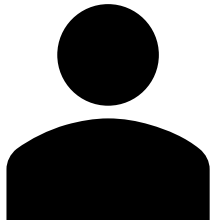
LOTUS: Enabling Bulk Semantic Processing with LLMs Using Semantic Operators

Liana Patel, Sid Jha, Parth Asawa, Melissa Pan, Carlos Guestrin, and Matei Zaharia

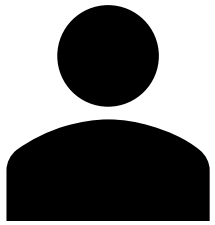
How should we ask questions over large amounts of data?

How should we ask questions over large amounts of data?



Title	Arxiv URL	Abstract	Authors	H-Index	Date Published	Categories

How should we ask questions over large amounts of data?



How many papers claim to achieve SOTA on BIRD?

Summarize recent papers on vector databases

Which paper about vector databases has the funniest title?

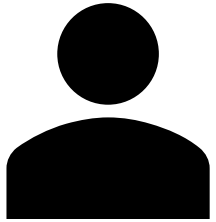
Which papers obtains the highest MMLU score?

Which papers contradict claims made by other papers?

Which papers contradict the claim that LMs can efficiently use long context?

Title	Arxiv URL	Abstract	Authors	H-Index	Date Published	Categories

How should we ask questions over large amounts of data?



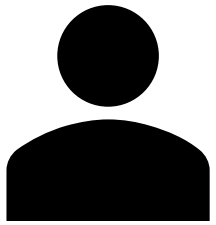
Which paper about vector databases has the funniest title?

Title	Arxiv URL	Abstract	Authors	H-Index	Date Published	Categories
			...			

How should we ask questions over large amounts of data?

RAG

Good for point lookups, not bulk processing



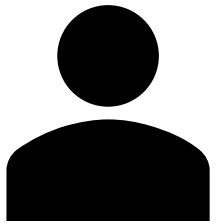
Which paper about vector databases has the funniest title?

Title	Arxiv URL	Abstract	Authors	H-Index	Date Published	Categories
...						

How should we ask questions over large amounts of data?

RAG

Good for point lookups, not bulk processing

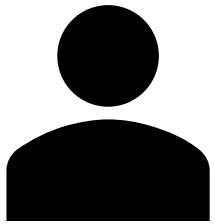


Which paper about vector databases has the funniest title?

Title	Arxiv URL	Abstract	Authors	H-Index	Date Published	Categories
Full Text Indexing in Open Source **DBMS**		...A **vector** of tokens...Let's say we have a **database**	...			
How can **Catchy Titles** be Generated		...which we remember, a **funny title** for example				

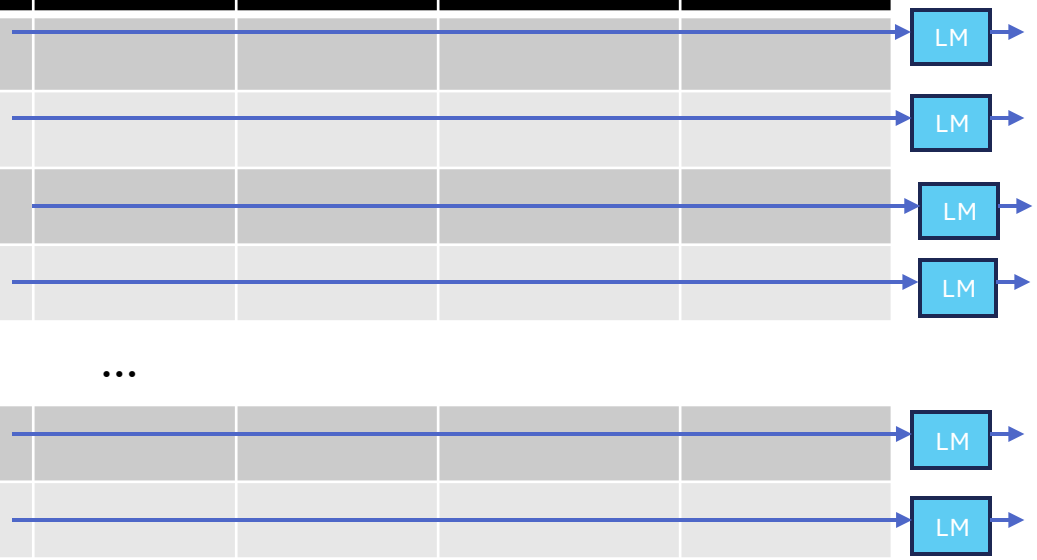
→ LLM

How should we ask questions over large amounts of data?



Which paper about vector databases has the funniest title?

Title	Arxiv URL	Abstract	Authors	H-Index	Date Published	Categories
...						



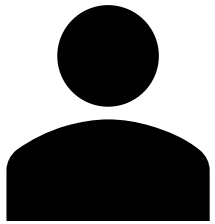
RAG

Good for point lookups, not bulk processing

SQL with LLM UDF

Limited to row-wise LLM execution

How should we ask questions over large amounts of data?



Which paper about vector databases has the funniest title?

Title	Arxiv URL	Abstract	Authors	H-Index	Date Published	Categories
...						

→ LLM

RAG

Good for point lookups, not bulk processing

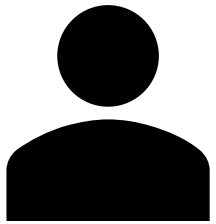
SQL with LLM UDF

Limited to row-wise LLM execution

LLM( )

Scaling issues

How should we ask questions over large amounts of data?



Which paper about vector databases has the funniest title?

Title	Arxiv URL	Abstract	Authors	H-Index	Date Published	Categories
Social Choice for Heterogeneous Fairness						
...						

→ LLM

RAG

Good for point lookups, not bulk processing

SQL with LLM UDF

Limited to row-wise LLM execution

LLM( )

Scaling issues

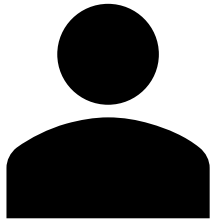
What is LOTUS?

LLMs **O**ver **T**ables of **U**nstructured & **S**tructured data

A query engine for reasoning over large corpuses of data

What is LOTUS?

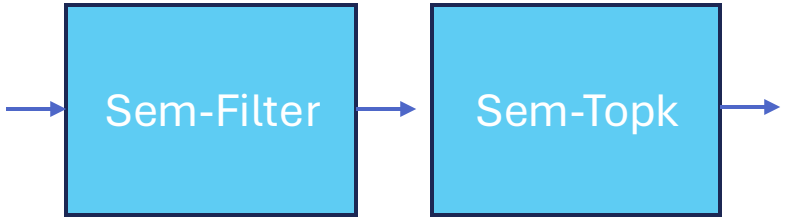
A query engine for reasoning over large corpuses of data



Which paper about vector databases has the funniest title?

Title	Arxiv URL	Abstract	Authors	H-Index	Date Published	Categories
✗	✗	✗	✗	✗	✗	✗
3						
✗	✗	✗	✗	✗	✗	✗
1						
2						

Bulk semantic processing

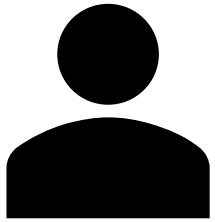


Select papers discussing VDBs

Rank papers by funniest title

What is LOTUS?

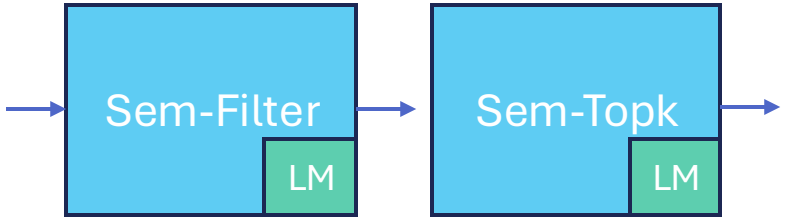
A query engine for reasoning over large corpuses of data



Which paper about vector databases has the funniest title?

Title	Arxiv URL	Abstract	Authors	H-Index	Date Published	Categories
✗	✗	✗	✗	✗	✗	✗
3						
✗	✗	✗	✗	✗	✗	✗
1						
2						

Bulk semantic processing



Select papers discussing VDBs

Rank papers by funniest title

LOTUS: Two-fold Goals

A query engine for reasoning over large corpuses of data

Ease of Use

Efficiency

LOTUS: Key Ideas

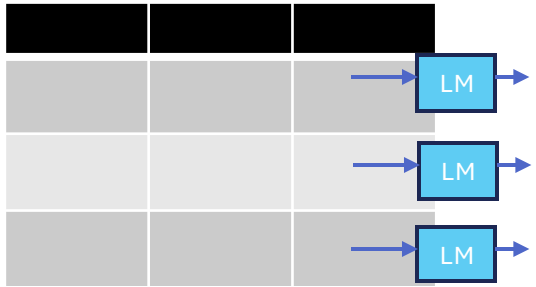
- Declarative programming model
- Automatic optimization
- Leverage structured data, embeddings, and unstructured data together

Programming with LOTUS

pandas

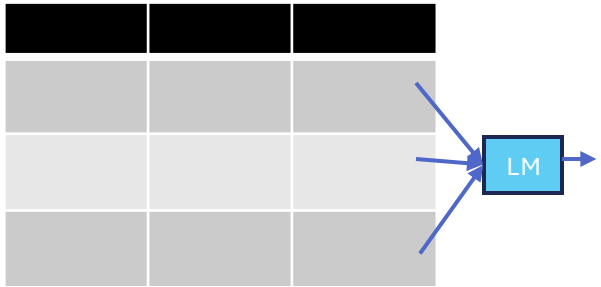
Key idea: ***semantic operators*** provide a declarative programming model that seamlessly extends the relational model

LLM Transformations



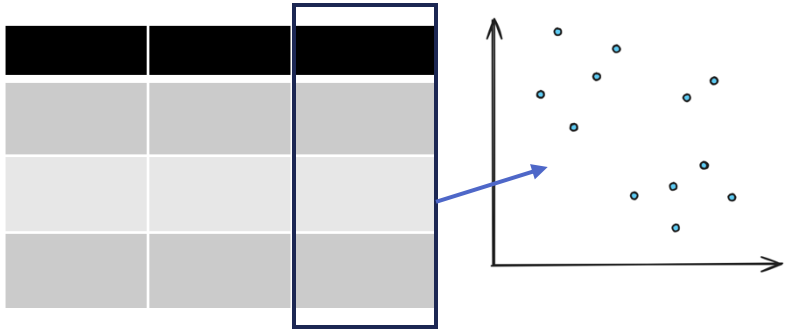
Eg) sem_map, sem_filter, sem_join

LLM Aggregations



Eg) sem_agg, sem_topk

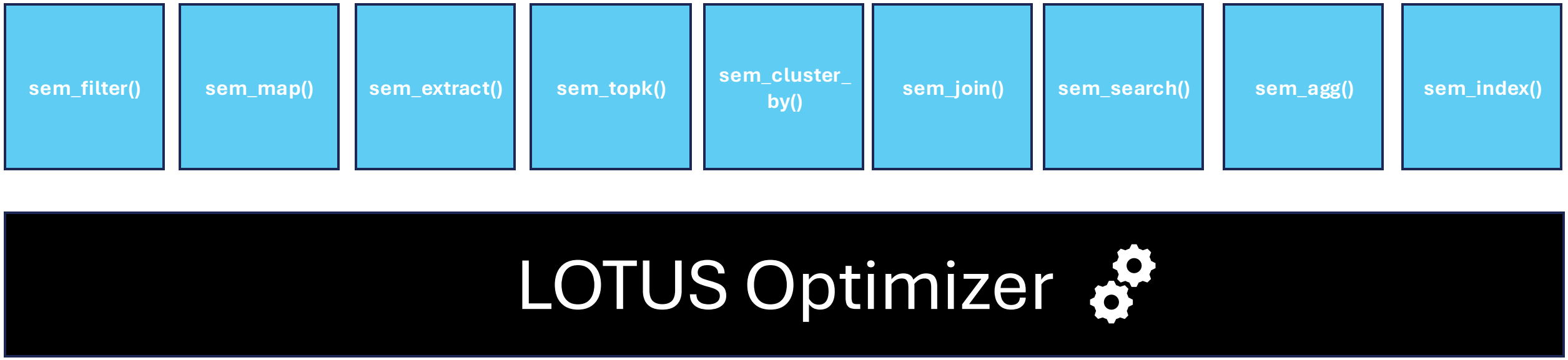
Similarity-based Operations



Eg) sem_search, sem_cluster_by

Under-the-hood: LOTUS Optimizer

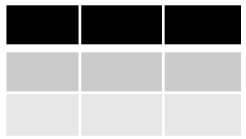
Semantic operators create a rich design space and can be ***transparently optimized***



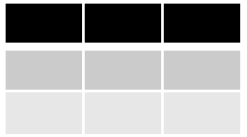
Efficient batch processing, algorithmic approximations, clustering, etc.

Under-the-hood: LOTUS Optimizer

Eg) Algorithmic Approximations for Sem-Join

T1 

Arxiv Papers

T2 

Wikipedia Articles

sem_join()

Does the {arxiv_paper} contradict a claim made by the {wiki_article}?

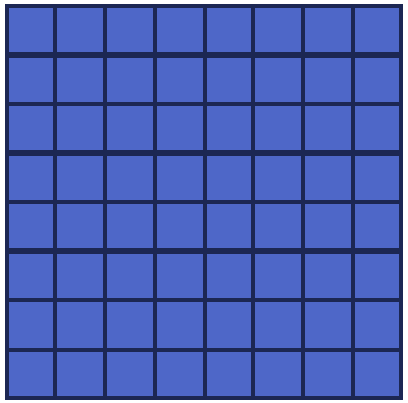
Under-the-hood: LOTUS Optimizer

Eg) Algorithmic Approximations for Sem-Join

Exact Implementation

T2
Wikipedia Articles

T1
Arxiv Papers

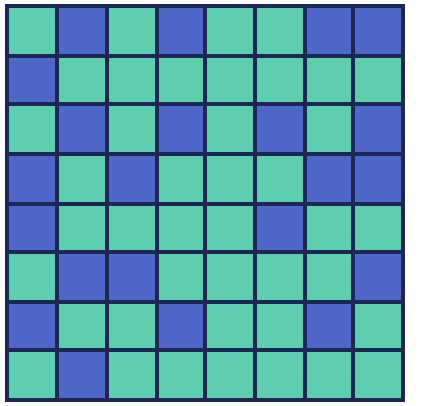


LM-based predicate evaluation

Learned approximation with probabilistic guarantees on accuracy!

T2
Wikipedia Articles

T1
Arxiv Papers

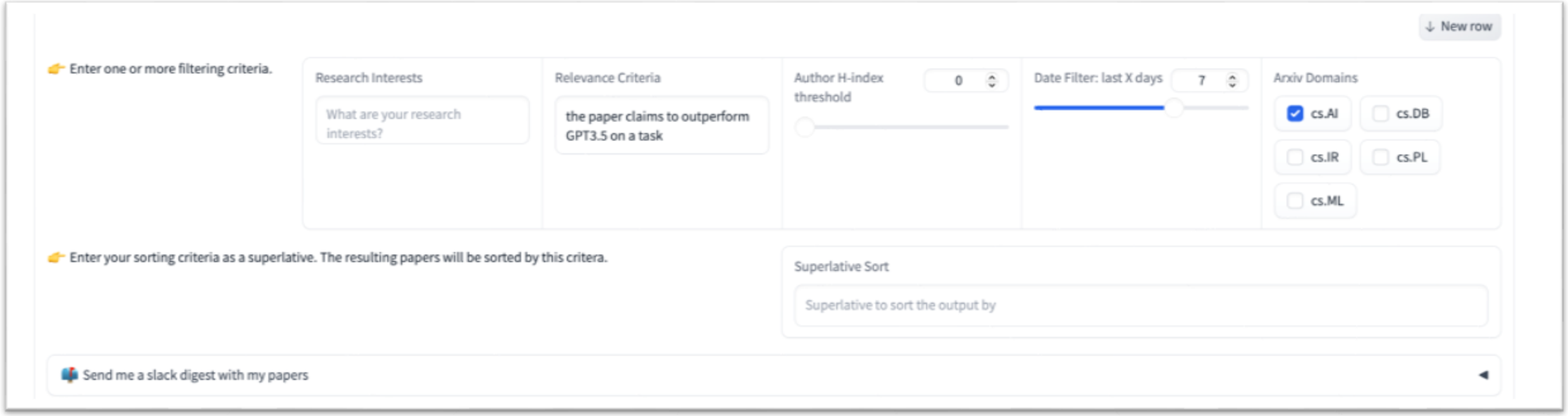


LM-based predicate evaluation

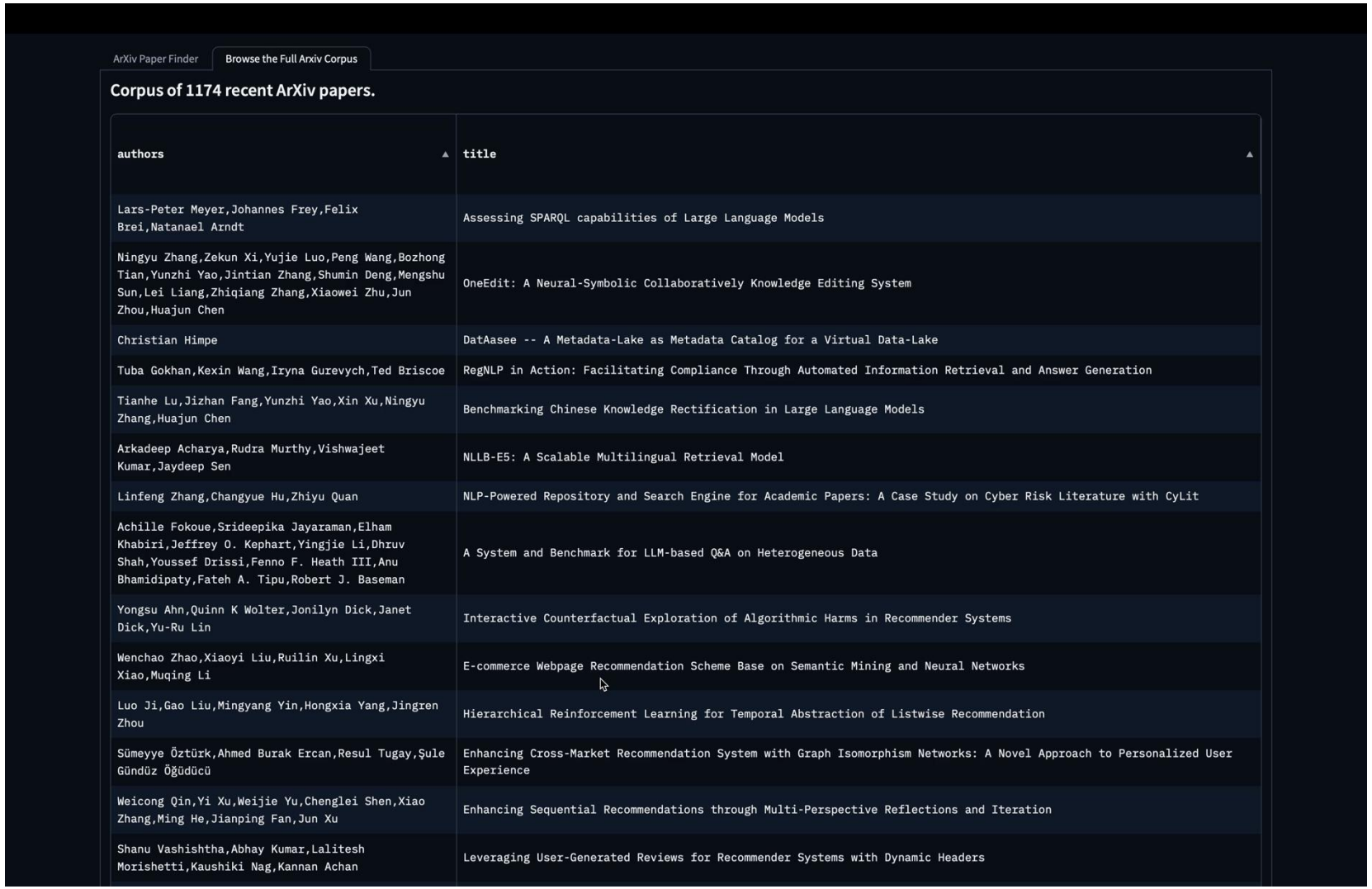
Inexpensive proxy

Demo: ArXiv Paper Search

- https://b473ef4e57d606fb03.gradio.live



Demo: ArXiv Paper Search

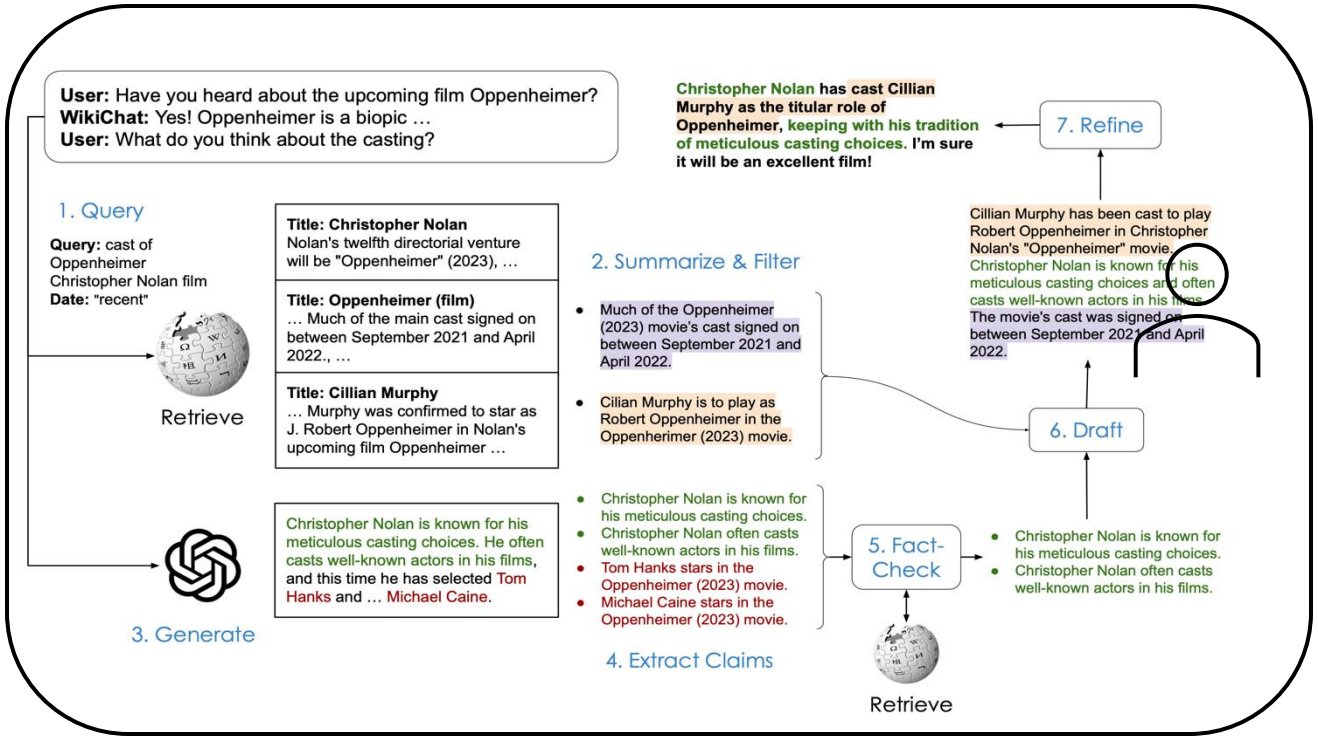


Programming with LOTUS

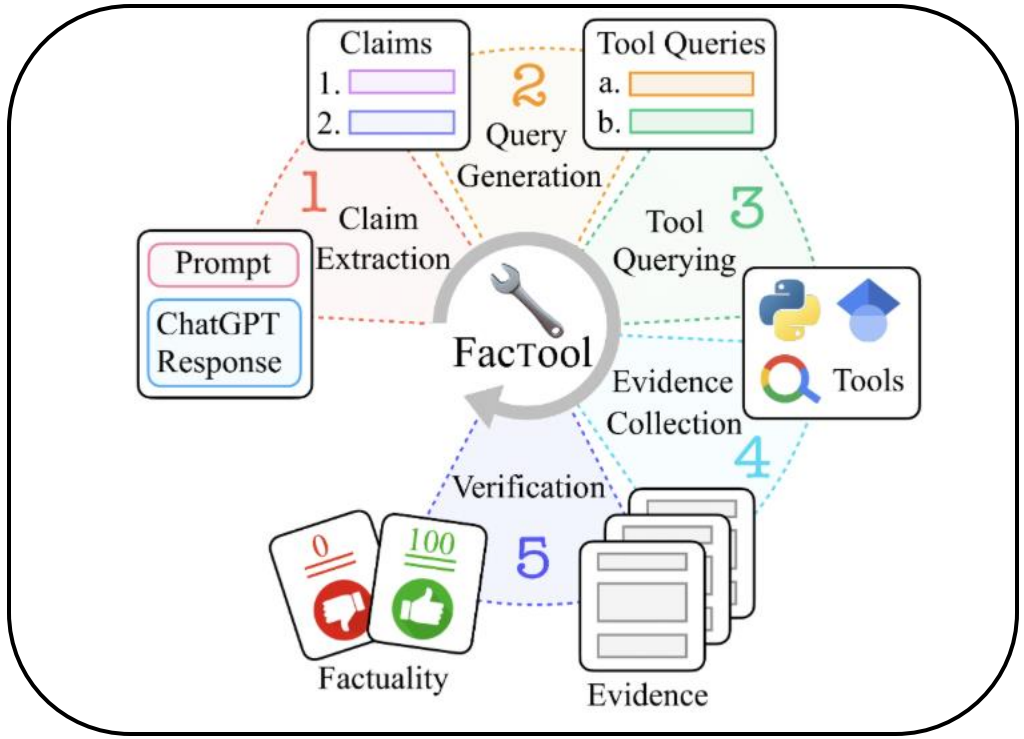
- Query logic is expressed in <20 lines of code!

```
# filter by hindex
papers_df = papers_df[papers_df["max_author_hindex"] > hindex_threshold]

# filter by domain on categories in papers_df
papers_df = papers_df[papers_df["categories"].apply(lambda x: any([d in x for d in domain_filter]))]

# filter by date
now = datetime.datetime.now(pytz.utc)
papers_df = papers_df[papers_df["date_published"] > now - pd.Timedelta(days=days_filter)]

# filter by topic and relevance criteria
papers_df.sem_search("abstract", research_topics, 100)\
    .sem_filter(f"Based on the paper {{title}} and {{abstract}} of each paper, the paper is likely to be highly relevant to the use
    .sem_filter(f"Based on the paper {{title}} and {{abstract}}, the paper meets the following critera: {relevance_criteria}")\
    .sem_topk(f"Which {{abstract}} is{sort_query}", K=20)\
    .sem_agg(f"You are writing a digest for a user who wants to catch up on recent papers. Write a summary discussing each the impo
```

Use Case: fact-checking



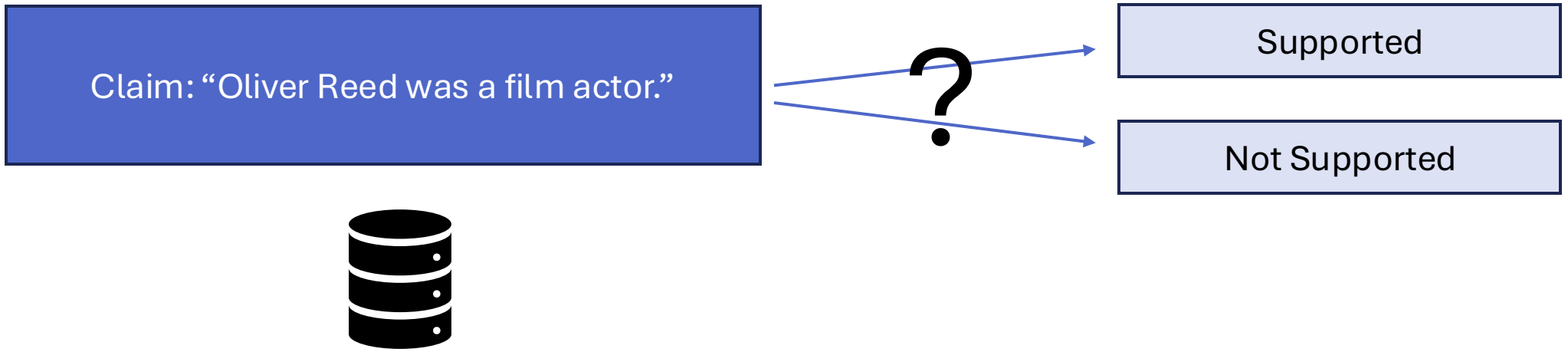
Wikichat (2023)



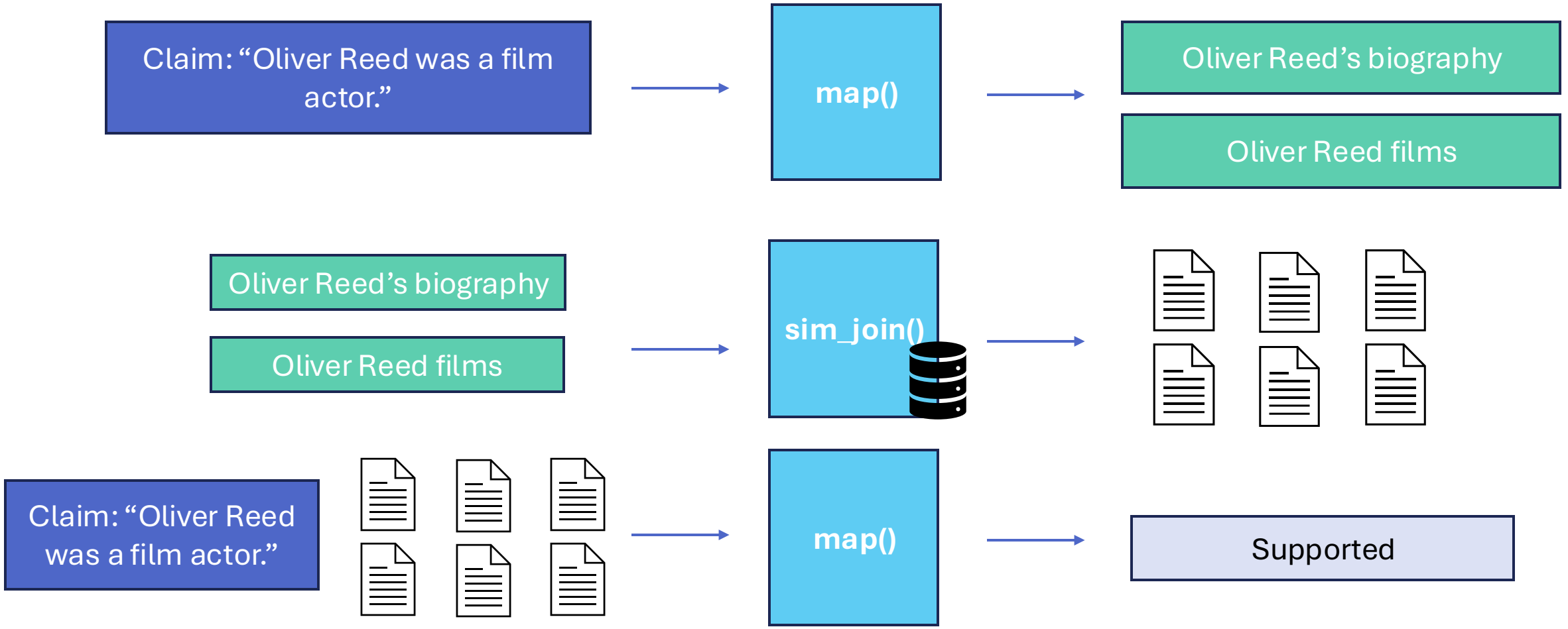
FacTool (2023)

Fact-Checking with FacTool

Method	Accuracy
FacTool	83.5



Fact-Checking with LOTUS



Fact-Checking with LOTUS

map()

sim_join()

map()

```
1 wiki_df.load_sem_index("article", "index_dir")
2
3 claim_df.sem_map("write 2 search queries given the {
      claim}", name="query")\
4     .sem_sim_join(wiki_df, left_on="query", right_on="
      articles",  K=10)\
5     # concatenate articles for each claim
6     .groupby(["claim"]).apply(lambda x: "\n".join(x["
      articles"]))\
7     .sem_map("Identify whether there are any factual
      errors in the {claim} based on the {articles}.
      Include your resasoning, any errors found in the
      claim, and the factuality of the claim.")
```

LOTUS-FacTool pipeline (<50 LoC)

Fact-Checking with LOTUS

map()

sim_join()

filter()

```
wiki_df.load_sem_index("article", "index_dir")

claim_df.sem_map("write 2 search queries given the {claim}", name="query")\
    .sem_sim_join(wiki_df, left_on="query", right_on="articles",  K=10)\
    # concatenate articles for each claim
    .groupby(["claim"]).apply(lambda x: "\n".join(x["articles"]))\
    .sem_filter("given the {context}, the {claim} is factual.", confidence_threshold=0.9)
```

LOTUS-fact-filter pipeline (<50 LoC)

Fact-Checking with LOTUS

sim_join()

map()

join()

```
wiki_df.load_sem_index("article", "index_dir")

for claim in claims_df["claim]:
    df = pd.DataFrame({"claim": [claim]})\
        .sem_map("what sub-claims are made in the {claim}", name=claimed_facts")
        .apply(lambda x: x[claimed_facts].split(","))
     claimed_facts_df = pd.DataFrame({"claims": df[claims]})
        .sem_sim_join(wiki_df, left_on="claim", right_on="articles",  K=20, n_rerank=10)\
        .sem_map("summarize the important facts in the {article}", name=facts)\
        .sem_join(claimed_fact_df, "is the {claimed_facts:right} verified by the {facts:left}")
```

LOTUS-fact-join pipeline (<50 LoC)

Fact-Checking with LOTUS

	Method	Accuracy
> 750 LoC	FacTool	83.5
< 50 LoC	LOTUS-Factool	90.0
	LOTUS-fact-filter	**90.5**
	LOTUS-fact-join	86.5

sem_search, sem_map, sem_filter, sem_join

Fact-Checking with LOTUS

	Method	Accuracy	Execution Time (s)
> 750 LoC	FacTool	83.5	1,174.8
< 50 LoC	LOTUS-Factool	90.0	111.48
	LOTUS-fact-filter	**90.5**	**64.28**
	LOTUS-fact-join	86.5	2394.2

sem_search, sem_map, sem_filter, sem_join

Fact-Checking with LOTUS

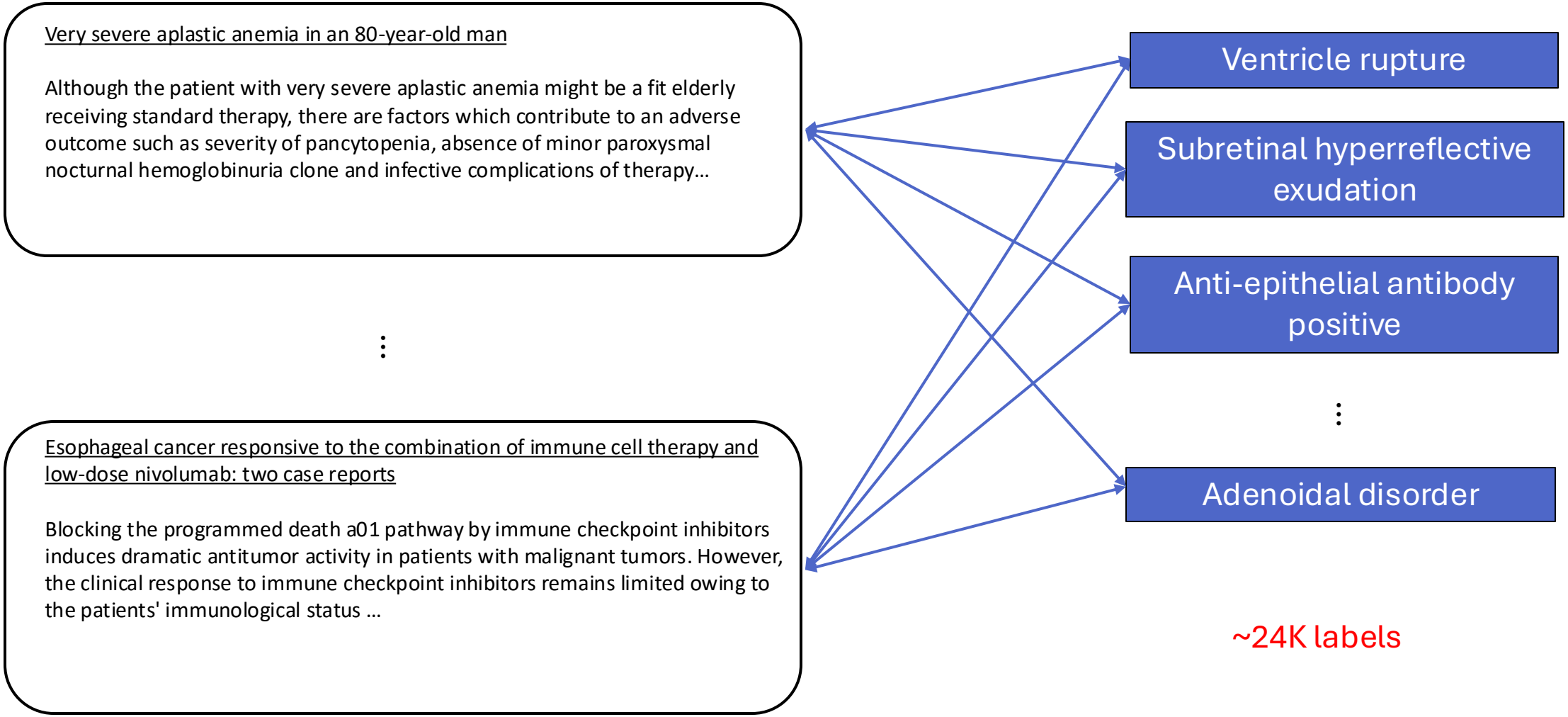
> 750 LoC

< 50 LoC

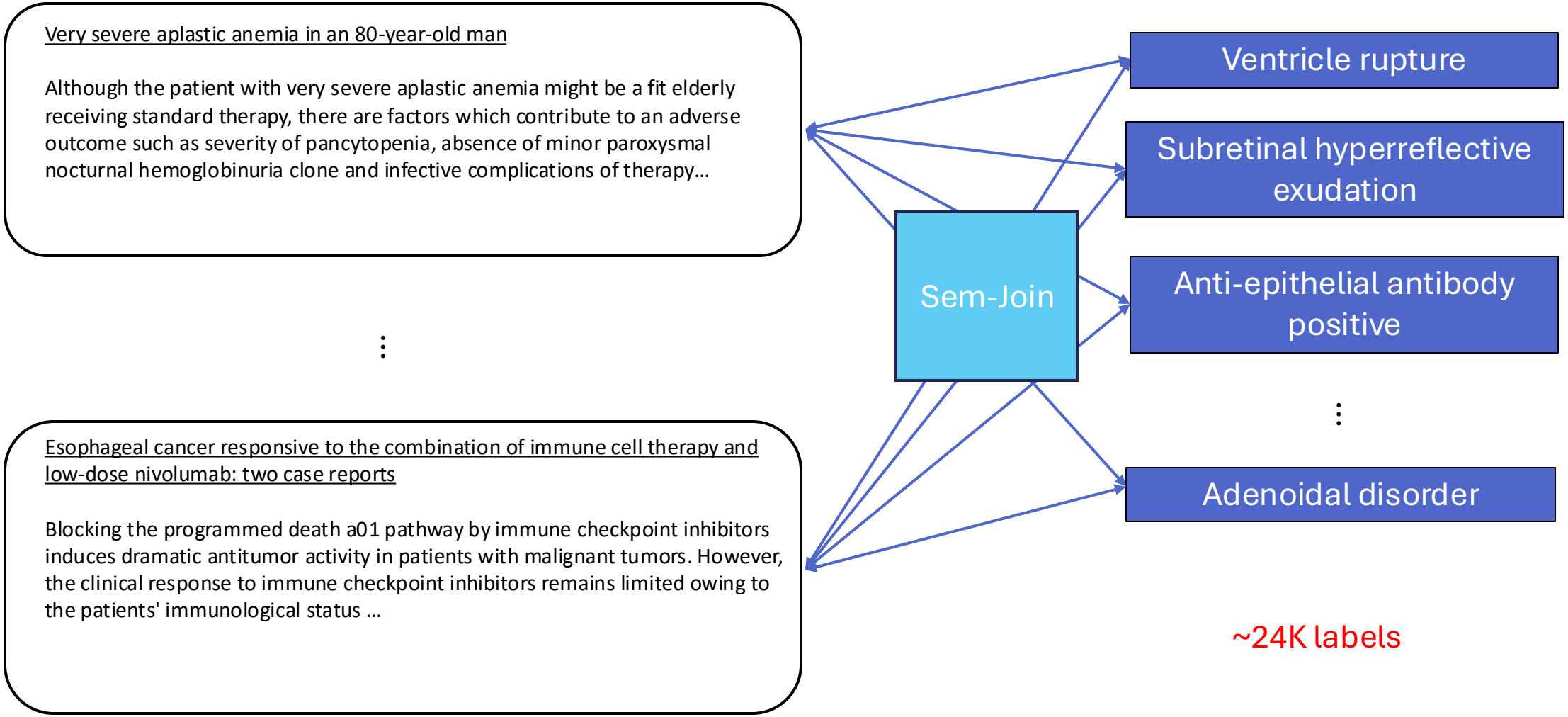
Method	Accuracy	Execution Time (s)
FacTool	83.5	1,174.8
LOTUS-Factool	90.0	111.48
LOTUS-fact-filter	90.5	64.28
LOTUS-fact-filter (+cascades)	**93**	**34.09**
LOTUS-fact-join	86.5	2394.2

sem_search, sem_map, sem_filter, sem_join

Use Case: extreme multi-label classification



Extreme Multi-Label Classification with LOTUS



Extreme Multi-Label Classification with LOTUS

Method	Rank-Precision@10	Execution Time (s)	Number LM Calls
Semantic Similarity Join	.120	2.91	0.00

Extreme Multi-Label Classification with LOTUS

Method	Rank-Precision@10	Execution Time (s)	Number LM Calls
Semantic Similarity Join	.120	2.91	0.00
LOTUS Semantic Join (nested-loop pattern)	N/A	2,144,560 *	6,092,500

* Estimated under linear-scaling assumption in number of batched calls

Extreme Multi-Label Classification with LOTUS

Method	Rank-Precision@10	Execution Time (s)	Number LM Calls
Semantic Similarity Join	.120	2.91	0.00
LOTUS Semantic Join (nested-loop pattern)	N/A	2,144,560 *	6,092,500
LOTUS Semantic Join (map-search-filter pattern)	**.258**	2,762	7,750
LOTUS Semantic Join (search-filter pattern)	.186	2,640	7,500

* Estimated under linear-scaling assumption in number of batched calls

Extreme Multi-Label Classification with LOTUS

Method	Rank-Precision@10	Execution Time (s)	Number LM Calls
Semantic Similarity Join	.120	2.91	0.00
LOTUS Semantic Join (nested-loop pattern)	N/A	2,144,560 *	6,092,500
LOTUS Semantic Join (map-search-filter pattern)	**.258**	2,762	7,750
LOTUS Semantic Join (search-filter pattern)	.186	2,640	7,500

* Estimated under linear-scaling assumption in number of batched calls

LOTUS transparently optimizes over this design space!

Summary

- LOTUS provides an ***optimized query engine*** for serving bulk-semantic processing over data

- ***Semantic operators*** provide a powerful declarative programming model that can efficiently capture wide-ranging LLM applications

Try it out!

https://github.com/TAG-Research/lotus /

```
pip install lotus-ai
```

Full paper: https://arxiv.org/abs/2407.11418

Please reach out!

@lianapatel_

lianapat@stanford.edu

Thank you!