DSC-204A: Scalable Data Systems, Fall 2025

15: Rethinking Data Systems for the Age of AI

Lecturer: Shreya Shankar Scribe: Zhaowei Zhang, Keerthana Senthilnathan, Yangqi Huang, Tongle Shen

1 Introduction: Motivation and Challenges in Unstructured Data Processing

1.1 Context and Motivation

The lecture begins with an overview of why unstructured data processing has become a critical problem in the age of large language models (LLMs). Thanks to modern LLMs, it is now possible to perform complex reasoning over massive corpora of unstructured text. This capability has transformative potential in domains such as law, healthcare, public policy, journalism, and many others.

However, as the speaker emphasizes, although LLMs enable new forms of analysis, making these capabilities *reliable*, *scalable*, and *cost-efficient* within real data systems remains enormously challenging. The talk illustrates this through two motivating workloads contributed by real users.

1.2 Motivating Workload 1: Climate Intervention Analysis

A government organization (the Scottish Climate Intelligence Service) aims to study climate initiatives undertaken by companies in response to the Climate Change Act of 2009. Their data consists of:

- heterogeneous documents such as sustainability reports, regulatory filings, and meeting minutes,
- unknown and inconsistently represented climate interventions (e.g., "EV subsidy", "commuter benefits", "lighting upgrades"),
- tens of thousands of pages of new documents each month.

Running an LLM naïvely (e.g., GPT-4.1) over this volume is prohibitively expensive—easily costing thousands of dollars per monthly batch. Small government organizations do not have budgets for such analysis, despite the importance of the insights.

This example illustrates not only the scale but also the semantic ambiguity of tasks (e.g., what counts as a climate "intervention"), making them difficult to express and execute reliably.

1.3 Motivating Workload 2: Detecting Implicit Bias in Legal Documents

Another real workload comes from public defenders in California who seek to detect explicit and implicit racial bias across:

- court transcripts,
- police reports,
- news articles,
- and related case materials.

For a single defendant, hundreds of pages of related documents must be examined. For statistical analysis across many cases, this expands to *millions of pages*. A naïve LLM workflow could cost millions of dollars per year, which is not feasible.

This workload requires subtle reasoning—e.g., identifying context-dependent indicators of bias, combining evidence across documents, and synthesizing summaries. These tasks are highly open-ended, subjective, and require cross-document aggregation.

1.4 Unifying Characteristics of These Workloads

Across these domains, the speaker identifies three common properties that make unstructured data processing uniquely difficult:

- **Complex reasoning**: Tasks require high-level interpretation (e.g., determining bias, identifying interventions) rather than simple pattern matching.
- **Whole-corpus analysis**: Correct answers require examining all documents, often combining information across large collections.
- **Open-ended outputs**: Operators frequently produce narrative summaries, explanations, or extracted statements—not simple relational values.

These properties mean that traditional database techniques alone are insufficient, while LLMs must be carefully integrated into structured workflows in order to be usable and trustworthy.

1.5 Three Core Research Questions

The speaker frames the rest of the talk around three fundamental questions that arise when integrating AI into data systems:

1. **How do we encode LLM capabilities into data systems?** In other words, what is the abstraction that allows LLMs to be embedded into query plans, similarly to relational operators?
2. **How do users steer and debug AI-powered data systems?** LLMs behave stochastically and opaquely; users need interfaces to understand outputs, adjust prompts, and diagnose failures.
3. **How can we ensure that analyses powered by LLMs are reliable at scale?** When analyses feed into policy decisions, journalism, and legal workflows, users must understand error modes and trust the system.

These questions define the core challenges for AI-powered data systems.

1.6 Overview of the Technical Contributions

The speaker introduces three main system components developed in the PhD work:

- **DocETL**: a declarative data system for executing semantic operators over text using LLMs.
- **DocWrangler**: an interactive interface that helps users understand their data, refine semantic operators, and debug results.
- **Evaluation algorithms**: methods for generating evaluation criteria, identifying LLM failure modes, and quantifying reliability.

Together, these systems form a "full-stack" approach that spans database research, AI systems, and HCI.

2 DocETL Operators

2.1 The accuracy problem

Issue 1: Accuracy Outliers

Problem: When scaling up, the probability of error is non-zero.

Solution: Fix it by supporting user-defined validation logic, and retry failed operations.

Issue 2: Persistent Low Accuracy

Problem: When the task is too complicated for LLM.

Solution(but this does not work well): Tried sorting, ensembles, prompting and few-shot tuning.

Observation: Semantic operators are rarely scoped optimally for LLMs, so before optimization, we need to rescope the tasks for LLMs.

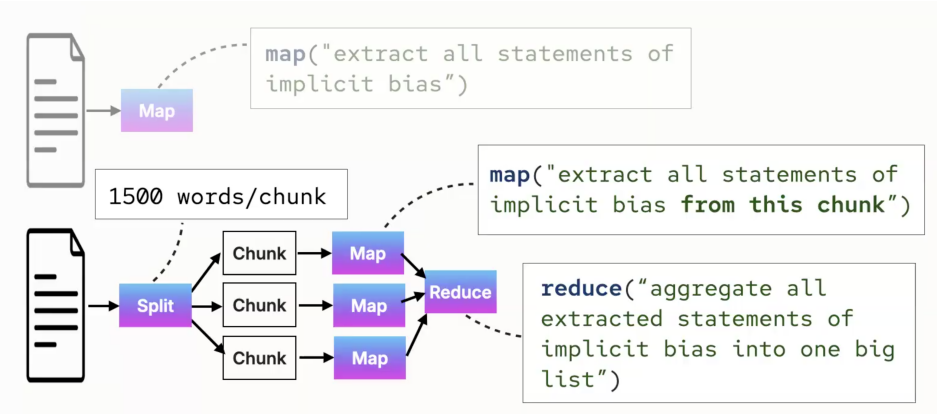

Figure 1: Chunk Task Decomposition

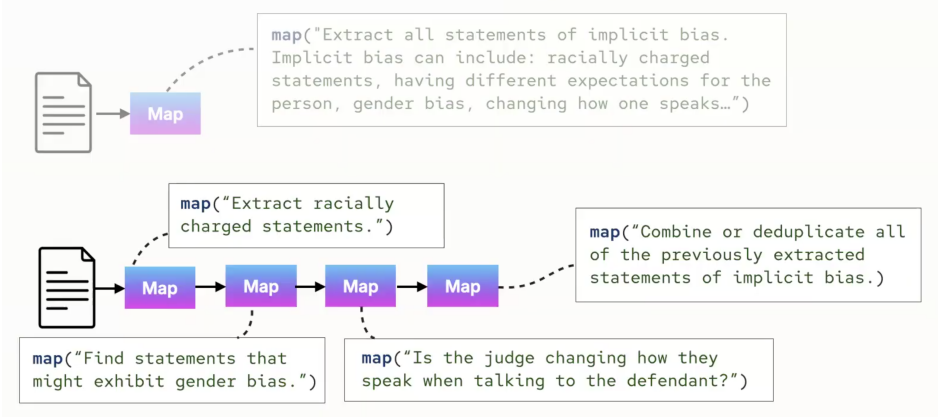

Figure 2: Chain Task Decomposition

2.2 Key Observation: Decomposition

Two types of decomposition are illustrated in Figure 1 and Figure 2

- Chunk Decomposition: In Figure 1, the task on the extra large file is decomposed into several mapping tasks, then one agent will perform reducing operator to gather the result.
- Chain Decomposition: In Figure 2, a complicated mapping task is decomposed into a pipeline of tasks

Solution: rewrite directives

Idea: Templates that describe how to rewrite a subsequence of operators

What they do:

- Encode reusable transformation patterns
- Instantiated by LLM agents
- Require new operators(e.g. split, gather, resolve)

LLMs are actually good at reading large inputs and reasoning about the sequence of operators, such as:

- map $\Rightarrow$ split $\rightarrow$ map $\rightarrow$ reduce
- map $\Rightarrow$ map^{+}:transform a single map into a pipeline of maps
- op $\Rightarrow$ map $\rightarrow$ op: when we need to do preprocessing on the documents

2.3 Cost-Oriented Rewrite Directives

Here are the designs of cost-oriented rewrite directives:

- **Operator fusion**: op $\rightarrow$ op $\Rightarrow$ op: when the adjacent tasks are similar, then reduce two agents to one.

- **Replace with Code**: op $\Rightarrow$ code_op: replace the agent with a deterministic function.
- **Change Models**: op(m) $\Rightarrow$ op(m'): change into a cheaper model while maintaining the same accuracy as before.
- **Add a compression step & push it down**: op$^+$ $\Rightarrow$ map $\rightarrow$ op$^+$: drop irrelevant text from documents, and leave the relevant parts for other operations.

2.4 New operators for DocETL

Implementation of two new operators are discussed in the lecture:

Gather: When splitting documents into chunks, each chunk would lose some context information(e.g. pronounces, he), then agents need to search other chunks with the following strategy:

- Previous/Next Chunks
- Transformed Content: such as a summary of a long prefix
- Document Metadata: such as content, to get information of the section hierarchy

Resolve: LLM generated summaries would be inconsistant(e.g. name), so we need to do entity resolution before we perform reduce operator.

2.5 Searching Over Rewrite Directives

DocETL has implemented 30+ directives that reshape the pipeline, and how do we search the directives and compose them into a pipeline?

One search procedure is to do **recursive tree searching**: an algorithm recursively decomposing the task and finding the best decomposition combination of operators to replace the current operator. When it decides a fusion of operators is better, then it rewrites two operator into one. This procedure is successful in finding the most accurate pipeline.

3 Problem: Optimal Substructure

Classical query optimizers rely on the idea that the optimal plan is composed of the optimal sub-plans. This is the problem of greedy search and what traditional database researchers are working with. However, this idea is not good for the semantic LLM operators, because LLM behavior is context-sensitive and fuzzy. Sometimes we cannot have a clean composition.

3.1 Solution of the Optimal Substructure Problem

Solution: Algorithm of Rewriting Pipelines

The input-output schedule:

- Input: a user's semantic operator pipeline and some "default" LLM for execution.

- Output: a Pareto frontier of plans with the same or higher accuracy and varying costs.
- Algorithm: rewrite the pipeline, "globally", with a limited budget.

The general method of the algorithm:

First - select pipeline

- Selection is deterministic and not LLM-based
- Select a node to rewrite based on itself and it's children's contributions to the frontier

Second and Third - rewrite pipeline and evaluate on samples

- LLM agent decides what directive to apply and creates the new operators (e.g., prompts, code, etc)
- Evaluate the new plan on a sample, recording cost and accuracy

Results - agentic query optimization:

Baseline	Accuracy (@ baseline mac accuracy)	Cost (@ baseline max accuracy)
LOTUS	2.10x	0.487x
PZ (ABACUS)	1.38x	0.575x
GPT-5 Agent	1.95x	0.375x

Findings - what strategies does the optimizer discover

- Model diversity: 100% of optimal plans use different model than the starting model.
- Context pruning: 83% add map operators to remove irrelevant text; some common mechanisms include embedding similarity, keyword filters, etc.
- Agent-authored code: 55% include non-obvious code operations synthesized by the agent.

4 DocWrangler

Problems and solutions when using DocWrangler:

Problem 1: why is semantic data processing so hard and how to solve it?

Solution 1: develop an IDE with pipeline editor and input & output inspector

Problem 2: semantic operators are very expressive, but sometimes it is difficult to know what and how to express

Solution 2: implement an interactive assistant to improve semantic operator prompts

Problem 3: as people iterate, they forget what they meant to improve. How to map the space of user challenges?

Core Idea: Three Gulfs Model

5 Semantic Data Processing & The Three Gulfs Model

5.1 Motivation

The central question motivating the talk is:

> *How do we comprehensively map the space of challenges users face in semantic data processing?*

To approach this question, the speaker references related work, particularly in the domain of **data cleaning**, which shares conceptual similarities with LLM-powered semantic data processing.

5.2 Background: Challenges in Data Cleaning

Traditional data cleaning tasks involve two major categories:

1. **Error Identification:** Locating errors in the dataset.
2. **Error Resolution:** Writing code or pipelines to fix identified errors.

A variety of tools—visualization dashboards, summary statistics, predictive interaction systems, Jupyter notebooks, FlashFill—help address these challenges.

However, these categories do not cleanly transfer to LLM-based semantic data processing, where semantic operators introduce a new abstraction layer that is detached from both the user and the underlying data.

5.3 The Three Gulfs Model

To more accurately describe user challenges in semantic data processing, the speaker introduces the **Three Gulfs Model**, which highlights the cognitive and operational gaps users must bridge. The model is intended as a framework for reasoning about the ecosystem of tools for unstructured data. Not every tool must address all gulfs.

5.3.1 Gulf 1: Comprehension (User $\leftrightarrow$ Data)

Users often do not fully understand the contents of their documents, and LLMs generate additional data requiring human interpretation. Tools must therefore help users comprehend their data at scale.

5.3.2 Gulf 2: Specification (User $\leftrightarrow$ Semantic Operator)

Once users understand their data, they must specify the exact task they want the semantic operator to perform. For example, distinguishing between over-the-counter and prescription medications within the data.

5.3.3 Gulf 3: Generalization (Semantic Operator ↔ Dataset)

After specifying the task, the LLM must generalize correctly across the entire dataset. Because LLMs can be error-prone, this often requires techniques such as DocuTL-style rewrites to improve operator reliability and consistency.

5.4 User Studies and Validation

The research team conducted user studies to validate whether real user behavior maps to the Three Gulfs Model.

Key findings:

- Users naturally developed strategies to bridge each gulf. Examples include:
  - Writing throwaway pipelines to explore and understand the data.
  - Repurposing semantic operators for debugging or validation.
  - Using visualization workflows to interpret operator outputs.
- The Doc Wrangler system was used across many countries and at least nine languages.
- Observed behaviors generalized well across diverse populations and organizations.
- The online Doc Wrangler deployment remains available, requiring users to bring their own API keys.

5.5 Community Growth

Following the open-sourcing of DocETL and Doc Wrangler, the surrounding community has grown substantially:

- Increased GitHub activity and contributions
- Active user discussions on Discord
- Community-generated tutorials, YouTube videos, and blog posts

This growing interest underscores the value of developing robust tools for semantic data processing.

5.6 Evaluation at Scale

The speaker briefly discusses evaluation methodologies for semantic operators, an area supported by parallel research efforts.

5.6.1 Insight 1: Identifying Meaningful Evaluation Criteria

By mining real analyst prompt histories, the team can identify evaluation criteria that matter to users. Examples include detecting sensitive information or maintaining professional tone.

An algorithm was developed to:

1. Extract prompts from user histories,
2. Generate candidate evaluation functions,
3. Select a minimal set of evaluations that cover common LLM failure patterns.

An open-source dataset of developer prompts for semantic operators was also released.

5.6.2 Insight 2: Using Front-End Wait Time for Adaptive Evaluation

Running semantic operators can be slow, causing user wait time in tools like Doc Wrangler. The team explored using this idle time to collect user labels that improve operator performance.

Findings:

- This approach works and was described in a WISP paper.
- Real-time user labeling revealed **criteria drift**:
  - Users may initially label certain outputs as incorrect (e.g., extracting hashtags as entities),
  - Later, after seeing more examples, they may change their minds.
- Evaluation rubrics evolve as users see more LLM behavior:
  - Users add new criteria,
  - Users reinterpret existing criteria,
  - Users adapt their expectations based on scale.
- Implication: It may be unwise to run semantic operators across the full dataset before establishing alignment with the user on a smaller sample.

5.7 Conclusion

The speaker concludes by summarizing the contributions of the work:

- Exploration of key research questions in semantic data processing,
- Introduction of system primitives such as semantic operators and rewrite directives,
- Development of conceptual frameworks like the Three Gulfs Model,
- Emphasis on co-designing systems and interfaces to support new workflows in data processing.