Towards Agentic Data Processing with DocETL

docetl.org

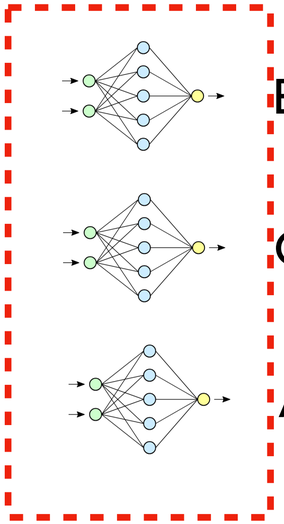
Shreya Shankar*, Aditya G. Parameswaran, Eugene Wu
UC Berkeley, EECS
October 2024

EPIC DATA lab
UC Berkeley
COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Analyzing Unstructured Data is Hard

- Organizations have lots of data and *intelligent data analysis* needs
  - These require teams of human annotators 👨
- Consider a big hospital 🏥
- Lots of data, e.g.,
  - Patient records
  - Documentation (medicines, illnesses, etc.)
- ML is promising but <u>hard</u>

MLOps challenges

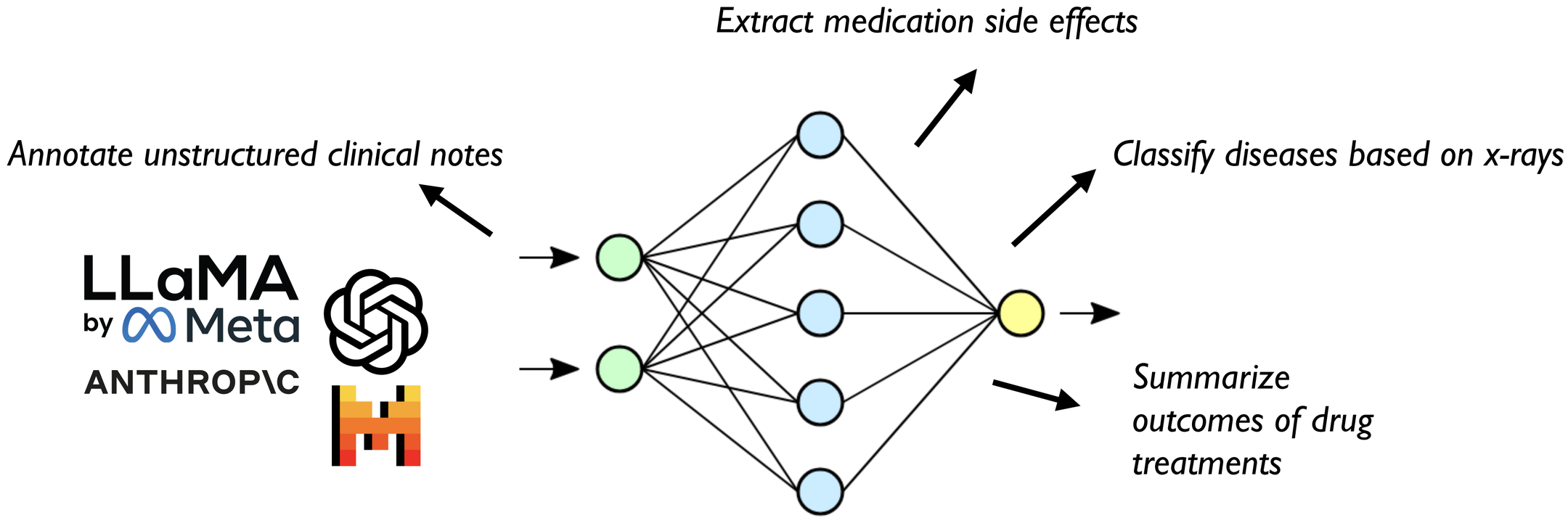


Extract medication side effects

Classify diseases based on x-rays

Annotate unstructured clinical notes

LLMs Enable Unstructured Data Analysis



- LLMs enable intelligent data processing pipelines *without training models*
- Significantly simplifies the ML lifecycle

Example: *Map* Operation on Doctor-Patient Visit Transcripts

Input Document

[patient] i'm doing well . i've been traveling a lot recently since things have , have gotten a bit lighter . and i got my , my vaccine , so i feel safer about traveling . i've been doing a lot of hiking . uh , went to washington last weekend to hike in northern cascades, like around the mount baker area

Prompt Template

Please review the following patient transcript and extract two pieces of information:

1. *All medications mentioned by the patient*
2. *Any side effects the patient reports experiencing from these medications*

LLM(s)

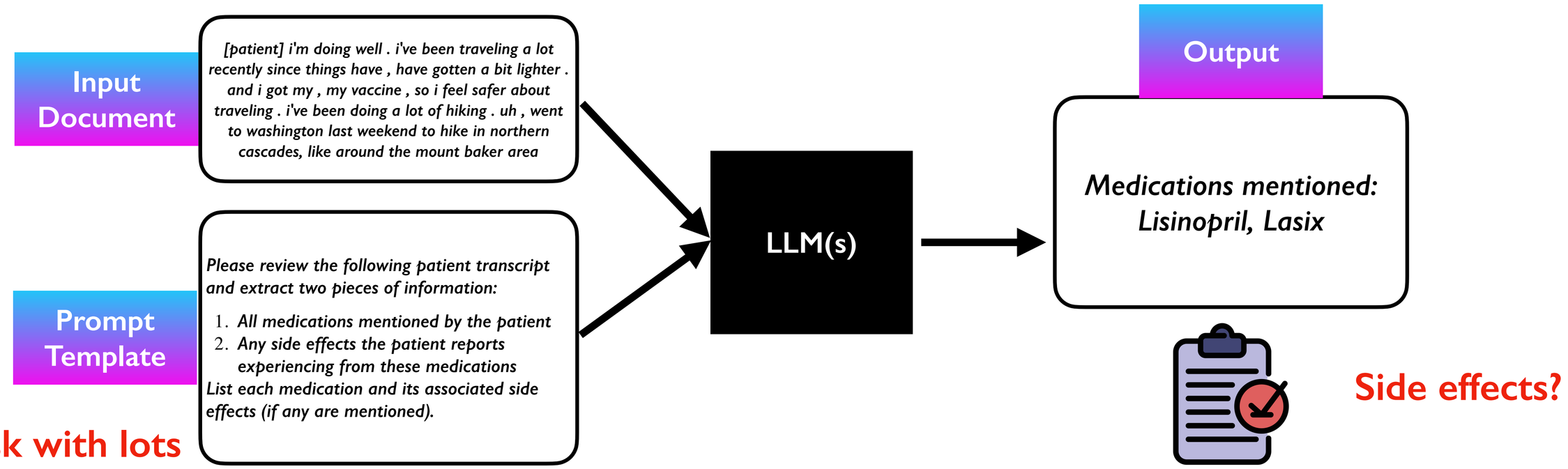
"Map" = 1 input to 0 or more outputs

Output

Medications mentioned: Lisinopril, Lasix
Side effects: No specific side effects were reported.

LLMs Make Unpredictable Mistakes

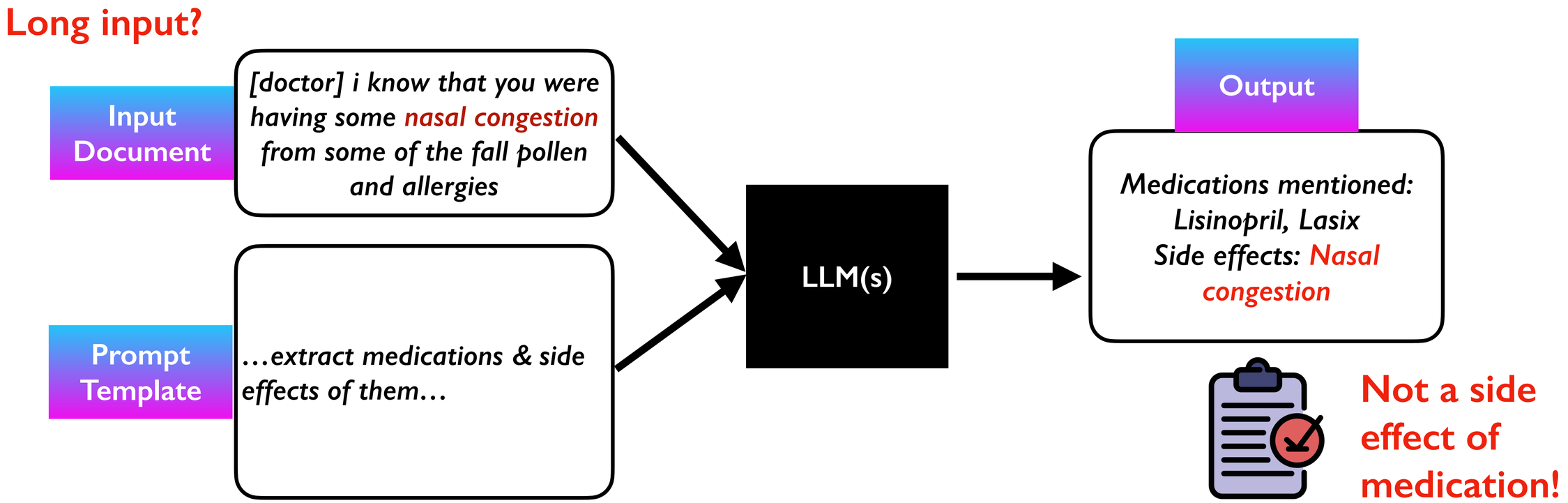
- Hallucinations, bad formatting, ignoring instructions, & more. Tasks can be hard.



Hard task with lots of instructions?

LLMs Make Unpredictable Mistakes

- Hallucinations, bad formatting, ignoring instructions, & more. Data can be hard.



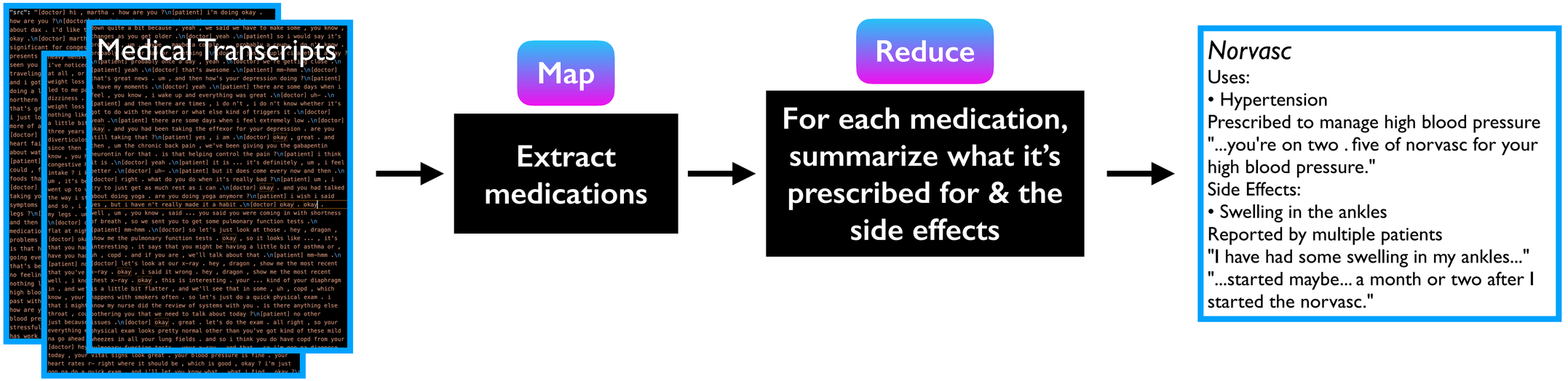
Challenges in AI-Powered Document Processing

- **Tasks** can be hard. **Data** can be hard. Sometimes both!
- Accuracy also depends on **models**.

- How do we take a complex task on complex documents & break it down?
  - Today: we propose *agentic* rewriting—have an LLM rewrite the pipeline for better accuracy!

DocETL

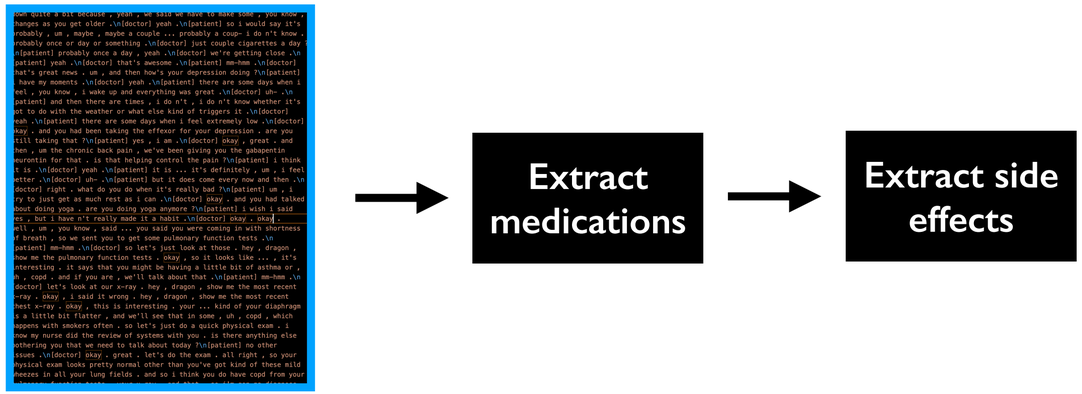
- DocETL (docetl.org) is a declarative system for optimizing & executing complex document processing pipelines



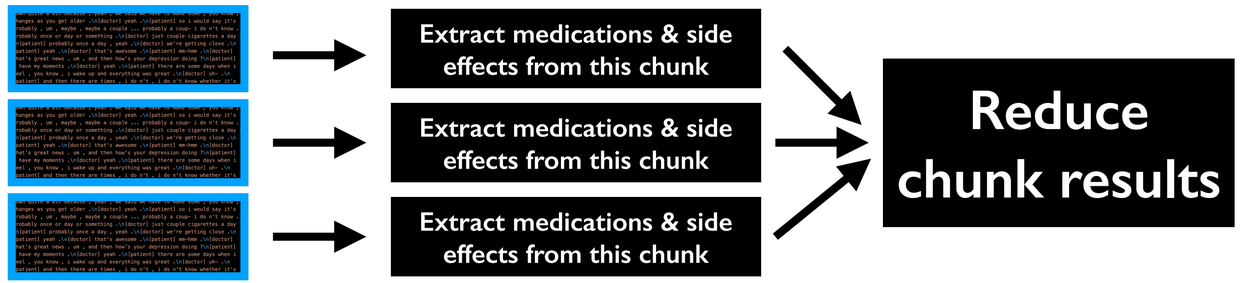
- Declarative = users specify only the ops & DocETL optimizes it to work *correctly* & *efficiently*
- LLM operators (map, reduce, resolve, filter, equijoin) & non-LLM operators (split, gather, unnest)

Unclear How To Make Pipelines More Accurate

- For our map operation (extract meds & side effects), we could *rewrite* it many ways. E.g.,



Map task one → map task two

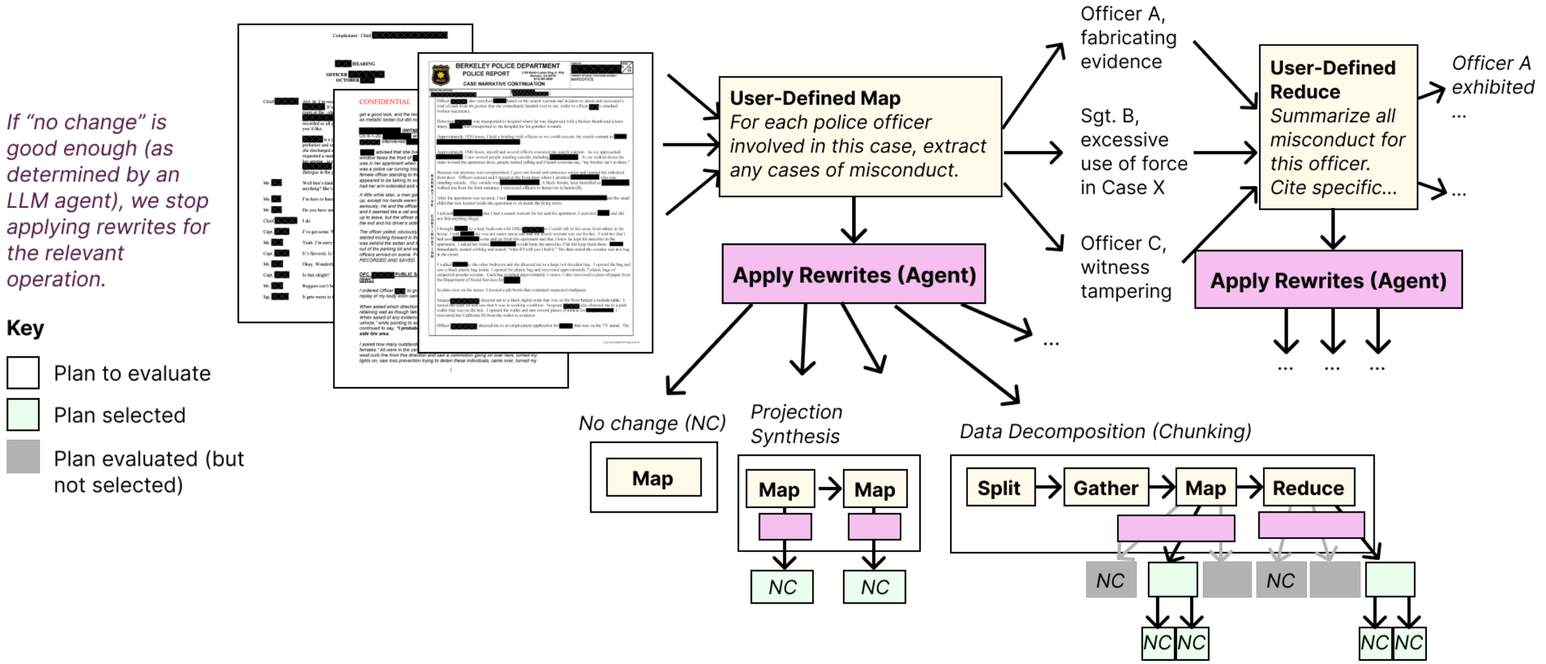


Chunk up data → apply map to each chunk → reduce the results

- How do we decompose the task? How do we validate accuracy? Automating this is hard!

Agentic Optimizer for Intelligent Data Processing

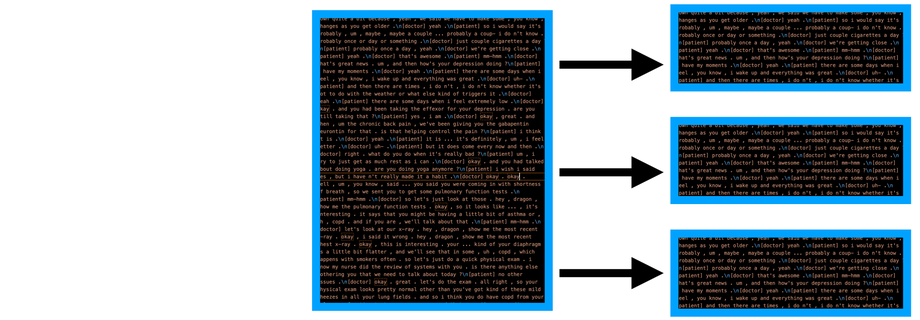
- 3 (new!) ingredients to the optimizer
  - Rewrite directives (implemented by LLM agents)
  - Validation agents to come up with task-specific evaluation criteria
  - Comparing & evaluating candidate plans

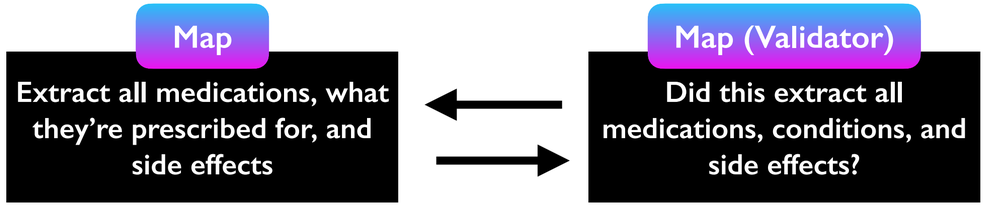


Optimizing a pipeline for identifying officer misconduct in police records obtained via FOIA requests initiated by journalists at UC Berkeley.

Agentic Optimizer
1. Generating Plans with 13 Rewrite Directives

- **Data decomposition:** *breaking down the unit of data fed into the LLM call*
  - Document chunking (map)
  - Multi-level aggregation (reduce)
- **Projection synthesis:** *breaking down the task (i.e., prompt) into helpful intermediate step(s)*
- **LLM-centric improvements:** iterative refinement (i.e., looping until the output is good), duplicate key/entity resolution





How to come up with these validators?

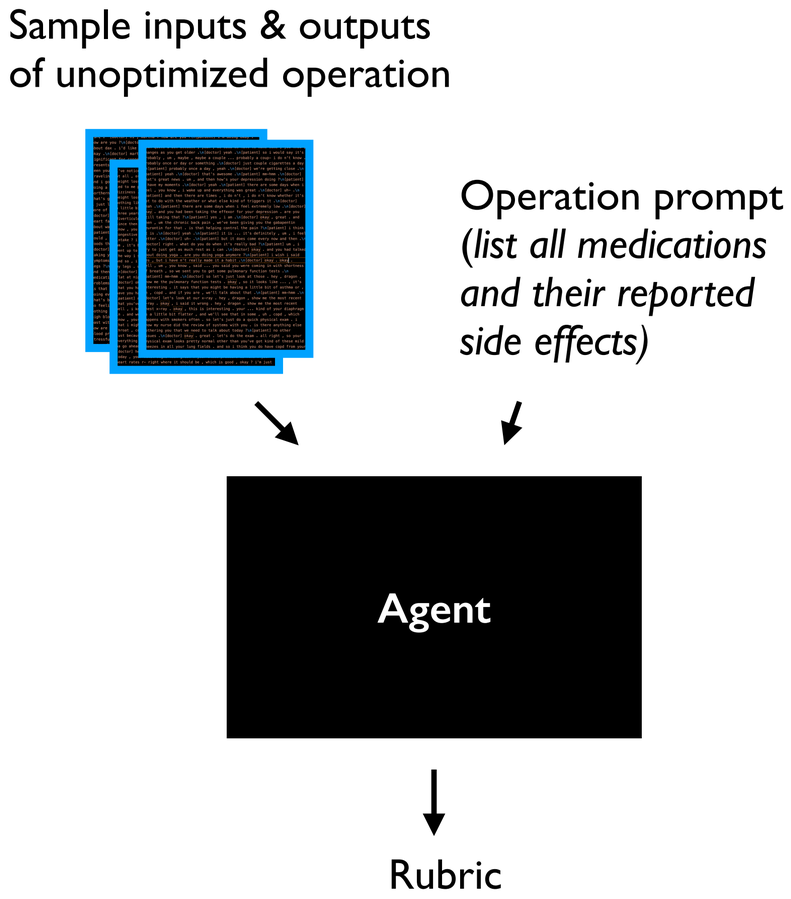
More generally: how do you pick the right plan?

Agentic Optimizer
2. Validation Agents

- In addition to generation, agents also **validate** plans
- To validate an operator, we use LLM agents to:
  - Synthesize validation criteria
  - Evaluate a candidate plan's sample outputs given this criteria

Example rubric:

- Are *all* medications in the input reflected in the output?
- For each medication, are its side effects reported?
- Are there medications or side effects in the output not in the input document?



Agentic Optimizer
3. Evaluating (Ranking) Candidate Plans

- Picking the best plan involves:
  - Rating each plan's output on 1 (bad) to 5 (good) scale
  - Pairwise comparisons of the top k plans

- Top-performing plan is listed 11th!
- Coarse-grained ratings are not perfect, but they are scalable in O(n)
- Pairwise comparisons are too expensive in O(n^2)
- Best of both worlds: O(n) + O(k^2)

```
Comparison Details:
Input 1: "plan_1" because Plan 1 output provides a clear and concise extraction of the medications,
their purposes, and side effects mentioned in the transcript, without unnecessary duplication. In
contrast, Plan 2 repeats entries for each medication multiple times with slight variations, leading
to redundancy and potential confusion. Plan 1's approach is more organized and aligns better with
the task requirement to list medications and their details accurately and succinctly.
Input 2: "plan_2" because Plan 2 provides a more comprehensive output by including both the
ibuprofen and wrist splint in the list of interventions for carpal tunnel syndrome. The task prompt
asks for a list of medications and their uses, and while the wrist splint is not a medication, it
is a significant part of the treatment plan discussed in the conversation. Therefore, Plan 2
captures a fuller picture of the treatment strategy outlined in the transcript.
Input 3: "plan_2" because Plan 2 performed better because it identified additional medications
mentioned in the transcript, specifically the MMR vaccine and the Shingles vaccine, which were both
overlooked by Plan 1. Plan 2 also provided a more detailed explanation of the vitamin D3 dosage.
Both plans maintained a clear format, but Plan 2 had a more comprehensive extraction of medications
discussed in the conversation.
Input 4: "plan_2" because Plan 2 is better because it captures both medications mentioned in the
conversation: Tylenol and the potential steroid injection. It also provides more detailed
descriptions of their uses and effects. Plan 1 only includes Tylenol and misses the mention of
steroid injection, which is a significant consideration for the patient's treatment plan.
Input 5: "plan_2" because Plan 2 provides a more detailed extraction of the medication use and
patient adherence to each, which aligns well with the requirement to list medications along with
their uses. It also captures the dosage of Prozac and the patient's stability on it, offering a
more comprehensive view of the medication management. Plan 1 is accurate but less detailed compared
```

"Plan 2 provides a more comprehensive output by including both the ibuprofen and wrist splint…"

optimizer.py:359

Plan Evaluation Results for extract_medications (map, 15 plans, 5 samples):

optimizer.py:376

Plan	Score	Runtime	Pairwise Wins
no_change	3.80	8.45s	5
gleaning_1_rounds	3.40	4.04s	6
chunk_size_190_peripheral_previous_tail_2_full_previous…	3.00	9.67s	4
chunk_size_549_peripheral_previous_tail_1_full	3.00	4.75s	8
chunk_size_909_peripheral_previous_tail_1_full	3.00	3.92s	3
chunk_size_190_peripheral_previous_tail_1_full_previous…	2.80	8.54s	4
chunk_size_190_peripheral_previous_tail_1_full_previous…	2.80	7.60s	3
chunk_size_190_peripheral_previous_tail_1_full	2.80	5.66s	2
chunk_size_190_peripheral_previous_tail_1_full_next_hea…	2.80	5.18s	1
chunk_size_909_peripheral_previous_tail_1_full_next_hea…	2.80	3.48s	2
chunk_size_1269_peripheral_previous_tail_1_full	2.80	2.48s	9
chunk_size_190_peripheral_previous_tail_2_full_next_hea…	2.60	8.99s	0
chunk_size_190_peripheral_previous_tail_2_full_previous…	2.60	6.73s	0
chunk_size_549_peripheral_previous_tail_1_full_next_hea…	2.60	6.56s	0
chunk_size_190_peripheral_previous_tail_2_full	2.60	6.46s	0

optimizer.py:377

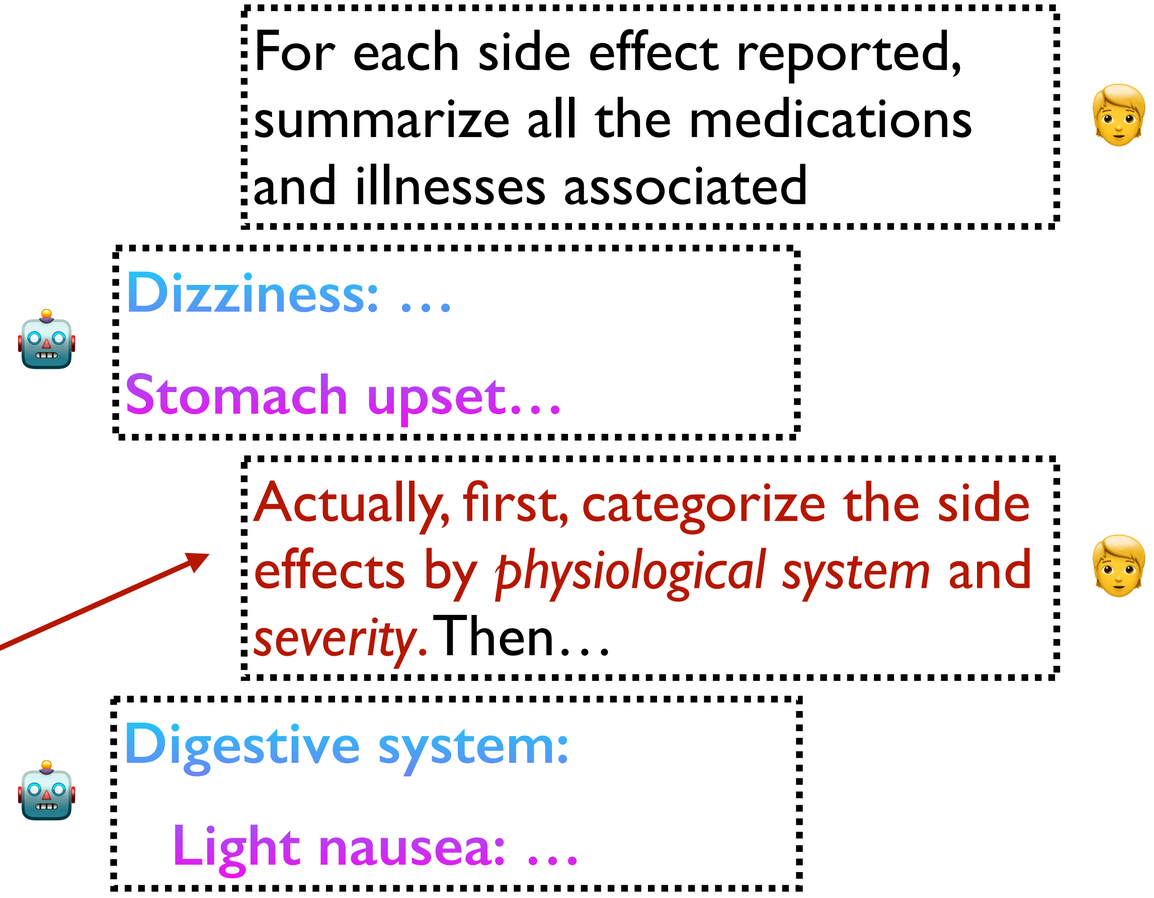
Choosing chunk_size_1269_peripheral_previous_tail_1_full for operation extract_medications (Score: 2.80, Runtime: 2.48s) optimizer.py:380

Towards Agentic Data Processing

This is more that just optimizing accuracy…

- Agent = LLM executing on behalf of human
- We are in a new world of…
  - Agents looking directly at the data
  - Agents synthesizing information across document boundaries

- Catch-22: Users need to see the outputs of an operation before they write the operation



Towards Agentic Data Processing

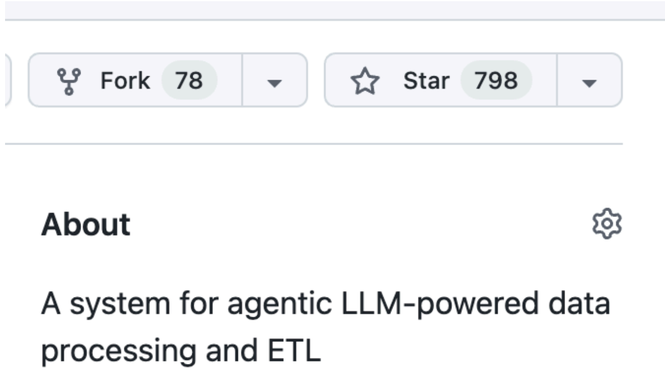
Roles of Humans; Role of Agents

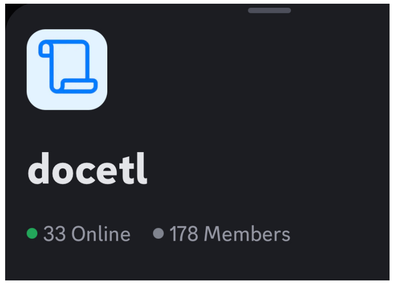
- What should the human do? *Steer the artifact towards their goal*
- What should the optimizer do? *Solve the steered (sub)-task as accurately and efficiently as possible*

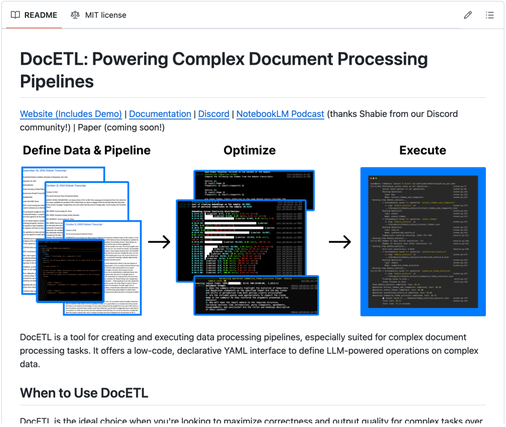
- DocETL pipelines are low-code (written in YAML) and optimized interactively in the terminal 📟
- Towards no-code
  - Natural language specification of pipelines 🥷
  - Interactive optimization in a web app 🌐

Growing Traction

- Launched 14 days ago!
- Some use cases so far
  - Civic engagement: reports on why & how council members voted on issues
  - Email analysis
  - Forensic psychiatry
  - Mining law articles
  - Mining climate reports to make policy recommendations
- Please join our community! Always looking for contributors & users.
- Exciting research & engineering ahead!







Takeaways

- Building data processing pipelines around LLMs requires **experimentation** and **validation**
- Optimizers have to be human-centered and agentic!
- DocETL is a low-code, declarative LLM-powered data processing system (docetl.org)

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

EPIC DATA lab
UC Berkeley

🌐 sh-reya.com 🐦 @sh_reya 📧 shreyashankar@berkeley.edu

Shankar, Shreya, Aditya G. Parameswaran, and Eugene Wu. "Complex Document Processing with DocETL: An Agent-Based Query Rewriting and Evaluation System." *In progress*.