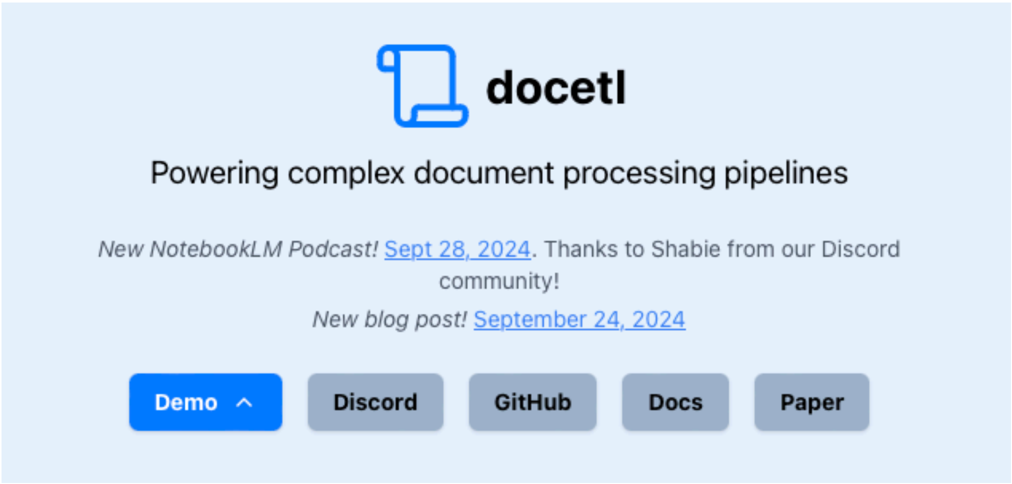
Unstructured Data Analysis with DocETL



docetl.org

Shreya Shankar[1], Aditya G. Parameswaran[1], Eugene Wu[2]
UC Berkeley EECS[1] and Columbia University[2]
November 2024

EPIC DATA lab
UC Berkeley
Columbia University in the City of New York

DocETL: A System for Unstructured Data Processing

Launched ~2 mos ago

github.com/ucbepic/docetl | 1.3k ⭐ | 300+

🧩 No/Low-Code Interface

Declarative YAML interface and operator suite that makes complex document processing **accessible to non-programmers**

🪄 Agentic Optimizer*

Improves output accuracy and quality by intelligently and automatically **decomposing complex tasks**

We're Just Getting Started! 🚀

- 🏛️ Civic Engagement
- 🔍 Forensic Psychiatry
- 📧 Email Analysis
- ⚖️ Mining Law Articles
- 📚 Summarizing educational resources

*We currently focus on optimizing accuracy, not cost.

Demo

Today's Goals 🎯

💡 **KEY INSIGHT**

LLM-powered query processing requires optimizing for **accuracy**, not just performance.

1️⃣ Why Optimize for Accuracy?

❌ Long documents break LLMs

⚠️ LLMs make mistakes on hard data processing tasks

🔄 Complex tasks require tedious decomposition

2️⃣ An Architecture for Such a Query Optimizer

📝 Novel rewrite directives for accuracy

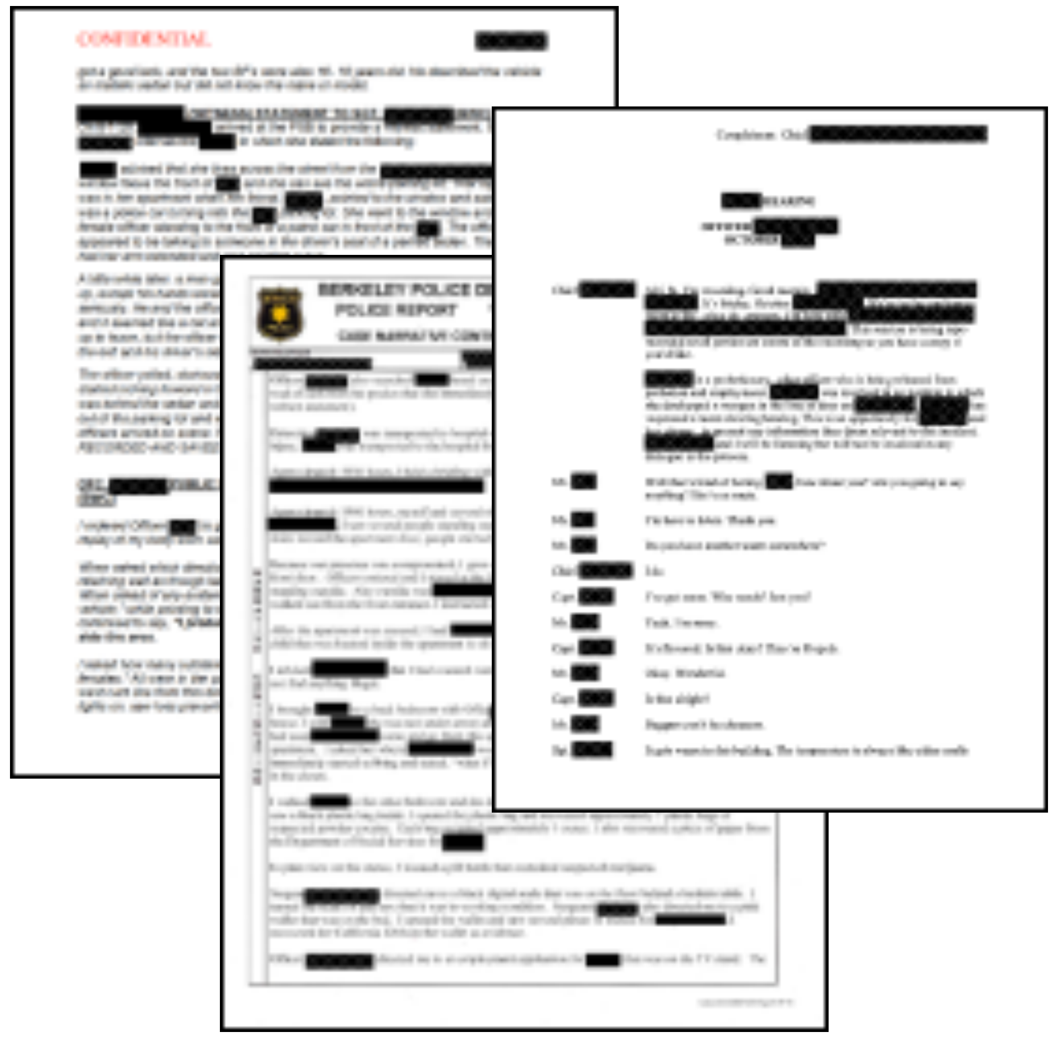
🤖 LLMs as accuracy judges in query optimization

📈 25-66% accuracy boosts across tasks

Complex Document Processing

https://bids.berkeley.edu/california-police-records-access-project

Police Records



⚠️ Challenges
Multiple document types (case reports, hearings, etc)
Very long & inconsistent

Required Analysis Types

🚓 Extract Misconduct

Identify instances of procedural violations and misconduct

👮 Officer Resolution

Link incidents involving the same officer across documents

⚠️ Challenges
Complex reasoning required
Cross-document analysis

Current Approaches

📋 Manual Review

Too time-consuming!

🧑‍🏫 Train Custom Models

Too resource-intensive!

🤖 Use LLMs

Error-prone

Hard to program

A Declarative Solution

```yaml
- name: extract_misconduct
  type: map
  output:
    schema:
      misconduct: "list[{officer: str, incident: str}]"
  prompt: |
    Analyze the following police record…
```

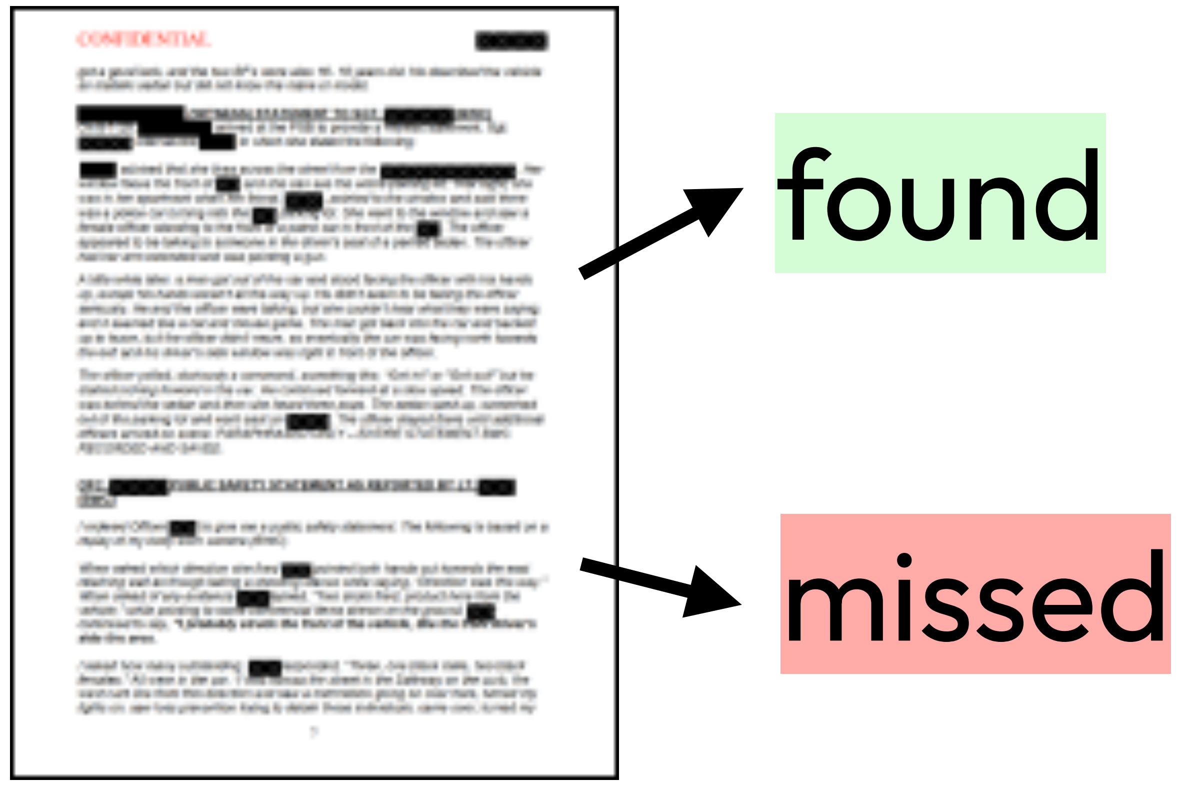
✅ Amenable to complex pipelines

✅ Human-friendly

✅ Automatic performance optimization

🤔 Is this all?? Are we done??

Still, Writing Reliable Complex Pipelines is Hard



Missed Information

LLMs ignore instances or give incorrect answers when docs are too long

Manual Validation

Users must verify correctness themselves

Experimentation

Users must figure out how best to decompose tasks

Unfortunately, LLM Mistakes are Here to Stay

Recent research shows these limitations are fundamental

💬 On Limitations of the Transformer Architecture

Peng, Narayanan, & Papadimitriou 2024

`Transformers can't solve certain compositional tasks`

💬 Calibrated Language Models Must Hallucinate

Kalai & Vempala, STOC 2024

`Good predictions require some hallucination`

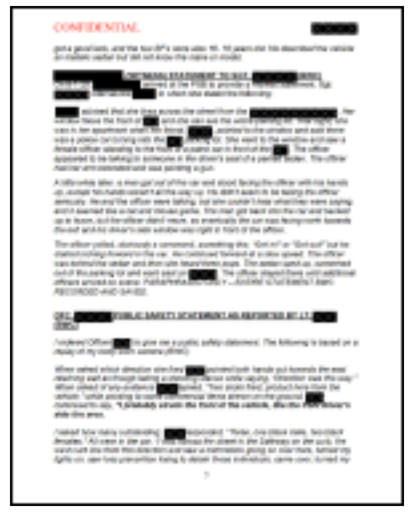
💬 Same Task, More Tokens: Impact of Input Length on LLM Reasoning

Levy, Jacoby & Goldberg, ACL 2024

`Longer inputs = worse performance, even when the task's inherent complexity is unchanged`

🔑 Key insight: complex tasks need to be broken down into smaller, well-scoped tasks to be correct.

Systems Should Rewrite Pipelines to Optimize Accuracy



→ For each police officer involved, extract any instances of misconduct.

Extract police officer names → Extract instances of misconduct

Split long doc into chunks → Extract officers & misconduct

Extract police officer names → Filter for officers who exhibited misconduct → Extract instances of misconduct

Which plan is best? What parameter choices?

Talk Roadmap

1 🧩 **DocETL Operators**
New operators for complex document processing

2 🔀 **Rewrite Directives**
A framework for **agentic** pipeline optimization to improve accuracy

3 🤖 **Optimizer Architecture**
Using **LLM-as-a-judge** to guide optimization decisions

4 👥 **Interactive Pipeline Development**
Vision for **human-AI collaboration** on DocETL with interactive latencies

DocETL Operators

8 operators for complex document processing

LLM-Powered (5)

🔮 **map**
Transform each document into 1+ results

📊 **reduce**
Aggregate multiple documents into a result

🔍 **filter**
Keep/drop documents based on fuzzy predicate

🔗 **equijoin**
Join documents on fuzzy condition

🎯 **resolve**
[New!] Entity resolution across documents

Utility (3)

🌳 **unnest**
Flatten nested arrays or documents

✂️ **split**
Divide documents into chunks

🧩 **gather**
[New!] Augment chunks with context

No-code; YAML

Reduce Operator

Physical implementation to handle infinite context

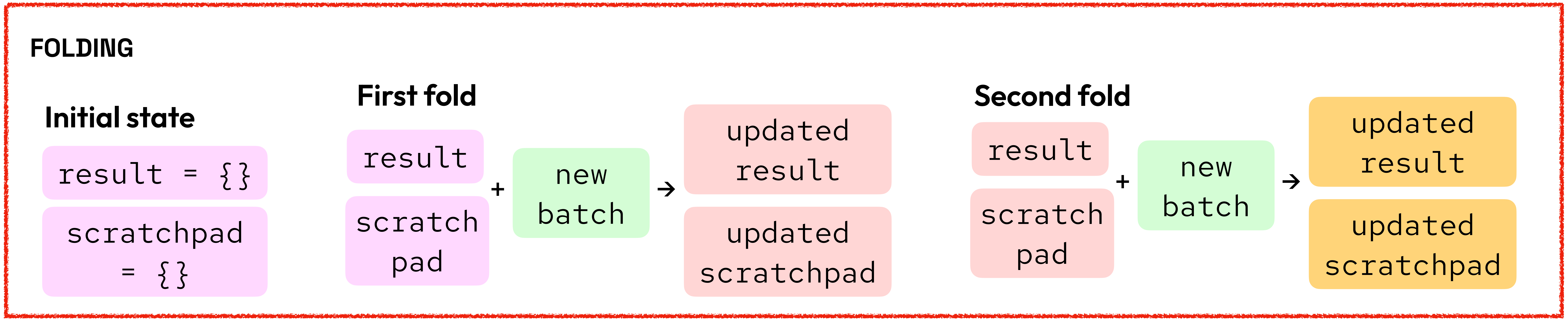
Task: Find types of misconduct Officer Quinnsworth exhibited multiple times

Report #127
"...excessive force during arrest..."

Report #89
"...evidence tampering..."

Report #45
"...excessive force complaint..."

+100s more docs for Officer Quinnsworth



Scratchpad permits the operator to be maintained incrementally

Resolve Operator

Raising the level of abstraction for users

Challenge

Document 1
👮 Officer X. Quinnsworth...

Document 2
👮 Sgt. Xander Quinnsworth was...

Document 3
👮 Officer Quinnswrth, badge #...

Quinnsworth

Officer Quinnsworth

Officer names are inconsistently referred to across documents!

Even if they are consistently represented, **an LLM might inconsistently extract them.**

Interface

```
- type: resolve
  key: officer_name
  comparison_prompt: |
    Do these names refer to
    the same officer? Name 1:  {{ input1.name }}
    and Name 2: {{ input2.name }}
  resolution_prompt: |
    Provide canonical name for:
    {% for input in inputs %}
    - {{ input.name }}
    {% endfor %}
```

✅ Comparison prompt to assess equality of two keys

✅ Resolution prompt to determine canonical name

Resolve Operator

Implementation

Three-Phase Resolution

1. Blocking
Automatically synthesize task-specific rules (e.g., find a blocking threshold via sampling; have LLM generate code)

Embedding-based blocking: Only compare pairs that meet a similarity threshold

LLM

2. Build Clusters
Compare all eligible pairs; merge clusters on LLM-determined equality and equality by transitivity

```
1. (J. Wilson, Officer Wilson)
2. (Officer Wilson, Det. Wilson)
3. (J. Wilson, Det. Wilson)
```

3 is equivalent by transitivity!

LLM

3. Canonicalize
For each cluster, invoke the LLM to determine the canonical form or name

Cluster 1:

- Officer J. Smith
- John Smith
- Smith, J.

→ Officer John Smith

Cluster 2:

- Officer Wilson
- Det. Wilson

→ Detective Wilson

Gather Operator

Augmenting chunks post-split

Challenge

Chunk 1
👮 Officer J. Smith responded…

Chunk 2
He then proceeded to…

Who is "he"?

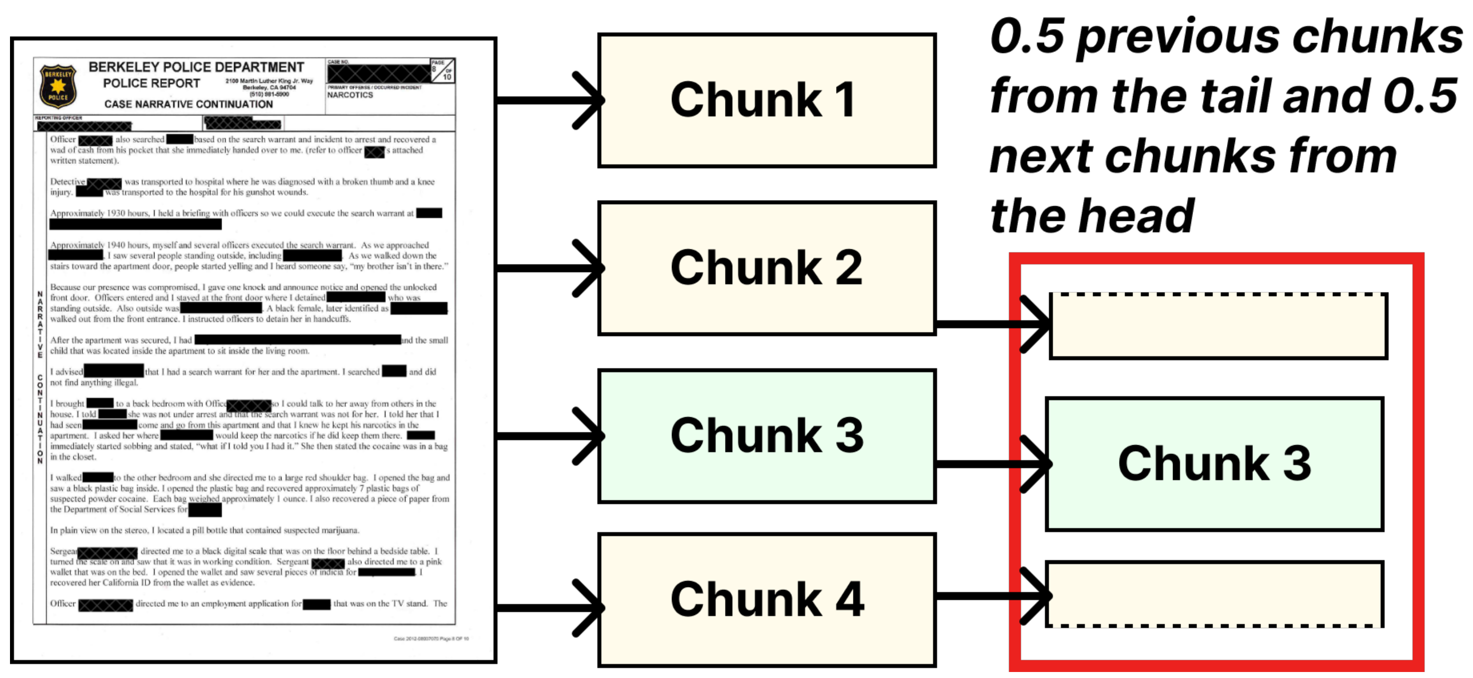
What happened before?

Context Types

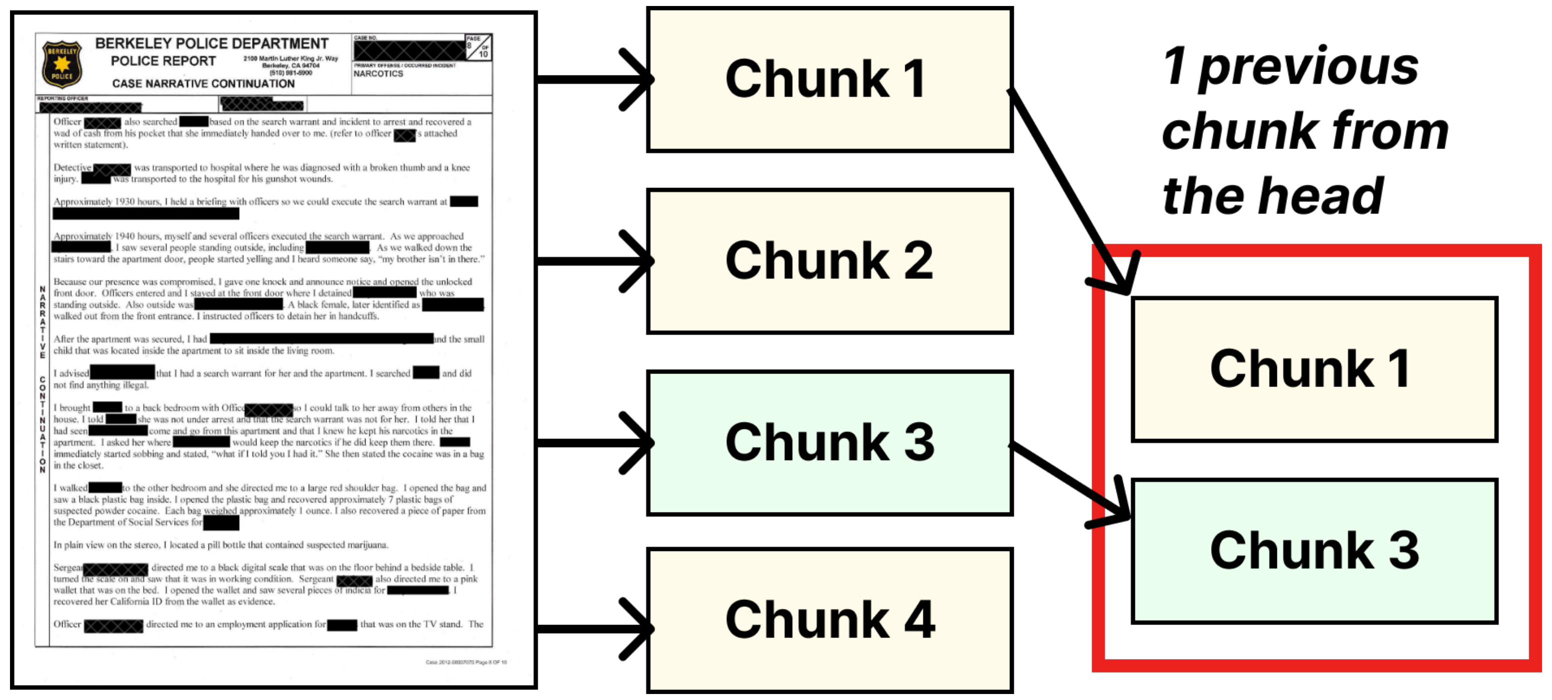
⬇️ **Previous/Next Chunks**	📝 **Transformed Content**	📄 **Document Metadata**
See Figure 2 on the next page for a detailed view of…	Previous 200 pages: "Suspect was last seen in Paris…"	1. Contract Terms 1.1 Licensing 1.1.2 Usage Rights
[Figure 2] Architecture diagram showing…	"He boarded a train to…"	The licensee shall…
Need next chunk for referenced context	Summary of a long prefix	Section hierarchy context

Gather Operator

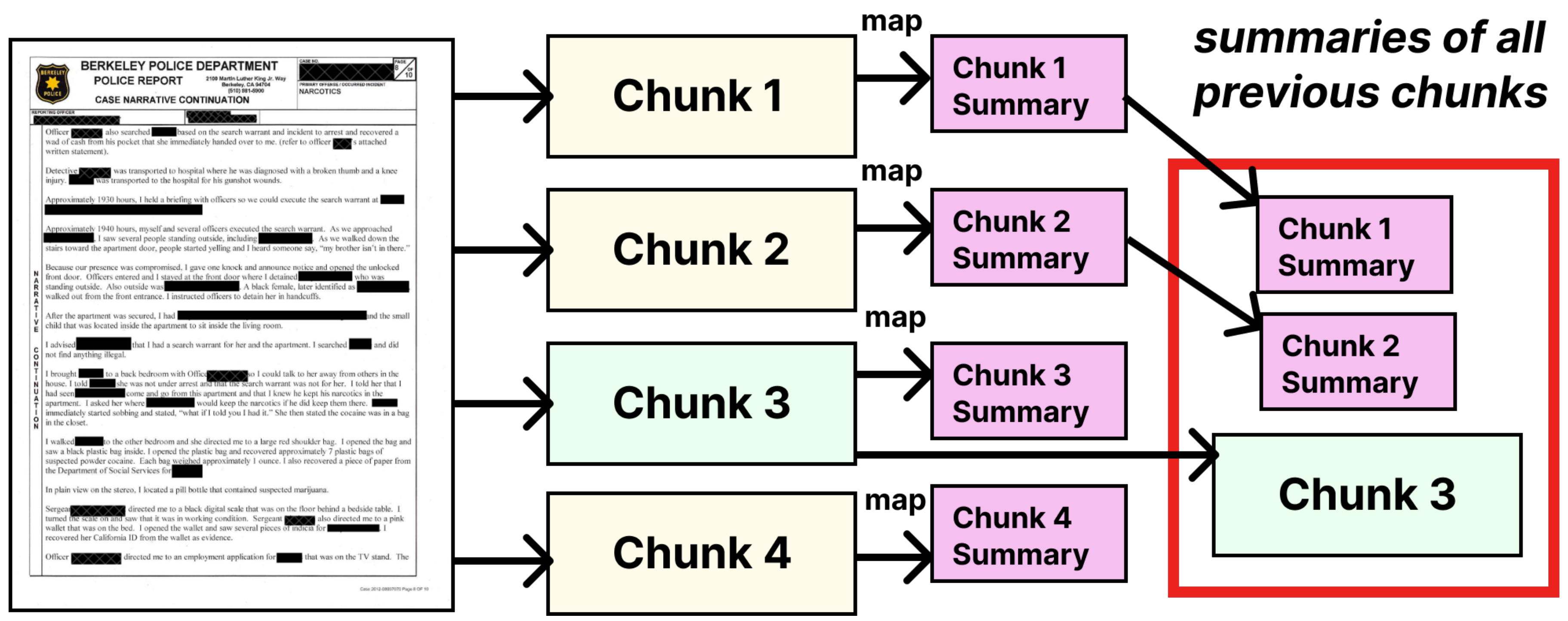
Illustrative Examples




```
-type: gather
 content_key: ..
 peripheral_chunks:
  previous:
   tail:
    count: 0.5
  next:
   head:
    count: 0.5
```




```
-type: gather
 content_key: ..
 peripheral_chunks:
  previous:
   head:
    count: 1
```




```
-type: gather
 content_key: chunk
 peripheral_chunks:
  previous:
   middle:
    content_key: chunk_summary
```

Talk Roadmap

1. 🧩 **DocETL Operators**
   New operators for complex document processing

2. 🔀 **Rewrite Directives**
   A framework for **agentic** pipeline optimization to improve accuracy

3. 🤖 **Optimizer Architecture**
   Using **LLM-as-a-judge** to guide optimization decisions

4. 👥 **Interactive Pipeline Development**
   Vision for **human-AI collaboration** on DocETL with interactive latencies

Why Rewrite Directives?

USER'S PIPELINE

map

For each document, extract officer name & describe the misconduct.

↓

reduce

Group by officer; summarize and analyze misconduct patterns.

NAIVE LLM DECOMPOSITION

reduce

For each document, summarize it into key points.

→ **original map** → **original reduce**

❌ Operator semantics can be wrong

❌ Initial operation loses critical details

❌ Not logically equivalent

Rewrite directives enable "safe" operator decomposition.

13 Rewrite Directives

Goal = Intelligently Decompose Tasks for Better Accuracy

✂️ Data Decomposition

Break down large inputs into manageable pieces

- Document chunking
- Multi-level aggregation

🎯 Projection Synthesis

Break down the task described in the prompt into 2+ prompts

- Chaining & Isolating
- Preprocessing

🔍 LLM-Centric Rules

Refine LLM-generated outputs

- Gleaning
- Duplicate detection

KEY PROPERTIES

- Abstract frameworks, not concrete rules
- Interpreted by LLM agents based on context
- Infinitely many possible instantiations!

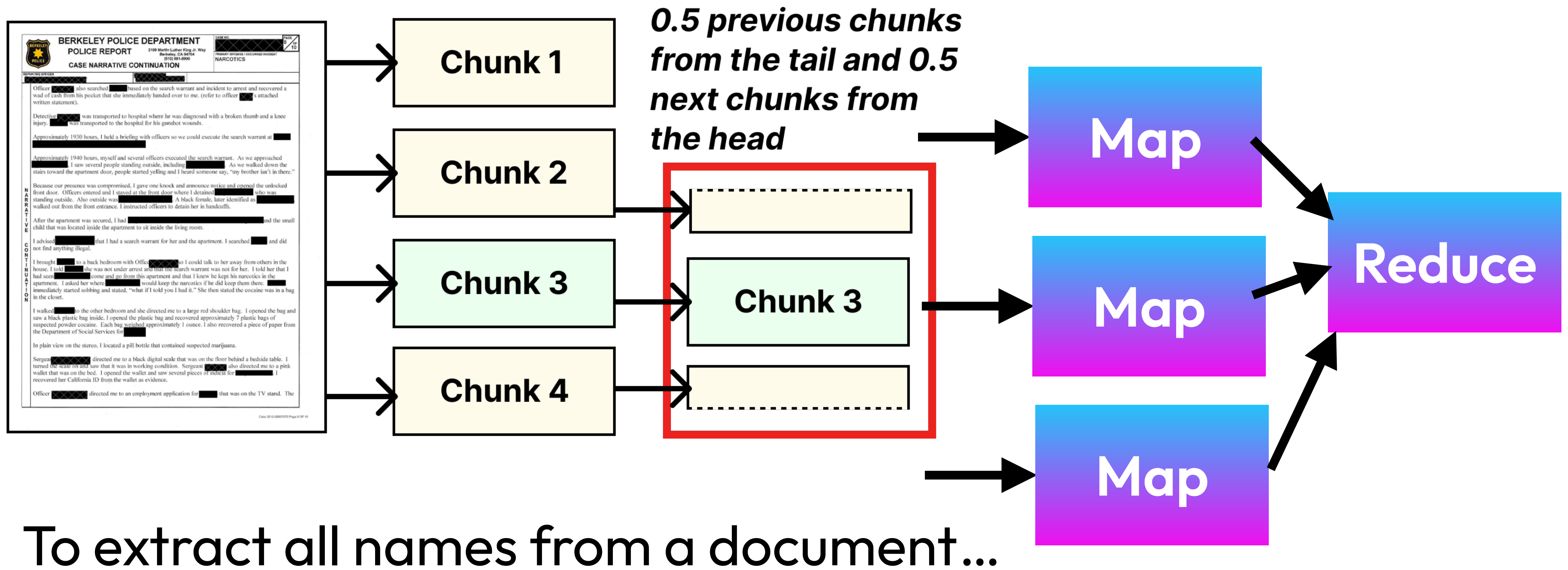
Data Decomposition Directives

Solving the “data is too hard” problem

DOCUMENT CHUNKING

+4

Map ⟹ Split → Gather → Map → Reduce



To extract all names from a document…

1. Split the document into chunks
2. Extract names from each chunk
3. Combine all the extracted names into one result

MULTI-LEVEL AGGREGATION

Reduce ⟹ Reduce → Reduce

Document	City	State
The news today is…	Berkeley	CA
Good morning! …	Dallas	TX
Happy November! …	Berkeley	CA
It’s another sunny …	Albany	NY
1000s more docs…		

To summarize documents for each state, we can…

1. Summarize documents for each city
2. Summarize the city summaries

Projection Synthesis Directives

Solving the “task is too hard” problem

CHAINING

$\text{Map} \Longrightarrow \text{Map} \rightarrow \text{Map}$

Original Complex Task

"Analyze this legal document and identify key clauses, risks, and provide recommendations"

Map 1: Extract Key Clauses

Map 2: Identify Risks of Clauses

Map 3: Provide Recs

ISOLATION

$\text{Map} \Longrightarrow (\text{Map} \mid\mid \text{Map}) \rightarrow \text{Reduce}$

Original Complex Task

"Analyze the app performance and customer service in these reviews: The checkout process was slow but the customer service was excellent..."

Map 1: Slow App Performance

Map 2: Excellent Customer Service

Reduce/Combine Analysis

GENERAL PREPROCESSING

$\text{Op} \Longrightarrow \text{Map} \rightarrow \text{Op}$

General use cases:

- Extract relevant fields
- Transform data format
- Add derived features

Before reduce: extract key info before aggregating

Before filter: compute explicit criteria fields

LLM-Centric Directives

Addressing LLM idiosyncrasies to improve outputs

GLEANING

$\text{Map} \Longrightarrow \text{Map} \rightarrow (\text{Map}_{\text{validator}} \rightarrow \text{Map}_{\text{generator}})^{<k}$

$\text{Reduce} \Longrightarrow \text{Reduce} \rightarrow (\text{Map}_{\text{validator}} \rightarrow \text{Reduce}_{\text{generator}})^{<k}$

DUPLICATE RESOLUTION

$\text{Reduce} \Longrightarrow \text{Resolve} \rightarrow \text{Reduce}$

1. Initial Operation

Extract all political views mentioned...

Output: “Healthcare reform, tax policy”

2. Validation and Feedback

"Missing environmental policy discussion from paragraph 3"

3. Refined Output

"Healthcare reform, tax policy, environmental regulations"

Raw Keys

New York City	Berkeley
NYC	Berkeley, CA

To summarize documents for each city...

1. Resolve the city names
2. Summarize as intended

Comparing Rewrite Directives

A sample of what we've learned thus far...

✂️ Data Decomposition

Good for:

- Long or many documents
- Outputs linear in # chunks or documents

Extracting all multiple choice questions from a test; finding all citations in a research paper

🎯 Projection Synthesis

Good for:

- **Ambiguous prompts**—where task criteria need better definition

"Extract interesting quotes" → Define interesting, then extract

- **Multi-aspect tasks**—when the prompt asks for many different things

Extracting 40 fields → Break into independent extractions

🔍 Gleaning

Good for:

- **Near-miss extractions**—when initial output is close but missing a few items
- **"Needle-in-a-haystack"**—finding specific, rare information in documents

Finding key statements or claims in research papers

Modern LLMs support 2M context window—quite permissive!

Requires document to fit in context window.

Talk Roadmap

1 🧩 **DocETL Operators**
New operators for complex document processing

2 🔀 **Rewrite Directives**
A framework for **agentic** pipeline optimization to improve accuracy

3 🤖 **Optimizer Architecture**
Using **LLM-as-a-judge** to guide optimization decisions

4 👥 **Interactive Pipeline Development**
Vision for **human-AI collaboration** on DocETL with interactive latencies

Agentic Optimizer

Improving accuracy in user-provided pipelines

📄 **User Pipeline**

Optimize one operator at a time

1. 🔍 **Validate Current Output**
   LLM judge determines if quality is sufficient
2. 🔍 **Generate & Test Plans**
   Apply rewrite directives as needed
3. ✨ **Select Best Plan**
   Two-stage evaluation by LLM judge

✅ **Optimized Pipeline**

Move to next operator

KEY INSIGHT

LLM agents generate, validate, and select plans to improve accuracy

Agentic Optimizer—1. Validation Agents

How do we determine whether an operator should be rewritten?

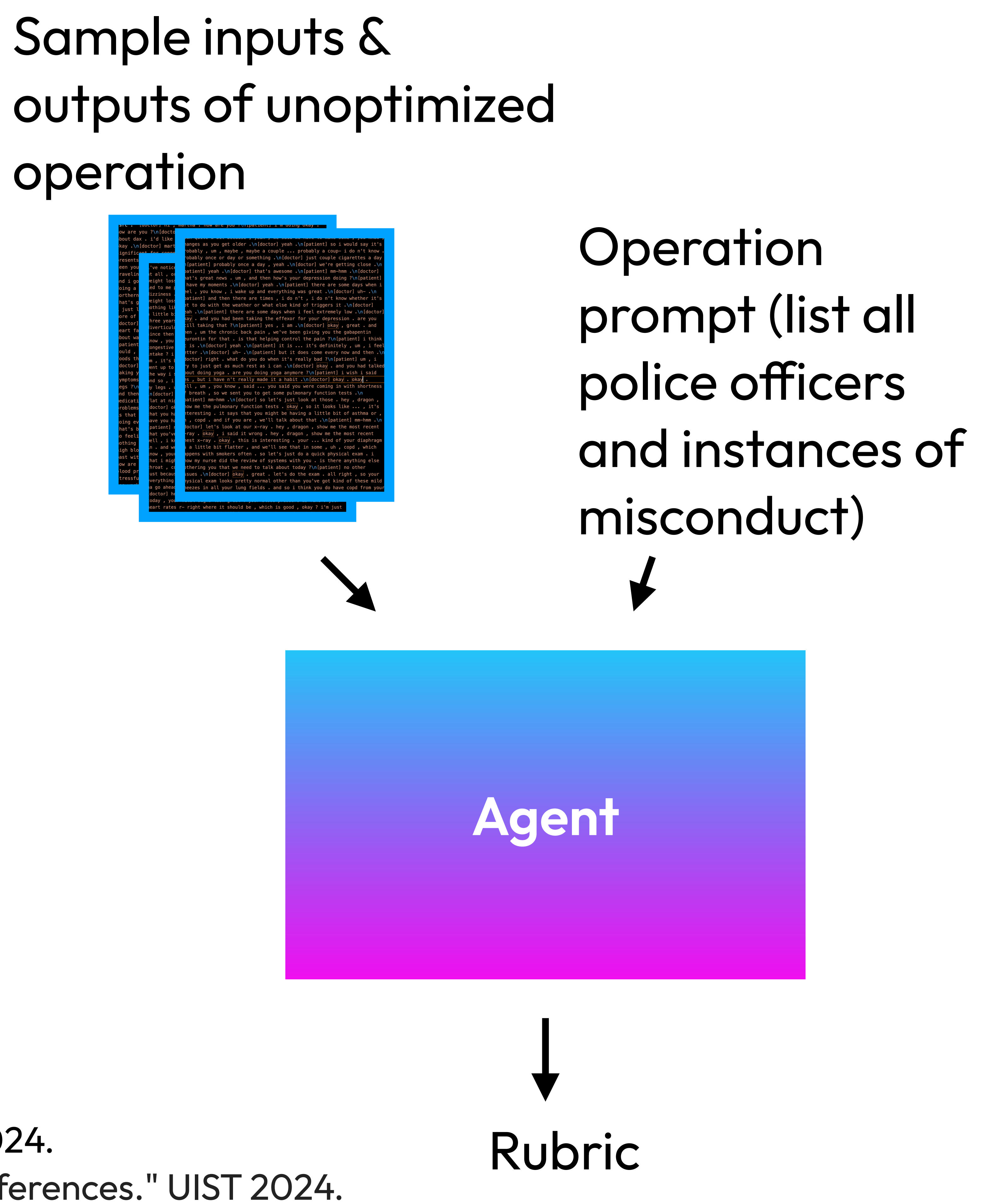
📋 LLM creates evaluation rubric [1, 2]

Example rubric:

- Are all instances of misconduct from the document captured?
- Are dates and locations included for each incident?
- Are there any misconduct claims in the output not supported by the document?

✅ LLM evaluates sample outputs

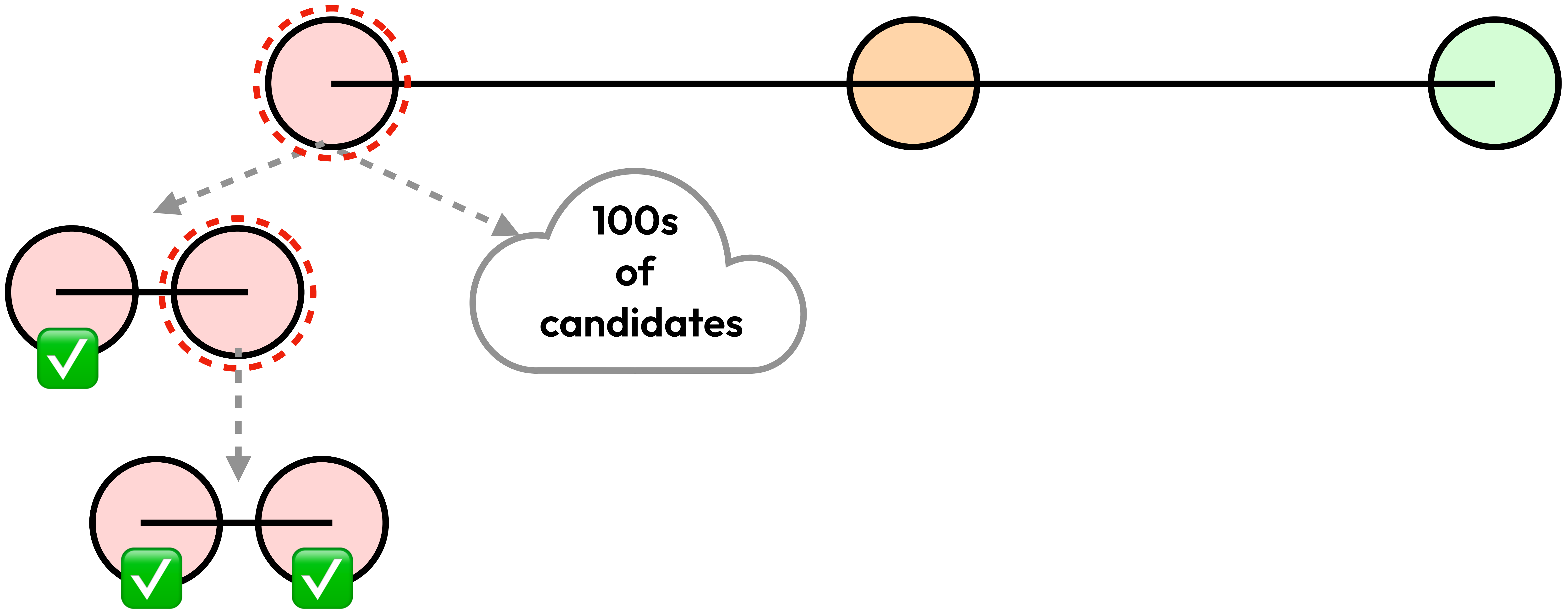
- If output meets all criteria → Move to next operator
- If not → Proceed to optimization



[1] Shankar, Shreya, et al. "SPADE: Synthesizing Data Quality Assertions for Large Language Model Pipelines." VLDB 2024.
[2] Shankar, Shreya, et al. "Who validates the validators? Aligning llm-assisted evaluation of llm outputs with human preferences." UIST 2024.

Agentic Optimizer—2. Plan Generation

How do we come up with specific rewrites?



Agentic Optimizer—3. Ranking Candidate Plans

How do we determine the best rewrite?

Two-Stage Evaluation:

1. Initial rating (1-5) of each plan's outputs
2. Pairwise comparisons of top k plans

Example Comparison

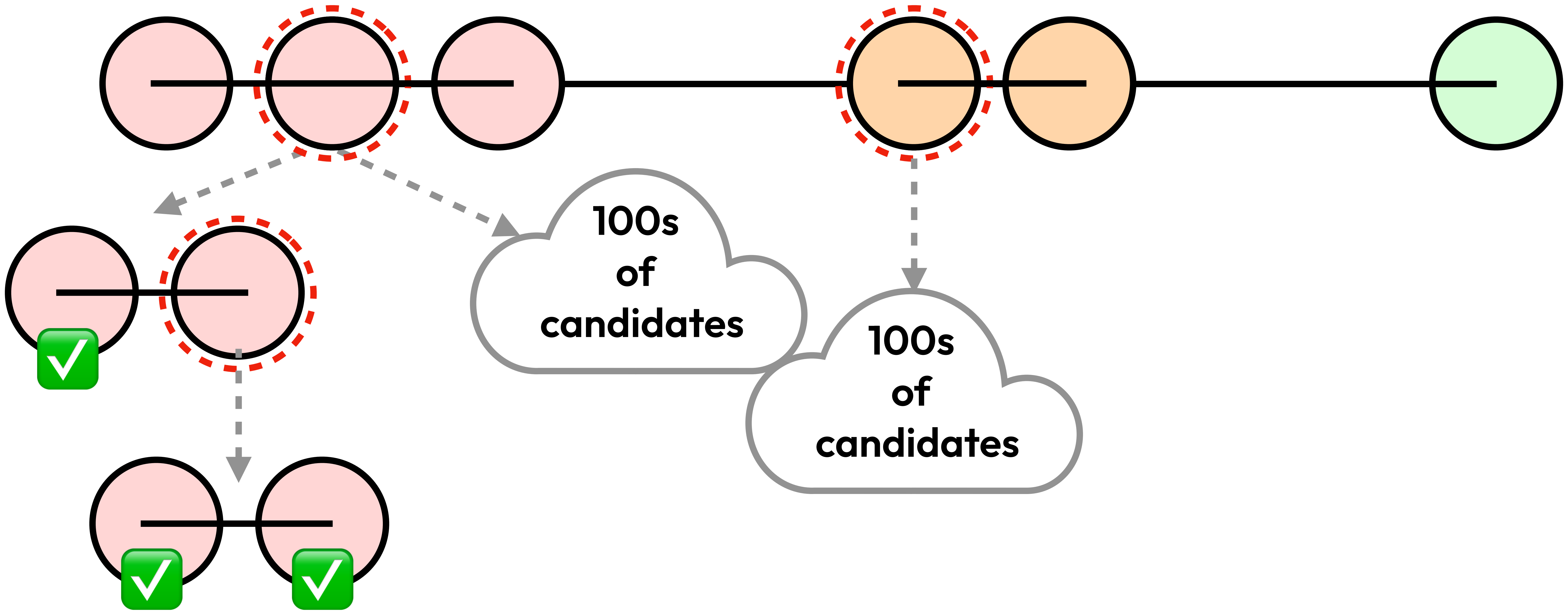
"Plan B is better because it includes all instances of misconduct with proper attribution, while Plan A misses several incidents from pages 45-48..."

Hybrid approach balances thoroughness with computational efficiency: $O(n) + O(k^2)$

SAMPLE PLAN RATINGS

Basic Map	**3.2**
Projection Synthesis A	**4.2**
Projection Synthesis B	**3.9**

Agentic Optimizer



Evaluation

25-66% improvements across tasks

⚖️ **LEGAL DOCUMENT ANALYSIS**

Task: Extract 40 types of clauses from legal documents

Method	Precision	Recall	F1
DocETL (Unopt)	0.305	0.451	0.364
LOTUS	0.350	0.473	0.379
Palimpzest	0.059	0.013	0.022
DocETL (Opt)	0.394	0.731	0.474

+25% improvement in F1 score

+55% improvement in recall

17x the cost of LOTUS or DocETL (unoptimized)

Evaluation

25-66% improvements across tasks

👽 **DECLASSIFIED ARTICLE ANALYSIS**

Task: Find distinct locations of paranormal events

Requires Entity Resolution

🕹️ **GAME REVIEW ANALYSIS**

Task: Create a timeline of positive and negative reviews for video games

Long Review Docs

-99.4% pairwise comparisons eliminated

+66% improvement in recall

1.2x cost

Task	Metric	Baseline	DocETL (Opt)
Declassified Articles	Location Precision	0.994	1.000
	Location Recall	163	270
Game Reviews	Hallucination Rate	0.465	0.312
	Sentiment Accuracy	0.664	0.650
	Kendall's Tau	0.470	0.631

-33% reduction in hallucinations

+34% improvement in ordering

12x cost

Case Study: Police Misconduct Task

227 documents, avg. 12.5K tokens, 2% exceed context limit

👮 **TASK**

Generate detailed misconduct summaries for each officer, including: officer's name, types of misconduct, comprehensive summary with dates and locations.

Metric	Baseline	DocETL S	DocETL T	DocETL O
The officers name is a specific name, not generic	0.84	0.93	0.89	0.87
The summary contains a date and location	0.67	0.10	0.91	0.92
The summary does not omit any instance of misconduct	0.42	0.78	0.76	0.80

VALIDATION

Human evaluation on 100 random samples •
96-97% agreement with LLM judge

Up to +90% improvements!

0.6x the cost of the baseline*

*Baseline includes entire documents in the reduce operation, while S, T, and O apply projection synthesis & are cheaper.

📈 Takeaway 1: Our optimizer can find plans with **much higher accuracy** (25-90% in our evals).

💰 Takeaway 2: Higher-accuracy plans are **not always** more expensive.

Talk Roadmap

1 🧩 **DocETL Operators**
New operators for complex document processing

2 🔀 **Rewrite Directives**
A framework for **agentic** pipeline optimization to improve accuracy

3 🤖 **Optimizer Architecture**
Using **LLM-as-a-judge** to guide optimization decisions

4 👥 **Interactive Pipeline Development**
Vision for **human-AI collaboration** on DocETL with interactive latencies

Towards Agentic Data Processing

Beyond Accuracy: The Challenge of Ambiguity

"Extract instances of police misconduct"

🤖 **LLM Output**
"Officer Thompson raised his voice during questioning. Officer Miller arrived 10 minutes late to the scene."

→

👤 **Human Refinement**
"Minor behavioral issues aren't misconduct. Focus on violations of policy, use of force, or civil rights."

🤖 **LLM Output**
"Officer Wilson detained suspect for 48 hours without charges. Officer Davis conducted search without warrant."

→

👤 **Human Refinement**
"Good, but also note if these actions were justified by department policy exceptions."

KEY INSIGHT

LLM-powered data processing requires **sensemaking**!

Human-Centered Research Questions

🤔 **INTENT UNDERSTANDING**

When do we optimize vs refine operator definitions?

- Did they mean excessive force or any force?
- Is this a prompt issue or optimization issue?

🤝 **HUMAN-IN-THE-LOOP**

When & how should humans steer the LLM?

- During optimization? How so?

👁️ **VISUALIZATION**

How do we visualize unstructured operations?

- Data flows between operators? Data in the outputs?

🌊We're riding a wave of unprecedented capabilities in data processing. It is very exciting! ✨

Data Systems Research Questions

Towards Cheap, Fast, and Accurate Queries

💲 **COST & ACCURACY OPTIMIZATION**

What rewrite directives optimize runtime and cost without sacrificing accuracy?

- Operator fusion, hybrid cost models, retrieval/RAG, etc.

🤝 **EFFICIENT OPERATOR EXECUTION**

How can we adapt plans to data characteristics during execution?

- Expand the set of models we consider for model cascades
- Training binary classification models on-the-fly for resolve, equijoin, filter

👁️ **INTERACTIVE LATENCIES IN THE UI**

How can users quickly iterate on their prompts?

- Sampling, approximate query processing, etc.

🌊We're riding a wave of unprecedented capabilities in data processing. It is very exciting! ✨

Takeaways 💡

📄 **Complex Documents Need Better Tools**
Traditional systems struggle with long, unstructured documents

🧩 **New Operators for New Challenges**
Gather for context, resolve for entity variations

🤖 **Agentic Optimization Works**
25 to 66% improvements across case studies

🤝 **Human-AI Collaboration is Key**
Support evolving understanding between human and AI

Try DocETL today! 🚀

⭐ 1.3k+ GitHub Stars

👥 300+ Discord Members

🌐 docetl.org

📧 shreyashankar@berkeley.edu

Thanks to:
Aditya G. Parameswaran, Eugene Wu
CLEAN team at UC Berkeley
Bhavya Chopra, Mawil Hasan, Stephen Lee, James Smith,
Bjoern Hartmann
Yiming Lin, Sep Zeighami

Shankar, Shreya, Aditya G. Parameswaran, and Eugene Wu. "DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing." In progress.