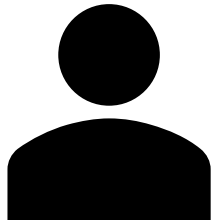


# LOTUS: Enabling Bulk Semantic Processing with LLMs Using Semantic Operators

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

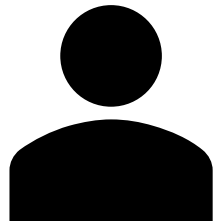
How should we ask questions over large amounts of data?

# 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?

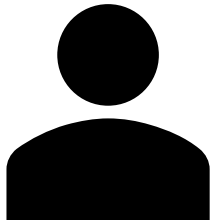
Which papers obtains the highest MMLU score?

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?

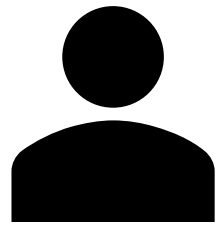


Which paper about vector databases has the funniest title?

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



# How should we ask questions over large amounts of data?



Which paper about vector databases has the funniest title?

RAG

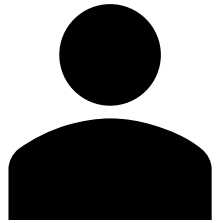
Good for point lookups,  
not bulk processing

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

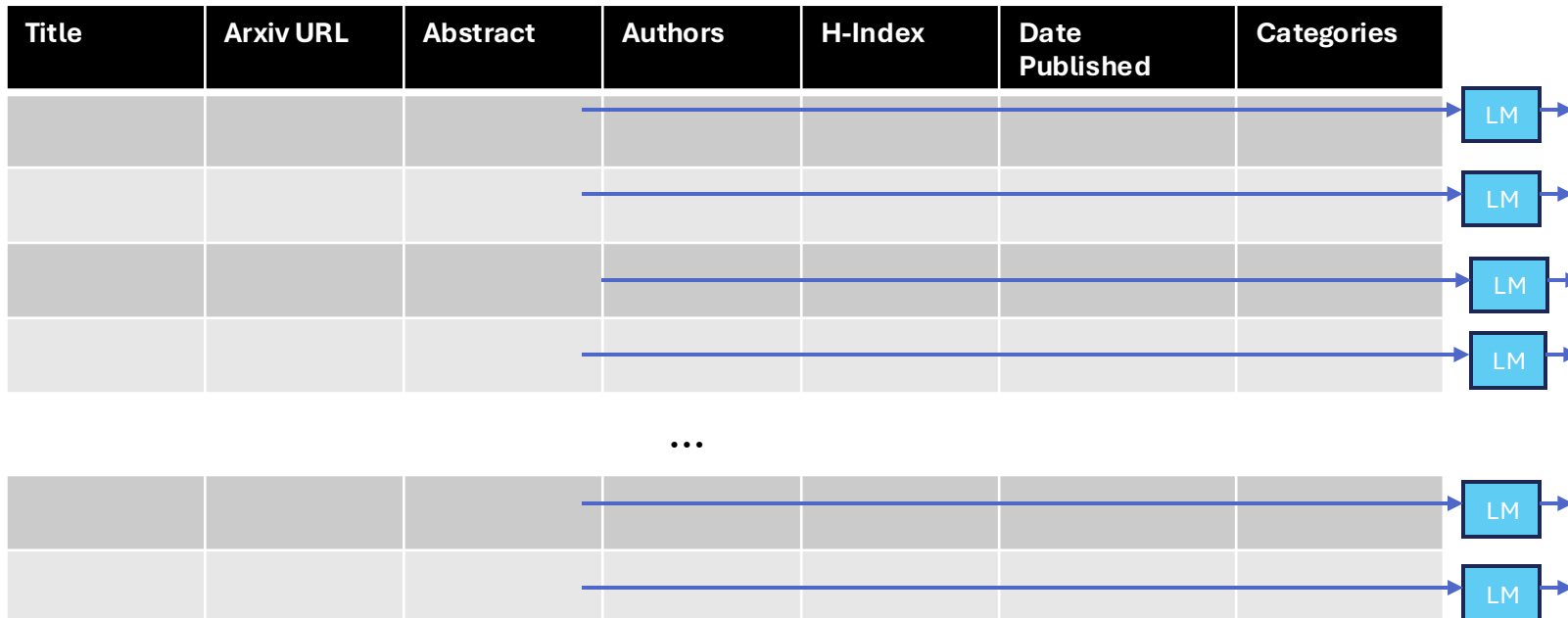


LLM

# How should we ask questions over large amounts of data?



Which paper about vector databases has the funniest title?



**RAG**

Good for point lookups,  
not bulk processing

**SQL with LLM  
UDF**

Limited to row-wise  
LLM execution







# What is LOTUS?

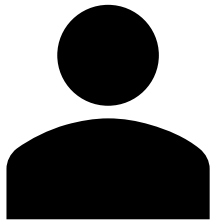
LLMs **O**ver **T**ables of **U**nstructured & **S**tructured data

A query engine for  
reasoning over large  
corpuses of data



# What is LOTUS?

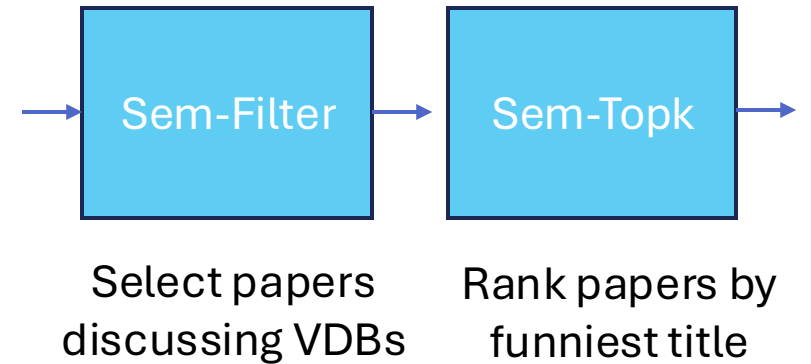
A query engine for reasoning over large corpuses of data



Which paper about vector databases has the funniest title?

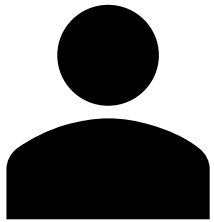
Title	Arxiv URL	Abstract	Authors	H-Index	Date Published	Categories
×	×	×	×	×	×	×
3						
×	×	×	×	×	×	×
1						
2						

Bulk semantic processing



# What is LOTUS?

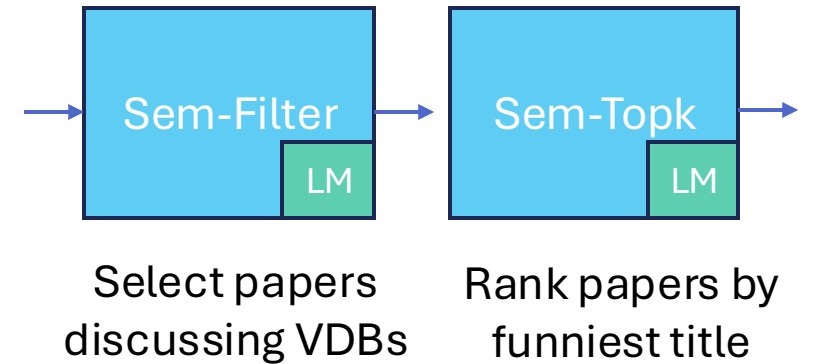
A query engine for reasoning over large corpuses of data



Which paper about vector databases has the funniest title?

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

Bulk semantic processing



# LOTUS: Two-fold Goals

A query engine for reasoning over large corpuses of data



Ease of Use

Efficiency

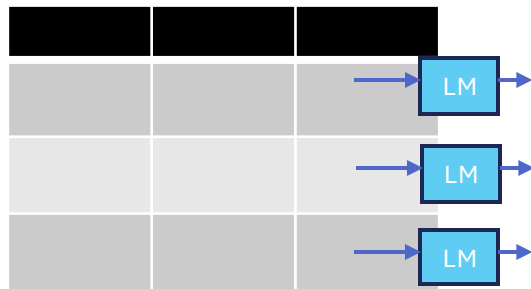
# LOTUS: Key Ideas

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

# Programming with LOTUS

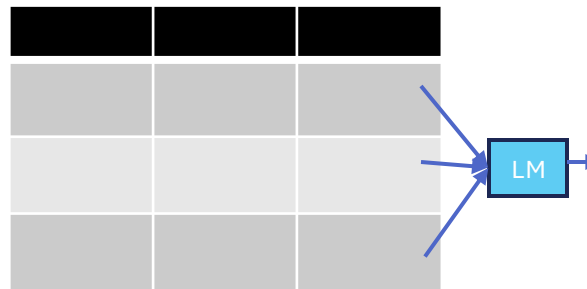
Key idea: ***semantic operators*** provide a declarative programming model that seamlessly extends the relational model

LLM Transformations



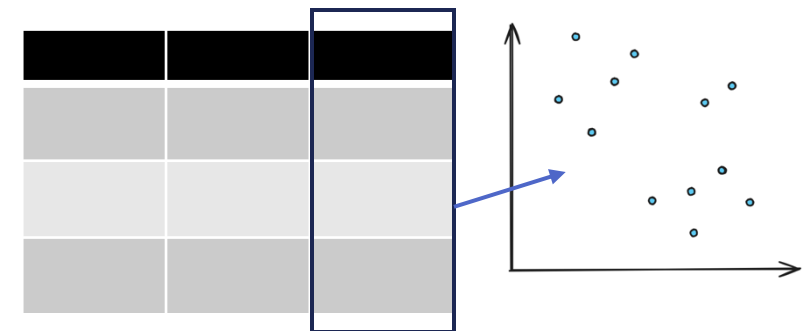
Eg) `sem_map`, `sem_filter`, `sem_join`

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*

sem\_filter()

sem\_map()

sem\_extract()

sem\_topk()

sem\_cluster\_  
by()

sem\_join()

sem\_search()

sem\_agg()

sem\_index()

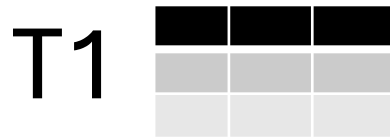
LOTUS Optimizer



Efficient batch processing, algorithmic approximations, clustering, etc.

# Under-the-hood: LOTUS Optimizer

Eg) Algorithmic Approximations for Sem-Join



Arxiv Papers



Wikipedia Articles

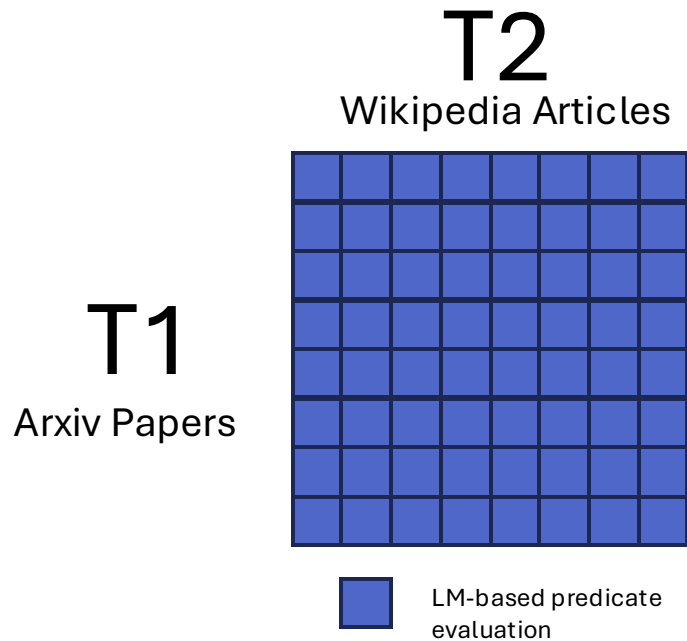


Does the {arxiv\_paper} contradict a claim made by the {wiki\_article}?

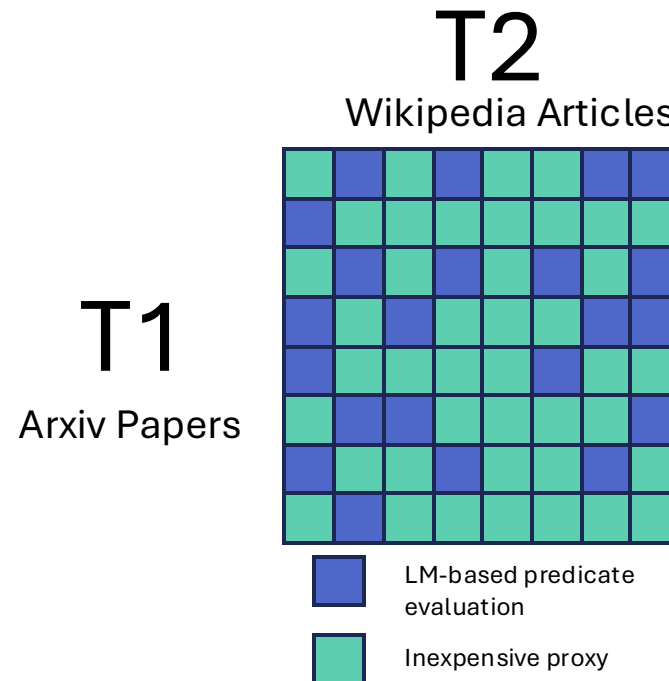
# Under-the-hood: LOTUS Optimizer

Eg) Algorithmic Approximations for Sem-Join

Exact Implementation



Learned approximation with probabilistic guarantees on accuracy!



# Demo: ArXiv Paper Search

- <https://b473ef4e57d606fb03.gradio.live>

Enter one or more filtering criteria.

<p>Research Interests</p> <input type="text" value="What are your research interests?"/>	<p>Relevance Criteria</p> <input type="text" value="the paper claims to outperform GPT3.5 on a task"/>	<p>Author H-index threshold</p> <input type="text" value="0"/> <input type="range"/>	<p>Date Filter: last X days</p> <input type="text" value="7"/> <input type="range"/>	<p>Arxiv Domains</p> <input checked="" type="checkbox"/> cs.AI <input type="checkbox"/> cs.DB <input type="checkbox"/> cs.IR <input type="checkbox"/> cs.PL <input type="checkbox"/> cs.ML
--	--	---	---	--

Enter your sorting criteria as a superlative. The resulting papers will be sorted by this criteria.

Superlative Sort

Send me a slack digest with my papers

New row

# Demo: ArXiv Paper Search

ArXiv Paper Finder [Browse the Full Arxiv Corpus](#)

Corpus of 1174 recent ArXiv papers.

authors	title
Lars-Peter Meyer, Johannes Frey, Felix Brei, Natanael Arndt	Assessing SPARQL capabilities of Large Language Models
Ningyu Zhang, Zekun Xi, Yujie Luo, Peng Wang, Bozhong Tian, Yunzhi Yao, Jintian Zhang, Shumin Deng, Mengshu Sun, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, HuaJun Chen	OneEdit: A Neural-Symbolic Collaboratively Knowledge Editing System
Christian Himpe	DatAasee -- A Metadata-Lake as Metadata Catalog for a Virtual Data-Lake
Tuba Gokhan, Kexin Wang, Iryna Gurevych, Ted Briscoe	RegNLP in Action: Facilitating Compliance Through Automated Information Retrieval and Answer Generation
Tianhe Lu, Jizhan Fang, Yunzhi Yao, Xin Xu, Ningyu Zhang, HuaJun Chen	Benchmarking Chinese Knowledge Rectification in Large Language Models
Arkadeep Acharya, Rudra Murthy, Vishwajeet Kumar, Jaydeep Sen	NLLB-E5: A Scalable Multilingual Retrieval Model
Linfeng Zhang, Changyue Hu, Zhiyu Quan	NLP-Powered Repository and Search Engine for Academic Papers: A Case Study on Cyber Risk Literature with CyLit
Achille Fokoue, Srideepika Jayaraman, Elham Khabiri, Jeffrey O. Kephart, Yingjie Li, Dhruv Shah, Youssef Drissi, Fenno F. Heath III, Anu Bhamidipaty, Fateh A. Tipu, Robert J. Baseman	A System and Benchmark for LLM-based Q&A on Heterogeneous Data
Yongsu Ahn, Quinn K Wolter, Jonilyn Dick, Janet Dick, Yu-Ru Lin	Interactive Counterfactual Exploration of Algorithmic Harms in Recommender Systems
Wenchao Zhao, Xiaoyi Liu, Ruilin Xu, Lingxi Xiao, Muqing Li	E-commerce Webpage Recommendation Scheme Base on Semantic Mining and Neural Networks
Luo Ji, Gao Liu, Mingyang Yin, Hongxia Yang, Jingren Zhou	Hierarchical Reinforcement Learning for Temporal Abstraction of Listwise Recommendation
Sümeyye Öztürk, Ahmed Burak Ercan, Resul Tugay, Şule Gündüz Öğüdücü	Enhancing Cross-Market Recommendation System with Graph Isomorphism Networks: A Novel Approach to Personalized User Experience
Weicong Qin, Yi Xu, Weijie Yu, Chenglei Shen, Xiao Zhang, Ming He, Jianping Fan, Jun Xu	Enhancing Sequential Recommendations through Multi-Perspective Reflections and Iteration
Shanu Vashishtha, Abhay Kumar, Lalitesh Morishetti, Kaushiki Nag, Kannan Achan	Leveraging User-Generated Reviews for Recommender Systems with Dynamic Headers

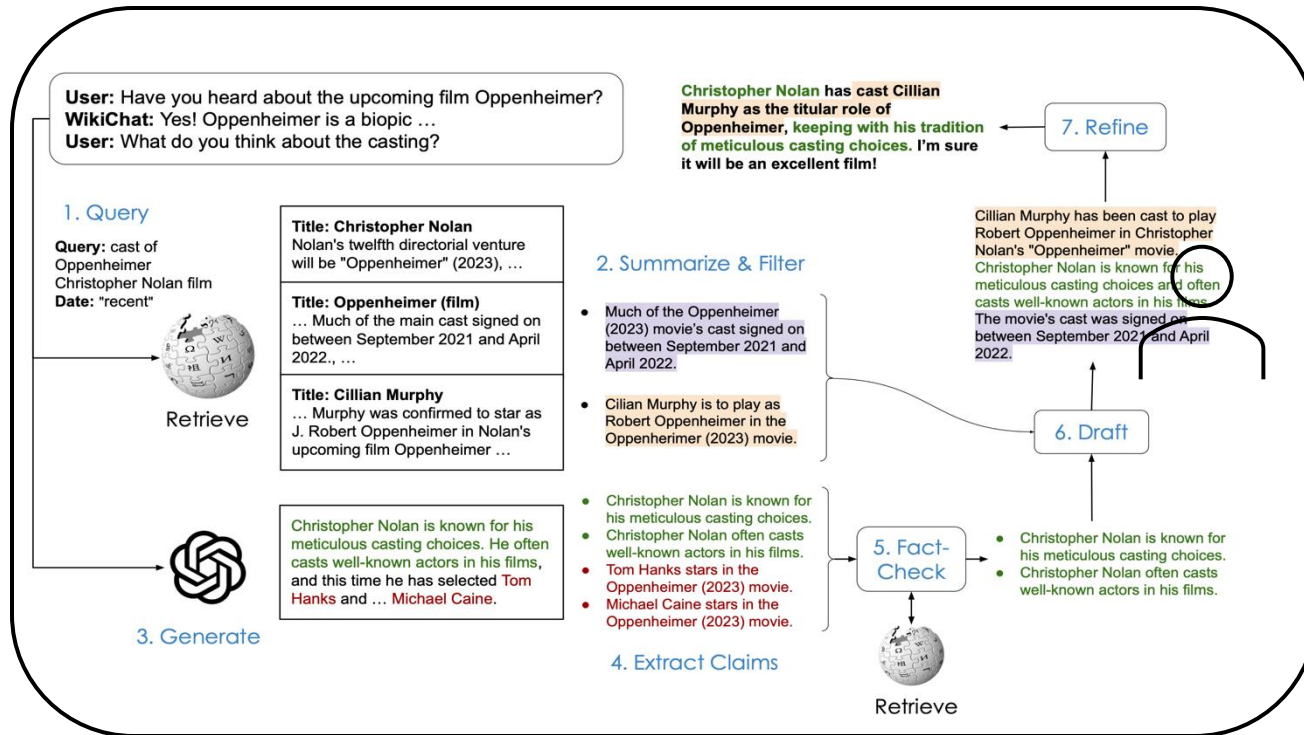
# Programming with LOTUS



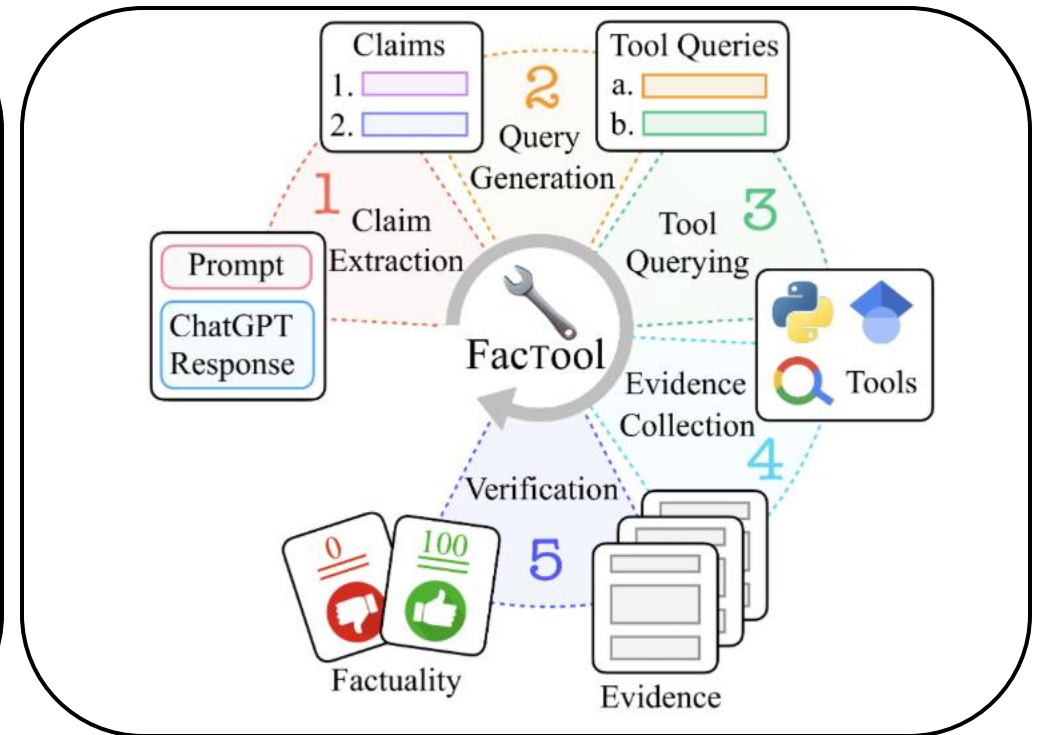
- Query logic is expressed in <20 lines of code!

```
1 # filter by hindex
2 papers_df = papers_df[papers_df["max_author_hindex"] > hindex_threshold]
3
4 # filter by domain on categories in papers_df
5 papers_df = papers_df[papers_df["categories"].apply(lambda x: any([d in x for d in domain_filter]))]
6
7 # filter by date
8 now = datetime.datetime.now(pytz.utc)
9 papers_df = papers_df[papers_df["date_published"] > now - pd.Timedelta(days=days_filter)]
10
11 # filter by topic and relevance criteria
12 papers_df.sem_search("abstract", research_topics, 100)\
13     .sem_filter(f"Based on the paper {{title}} and {{abstract}} of each paper, the paper is likely to be highly relevant to the use\
14     .sem_filter(f"Based on the paper {{title}} and {{abstract}}, the paper meets the following criteria: {relevance_criteria}")\
15     .sem_topk(f"Which {{abstract}} is{sort_query}", K=20)\
16     .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



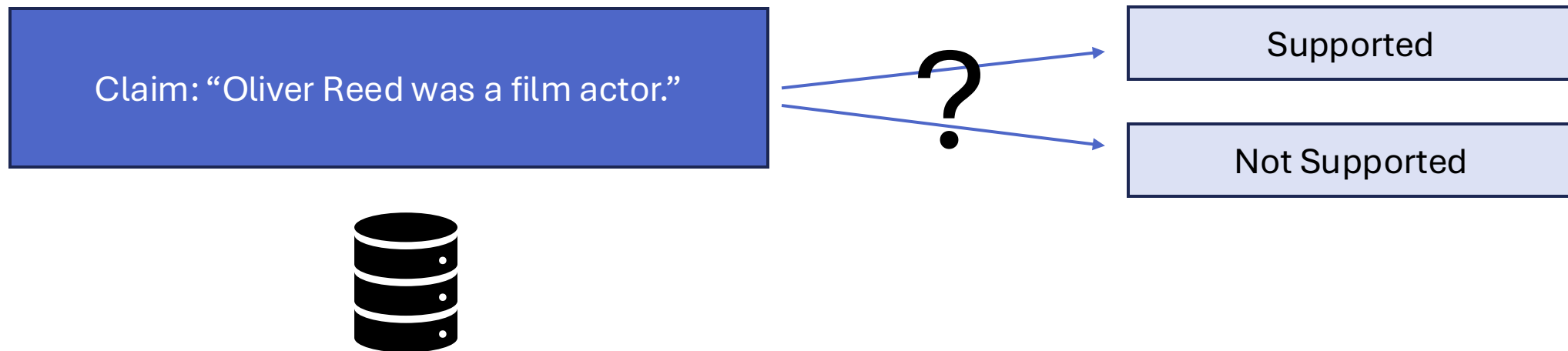
Wikichat (2023)



FacTool (2023)

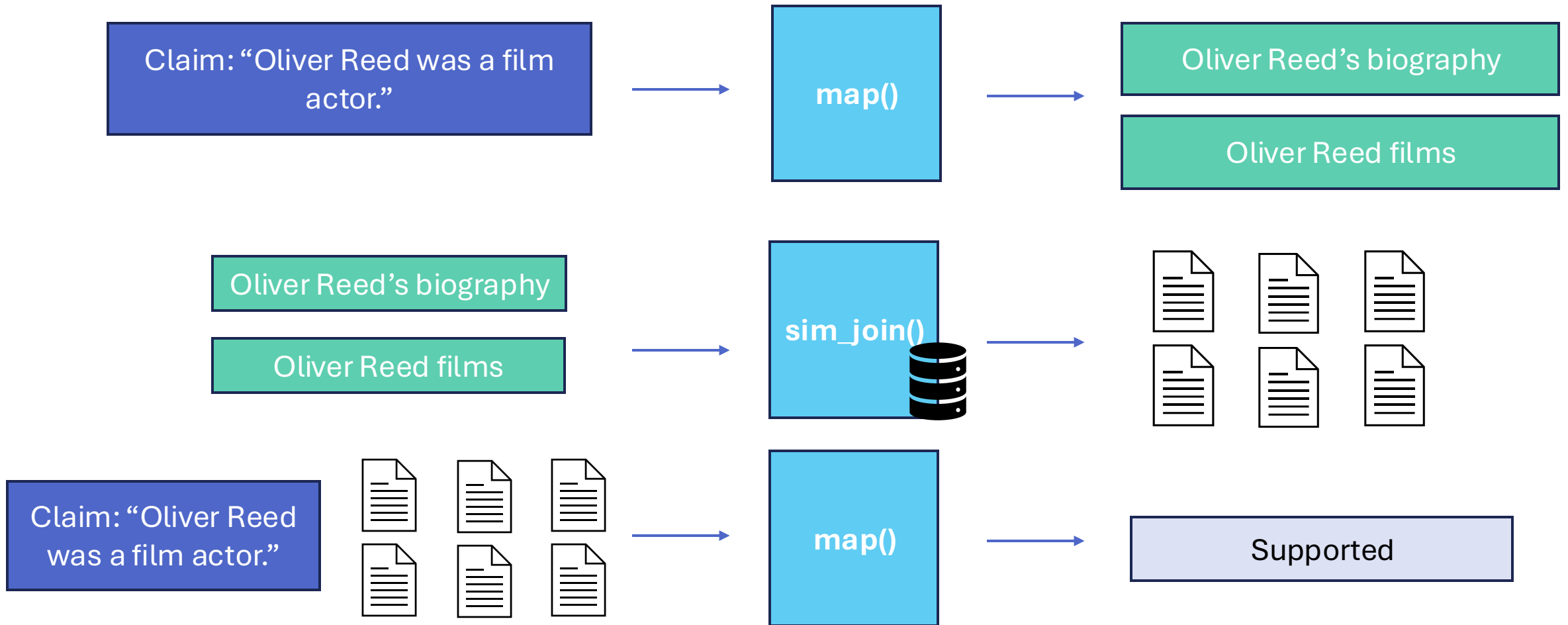
# Fact-Checking with FacTool

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 {
4   claim}", name="query")\
5   .sem_sim_join(wiki_df, left_on="query", right_on="
6     articles", K=10)\
7     # concatenate articles for each claim
8     .groupby(["claim"]).apply(lambda x: "\n".join(x["
9       articles"]))\
10    .sem_map("Identify whether there are any factual
11      errors in the {claim} based on the {articles}.
12      Include your reasoning, any errors found in the
13      claim, and the factuality of the claim.")
```

LOTUS-FacTool pipeline (<50 LoC)

# Fact-Checking with LOTUS

map()

sim\_join()

filter()

```
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_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()

```
1 wiki_df.load_sem_index("article", "index_dir")
2
3 for claim in claims_df["claim"]:
4     df = pd.DataFrame({"claim": [claim]})\
5         .sem_map("what sub-claims are made in the {claim
6                 }", name=claimed_facts")
7         .apply(lambda x: x[claimed_facts].split(","))
8     claimed_facts_df = pd.DataFrame({"claims": df[
9         claims]})
10    .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	<b>90.5</b>
	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	<b>90.5</b>	<b>64.28</b>
	LOTUS-fact-join	86.5	2394.2

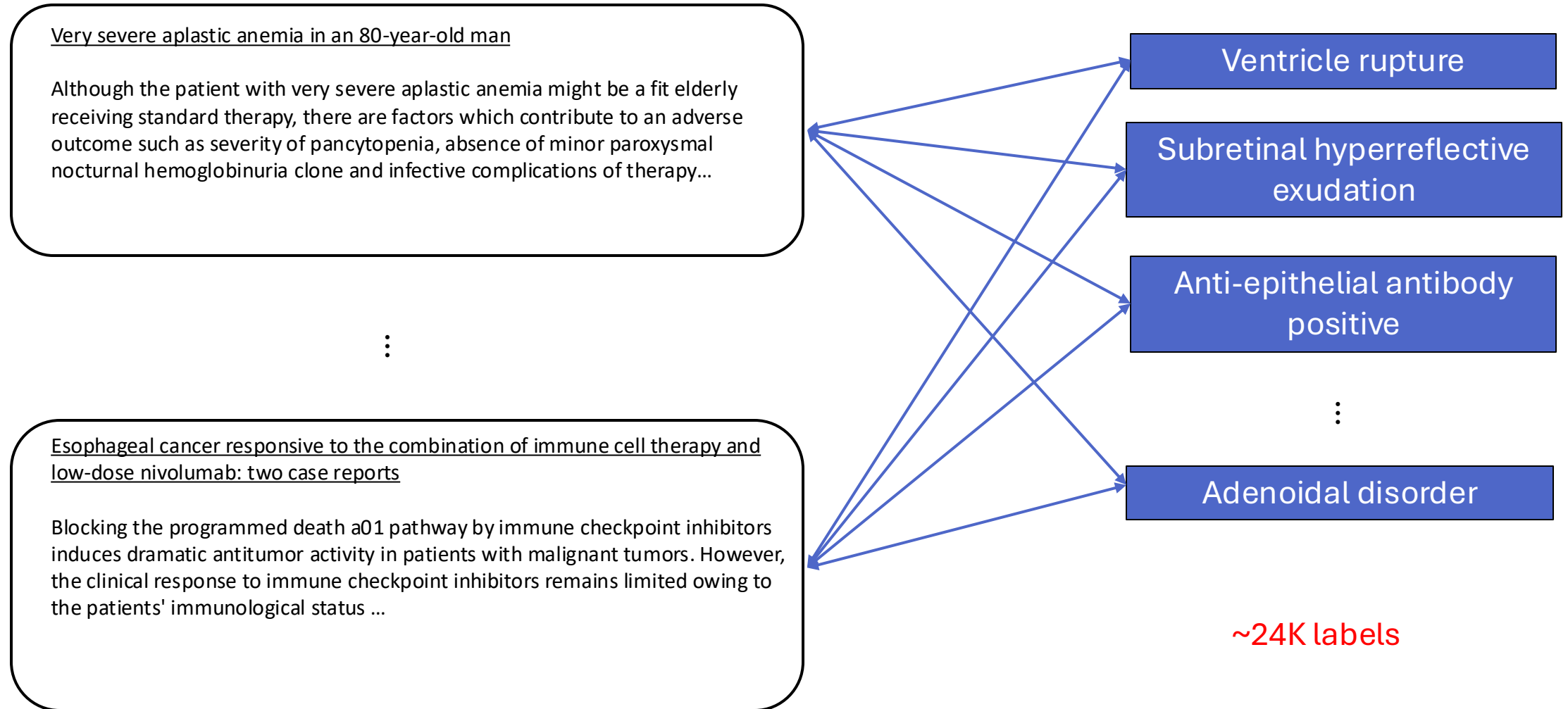
sem\_search, sem\_map, sem\_filter, sem\_join

# Fact-Checking with LOTUS

	Method	Accuracy	Execution Time (s)
> 750 LoC	FacTool	83.5	1,174.8
	LOTUS-Factool	90.0	111.48
< 50 LoC	LOTUS-fact-filter	90.5	64.28
	LOTUS-fact-filter (+cascades)	<b>93</b>	<b>34.09</b>
	LOTUS-fact-join	86.5	2394.2

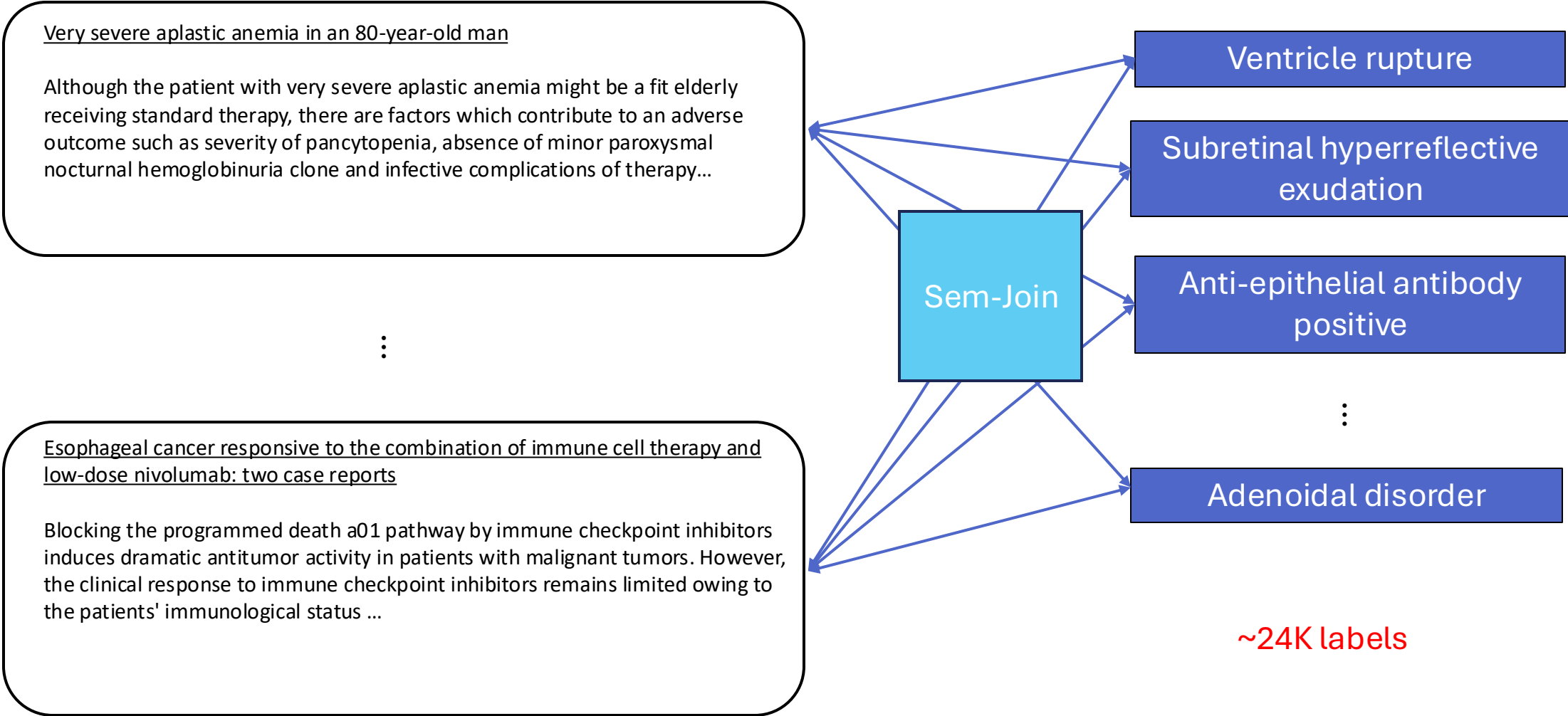
sem\_search, sem\_map, sem\_filter, sem\_join

# 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)	<b>.258</b>	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)	<b>.258</b>	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!**