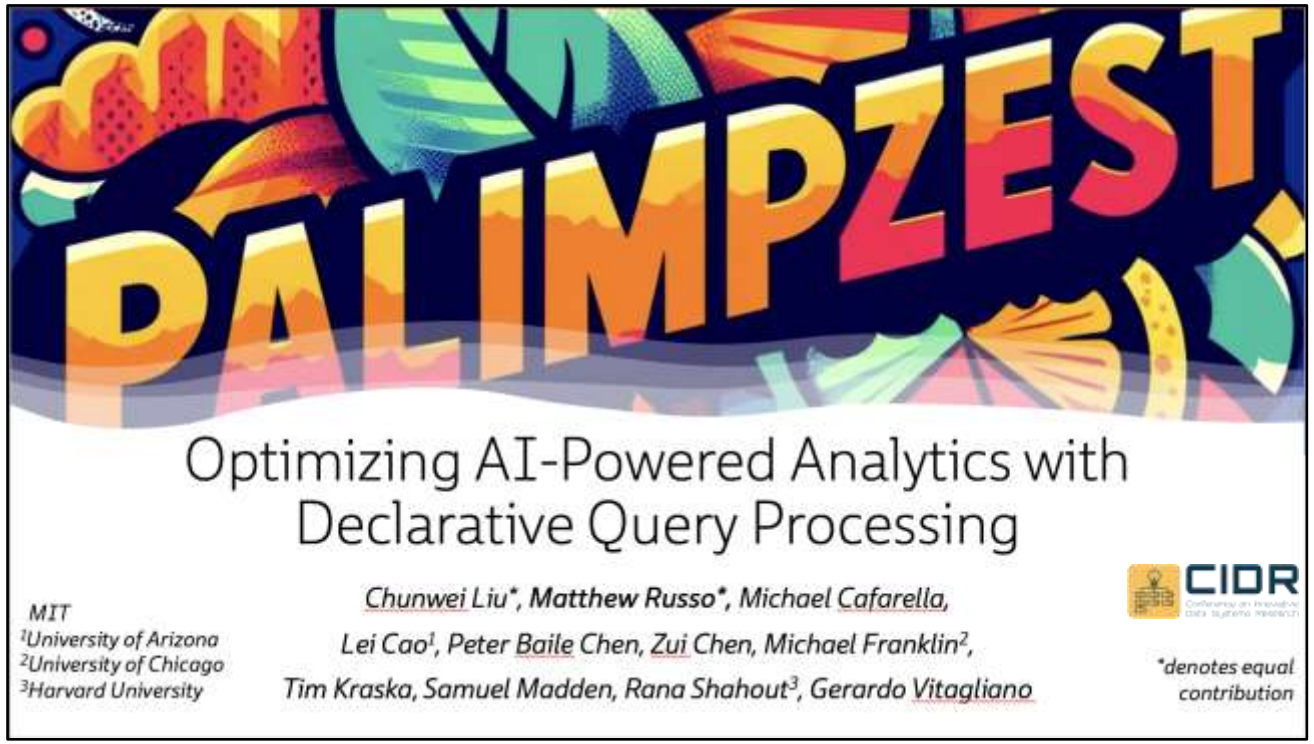
# Deep Research is the New Analytics System: Towards Building the Runtime for AI-Driven Analytics

**Matthew Russo, Tim Kraska**

MIT CSAIL Computer Science & Artificial Intelligence Laboratory

# Refresher from CIDR '25: Palimpzest & Semantic Operators

**Palimpzest:** declarative programming framework for **semantic operators** (i.e. AI-powered data transformations).





Chunwei Liu*


Matthew Russo*


Michael Cafarella


Lei Cao


Peter Baile Chen


Zui Chen


Michael Franklin


Tim Kraska


Sam Madden


Rana Shahout

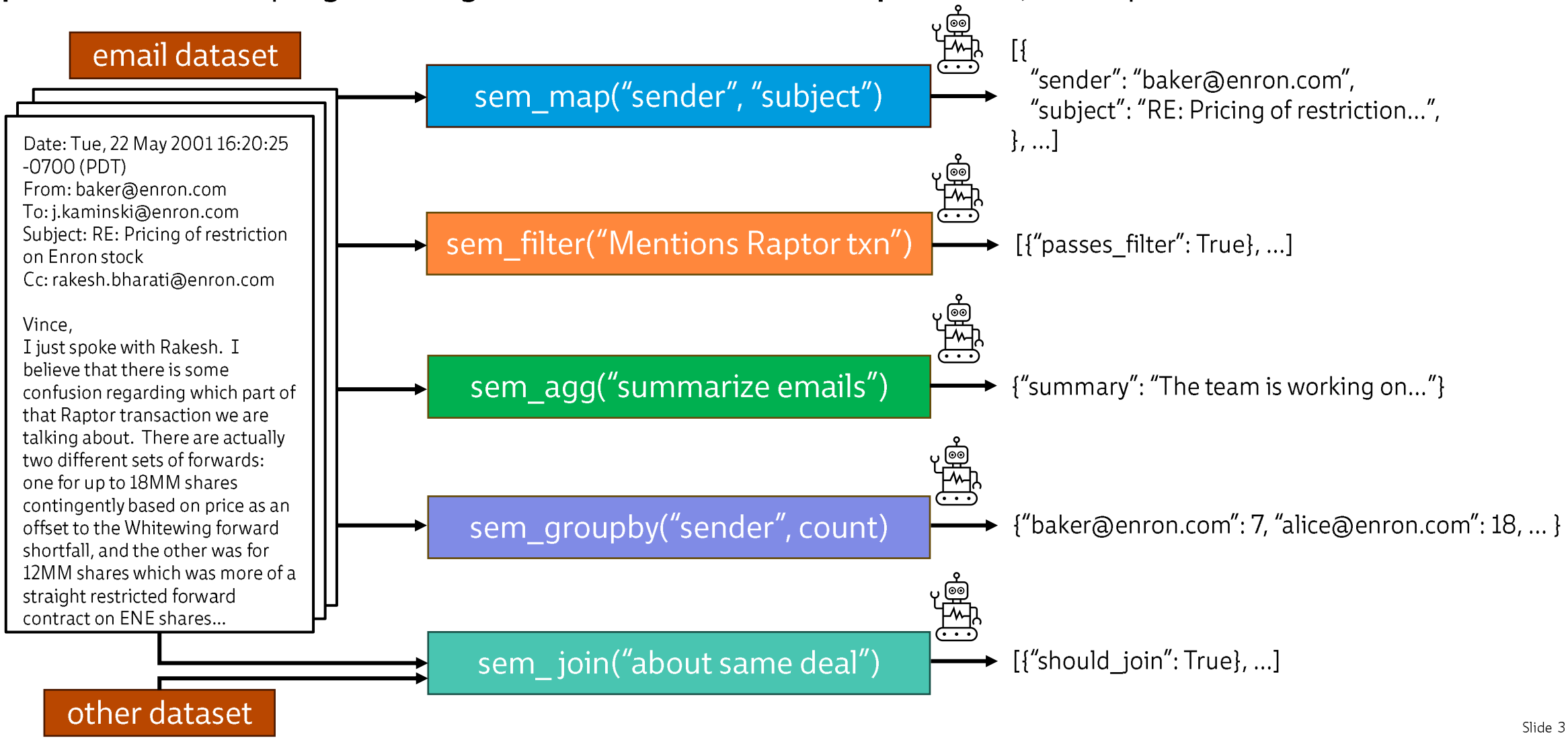

Gerardo Vitagliano

Paper

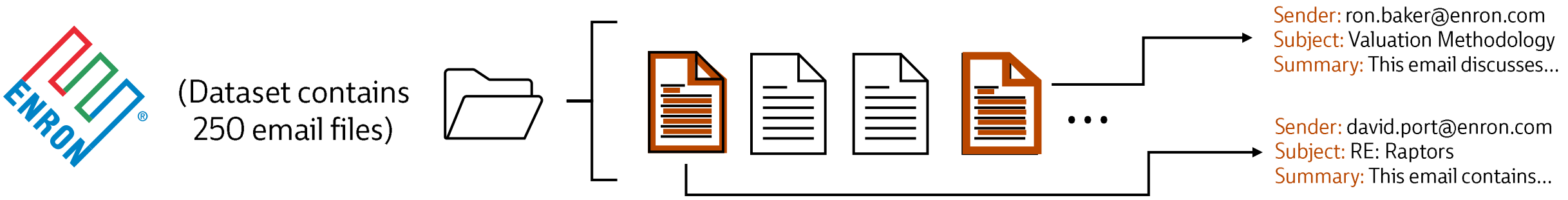
[1] Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing, Liu et al., CIDR'25

# Refresher from CIDR '25: Palimpzest & Semantic Operators

**Palimpzest:** declarative programming framework for **semantic operators** (i.e. AI-powered data transformations).



# Ex: Filtering Enron Emails with Palimpzest

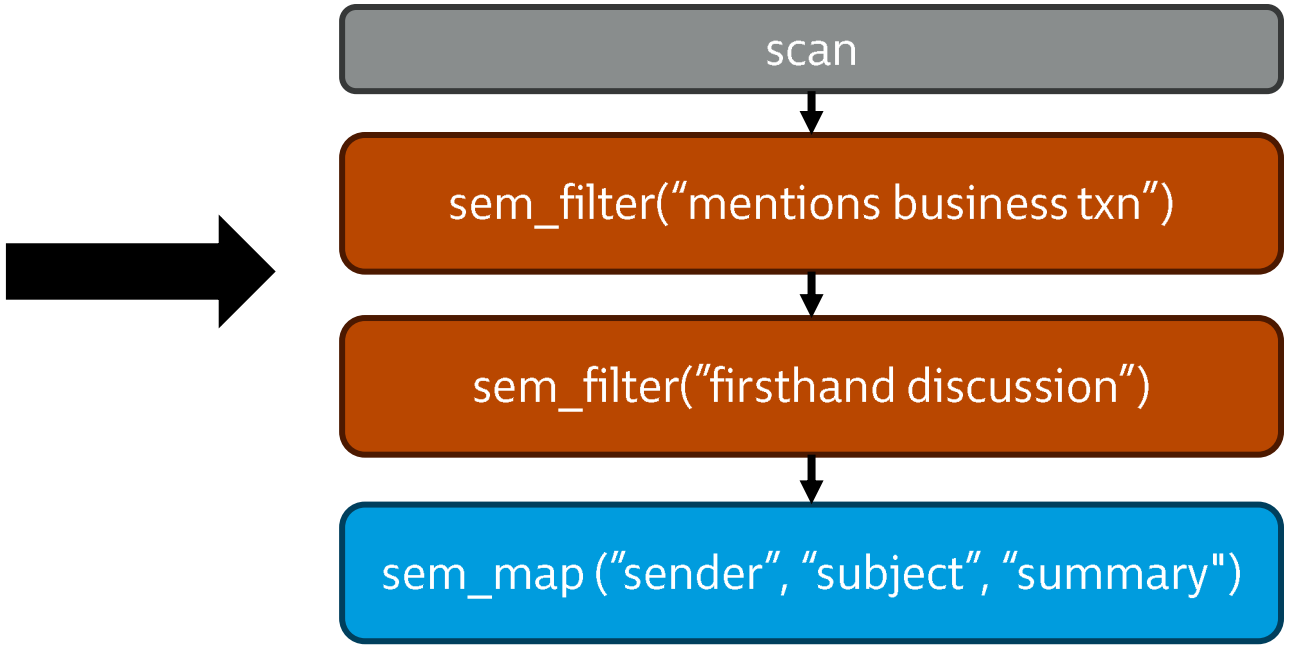



**Q1:** "Compute the sender, subject, and a summary for all emails which contain firsthand discussion of these four business transactions: Raptor, Chewco, Fat Boy, and Deathstar"

**A:**

```
# write PZ program for query
ds = pz.Dataset("emails/")
ds = ds.sem_filter("business txns")
ds = ds.sem_filter("firsthand disc.")
ds = ds.sem_map([
   "sender", "subject", "summary"
])
output = ds.optimize_and_run()

print(output.to_df())
```



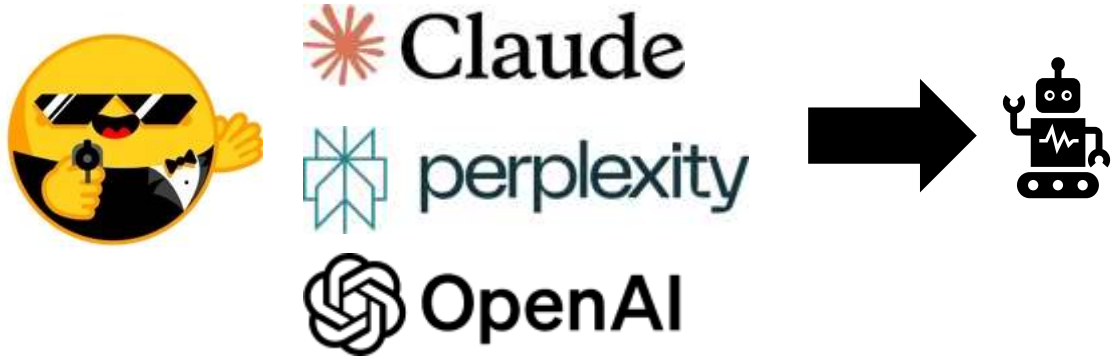
# Deep Research: The AI Community's Answer for AI-Driven Analytics

**High-Level Idea:** let an LLM plan, reason, use tools, and/or execute code to solve tasks in multi-step fashion


**Q1:** "Compute the sender, subject, and a summary for all emails which contain firsthand discussion of these four business transactions: Raptor, Chewco, Fat Boy, and Deathstar"

**A:**

**Deep Research Agent**




**Step 1:**

Here's my plan:
1. List the files in the dataset with the list_filepaths() tool
2. Filter for files with "Raptor", "Chewco", ..., in their text
3. Print each file to check whether it has firsthand discussion of the transaction
4. Return the final set of files

**Step 2:**
```
# list all files in dataset
list_filepaths()
```

**Step 3:**
```
# filter for files with "Raptor"
raptor_files = []
for file in os.listdir("emails/"):
    if "raptor" in file.lower():
        ...
```

**Step 4:**
```
# spot-check files with "Raptor"
for file in raptor_files[:3]:
    print(file)
```

**Step 5:**
```
# set final files; compute fields
final_files = ["file1.txt", ...]
...
```


[1] SmolAgents. 2025. Open-source DeepResearch – Freeing our search agents. HuggingFace Team.

# Deep Research: The AI Community's Answer for AI-Driven Analytics

**High-Level Idea:** let an LLM plan, reason, use tools, and/or execute code to solve tasks in multi-step fashion

Which approach is better?

Should we use Semantic Operators? Or is Deep Research all you need?

# Let's Test Semantic Operators and Deep Research on Two Simple Queries

(Dataset contains 250 email files)


**Q1:** "Compute the sender, subject, and a summary for all emails which contain firsthand discussion of these four business transactions: Raptor, Chewco, Fat Boy, and Deathstar"

---

KramaBench

(Dataset contains 132 CSV files)

1. Find the right file
2. Parse table w/per-year identity theft counts
3. Compute the ratio

**Q2:** "What is the ratio of identity theft reports in 2024 vs. 2001? Round to 4 decimal places."

[1] KramaBench: A Benchmark for AI Systems on Data-to-Insight Pipelines over Data Lakes, Lai et al., (In Submission)

# Is One System Dominant for Unstructured Analytics?

Q1: Compute the sender, subject, and a summary for all emails which contain firsthand discussion of these four...

**Semantic Operators**

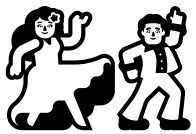
scan

sem_filter("mentions business txn")

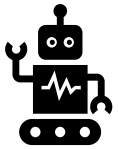
sem_filter("firsthand discussion")

sem_map ("sender", "subject", "summary")

F1: 0.99 || Cost: $0.87 || Time: 546s



**Deep Research**

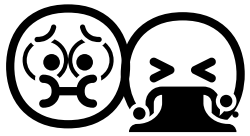


```
# filter for files with "Raptor"
raptor_files = []
for file in os.listdir("emails/"):
    if "raptor" in file.lower():
        ...
```

```
# spot-check files with "Raptor"
for file in raptor_files[:3]:
    print(file)
```

```
# set final files and compute fields
final_files = ["file1.txt", ...]
...
```

F1: 0.51 || Cost: $0.08 || Time: 37s



Q2: What is the ratio of identity theft reports in 2024 vs. 2001? Round the answer to 4 decimal places

**Semantic Operators**

scan

scan

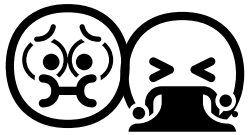
sem_filter("2024 identity thefts")

sem_filter("2001 identity thefts")

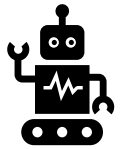
join

map ("ratio")

Pct. Err: 17.0% || Cost: $1.7 || Time: 215s



**Deep Research**

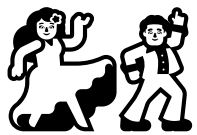


```
# list all files in dataset
list_filepaths()
```

```
# read files of interest
files = [...]
for file in files:
    print(file)
```

```
# compute ratio in Python
2001_thefts = ...
2024_thefts = ...
ratio =
2024_thefts/2001_thefts
print(f"{ratio:.4f}")
```

Pct. Err: 0.0% || Cost: $0.05 || Time: 30s



# Is One System Dominant for Unstructured Analytics?

Q1: Compute the sender, subject, and a summary for all emails which contain firsthand discussion of these four...

**Semantic Operators**

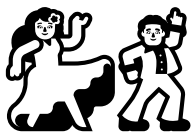
scan

sem_filter("mentions business txn")

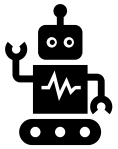
sem_filter("firsthand discussion")

sem_map ("sender", "subject", "summary")

F1: 0.99 || Cost: $0.87 || Time: 546s



**Deep Research**

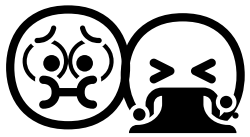


```
# filter for files with "Raptor"
raptor_files = []
for file in os.listdir("emails/"):
    if "raptor" in file.lower():
        ...
```

```
# spot-check files with "Raptor"
for file in raptor_files[:3]:
    print(file)
```

```
# set final files and compute fields
final_files = ["file1.txt", ...]
...
```

F1: 0.51 || Cost: $0.08 || Time: 37s



Q2: What is the ratio of identity theft reports in 2024 vs. 2001? Round the answer to 4 decimal places

**Semantic Operators**

scan

scan

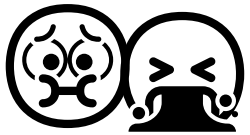
sem_filter("2024 identity thefts")

sem_filter("2001 identity thefts")

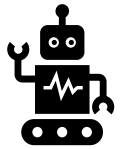
join

map ("ratio")

Pct. Err: 17.0% || Cost: $1.7 || Time: 215s



**Deep Research**



```
# list all files in dataset
list_filepaths()
```

```
# read files of interest
files = [...]
for file in files:
    print(file)
```

```
# compute ratio in Python
2001_thefts = ...
2024_thefts = ...
ratio =
2024_thefts/2001_thefts
print(f"{ratio:.4f}")
```

Pct. Err: 0.0% || Cost: $0.05 || Time: 30s

Pct. Err: 87.8% || Cost: $0.05 || Time: 121s

# Is One System Dominant for Unstructured Analytics?

Q1: Compute the sender, subject, and a summary for all emails which contain firsthand discussion of these four...

Semantic Operators | Deep Research

Q2: What is the ratio of identity theft reports in 2024 vs. 2001? Round the answer to 4 decimal places

Semantic Operators | Deep Research

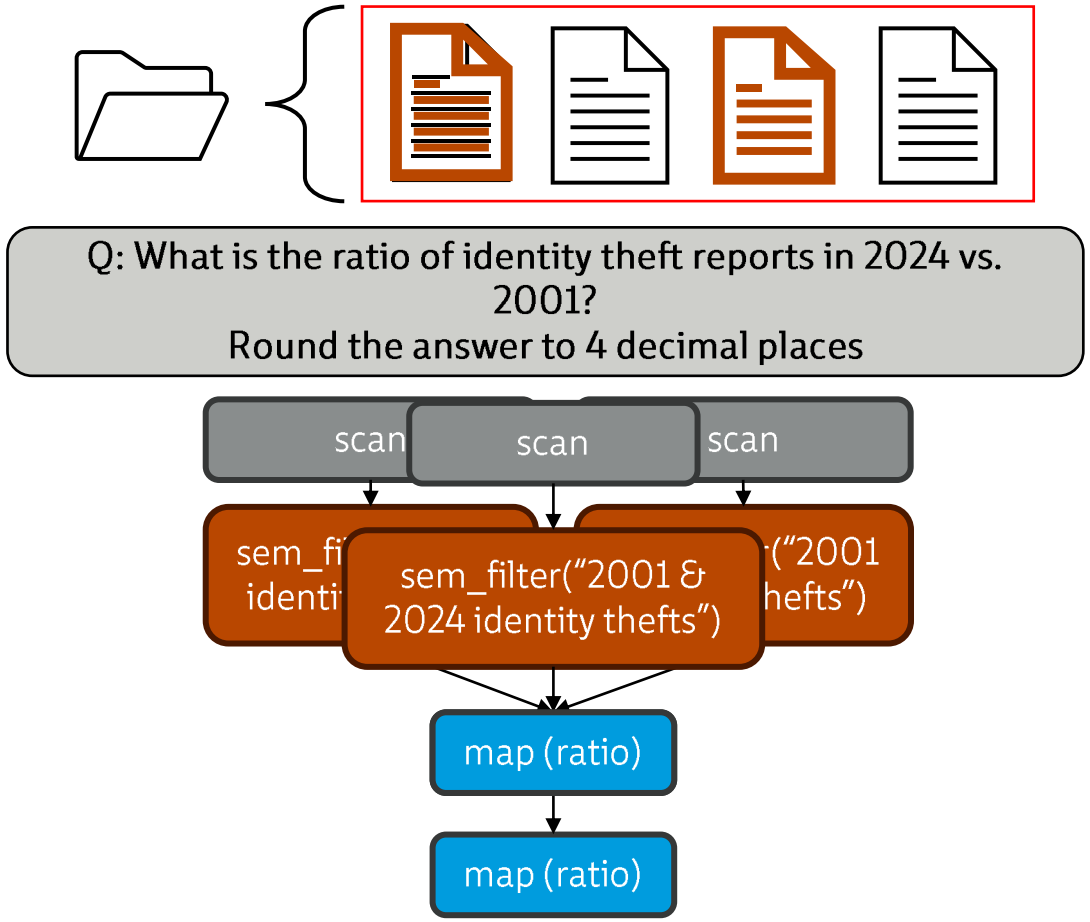
Neither system "wins" on both queries!

What causes each system to underperform?

# Observation 1: Semantic Operators Lack Flexibility in their Execution Model and Can be Difficult to Program With

- Iterator execution (i.e. scan operator) forces plan to process every file (twice)
  - Cannot simply stop once we find the right file!

- **Correctness** of semantic operator programs is **data dependent**
  - E.g. without knowing the data layout, writing the correct NL filter predicate is challenging

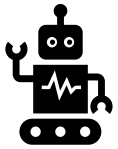


# Observation 2: Deep Research Agents Can Write Good Query Plans, But Suffer from Poor Query Execution

- The high-level query plan generated by the Deep Research agent is quite reasonable:
  1. Filter for files with keywords
  2. Read the files to confirm they satisfy filters
  3. Extract subject, sender, summary for final files

- However, the query execution is poor:
  - Keywords are good, but not sufficient
  - Reads only a subset of files in step (2.)

Q: Compute the sender, subject, and a summary for all emails which contain firsthand discussion of these four...

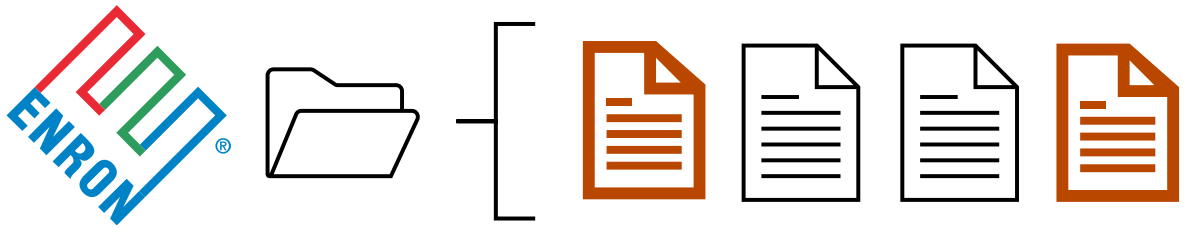


```
# filter for files with "Raptor"
raptor_files = []
for file in os.listdir("emails/"):
    if "raptor" in file.lower():
        ...
```

```
# spot-check files with "Raptor"
for file in raptor_files[:3]:
    print(file)
```

```
# set final files and compute fields
final_files = ["file1.txt", ...]
...
```

# Simple Idea: Allow Deep Research Agents to Write PZ Programs



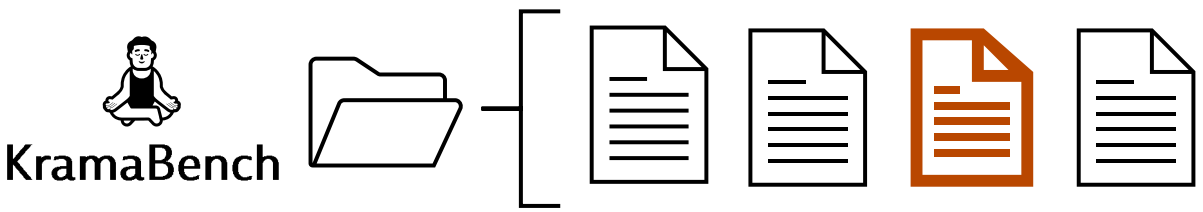

Q1: Compute the sender, subject, and a summary for all emails which contain firsthand discussion of these four...

```
# filter for files with "Raptor"
raptor_files = []
for file in os.listdir("emails/"):
    if "raptor" in file.lower():
        ...

# spot-check files with "Raptor"
for file in raptor_files[:3]:
    print(file)

# set final files and compute
```

scan → sem_filter 1 → sem_filter 2 → sem_map




Q2: What is the ratio of identity theft reports in 2024 vs. 2001? Round the answer to 4 decimal places

```
# list all files in dataset
tool_list_filepaths()

# read files of interest
files = [...]
for file in files:
    print(file)

# compute ratio in Python
2001_thefts = ...
2024_thefts = ...
ratio =
2024_thefts/2001_thefts
print(f"{ratio:.4f}")
```

```
# list all files in dataset
tool_list_filepaths()

# filter for files of interest
ds = pz.Dataset("legal/")
ds = ds.sem_filter("theft #s")
ds.optimize_and_run()

# compute ratio in Python
2001_thefts = ...
2024_thefts = ...
ratio = 2024_thefts/2001_thefts
print(f"{ratio:.4f}")
```

# Evaluation Setup

We evaluate (a subset of) the following systems on each query:

1. **PZ Plan:** a handwritten PZ plan
2. **CodeAgent:** Naïve Deep Research Agent (from HF's SmolAgents library)
3. **CodeAgent + Sem. Ops.:**
    1. Sem_map + sem_filter with GPT-4o
    2. Can write python code which uses these tools to map / filter over input files
4. **Prototype:** CodeAgent + instruction & demonstrations for how to write PZ

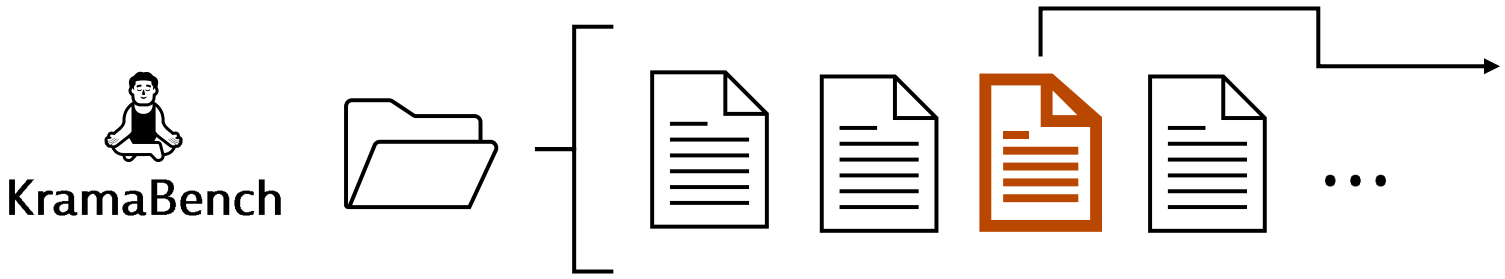
# Eval Q1: Can our Prototype Mitigate Deep Research's Weaknesses?


**Q:** "Compute the sender, subject, and a summary for all emails which contain firsthand discussion of these four business transactions: Raptor, Chewco, Fat Boy, and Deathstar"

| | Total Cost ($) | Runtime (s) | F1 | Recall | Precision |
|---|---|---|---|---|---|
| CodeAgent | $0.08 | 37.0 | 50.53% | 46.15% | 88.89% |
| CodeAgent + Sem. Ops. | $3.76 | 1,999.9 | 98.67% | 97.44% | 100% |
| **Prototype** | $0.87 | 546.2 | 98.67% | 97.44% | 100% |

PZ's Optimizer (Abacus) is crucial for optimizing semantic operator performance in Code Agent's plan!

# Eval Q2: Can our Prototype Mitigate Semantic Operators' Weaknesses?




1. Find the right file
2. Parse table w/per-year identity theft counts
3. Compute the ratio

**Q:** "What is the ratio of identity theft reports in 2024 vs. 2001? Round to 4 decimal places."

| | Total Cost ($) | Runtime (s) | Pct. Err |
|---|---|---|---|
| PZ Plan | $1.66 | 215.2 | 17.00% |
| CodeAgent | $0.03 | 77.0 | 27.56% |
| **Prototype** | $1.17 | 583.0 | **0.02%** |

Avg. of sometimes correct, sometimes wildly wrong!

(1) PZ benefits from Deep Research's dynamic execution
(2) Deep Research benefits from semantic filter finding the right file

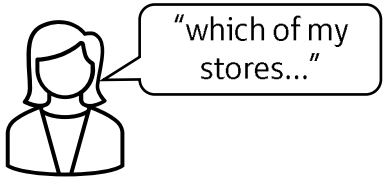
# Eval Q2: Can This Approach Mitigate Semantic Operators' Weaknesses?

So that's it? Have we fully solved the problem of building the runtime for AI-driven analytics?

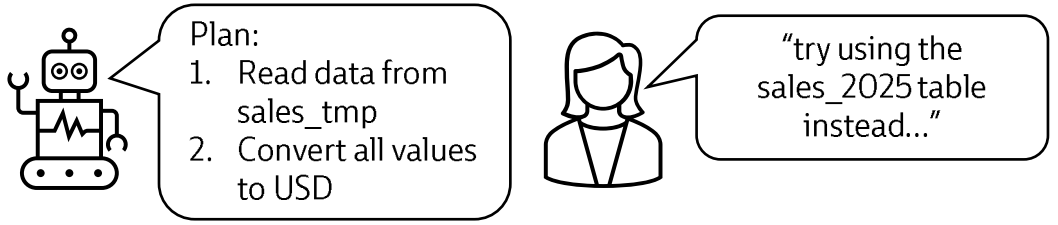
# What do Users Need from an AI-Driven Analytics System?

**Desiderata:**

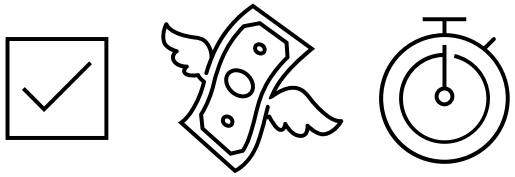
1. Natural Language Interface




2. Interpretable / Interactive Execution Plan(s)




3. Performance (quality, cost, latency)



4. Scalability to large datasets (e.g. 100k+ files)

# Can We Build a System Which Achieves our Desiderata?

| | Optimized Execution | Dynamic Execution | Supports NL Queries | Interpretable Execution | Supports Unstructured Data | Scalability to Large Datasets |
|---|---|---|---|---|---|---|
| OLAP Databases | ☑ | ✗ |  | ☑ | ✗ | ☑ |
| Semantic Operator Systems | ☑ | ✗ |  | ✗ | ☑ |  |
| Deep Research Systems | ✗ | ☑ | ☑ | ✗ | ☑ |  |
| Carnot | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |

# Thank you! (Q&A)

Email: mdrusso@mit.edu