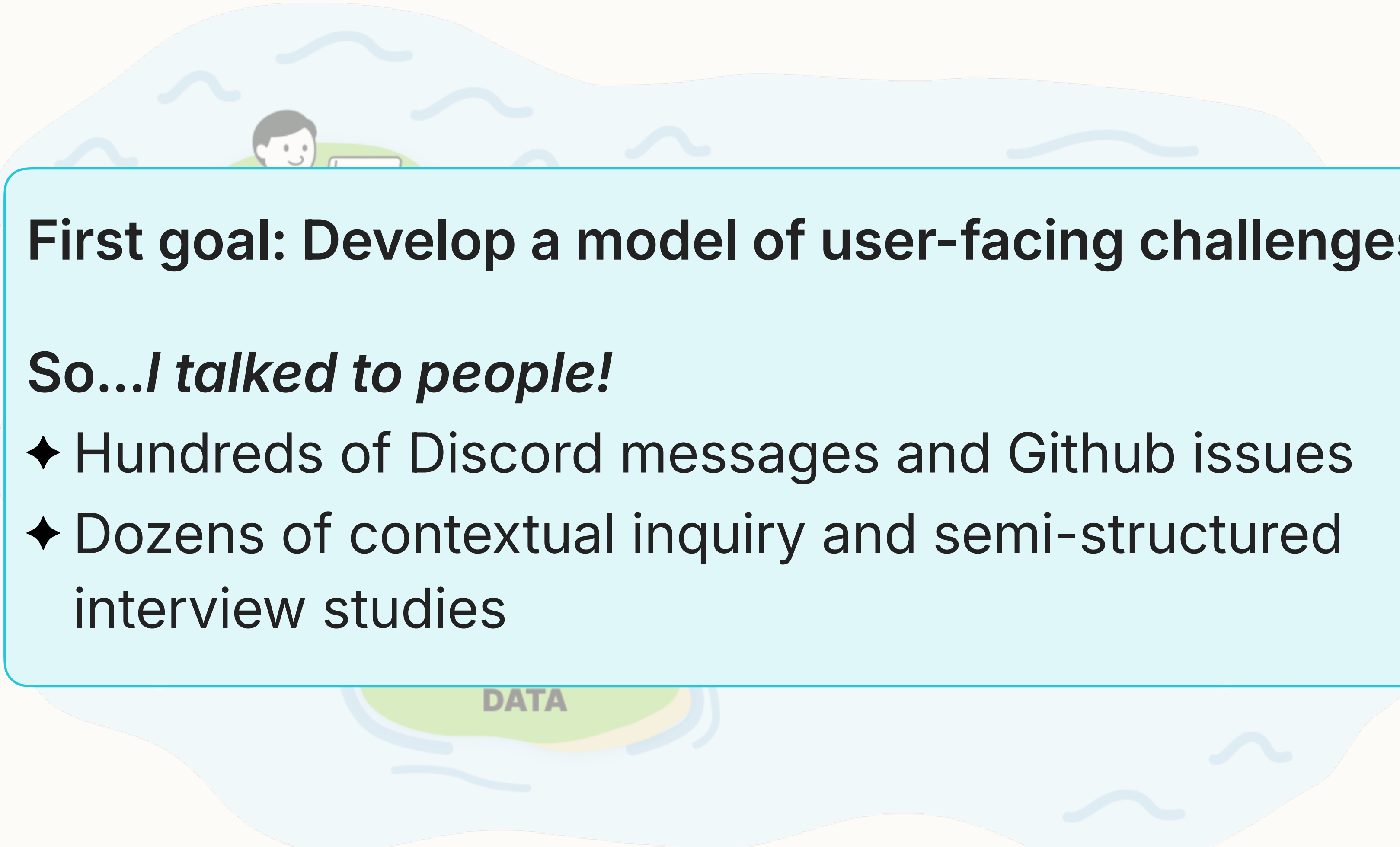
StopRun FreshRun

+ Add Operation

Speed > 20x

Key Idea: Three Gulfs Framework

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



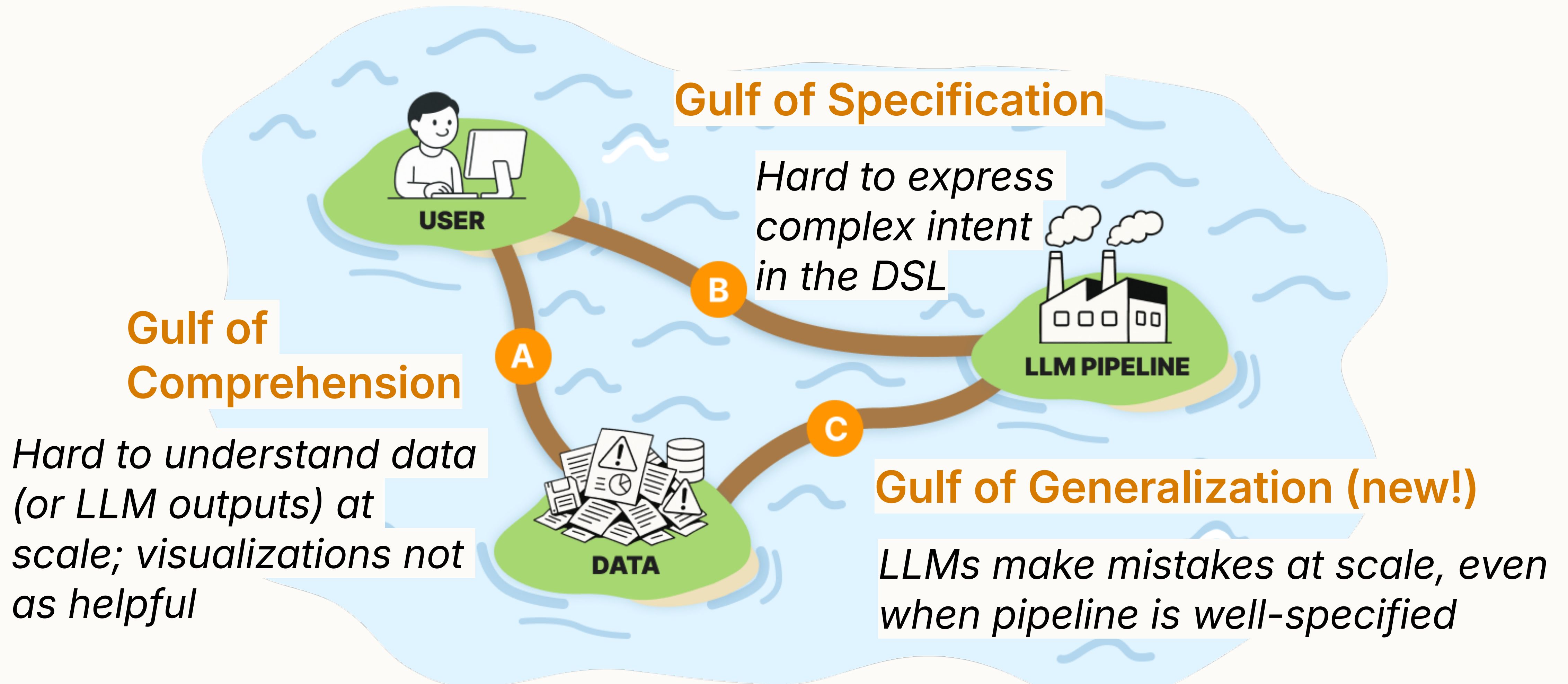
First goal: Develop a model of user-facing challenges.

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*. 🏆



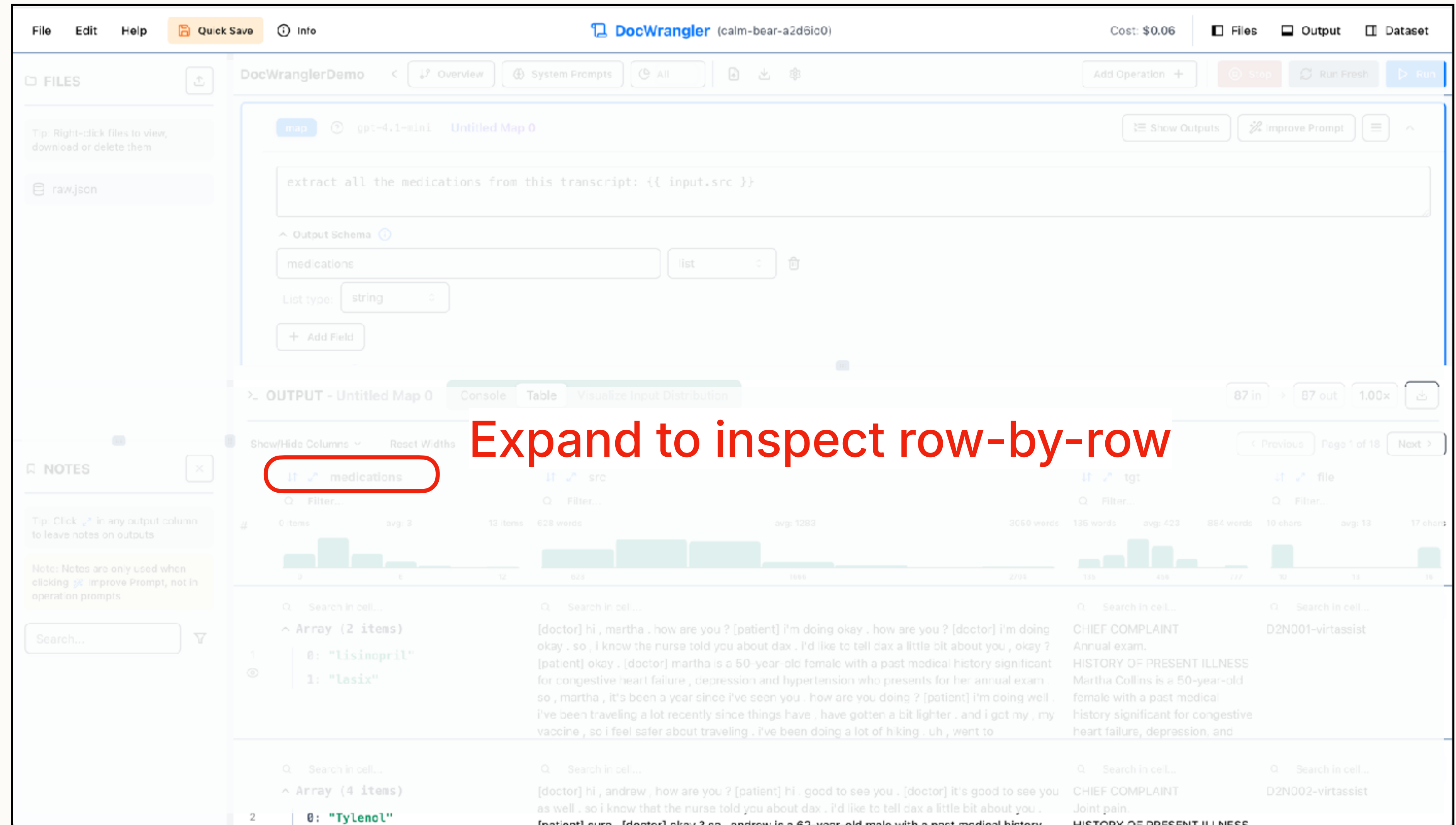
Illustrating the Gulf of Comprehension

Task

"Extract all  from each transcript"

Challenge

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



The screenshot shows the DocWrangler interface. At the top, there's a menu bar with File, Edit, Help, Quick Save, and Info. The main header displays 'DocWrangler (calm-bear-a2d6ic0)' and 'Cost: \$0.06'. Below the header, there's a 'FILES' section on the left with a tip: 'Tip: Right-click files to view, download or delete them'. The main area shows a 'map' operation with the prompt 'extract all the medications from this transcript: {{ input.src }}'. The 'Output Schema' section shows a field named 'medications' with a 'list' type and a 'string' list type. Below this, there's a 'Visualize Input Distribution' section with a table of statistics for various fields. The 'medications' field is highlighted with a red box, and a red arrow points to it with the text 'Expand to inspect row-by-row'. The table shows statistics for 'medications', 'src', 'tgt', and 'file'. The 'medications' field has 13 items, 628 words, and an average length of 1283. The 'src' field has 3069 words, the 'tgt' field has 135 words, and the 'file' field has 10 items and 17 words. The bottom section shows a preview of the output, which is an array of 2 items: 'Lisinopril' and 'lasix'.

Field	Items	Words	Avg
medications	13 items	628 words	avg: 1283
src	3069 words		
tgt	135 words	avg: 423	884 words
file	10 items	avg: 13	17 words

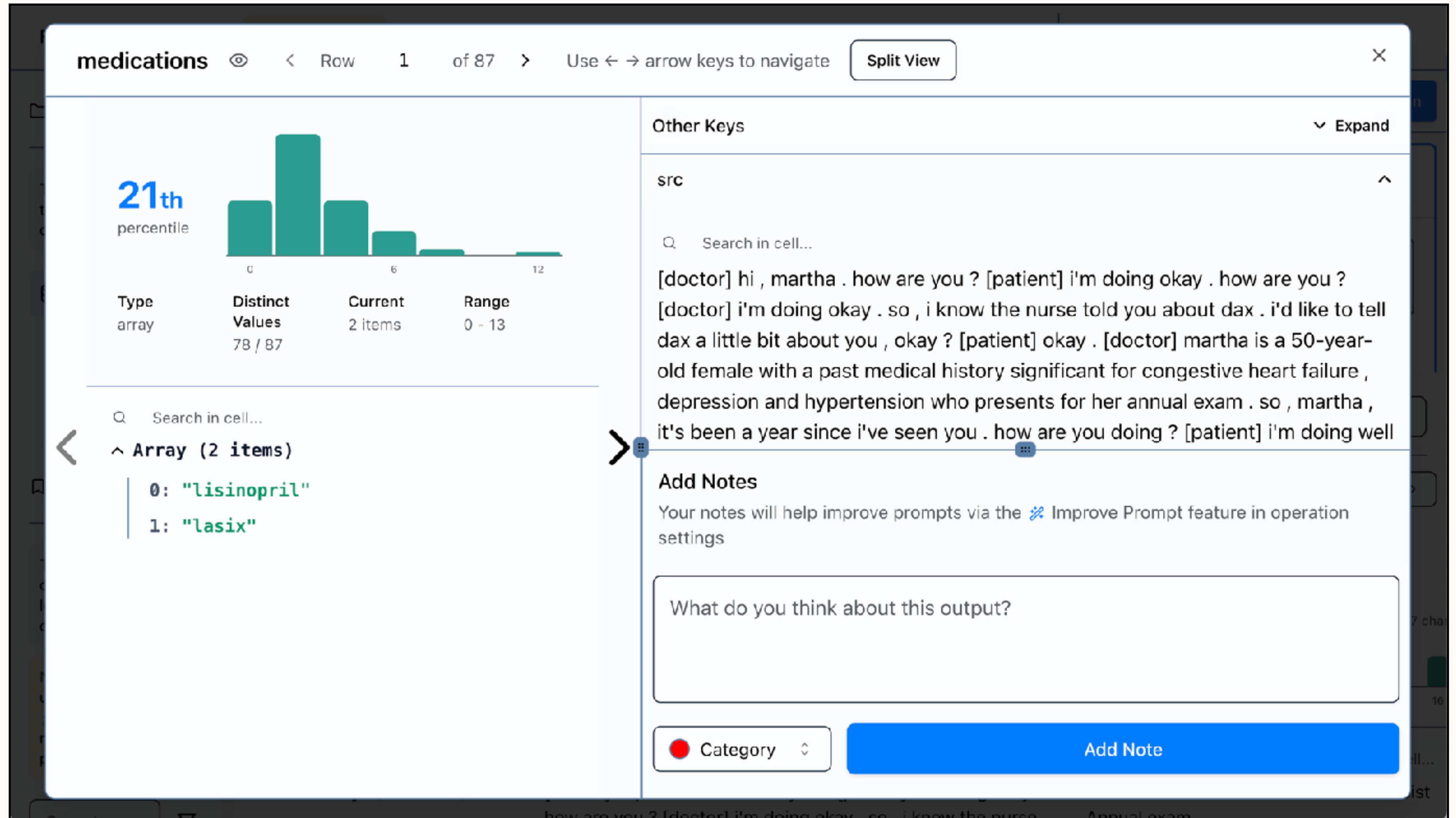
In-Situ Notetaking to *Comprehend* Data Better

Task

"Extract all  from each transcript"

Challenge

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



The screenshot displays a data analysis interface for a dataset named "medications". At the top, it shows "Row 1 of 87" and a "Split View" button. The main area is divided into three sections:

- Histogram:** A bar chart showing the distribution of values. The x-axis is labeled with 0, 6, and 12. The y-axis is labeled "21th percentile".
- Statistics Table:** A table with four columns: Type, Distinct Values, Current, and Range. The data is as follows:

Type	Distinct Values	Current	Range
array	78 / 87	2 items	0 - 13
- Text Analysis Panel:** A panel on the right titled "Other Keys" with an "Expand" button. It contains a search bar "Search in cell..." and a text input area with the following text:

[doctor] hi , martha . how are you ? [patient] i'm doing okay . how are you ?
[doctor] i'm doing okay . so , i know the nurse told you about dax . i'd like to tell
dax a little bit about you , okay ? [patient] okay . [doctor] martha is a 50-year-
old female with a past medical history significant for congestive heart failure ,
depression and hypertension who presents for her annual exam . so , martha ,
it's been a year since i've seen you . how are you doing ? [patient] i'm doing well

Below the text input area, there is an "Add Notes" section with a text box containing "What do you think about this output?". At the bottom, there is a "Category" dropdown menu and a blue "Add Note" button.

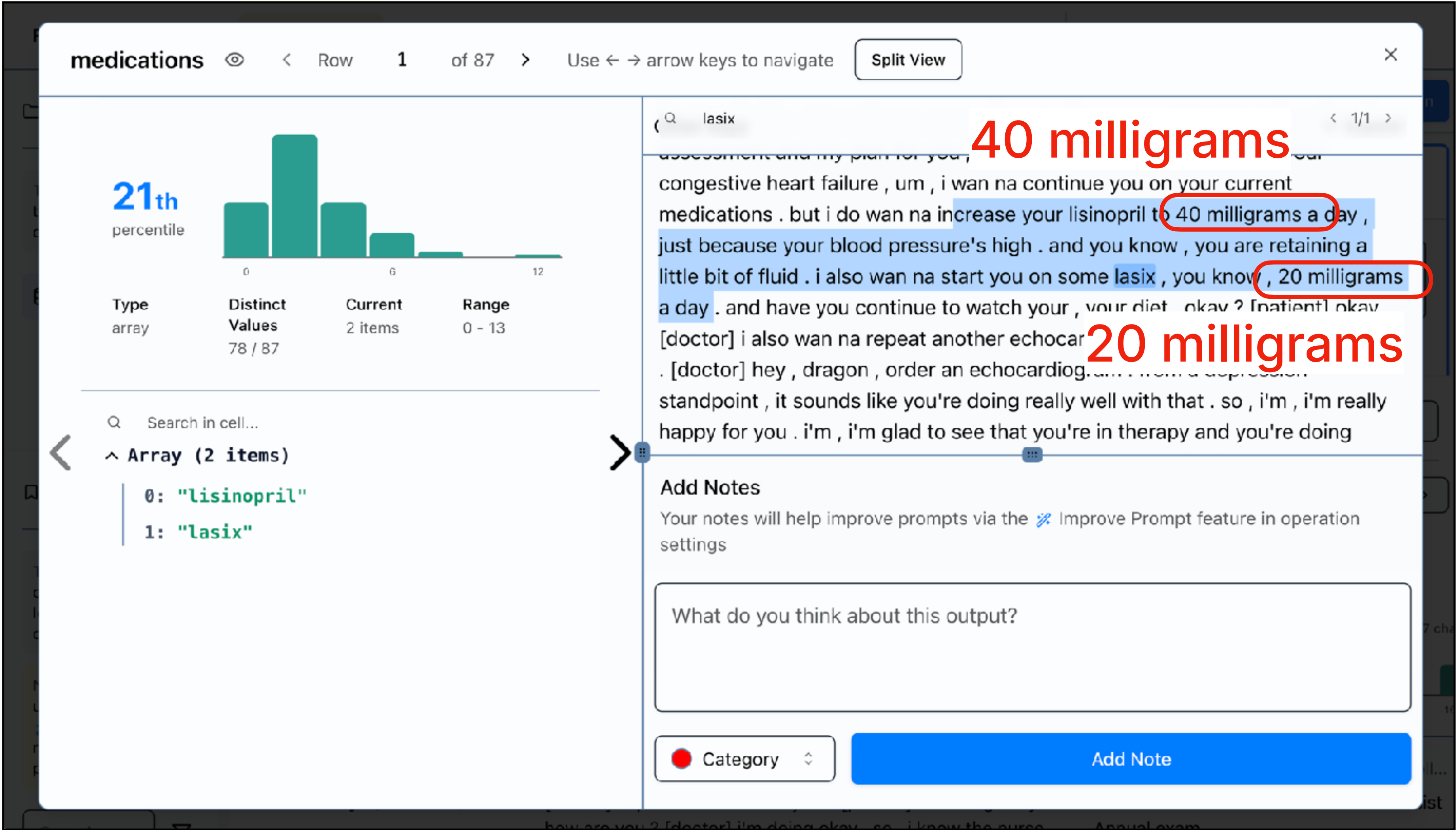
In-Situ Notetaking to *Comprehend* Data Better

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

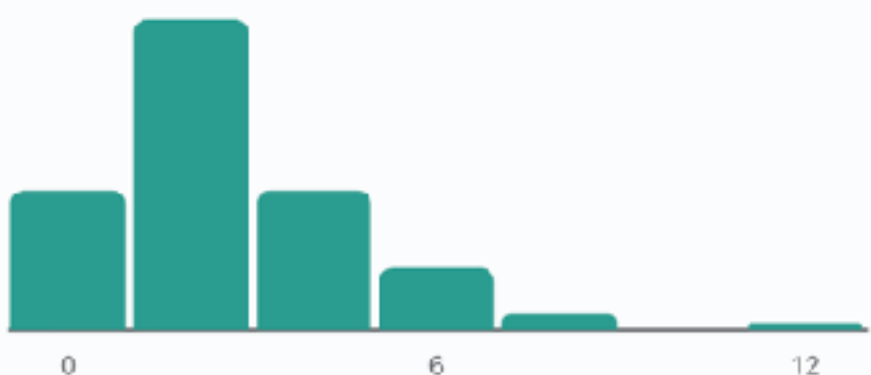
Discovering what you really want & the long tail of LLM behaviors

medications

Row 1 of 87

Split View

21th percentile



Type

array

Distinct Values

78 / 87

Current

2 items

Range

0 - 13

Search in cell...

Array (2 items)

0: "lisinopril"

1: "lasix"

lasix

40 milligrams

20 milligrams

Add Notes

i want the dosages included as well, like "lisinopril 40 mg/day" and "lasix 20 mg/day"

Category

Add Note

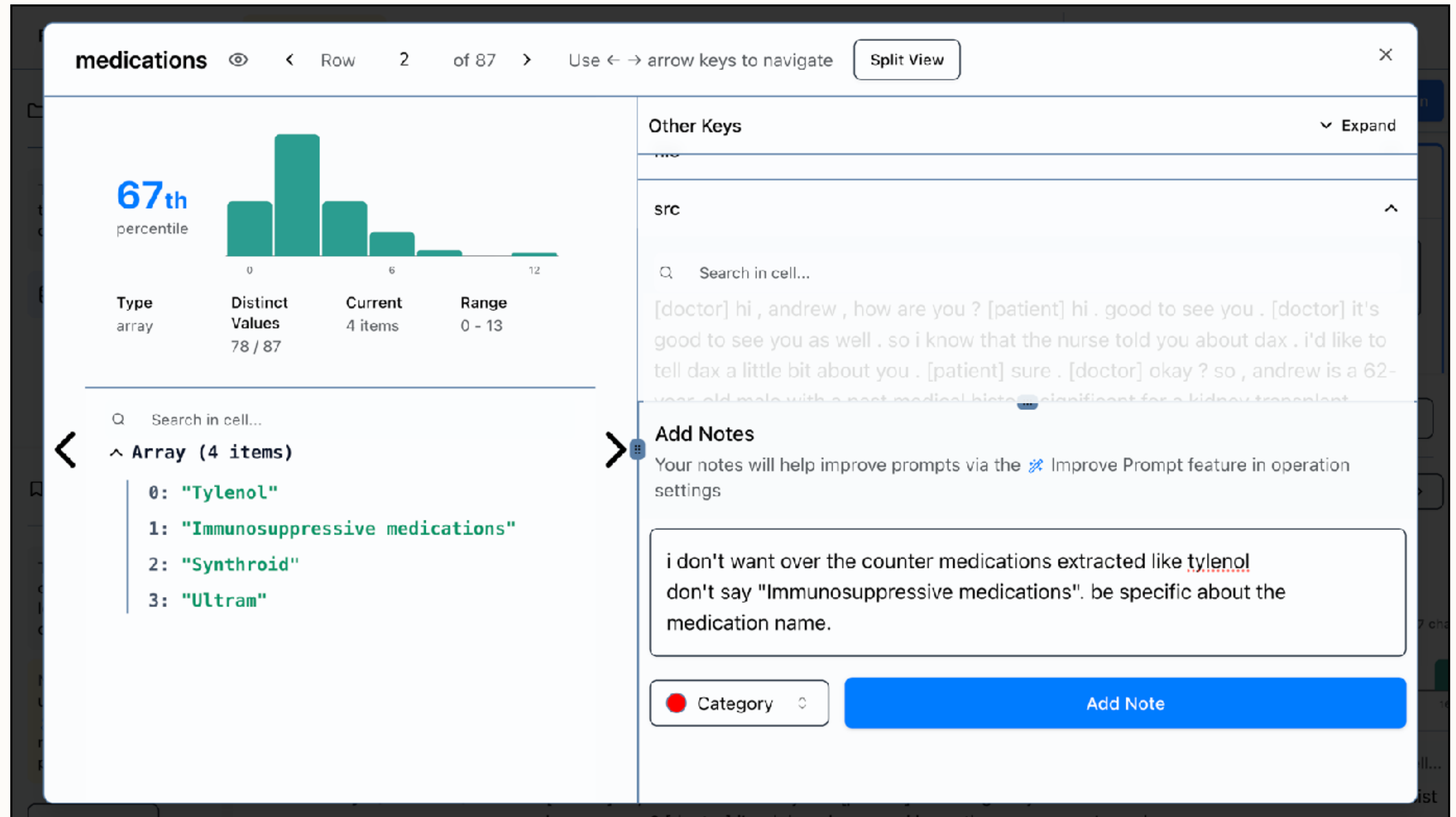
In-Situ Notetaking to *Comprehend* Data Better

Task

"Extract all  from each transcript"

Challenge

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



The screenshot displays a data visualization tool interface. At the top, it shows 'medications' with navigation controls and a 'Split View' button. The main area features a histogram titled '67th percentile' showing the distribution of medication counts. Below the histogram, a table provides summary statistics: Type (array), Distinct Values (78 / 87), Current (4 items), and Range (0 - 13). A search bar is present above a list of medications: 0: "Tylenol", 1: "Immunosuppressive medications", 2: "Synthroid", and 3: "Ultram". On the right side, there is a section for 'Other Keys' with an 'Expand' button. Below this, a search bar is followed by a text snippet from a transcript. At the bottom right, there is an 'Add Notes' section with a text input area containing a note about not wanting over-the-counter medications like Tylenol and a blue 'Add Note' button.

Type	Distinct Values	Current	Range
array	78 / 87	4 items	0 - 13

0: "Tylenol"
1: "Immunosuppressive medications"
2: "Synthroid"
3: "Ultram"

Add Notes
Your notes will help improve prompts via the [Improve Prompt](#) feature in operation settings

i don't want over the counter medications extracted like tylenol
don't say "Immunosuppressive medications". be specific about the medication name.

Category 

Add Note

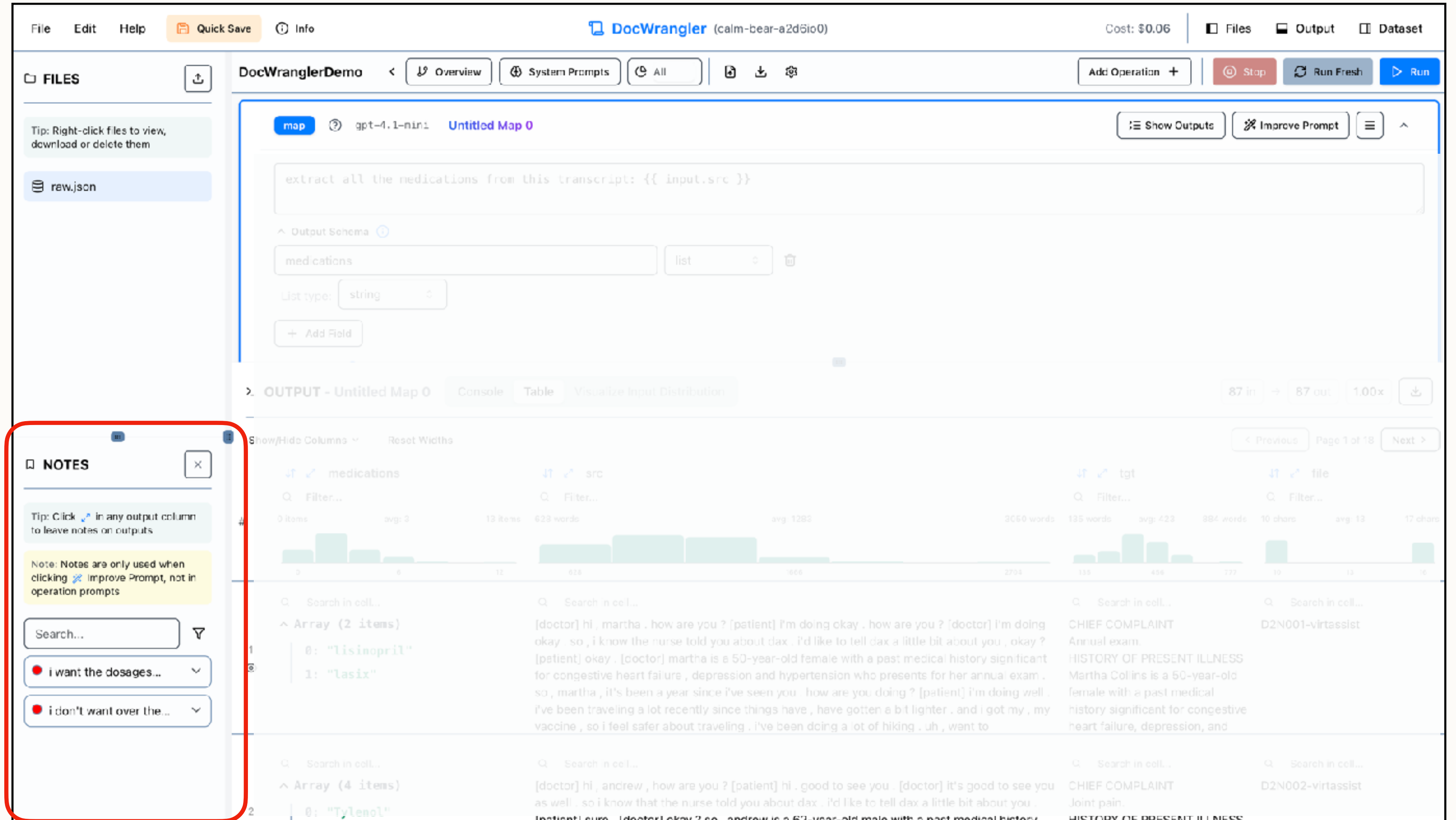
In-Situ Notetaking to *Comprehend* Data Better

Task

"Extract all  from each transcript"

Challenge

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



The screenshot displays the DocWrangler interface. At the top, there's a menu bar with 'File', 'Edit', 'Help', 'Quick Save', and 'Info'. The main header shows 'DocWranglerDemo' with navigation tabs for 'Overview', 'System Prompts', and 'All'. A 'Cost: \$0.06' indicator is present. On the left, a 'FILES' sidebar shows a file named 'raw.json'. The central workspace is titled 'Untitled Map 0' and contains a map operation with a prompt: 'extract all the medications from this transcript: {{ input.src }}'. Below the prompt, the 'Output Schema' is defined with a field 'medications' of type 'list'. The 'OUTPUT - Untitled Map 0' section shows a table view with columns for 'medications', 'src', 'tgt', and 'file'. Each column has a histogram and a search bar. A sidebar on the left, titled 'NOTES', is highlighted with a red box. It contains a search bar and two notes: 'i want the dosages...' and 'i don't want over the...'. The main table displays two rows of data. The first row shows medications 'lisinopril' and 'lasix' for a patient named martha. The second row shows 'Tylenol' for a patient named andrew.

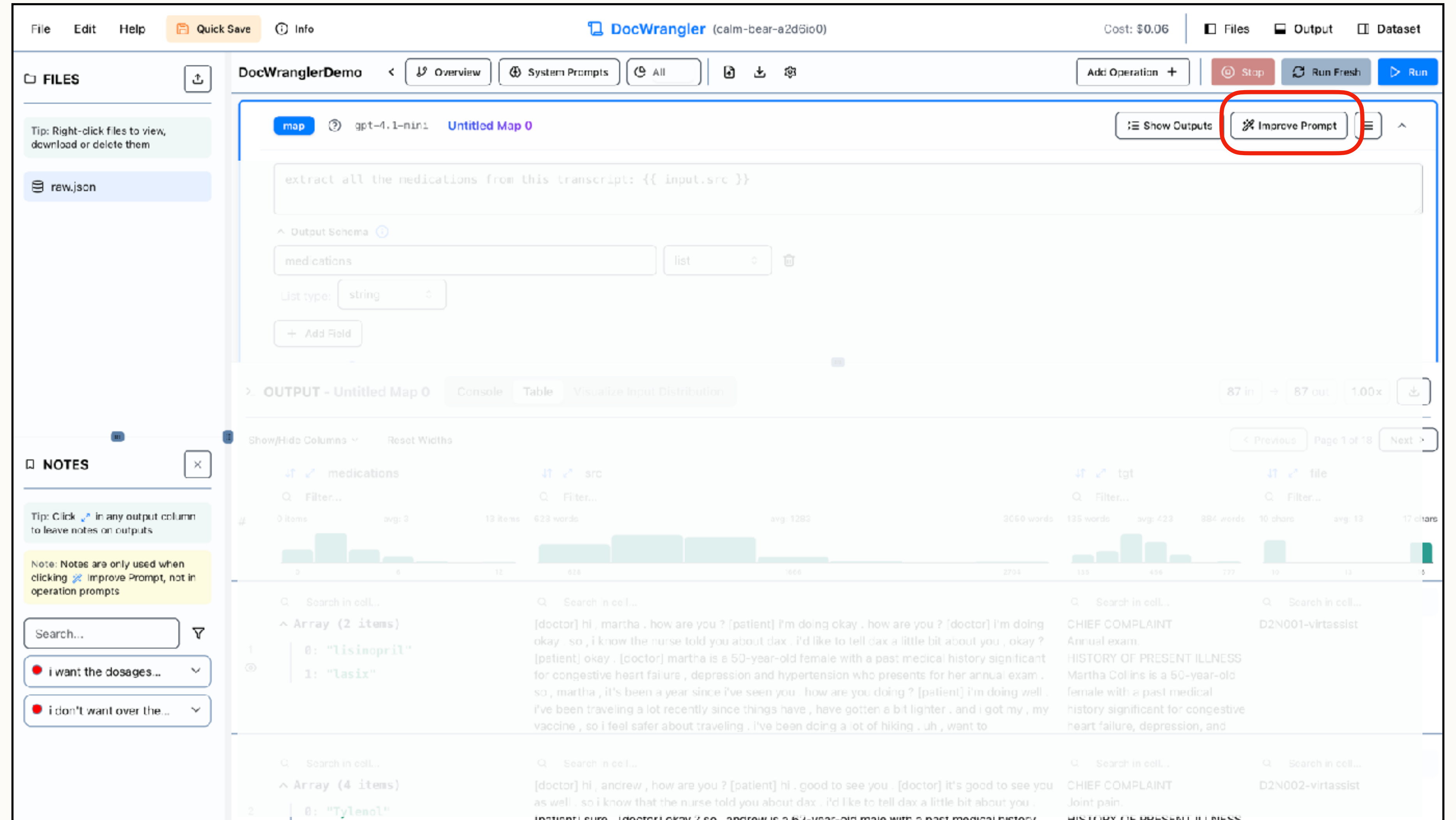
Illustrating the Gulf of Specification

Task

"Extract all  from each transcript"

Challenge

Turning your notes and requirements into better prompts



The screenshot displays the DocWrangler interface. At the top, the title bar shows 'DocWrangler (calm-bear-a2d5io0)' with a cost of '\$0.06'. The main workspace is titled 'DocWranglerDemo' and contains a 'map' operation named 'Untitled Map 0'. The prompt for this operation is 'extract all the medications from this transcript: {{ input.src }}'. The output schema is defined with a field 'medications' of type 'list', and the list type is set to 'string'. The output is displayed in a table view, showing two rows of data. The first row contains a list of medications: 'lisinopril' and 'lasix'. The second row contains 'Tylenol'. The interface also includes a 'FILES' sidebar on the left with a file named 'raw.json', and a 'NOTES' sidebar on the right with a search bar and two notes: 'i want the dosages...' and 'i don't want over the...'. A red circle highlights the 'Improve Prompt' button in the top right corner of the map operation's configuration area.

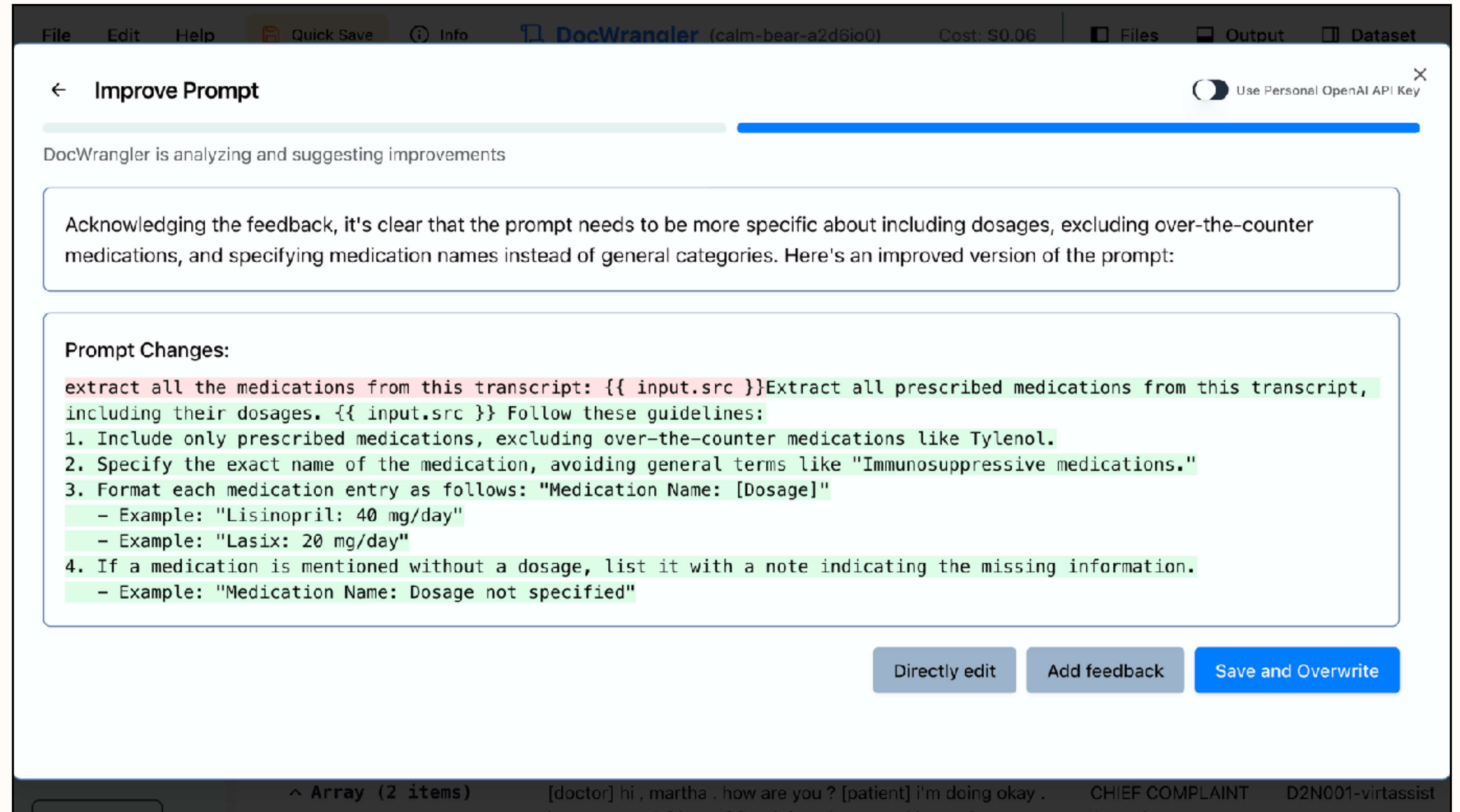
Interactive Assistance in *Specifying* Operations

Task

"Extract all 
from each
transcript"

Challenge

Turning your
notes and
requirements
into better
prompts



File Edit Help Quick Save Info DocWrangler (calm-bear-a2d6io0) Cost: \$0.06 Files Output Dataset

← Improve Prompt Use Personal OpenAI API Key

DocWrangler is analyzing and suggesting improvements

Acknowledging the feedback, it's clear that the prompt needs to be more specific about including dosages, excluding over-the-counter medications, and specifying medication names instead of general categories. Here's an improved version of the prompt:

Prompt Changes:

```
extract all the medications from this transcript: {{ input.src }}Extract all prescribed medications from this transcript, including their dosages. {{ input.src }} Follow these guidelines:
```

1. Include only prescribed medications, excluding over-the-counter medications like Tylenol.
2. Specify the exact name of the medication, avoiding general terms like "Immunosuppressive medications."
3. Format each medication entry as follows: "Medication Name: [Dosage]"
 - Example: "Lisinopril: 40 mg/day"
 - Example: "Lasix: 20 mg/day"
4. If a medication is mentioned without a dosage, list it with a note indicating the missing information.
 - Example: "Medication Name: Dosage not specified"

Directly edit Add feedback Save and Overwrite

^ Array (2 items) [doctor] hi , martha . how are you ? [patient] i'm doing okay . CHIEF COMPLAINT D2N001-virtassist

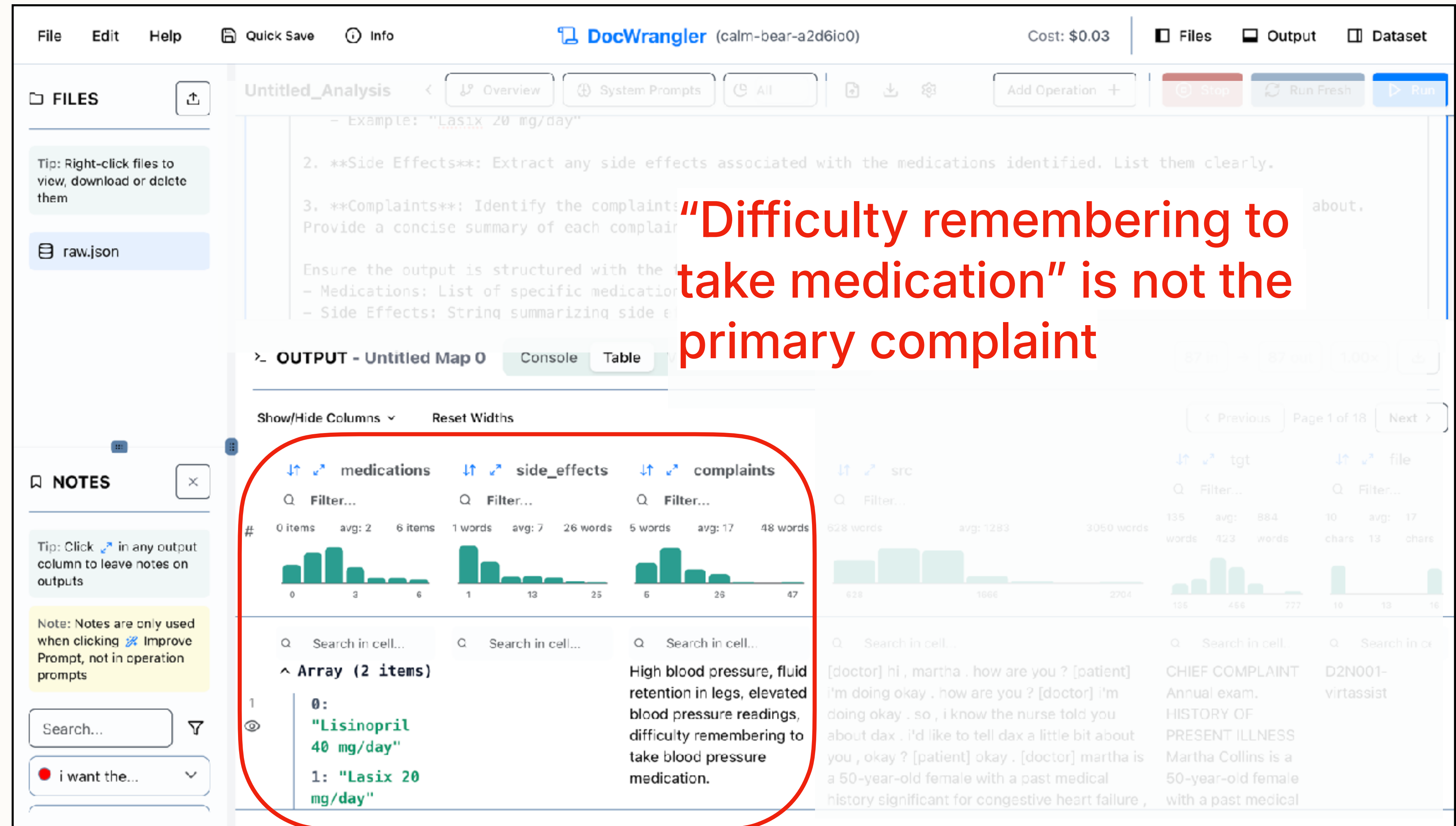
Illustrating the Gulf of Generalization

Task

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

Challenge

Too complex for a single LLM call



DocWrangler (calm-bear-a2d6io0) Cost: \$0.03

File Edit Help Quick Save Info

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...

i want the...

Untitled_Analysis

Overview System Prompts All

Add Operation + Stop Run Fresh Run

Example: "Lasix 20 mg/day"

2. ****Side Effects****: Extract any side effects associated with the medications identified. List them clearly.

3. ****Complaints****: Identify the complaints. Provide a concise summary of each complaint.

Ensure the output is structured with the following schema:

- Medications: List of specific medication names and dosages.
- Side Effects: String summarizing side effects.
- Complaints: String summarizing patient complaints.

OUTPUT - Untitled Map 0

Show/Hide Columns Reset Widths

medications	side_effects	complaints
0 items avg: 2 6 items	1 words avg: 7 26 words	5 words avg: 17 48 words
0 3 6	1 13 25	6 26 47
0: "Lisinopril 40 mg/day"	High blood pressure, fluid retention in legs, elevated blood pressure readings, difficulty remembering to take blood pressure medication.	about.
1: "Lasix 20 mg/day"		

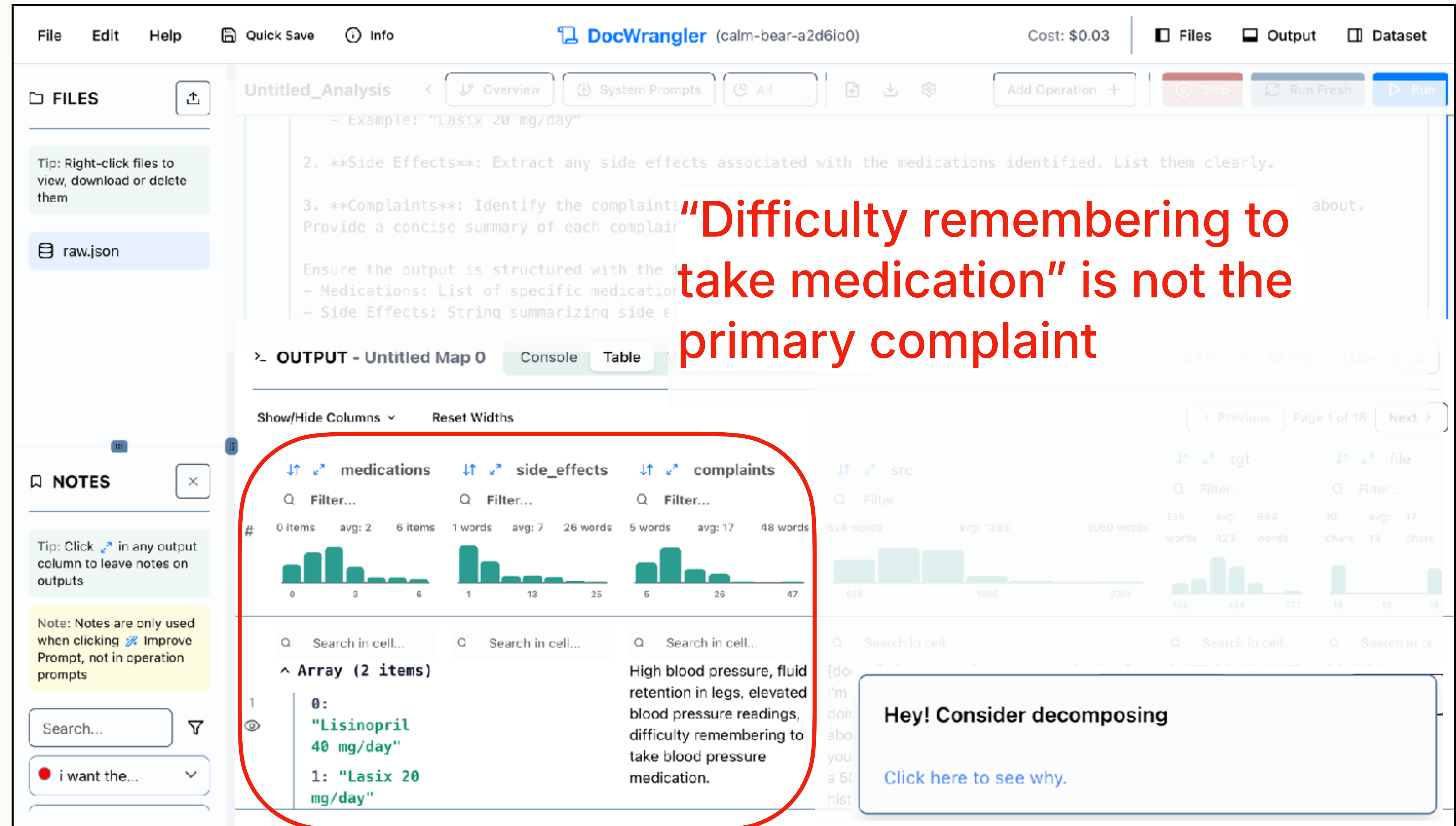
Automatic Decomposition to Help *Generalize*

Task

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

Challenge

Too complex for a single LLM call



The screenshot shows the DocWrangler interface with a task, a challenge, and a solution. The task is to extract side effects and patient complaints from transcripts. The challenge is that this is too complex for a single LLM call. The solution is to use automatic decomposition to break the task into smaller, manageable steps.

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

Challenge: Too complex for a single LLM call

Solution: Automatic decomposition to help *Generalize*

The interface shows a task description, a challenge, and a solution. The task is to extract side effects and patient complaints from transcripts. The challenge is that this is too complex for a single LLM call. The solution is to use automatic decomposition to break the task into smaller, manageable steps.

The interface also shows a list of files, a notes section, and a search bar. The main content area displays a table of results with columns for medications, side effects, and complaints. The table is filtered to show only the results for the task.

Medications: "Lisinopril 40 mg/day", "Lasix 20 mg/day"

Side Effects: High blood pressure, fluid retention in legs, elevated blood pressure readings, difficulty remembering to take blood pressure medication.

Complaints: (None listed)

Hey! Consider decomposing
[Click here to see why.](#)

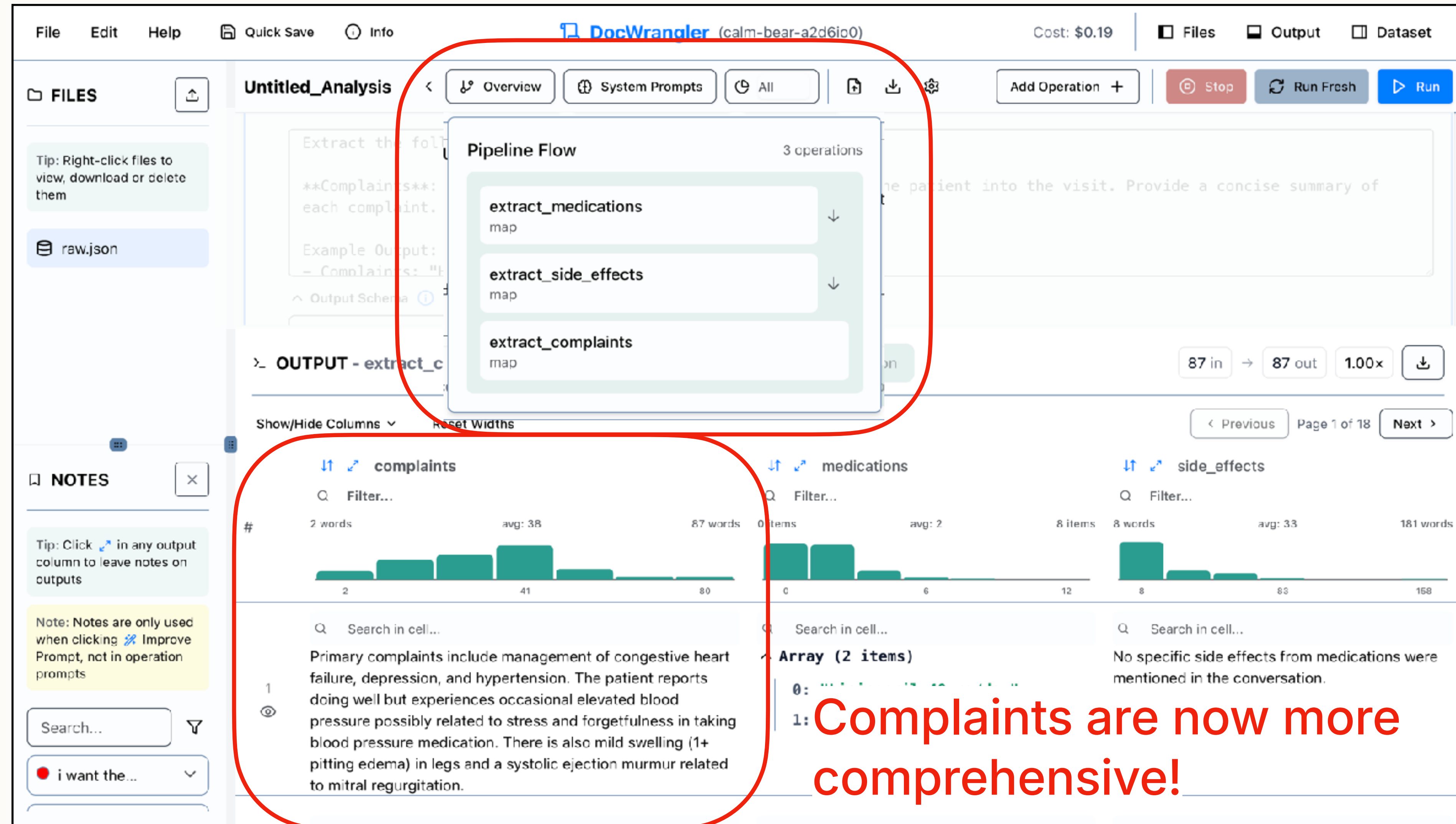
Automatic Decomposition to Help *Generalize*

Task

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

Challenge

Too complex for a single LLM call

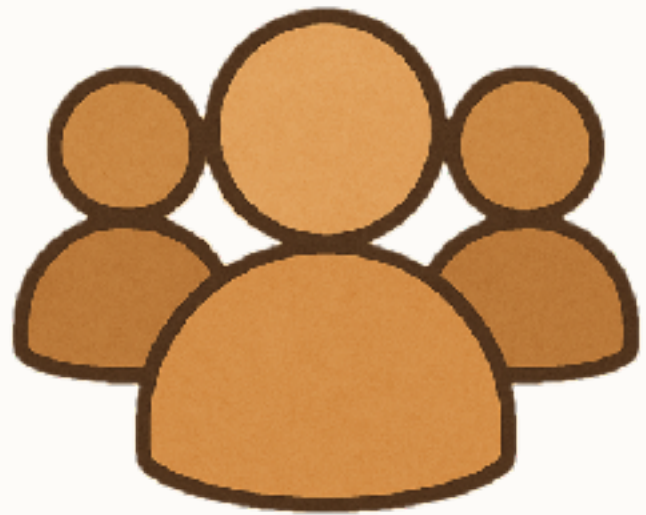


The screenshot displays the DocWrangler interface for an analysis titled "Untitled_Analysis". The interface includes a top navigation bar with "File", "Edit", "Help", "Quick Save", and "Info" options. A status bar at the top right shows the cost as "\$0.19" and tabs for "Files", "Output", and "Dataset". The main workspace is divided into several sections:

- FILES:** A sidebar on the left showing a file named "raw.json" with a tip: "Tip: Right-click files to view, download or delete them".
- System Prompts:** A section on the left containing a prompt: "Extract the following... **Complaints:**: each complaint. Example Output: - Complaints: '...'" and an "Output Schema" link.
- Pipeline Flow:** A central panel showing a sequence of three operations: "extract_medications" (map), "extract_side_effects" (map), and "extract_complaints" (map), connected by downward arrows.
- OUTPUT - extract_c:** A section showing the results of the pipeline, including a "Show/Hide Columns" dropdown and a "Reset Widths" button.
- NOTES:** A sidebar on the left with a tip: "Tip: Click [icon] in any output column to leave notes on outputs" and a note: "Note: Notes are only used when clicking [icon] Improve Prompt, not in operation prompts". It includes a search bar and a dropdown menu with the option "i want the...".
- complaints:** A table view showing a single row of data with a filter bar and a search bar. The data includes a primary complaint: "Primary complaints include management of congestive heart failure, depression, and hypertension. The patient reports doing well but experiences occasional elevated blood pressure possibly related to stress and forgetfulness in taking blood pressure medication. There is also mild swelling (1+ pitting edema) in legs and a systolic ejection murmur related to mitral regurgitation."
- medications:** A table view showing a single row of data with a filter bar and a search bar. The data includes a note: "No specific side effects from medications were mentioned in the conversation."
- side_effects:** A table view showing a single row of data with a filter bar and a search bar. The data includes a note: "No specific side effects from medications were mentioned in the conversation."

At the bottom right, a red text overlay states: "Complaints are now more comprehensive!"

DocWrangler User Study: Design



10 participant think-aloud lab study

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



Online deployment

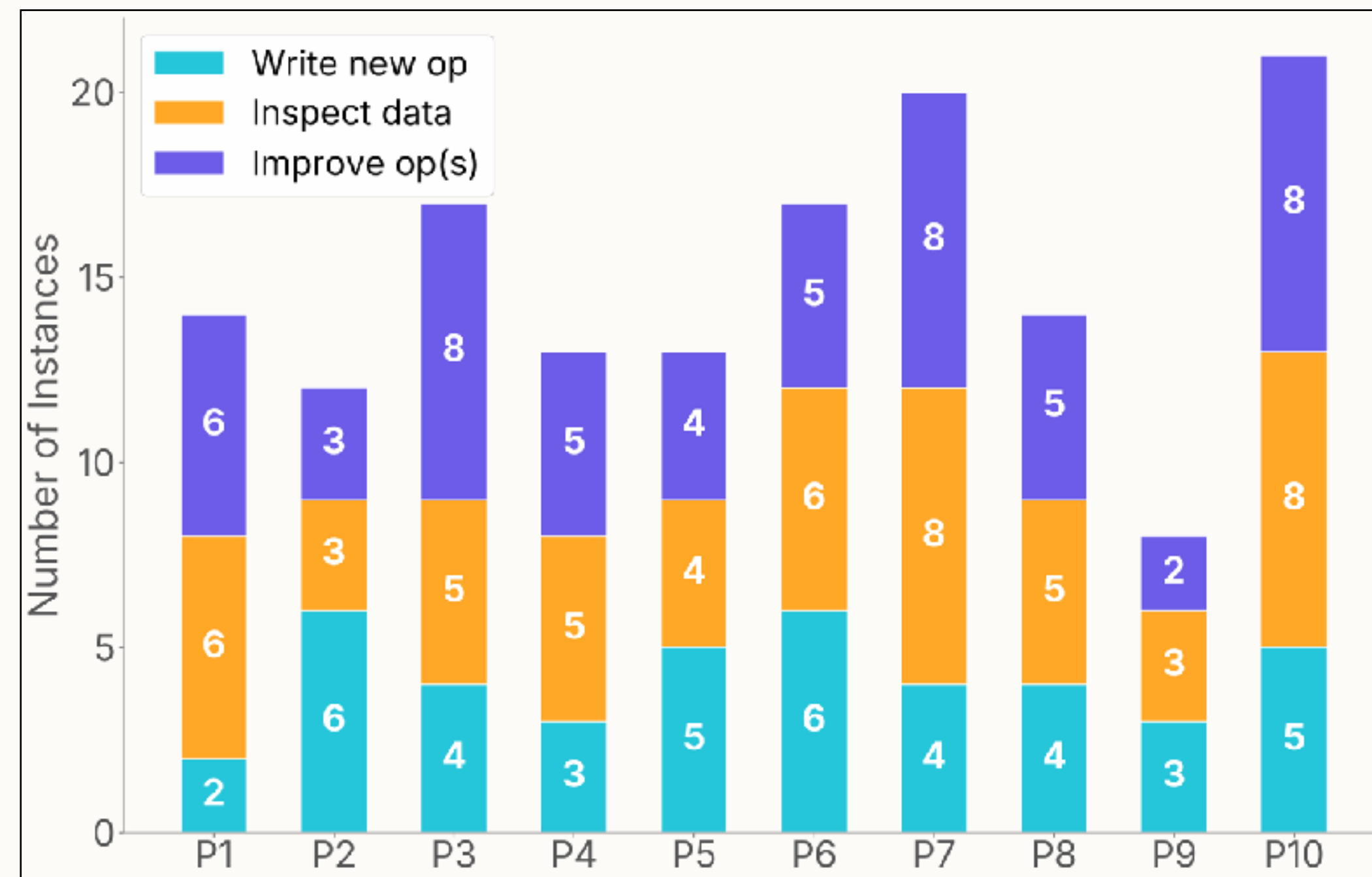
- ◆ BYO API keys
- ◆ ~4,500 pipelines; ~30GB of data (~5 billion words!)
- ◆ 30+ S&P 500 industries represented

Quantitative and qualitative analysis

What Did We Learn?

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

Participant Activity Distribution



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

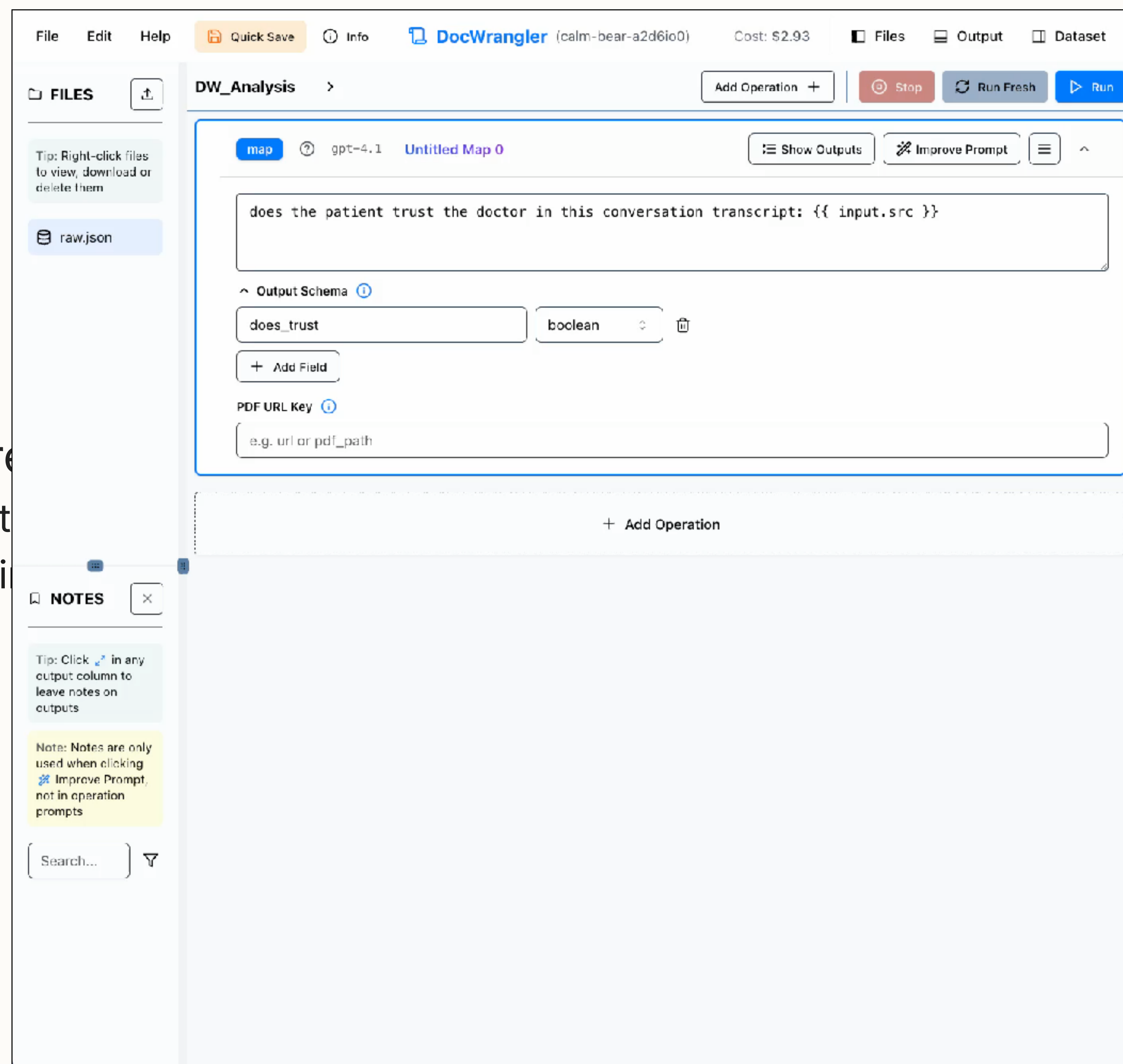
The screenshot shows a user interface for a query builder. At the top, there is a navigation bar with a blue 'map' button, a help icon, the text 'gpt-5-nano', a 'summary' button, and two buttons: 'Show Outputs' and 'Improve Prompt'. Below this is a large text input field containing the prompt 'summarize signs of distrust in {{input.src}}'. Under the input field is an 'Output Schema' section, which is currently expanded. It contains a table with one row: 'summary' in the first column and 'string' in the second column. To the right of the 'string' type is a trash icon. Below the table is a button labeled '+ Add Field'.

1. Users iterate

2. Users don't re

- * E.g., they creat

- * Even when tryi



ays

1. Users iterate

2. Users don't re

- ✱ E.g., they creat
- ✱ And open-ende

3. Underspecific

File Edit Help Quick Save Info DocWrangler (calm-bear-a2d6io0) Cost: \$3.32 Files Output Dataset

DW_Analysis Add Operation + Stop Run Fresh Run

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

raw.json

map gpt-4.1 Untitled Map 0 Show Outputs Improve Prompt

does the patient trust the doctor in this conversation transcript: {{ input.src }}

Output Schema

does_trust boolean reasoning string

+ Add Field

OUTPUT - Untitled Map 0 Console Table Visualize Input Distribution 87 in → 87 out 1.00x

Show/Hide Columns Reset Widths

does_trust reasoning src

Filter... Filter... Filter...

2 distinct values 59 words avg: 104 182 words 628 words

true false 58 113 167 628

Search in cell...

1 Throughout the conversation, the patient responds affirmatively and cooperatively to the doctor's questions, advice, and suggestions. The patient voluntarily shares personal and health information, admits difficulties (e.g., forgetting to take medication), honestly details symptoms, and engages in follow-up questions (e.g., about pill timing), indicating openness and willingness to accept the doctor's guidance. She accepts medical recommendations, confirms actions taken, and expresses appreciation at the end ('good seeing you too'). There is no evidence of

Search in cell...

2 evident by the patient's cooperative responses ('sure,' 'you got it,' 'perfect'), willingness to answer questions thoroughly, lack of hesitation or resistance to the doctor's suggestions, and no expressed doubt or concern about the doctor's decisions. The patient follows the doctor's lead, responds affirmatively to each part of the care plan, and shows comfort with the examination and assessment process. While the patient does not verbalize explicit trust, the patient's actions, tone, and consistent agreement to the doctor's recommendations (such as taking prescribed

Search in cell...

Search in cell...

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...

ays

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

ucbepic / docetl

<> Code

Issues 28

Pull requests

A system for agentic LLM-powered data processing ETL

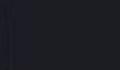
docetl.org

MIT license

3.1k stars 320 forks 26 watching

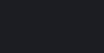
112 Branches 15 Tags Activity

Contributors 28



docetl

36 Online • 472 Members



Contributors 28



[+ 14 contributors](#)

Using DocWrangler to Process Blog Posts



vishal
1.02K subscribers



The screenshot shows the DocWrangler web application interface. The top navigation bar includes a search bar and several tabs. The main interface is divided into three panels. The left panel, titled 'FILES', contains a file explorer showing a list of documents, including '2021-04-12-fastai-chapter-6.md' and '2021-04-25-fastai-chapter-6-bias-classifier.md'. Below the file list is a 'NOTES' section with a list of notes, including 'The chunk is small enough where the takeaways are pretty...'. The center panel features a circular profile picture of a man with a beard and glasses. The right panel, titled 'Blog Post LLM Context', displays a prompt for summarizing information from a document. Below this, the '2. OUTPUT - Untitled Map 0' section shows a visualization of the output, including a bar chart and a table of data. The bar chart has a title 'key_points' and shows a distribution of values. The table below it has columns for 'key_points', 'filename', and 'markdown'. The first row of the table shows 'key_points' as 'The chunk is small enough where the takeaways are pretty...', 'filename' as '2021-04-12-fastai-chapter-6.md', and 'markdown' as 'The chunk is small enough where the takeaways are pretty...'. The second row shows 'key_points' as 'No explanatory text is in this chunk, so the LLM has inferred...', 'filename' as '2021-04-12-fastai-chapter-6.md', and 'markdown' as 'No explanatory text is in this chunk, so the LLM has inferred...'. The third row shows 'key_points' as 'The document title is not included in this set of key points.', 'filename' as '2021-04-12-fastai-chapter-6.md', and 'markdown' as 'The document title is not included in this set of key points...'. The fourth row shows 'key_points' as 'The key points are valid and they match the Markdown chunk...', 'filename' as '2021-04-12-fastai-chapter-6.md', and 'markdown' as 'The key points are valid and they match the Markdown chunk...'. The fifth row shows 'key_points' as 'The key points here are unnecessary and somewhat...', 'filename' as '2021-04-12-fastai-chapter-6.md', and 'markdown' as 'The key points here are unnecessary and somewhat...'. The sixth row shows 'key_points' as 'I want document title removed from the LLM prompt.', 'filename' as '2021-04-12-fastai-chapter-6.md', and 'markdown' as 'I want document title removed from the LLM prompt...'. The bottom of the interface shows a search bar and a list of documents, including '2021-04-12-fastai-chapter-6.md' and '2021-04-25-fastai-chapter-6-bias-classifier.md'.



@FKShunya

1 month ago

nice to see you back , dacewrangler +1







Reply



@alexkelly757

1 month ago

I was wondering about resolve myself , decided to go for reduce . It was interesting your thoughts process on map and reduce







Reply



@wisha_camer

1 month ago

Thanks I'll try out gp5 nano for sure, I need to experiment with more models. Regarding the final map: the reason aggregating across multiple documents, it was summarizing within a single document. But I'm sure conceptually I







1 reply

The screenshot shows a YouTube video player with the title "Building a FastHTML + LanceDB Search App Using Solvelt (Jeremy H...)". The video content is a code walkthrough for a search application. The code shown is as follows:

```
import lancedb
import numpy as np
from lancedb.pydantic import LanceModel, Vector
from lancedb.embeddings import get_registry

class Email(LanceModel):
    id: str
    text: str
    embedding: Vector[1536]
```

An iceberg floating in the ocean. The small tip above the water is labeled 'Structured data (Tables & SQL)'. The much larger, submerged part of the iceberg is labeled 'Unstructured data'.

Structured data
(Tables & SQL)

Unstructured data

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*

An iceberg floating in the ocean. The small tip above the water is labeled 'Structured data (Tables & SQL)'. The much larger, submerged part of the iceberg is labeled 'Unstructured data'.

Structured data
(Tables & SQL)

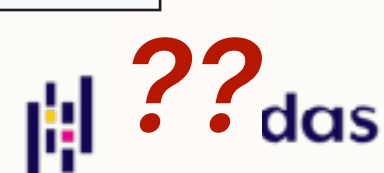
Unstructured data

Today's Talk

4. Where do we go from here?

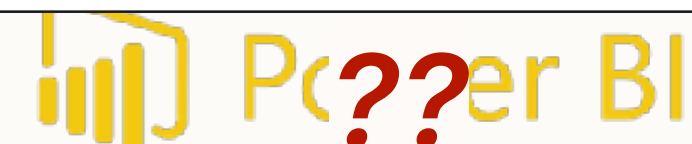
Today: *Cusp* of Unstructured Data Analysis

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



Inner Loop

Help the user ask the right questions



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

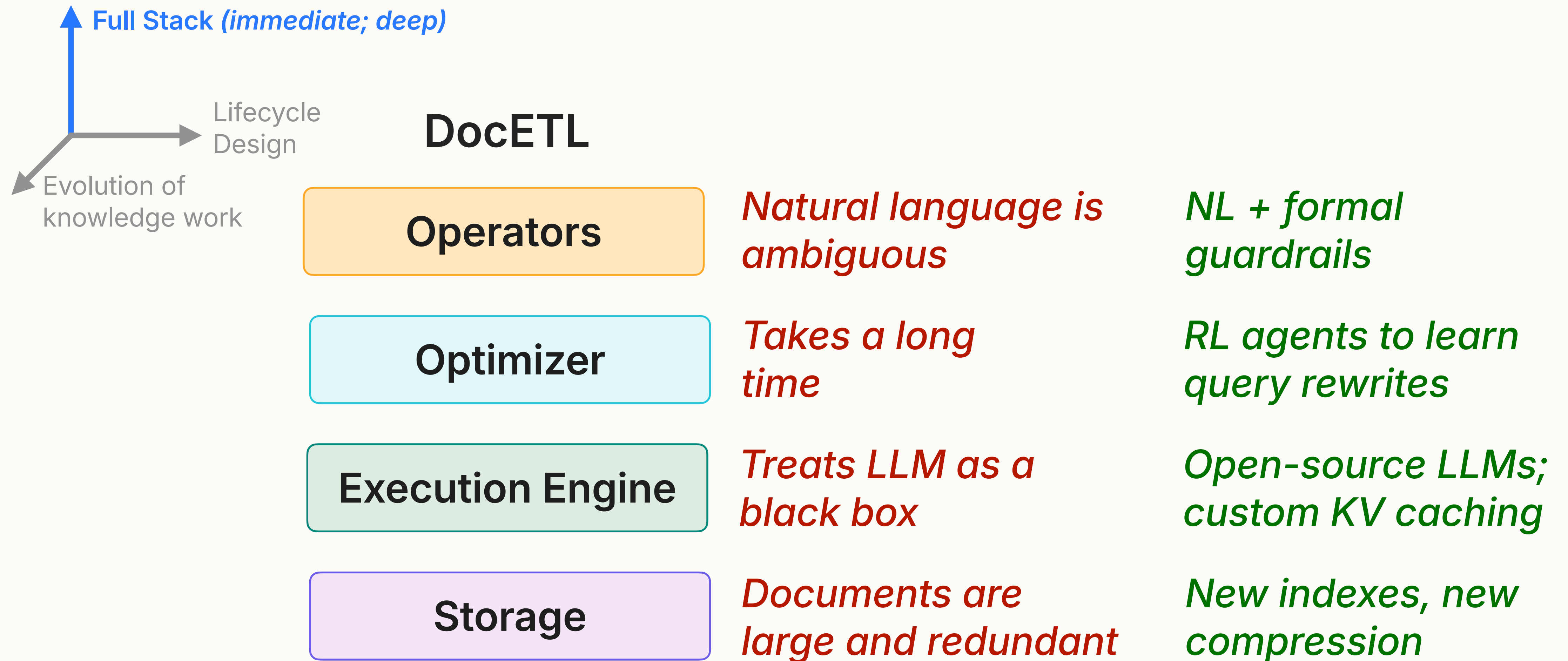
"Full-lifecycle" for data-oriented tasks

How is AI changing knowledge work?

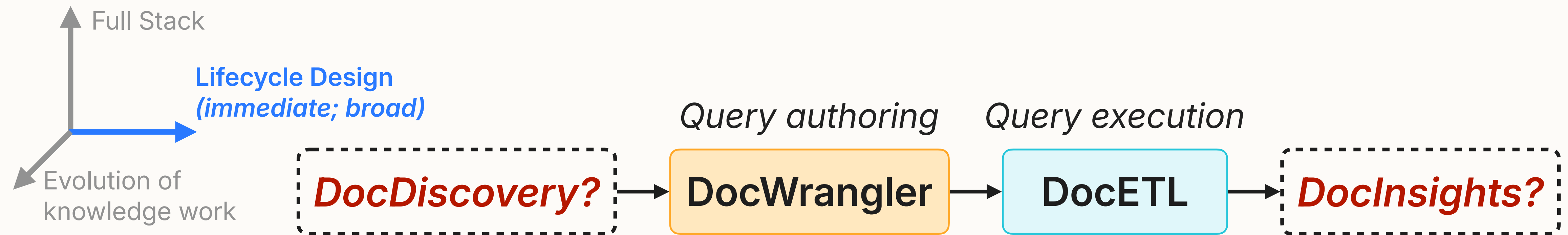


Field studies with domain experts, new research methodologies

Tomorrow: “Full-Stack” in Inner Loop



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.