Stanford CS224v Course

Conversational Virtual Assistants with Deep Learning

Lecture 10

Agentic AI for Knowledge Base Queries

Monica Lam & Shicheng Liu & Sina Semnani

Agentic AI

An AI system

that can make decisions and take actions on its own

to achieve a goal

Lecture Goal

Focus: Semantic Parsing for Complex Knowledge Bases

Agentic Approach

Computational Thinking

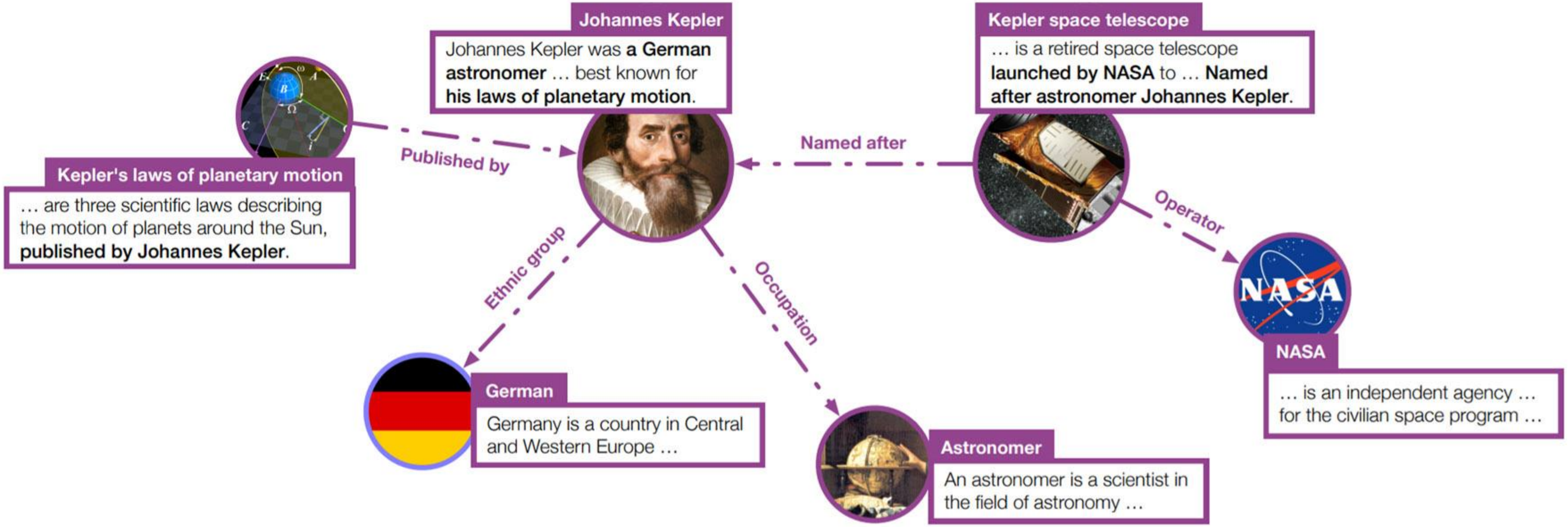
… so You can Apply them to your Project

Outline

- Knowledge Graphs – aka Knowledge Bases (KB)
  - Challenges of KBQA (KB Question Answering)
  - Prior Approaches
  - The Agentic Approach for KBs
  - Dataset and Evaluation
- Agentic Approach to SQL Databases
- Computational Thinking for KB Queries

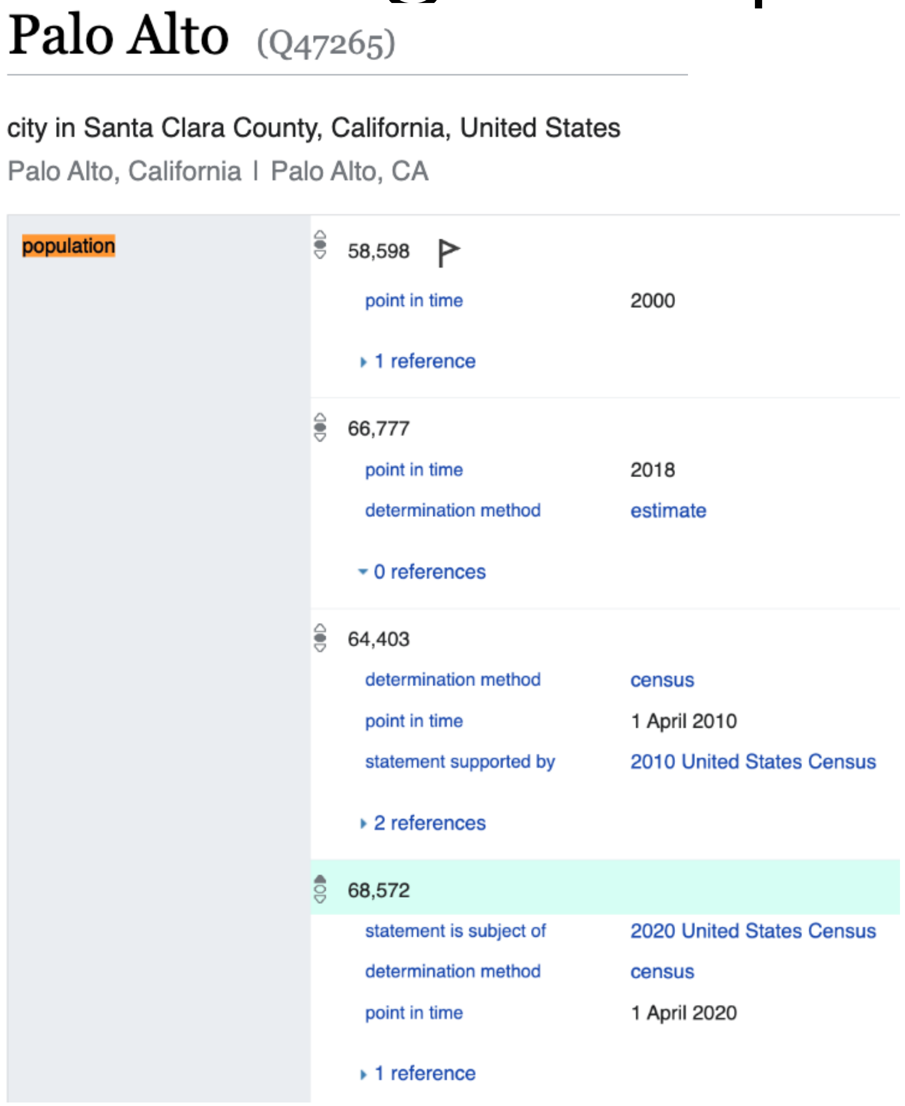
A Knowledge Graph

- A lot of information **cannot** be represented in tables
- Knowledge graph is also known as a semantic web
  - Nodes are entities
  - Edges are relationships



Wikidata: The Largest Live Knowledge Graph

- Stats:
  - 15B facts (1B more triples per year)
  - 100M entities
  - 10K properties (3000 of interest)
  - 25K contributors
- All entities in Wikipedia are in Wikidata
- Wikidata contains many more entities



Wikidata Representation – a RDF graph

- Representation: RDF triples
  - Nodes are entities (represented by unique QIDs)
    - There are many entities with exactly the same name
    - Same ID across all Wikipedia in all languages
  - Edges are properties (represented by unique PIDs)

Query with SPARQL

Who founded Stanford?

SELECT ?x WHERE

{ wd:Q41506 wdt:P112 ?x. }

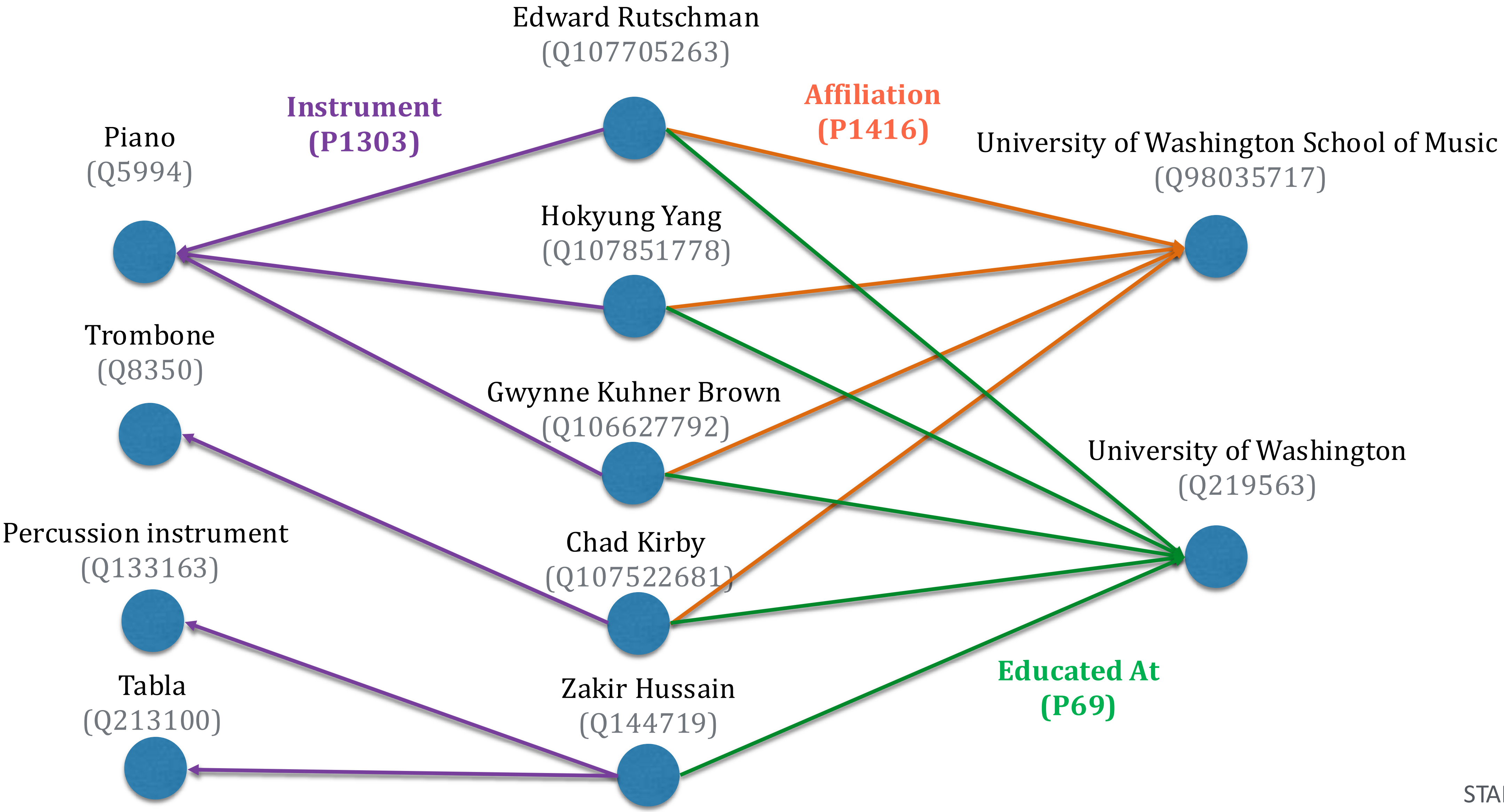
Stanford Founded by

A natural language interface can greatly expand access!c

Running Example: Music Instruments Played

Question: What are the musical instruments played by people who are affiliated with the University of Washington School of Music and have been educated at the University of Washington, and how many people play each instrument?

A Subset of the Knowledge Graph in Wikidata



```
>> execute_sparql("""
SELECT ?instrument ?instrumentLabel (COUNT(?student) AS ?count) WHERE {
  ?student wdt:P1303 ?instrument;
    wdt:P1416 wd:Q98035717;
    wdt:P69 wd:Q219563.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
GROUP BY ?instrument ?instrumentLabel
""")
Observation:
   | instrument    | instrumentLabel  | count  |
   |---------------|------------------|--------|
   | Q5994         | piano            | 99     |
   | Q1467960      | mbira            | 2      |
   | Q8350         | trombone         | 11     |
   | Q8338         | trumpet          | 8      |
   | Q17172850     | voice            | 32     |
   | ...           | ...              | ...    |
   | Q302497       | mandolin         | 1      |
   | Q187851       | recorder         | 1      |
   | Q185041       | cor anglais      | 1      |
   | Q83509        | piccolo          | 1      |

>> stop()
```

Writing the correct SPARQL query after correcting the mistake

P1303: instrument
P1416: affiliation
P69: educated at

Q98035717:
University of Washington School of Music

Q219563:
University of Washington

The Power of WikiData

- SPARQL allows relational algebra operations across the entire knowledge graph
  - Running example needs: filters, projections, joins, counts …
  - Query optimizations

- Dataset for research in **Mathematics**, **Biology**, **Education**, **Social Sciences**, **Linguistics**, ..

Quiz: Can we represent the data as tables?

Why Querying Wikidata with SPARQL is Difficult

- **No fixed schema:**
    - SPARQL and Neo4j are two most popular knowledge graph representations
    - Neo4j is typed – it has a fixed schema. All nodes of the same type have the same set of properties.
    - SPARQL is untyped – designed for extension across diverse fields (Adding properties need permission).
- **Many properties (3000):** Hard to memorize all the properties
- **Many similar properties:**
    - Often unclear which property or entity should be used
    - Questions on locations: “Where did Isaac Newton live?” “Where is Salesforce?”
    - Possible properties – depends on what is available for the node
        - *administrative territorial entity*
        - *residence*, *state*, *country, place of birth, place of death*
        - *headquarters location*, *location of formation*
- **Need to look at a node’s properties to determine the right SPARQL**
- **Queries can be complex**

Outline

- Knowledge Graphs – aka Knowledge Bases (KB)
  - Challenges of KBQA (KB Question Answering)
  - **Prior Approaches**
  - The Agentic Approach for KBs
  - Dataset and Evaluation

- Agentic Approach to SQL Databases

- Computational Thinking for KB Queries

Spider 2.0

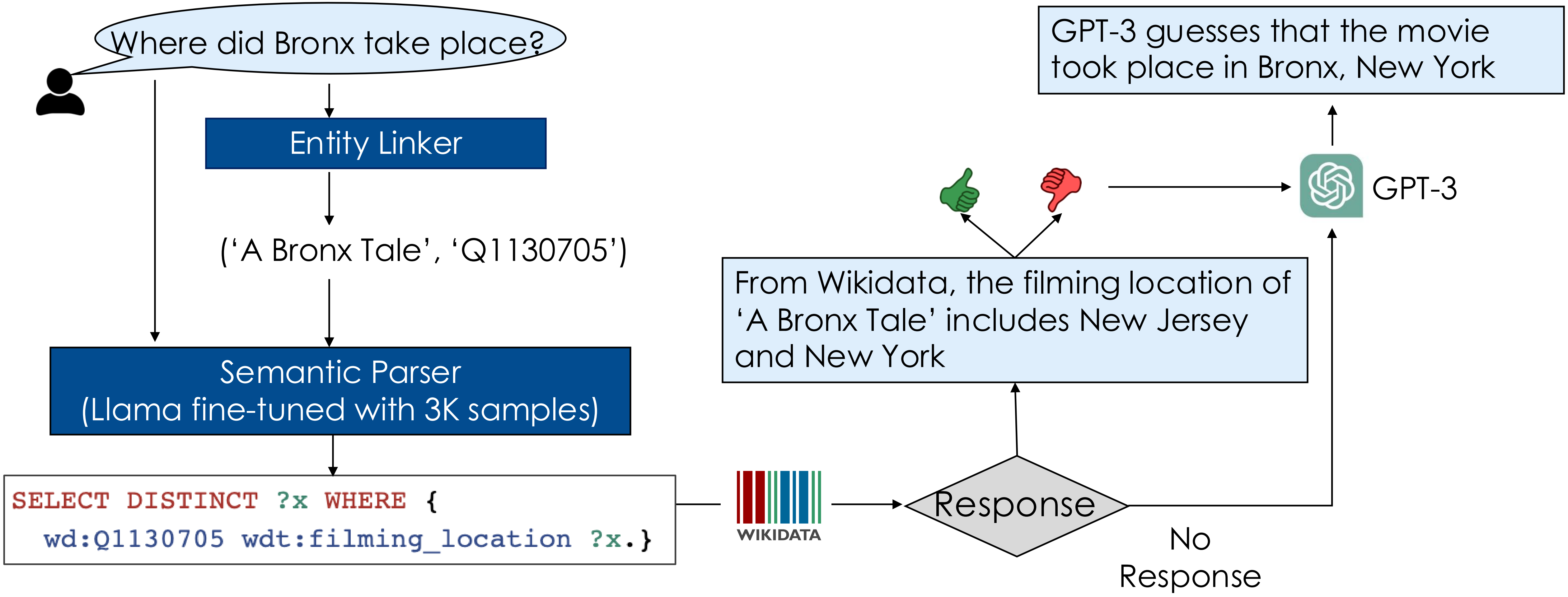
Previous Work

Knowledge Base Query Answering (KBQA)

Semantic Parser: Generating Sparql
(Fine-tuning, LLM)

Subgraph Traversal: Traversing the Knowledge Graph without Using SPARQL

1. Semantic Parsing with Fine-Tuning



Silei Xu et al. Fine-tuned LLMs Know More, Hallucinate Less with Few-Shot Sequence-to-Sequence Semantic Parsing over Wikidata.. EMNLP 2023

2. Semantic Parsing with Prompting LLMs

- Recall: Yelp semantic parse with LLM
  - LLMs know the syntax of SQL
  - Generates accurate SQL queries for simple tables directly
- LLMs know the syntax of SPARQL
  - And just a few PIDs and QIDs of entities
  - Cannot parse "research questions"
    (with many joins, group-bys, ranks, …)

3. Subgraph Retrieval (No SPARQL)

Retrieve a part of the graph based on a question

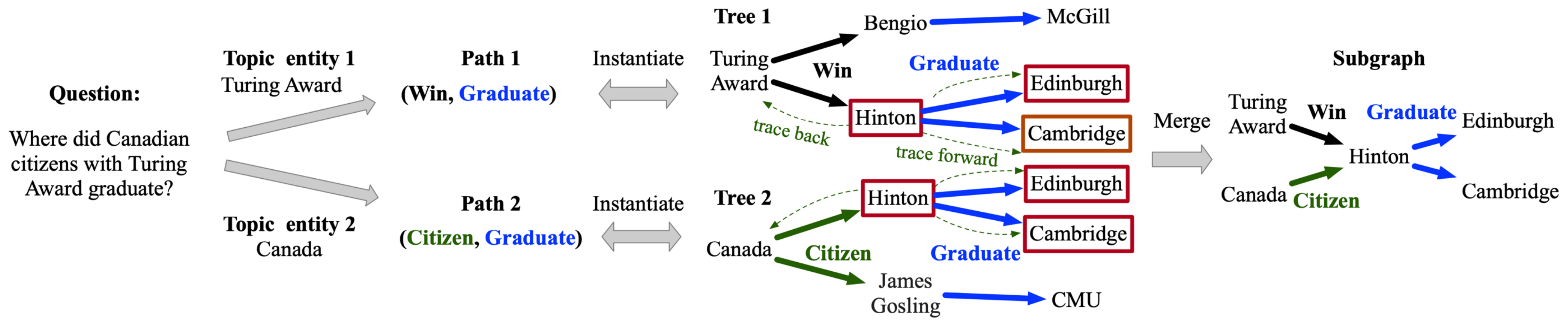


Figure 2: Illustration of the subgraph retrieving process. We expand a path from each topic entity as well as induce a corresponding tree, and then merge the trees from different topic entities to form a unified subgraph.

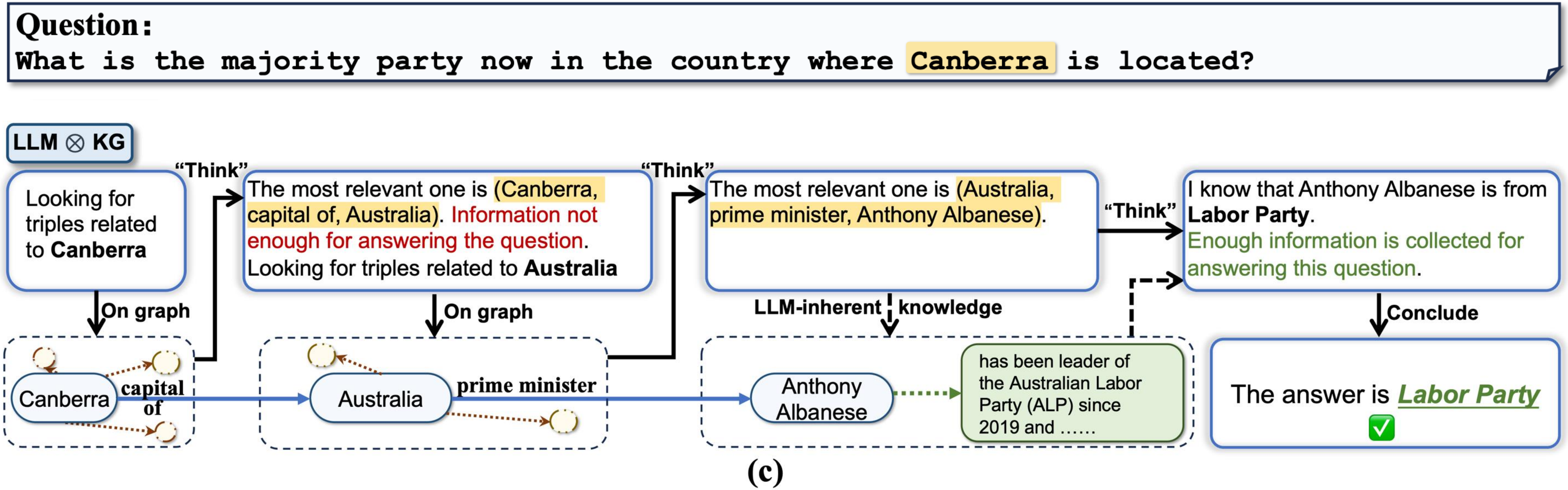
Quiz: Can this find the tallest mountain?

Quiz: Can this provide the full generality of SPARQL

[Subgraph Retrieval Enhanced Model for Multi-hop Knowledge Base Question Answering](#) (Zhang et al., ACL 2022)

4. LLM-Based Graph Exploration

Explore a sub-graph by walking the graph one edge at a time



Quiz: Can this provide the full generality of SPARQL

Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph, Sun et al, ICLR 2024

Outline

- Knowledge Graphs – aka Knowledge Bases (KB)
    - Challenges of KBQA (KB Question Answering)
    - Prior Approaches
    - **The Agentic Approach for KBs**
    - Dataset and Evaluation
- Agentic Approach to SQL Databases
- Computational Thinking for KB Queries

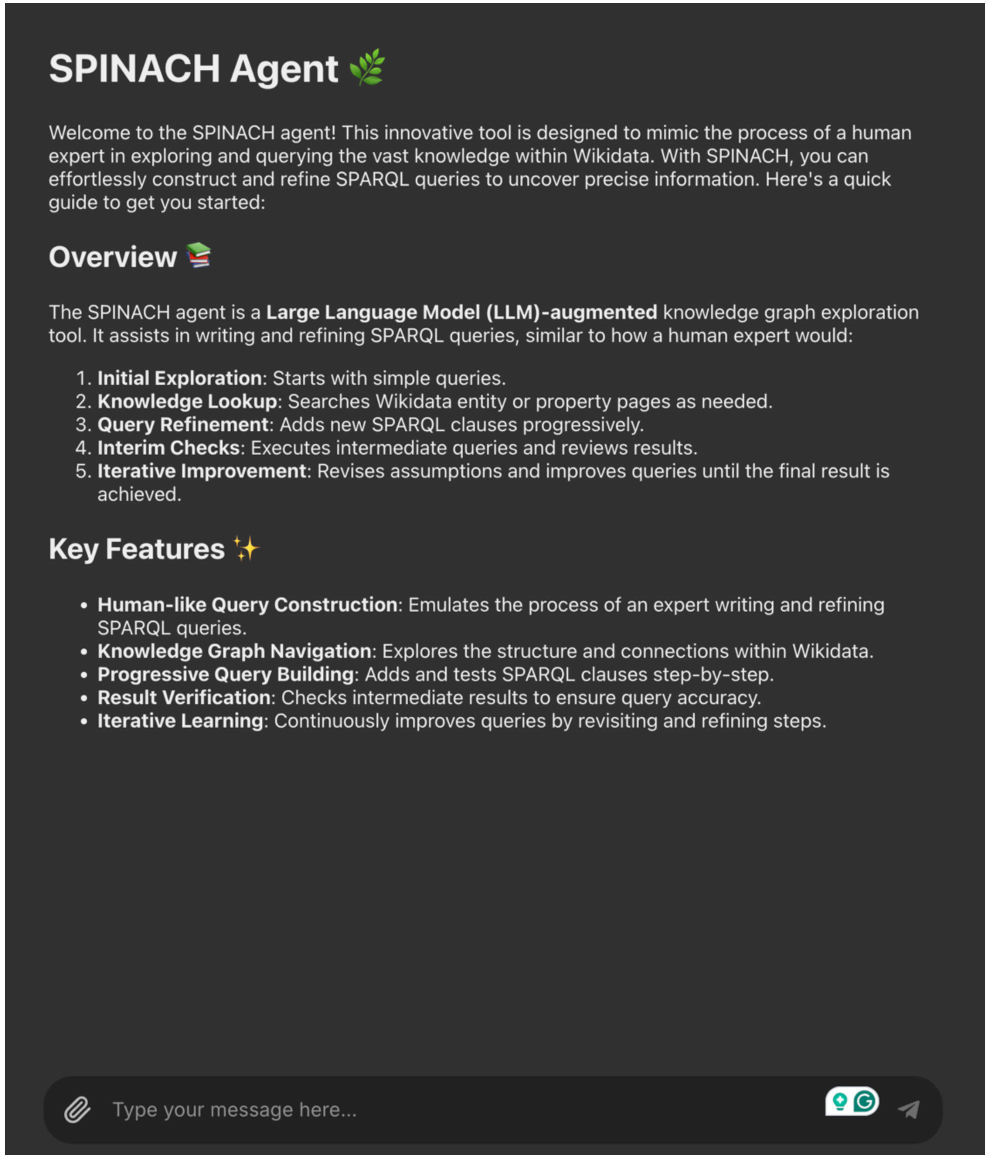
SPINACH: SPARQL-Based Information Navigation for Challenging Real-World Questions

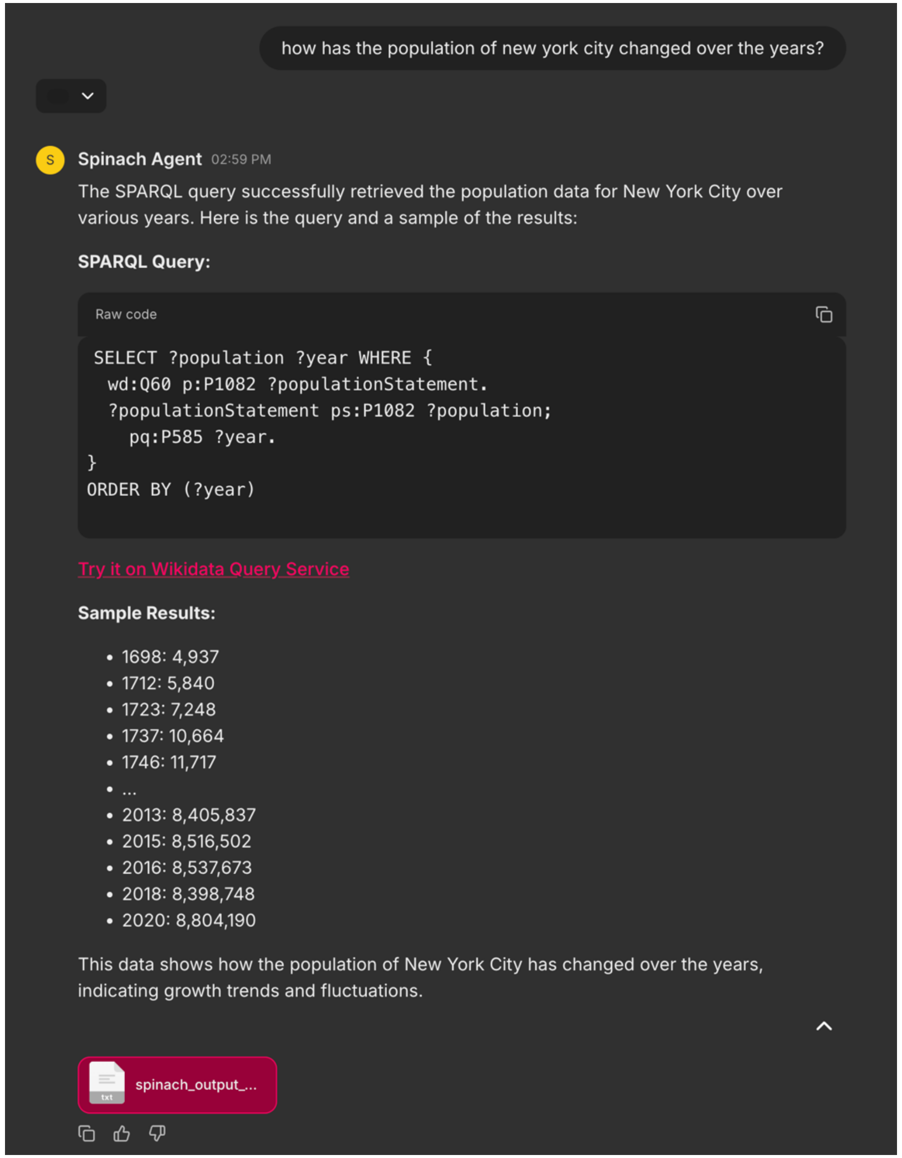
Shicheng Liu*   Sina J. Semnani*
Harold Triedman[1]   Jialiang Xu   Isaac Dan Zhao   Monica S. Lam
Stanford University

* Equal contribution
[1] Cornell Tech; Work conducted while at the Wikimedia Foundation

EMNLP 2024

Https://spinach.genie.stanford.edu





Combines the Best of Semantic Parsing (SPARQL), Subgraph Retrieval, & LLMs

- Semantic parsing to SPARQL:
  - Expressiveness, implementation/query optimizations
- Graph retrieval: exam actual properties in the data
- LLM agentic approach combines the 2 above
  - If human experts can do it: we can do it!
  - Automate the human expert's approach
  - Humans don't memorize the nodes and properties

    → we don't need fine-tuning! Just use In-Context Learning

How a Human Expert Write a SPARQL Query

1. Start by writing **simple SPARQL queries**;
2. **Look up** Wikidata **entity or property pages** when needed
   - To understand the structure of the knowledge graph
   - Check what properties exist for a node
3. **Add new SPARQL clauses** to build towards the final SPARQL

LLM Agent weaves together

- Knowledge inquiry
- Query writing
- Execution and evaluation of results (subgraph retrieval)

Actions Useful for Writing SPARQL

Provided by Wikidata

search_wikidata(string): http://wikidata.org (search bar):
text → returns QIDs and PIDs

get_wikidata_entry(QID): https://www.wikidata.org/wiki/QID:<QID>
QID → Wikidata page for entity

get_property_examples(PID): https://www.wikidata.org/wiki/Property:<PID>:
PID → examples of how property PID is used

execute_sparql(SPARQL): https://query.wikidata.org:
SPARQL → result

Running Example

Question: What are the musical instruments played by people who are affiliated with the University of Washington School of Music and have been educated at the University of Washington, and how many people play each instrument?

Action: **search_wikidata**

Question: What are the musical instruments played by people who are affiliated with the University of Washington School of Music and have been educated at the University of Washington, and how many people play each instrument?

```
search_wikidata("musical instrument")
```

```
- musical instrument (Q34379)
  device created or adapted to make musical sounds
- heraldic musical instrument (Q56877088)
  category of heraldic charges
- Musical instrument (Q102413357)
  Oil of canvas by Alla Grigoryan
- Musical Instrument (Q102413736)
  painting by Musical instrument
```

This step performs Named Entity Disambiguation (NED) with LLM!

Action: **search_wikidata**

Question: What are the musical instruments played by people who are affiliated with the University of Washington School of Music and have been educated at the University of Washington, and how many people play each instrument?

```
search_wikidata("affiliation")
```

affiliation (P1416)
organization that a person or organization is affiliated with (not necessarily member of or employed by)
27 statements, 0 sitelinks - 18:16, 28 October 2024

affiliation string (P6424)
qualifier to provide the published string form of affiliation attached to an author
14 statements, 0 sitelinks - 16:31, 1 October 2024

Action: **get_wikidata_entry**

Question: What are the musical instruments played by people who are affiliated with the University of Washington School of Music and have been educated at the University of Washington, and how many people play each instrument?

```
get_wikidata_entry
("Q98035717")
```

University of Washington School of Music (Q98035717)


```
{
  instance of (P31): {
    faculty (Q180958): {
      Qualifiers: [
        {
          of (P642): University of Washington (Q219563)
        }
      ]
    },
    music school (Q1021290): {}
  },
  field of work (P101): [
    higher education (Q136822),
    postgraduate education (Q141212),
    musicology (Q164204),
    music (Q638),
    research (Q42240),
    ethnomusicology (Q208365)
  ],
  located in the administrative territorial entity (P131): Seattle (Q5083),
  location (P276): Music Building (Q98690890),
  has subsidiary (P355): University of Washington Laboratory for Music
      Cognition, Culture & Learning (Q101157561),
  part of (P361): University of Washington College of Arts and Sciences
      (Q7896566),
  has part(s) (P527): University of Washington Laboratory for Music
      Cognition, Culture & Learning (Q101157561),
  parent organization (P749): University of Washington College of Arts and
      Sciences (Q7896566)
}
```

Action: `get_property_examples`

Question: What are the musical instruments played by people who are affiliated with the University of Washington School of Music and have been educated at the University of Washington, and how many people play each instrument?

```
get_property_examples("P1416")
```

```
Affiliation (P1416)

Raoul Bott -> Institute for Advanced Study
Hannah Sipe -> University of Washington School of Environmental and Forest Sciences
```

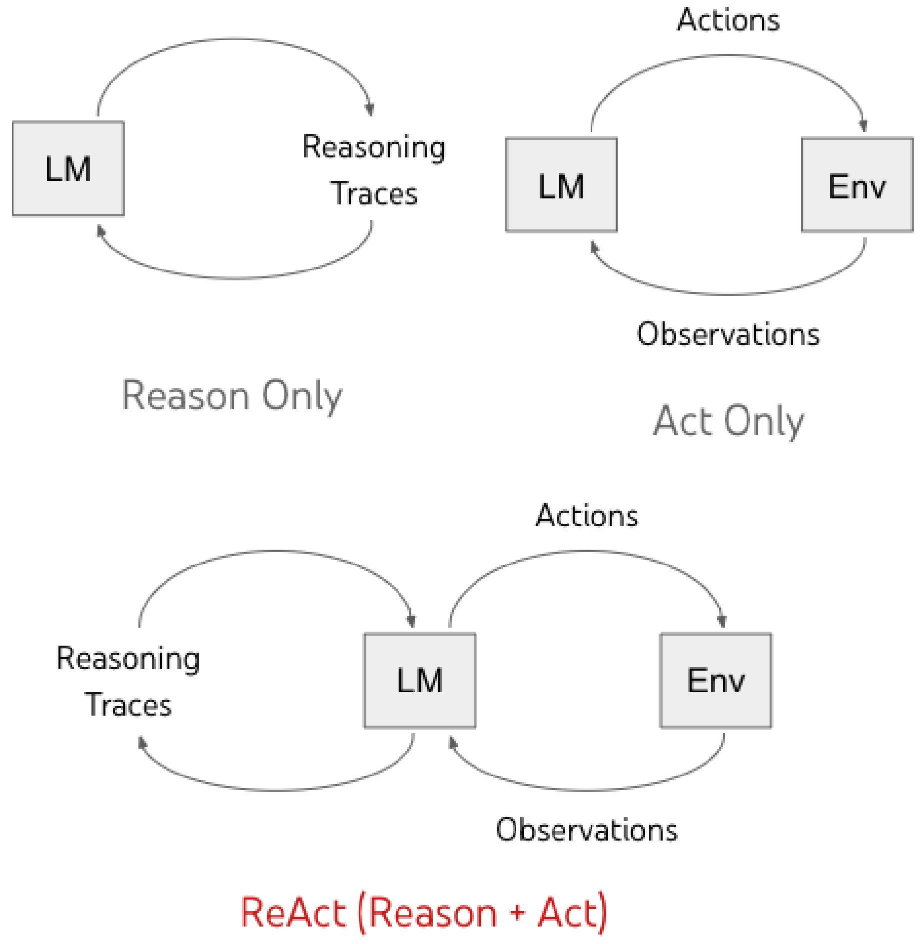
Action: **execute_sparql**

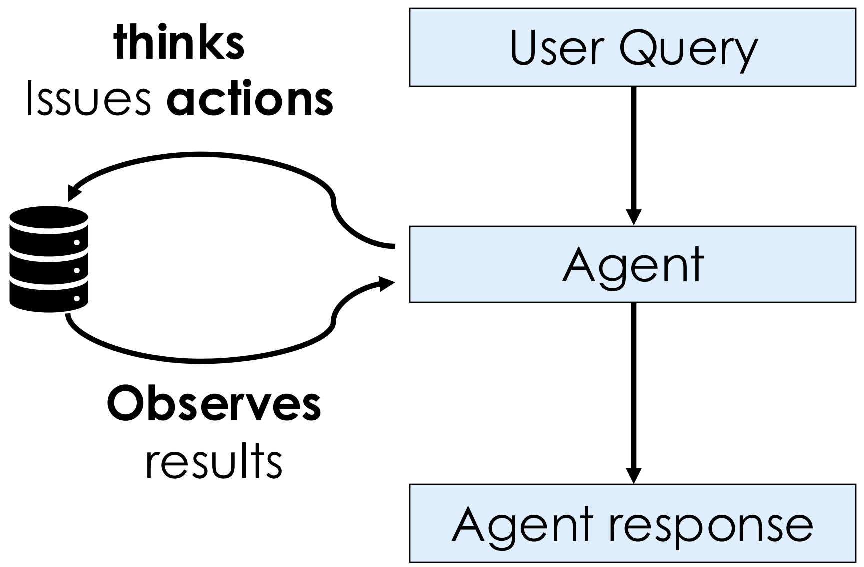
Question: What are the musical instruments played by people who are affiliated with the University of Washington School of Music and have been educated at the University of Washington, and how many people play each instrument?

```
execute_sparql("""
SELECT ?instrument ?instrumentLabel (COUNT(?student) AS ?count) WHERE {
  ?student wdt:P1303 ?instrument;
    wdt:P1416 wd:Q98035717;
    wdt:P69 wd:Q219563.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
GROUP BY ?instrument ?instrumentLabel
""")
```

instrument	instrumentLabel	count
Q5994	piano	99
Q1467960	mbira	2
Q8350	trombone	11
Q8338	trumpet	8
Q17172850	voice	32
...	...	...
Q302497	mandolin	1
Q187851	recorder	1
Q185041	cor anglais	1
Q83509	piccolo	1

Agentic Approach for Knowledge Bases





[ReAct: Synergizing Reasoning and Acting in Language Models](#), Yao et al, ICLR 2023

The SPINACH Agent

- Imitates what the user does with an agentic approach
- Uses the full expressiveness of SPARQL for exploration

- For N steps:
  - Given the history of agent actions,
    - Prompt LLM to generate a **thought** and an **action**
    - Execute an action against the KG
    - Add an **observation** to the history

Main agent code available in this file,
implemented with LangGraph (part of LangChain) in Python

Zero-Shot LLM Policy Prompt

instruction

Your task is to write a Wikidata SPARQL query to answer the given question. Follow a step-by-step process:

1. Start by constructing very simple fragments of the SPARQL query.
2. Execute each fragment to verify its correctness. Adjust as needed based on your observations.
3. Confirm all your assumptions about the structure of Wikidata before proceeding.
4. Gradually build the complete SPARQL query by adding one piece at a time.
5. Do NOT repeat the same action, as the results will be the same.
6. The question is guaranteed to have an answer in Wikidata, so continue until you find it.
7. If the user is asking a True/False question with only one answer, use ASK WHERE to fetch a True/False answer at the very end.
8. In the final SPARQL projections, do not only ask for labels.
   Ask for the actual entities whenever needed (e.g. instead of doing `SELECT xLabel`, do `SELECT x`).
9. If the final result was contained in last round's `get_wikidata_entry` and you are ready to stop,
   use `execute_sparql` and generate a SPARQL to retrieve that results.

Form exactly one "Thought" and perform exactly one "Action", then wait for the "Observation".

Possible actions are:

- get_wikidata_entry(QID): Retrieves all outgoing edges (linked entities, properties, and qualifrs) of a specified Wikidata entity using its QID.
- search_wikidata(string): Searches Wikidata for entities or properties matching the given string.
- get_property_examples(PID): Provides a few examples demonstrating the use of the specified property (PID) in Wikidata.
- execute_sparql(SPARQL): Runs a SPARQL query on Wikidata and returns a truncated result set for brevity.
- stop(): Marks the last executed SPARQL query as the final answer and ends the process.

```
# input
{% for i in range(0, conversation_history|length) %}
User Question: {{ conversation_history[i]["question"] }}
{% for j in range(0, conversation_history[i]["action_history"]|length) %}
{{ conversation_history[i]["action_history"][j] }}
{% endfor %}
--
{% endfor %}

User Question: {{ question }}

{% if action_history %}
{% for i in range(0, action_history|length) %}

{{ action_history[i] }}
{% endfor %}
{% endif %}

Output one "Thought" and one "Action":
```

Templates written in [jinja syntax]

Problem: Looping

- SPINACH needs to explore different queries during iteration
  - Run policy prompt with temperature 1.0
- LLM can fall into a loop of executing the same action

```
execute_sparql("SELECT ?x WHERE {wd:Q1 wdt:P1 ?x.}")
Did not find any results
execute_sparql("SELECT ?x WHERE {wd:Q1 wdt:P1 ?x.}")
Did not find any results
execute_sparql("SELECT ?x WHERE {wd:Q1 wdt:P1 ?x.}")
…
```

Quiz: Why would this happen?
Quiz: How to solve this?

Solution: Detecting & Stopping Loops

- If repeated actions are found,
  - Agent resets exploration state to the one before the repetition
  - Continues from there

- We cap number of actions by either:
  - 15 actions after taking “rollbacks” into account
  - a total of 30 actions including “rollbacks”

The SPINACH Agent in Action (1)

Question: What are the musical instruments played by people who are affiliated with the University of Washington School of Music and have been educated at the University of Washington, and how many people play each instrument?

```
Thought: First, I need to identify the properties and entities related to
people affiliated with the University of Washington School of Music ...

>> search_wikidata("University of Washington School of Music")
Observation: University of Washington School of Music (Q98035717)

[3 more time steps omitted]

>> search_wikidata("musical instrument")
```

Looking up relevant PIDs and QIDs

The SPINACH Agent in Action (2)

Question: What are the musical instruments played by people who are affiliated with the University of Washington School of Music and have been educated at the University of Washington, and how many people play each instrument?

```
>> execute_sparql("""
SELECT ?instrument (COUNT(*) AS ?count) WHERE {
  ?person wdt:P69 wd:Q1063349;
    wdt:P1416 wd:Q98035717;
    wdt:P1303 ?instrument.
}
GROUP BY ?instrument
""")
Obesrvation: Empty result.
```

Attempting to directly write the SPARQL fails due to the hallucinated Q1063349

The SPINACH Agent in Action (3)

```
Thought: Since my previous queries returned empty results, I need to
verify if there are any individuals in Wikidata who have been educated at
the University of Washington. This will help me confirm if I am using the
correct properties and entities.
>> get_wikidata_entry("Q1063349")
Observation:
  Spasskoye (Q1063349) {
    country (P17): Russia (Q159),
    ... more entries ...
  }

Thought: It seems like there is an issue with the QID I used for the
University of Washington. I should search for the correct QID.
>> search_wikidata("University of Washington")
Observation:
  - University of Washington (Q219563)
    public research university in Seattle, Washington, United States
```

Attempting to debug the issue by running simpler SPARQL queries

```
>> execute_sparql("""
SELECT ?instrument ?instrumentLabel (COUNT(?student) AS ?count) WHERE {
  ?student wdt:P1303 ?instrument;
    wdt:P1416 wd:Q98035717;
    wdt:P69 wd:Q219563.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
GROUP BY ?instrument ?instrumentLabel
""")
Observation:
   | instrument     | instrumentLabel  | count  |
   |----------------|------------------|--------|
   | Q5994          | piano            | 99     |
   | Q1467960       | mbira            | 2      |
   | Q8350          | trombone         | 11     |
   | Q8338          | trumpet          | 8      |
   | Q17172850      | voice            | 32     |
   | ...            | ...              | ...    |
   | Q302497        | mandolin         | 1      |
   | Q187851        | recorder         | 1      |
   | Q185041        | cor anglais      | 1      |
   | Q83509         | piccolo          | 1      |

>> stop()
```

Writing the correct SPARQL query after correcting the mistake

P1303: instrument
P1416: affiliation
P69: educated at

Q98035717:
University of Washington School of Music

Q219563:
University of Washington

Outline

- Knowledge Graphs – aka Knowledge Bases (KB)
  - Challenges of KBQA (KB Question Answering)
  - Prior Approaches
  - The Agentic Approach for KBs
  - **Dataset and Evaluation**
- Agentic Approach to SQL Databases
- Computational Thinking for KB Queries

KBQA Data Sets (Crowdsourcing)

- **Datasets with natural questions originally collected through search engines or crowdsourcing** ← Quiz: Is this OK?
  - WebQuestionSP (Yih et al., 2016)
  - QALD datasets (Usbeck et al., 2017, 2018, 2023; Perevalov et al., 2022)
  - RuBQ (Korablinov and Braslavski, 2020)
  - SimpleQuestions (Bordes et al., 2015)

Simple Queries

KBQA Data Sets (Synthesized)

- **Datasets with synthetically generated logical forms & questions**
  - ComplexWebQuestions (Talmor and Berant, 2018)
  - GrailQA (Gu et al., 2021)
  - KQA Pro (Cao et al., 2022a)
  - CFQ (Keysers et al., 2020)
  - CWQ (Talmor and Berant, 2018)
  - LC-QuAD2 (Dubey et al., 2019)

Limited NL variety
& Unique query patterns

Dataset	Size	UQPs	UQPs / Size
GrailQA (train+dev)	51100	116	0.227%
KQA-Pro	117970	1689	1.432%
CWQ	34689	402	1.159%
WikiWebQuestions	4316	176	0.041%
SPINACH	320	320	100.0%

Table 6: Comparison of Unique Query Patterns (UQPs) in SPINACH and prior works.

Wikidata SPARQL Forum

https://m.wikidata.org/wiki/Wikidata:Request_a_query

- To help Wikidata users write SPARQL queries
- People exchange conversations on how to write SPARQLs
- The queries are real, but difficult

Doctoral advisor

Hello, I would like to create a graph that links all the doctoral advisor (P184) and doctoral student (P185). It looks like a family tree but for doctoral advisor/student relations. I thought I could adapt from a family tree query but I am not able to find any. You can test on Leonhard Euler (Q7604). Thanks in advance. Pamputt (talk) 11:47, 9 April 2019 (UTC)

Initial Question

```
#defaultView:Graph

SELECT ?doctor ?doctorMaster ?doctorLabel ?doctorMasterLabel WHERE {
  ?root (wdt:P184*) ?doctor.
  ?doctor wdt:P184 ?doctorMaster.
  VALUES ?root {
    wd:Q7604
  }
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Try it!

Some examples : Wikidata:SPARQL query service/Wikidata Query Help/Result Views

@Pamputt: (I'm currently trying to format the tree from left to right from the root by default on the annotation, did not find the answer yet. You can play with the different layout buttons to change the display of the graph to something you like) author TomT0m / talk page 12:52, 9 April 2019 (UTC)

Response with SPARQL

A version including the doctor trained by Euler :

```
#defaultView:Graph#defaultView:Graph

SELECT ?doctor ?doctorMaster ?doctorLabel ?doctorMasterLabel ?docimage WHERE {
  { ?root (wdt:P184*) ?doctor. } union {?root
wdt:P185/wdt:P185?/wdt:P185?/wdt:P185? ?doctor .}
  ?doctor wdt:P184 ?doctorMaster.
  optional {?doctor wdt:P18 ?docimage }
  VALUES ?root {
    wd:Q7604
  }
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en".
}
}
```

Try it!

(depth caped at 4 trainees because the tree seems to explode overwise and the causes performance issues in the graph rendering, and for readability) author TomT0m / talk page 13:35, 9 April 2019 (UTC)

Refined SPARQL

Merci TomT0m et effectivement pour les grands noms scientifiques, ça peut partir en timeout. Pamputt (talk) 15:58, 9 April 2019 (UTC)

Acknowledgement

The SPINACH Dataset

From discussions on Wikidata Request Query forum (July 2016 – May 2024)

- **From the natural (real) SPARQL, we annotate its corresponding English question**
  - Disambiguate entities and properties
  - Natural verbalizations
  - Accurately capturing optional clauses and projections
- **We removed these categories of clauses:**
  - Wikimedia presentation queries
  - Questions on complex SPARQL code
  - Queries obscured by optimizations
  - Formatting clauses
- 155 validation examples and 165 test examples

← Quiz: These numbers seem small.
Are they enough?

SPINACH dataset:
natural questions + complex logical forms

	Avg. Clauses	Avg. Projs	Avg. Rels	Avg. Subjs	Avg. Preds	Avg. Objs	Avg. Lits
Natural questions w/ annotated logical forms							
WikiWebQuestion (Xu et al., 2023)	2.63	1	1.53	1.25	1.52	1.53	0.04
QALD-9 Plus (Perevalov et al., 2022)	3.14	1	1.77	1.26	1.70	1.78	0.05
QALD-10 (Usbeck et al., 2023)	2.38	1	1.27	1.19	1.17	1.32	0.05
RuBQ (Korablinov and Braslavski, 2020)	2.17	1	1.12	1.03	1.11	1.07	0.01
SimpleQuestionsWikidata (Diefenbach et al., 2017b)	2.00	1	1.00	1.00	1.00	1.00	0.00
Synthetic logical forms w/ synthetic or paraphrased questions							
CWQ (Talmor and Berant, 2018)	5.19	1	2.80	1.87	2.62	3.38	0.11
GrailQA (Gu et al., 2021)	7.10	1	3.02	1.97	2.43	3.90	0.08
KQA Pro (Cao et al., 2022a)	6.34	1	5.01	2.77	3.94	2.43	2.37
MCWQ (Cui et al., 2022)	6.34	1	5.09	2.67	3.53	3.37	0.00
LC-QuAD-2 (Dubey et al., 2019)	3.65	1	2.07	1.51	2.05	2.07	0.22
Natural logical form w/ annotated questions							
SPINACH **(Ours)**	8.89	2.50	4.03	1.76	3.55	4.53	0.46

Results on the SPINACH Dataset

See calculation of F1 for tables in the Appendix of this lecture

	Dev		Test	
	EM	F1	EM	F1
Direct GPT-4o Question Answering	0.0	3.9	0.0	4.0
GPT-4o Generating SPARQL	1.3	5.4	0.6	3.9
Fine-tuned model -> Fine-tuned WikiSP (Xu et al., 2023)	1.3	3.5	1.2	7.1
SOTA. Ask LLM to walk the graph -> 0-shot ToG (GPT-4) (Sun et al., 2024a)	3.9	9.8	1.8	7.2
0-shot SPINACH agent (GPT-4o) **(Ours)**	**21.4**	**46.4**	**16.4**	**45.3**

SPINACH agent achieves considerable gain over prior approaches!

Fine-tuned LLMs Know More, Hallucinate Less with Few-Shot Sequence-to-Sequence Semantic Parsing over Wikidata, Xu et al, EMNLP 2023
Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph, Sun et al, ICLR 2023

Results on Other Datasets

	QALD-7 (Task 4) Test		QALD-9 Plus (en) Test		QALD-10 (en) Full Test Set		QALD-10 (en) Subset in ToG		WikiWebQuestions Dev		WikiWebQuestions Test	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
STAGG (Yih et al., 2016)	-	19.0	-	-	-	-	-	-	-	-	-	-
GGNN (Sorokin and Gurevych, 2018)	-	21.3	-	-	-	-	-	-	-	-	-	-
LingTeQA (To and Reformat, 2020)	-	34.0	-	-	-	-	-	-	-	-	-	-
Baramiia et al. (2022)	-	-	-	-	-	42.8	-	-	-	-	-	-
Shivashankar et al. (2022)	-	-	-	-	-	49.1	-	-	-	-	-	-
QAnswer (Diefenbach et al., 2017a)	-	40.0	-	44.6	-	57.8	-	-	-	-	-	-
SPARQL-QA (Borroto et al., 2022)	-	-	-	-	-	59.5	-	-	-	-	-	-
Liu et al. (2024)	-	-	-	-	56.5	-	-	-	-	-	-	-
0-shot ToG (GPT-4) (Sun et al., 2024a)	-	-	-	-	-	-	54.7	-	-	-	-	-
Fine-tuned WikiSP (Xu et al., 2023)	38.0	43.6	-	-	-	-	-	-	**75.6**	**76.9**	**65.5**	**71.9**
0-shot SPINACH agent (GPT-4o) **(Ours)**	**62.2**	**74.6**	**58.3**	**71.6**	**63.1**	**69.5**	**64.7**	**72.4**	61.2	72.3	59.9	70.3

Zero-shot ICL (in-context learning) achieves new SOTA on QALD Wikidata datasets

Comes within 1.6 F1 on WikiWebQuestions to WikiSP, fine-tuned on WikiWebQuestions

The Importance of Agent Actions

	EM	F1
SPINACH agent	**21.4**	**46.4**
w/o `get_wikidata_entry`	11.7	36.4
w/o `get_property_examples`	10.4	29.4
w/o `search_wikidata`	4.6	25.3

All actions make meaningful contribution to the agent performance

Live Demo:

https://spinach.genie.stanford.edu/

Code:

https://github.com/stanford-oval/spinach

As a bot on Wikidata:

https://www.wikidata.org/wiki/User:SpinachBot

Error Analysis

40%: **Property-related problems**:
Fails to fetch the correct property or incorrectly uses a property

30%: **Complicated SPARQL**:
Fails to write complex SPARQL to fetch results.

15%: **Not enough exploration:**
Insufficient exploration within limit of actions allowed.

10%: **Inaccurate semantic parsing**:
LLM injecting an extra clause.

5%: **Formatting issues**

SPINACH Deployed on Wikidata Forum

https://m.wikidata.org/wiki/Wikidata:Request_a_query

- 600+ conversations in the wild: all real and hard queries!
- 198 randomly selected conversations
  - Success rate: 78% (154 cases) A higher success rate in practice!
  - Failures: 22% (44 cases)
    - 50% (22 cases) similar to queries in the dataset
    - 50% (22 cases) are not similar:
      underspecified queries, query correction/modification, string manipulation

Outline

- Knowledge Graphs – aka Knowledge Bases (KB)
  - Challenges of KBQA (KB Question Answering)
  - Prior Approaches
  - The Agentic Approach for KBs
  - Dataset and Evaluation
- **Agentic Approach to SQL Databases**
- Computational Thinking for KB Queries

The New York Times

Inside the Late-Night Parties Where Hawaii Politicians Raked In Money

After the state passed a law barring government contractors from donating to politicians, fund-raising parties showed just how completely the reform effort failed.

By Blaze Lovell, Eric Sagara and Irene Casado Sanchez

The reporters examined campaign contributions and government contracts for this article, part of a series about loopholes in Hawaii's pay-to-play laws, for The Times's Local Investigations Fellowship.

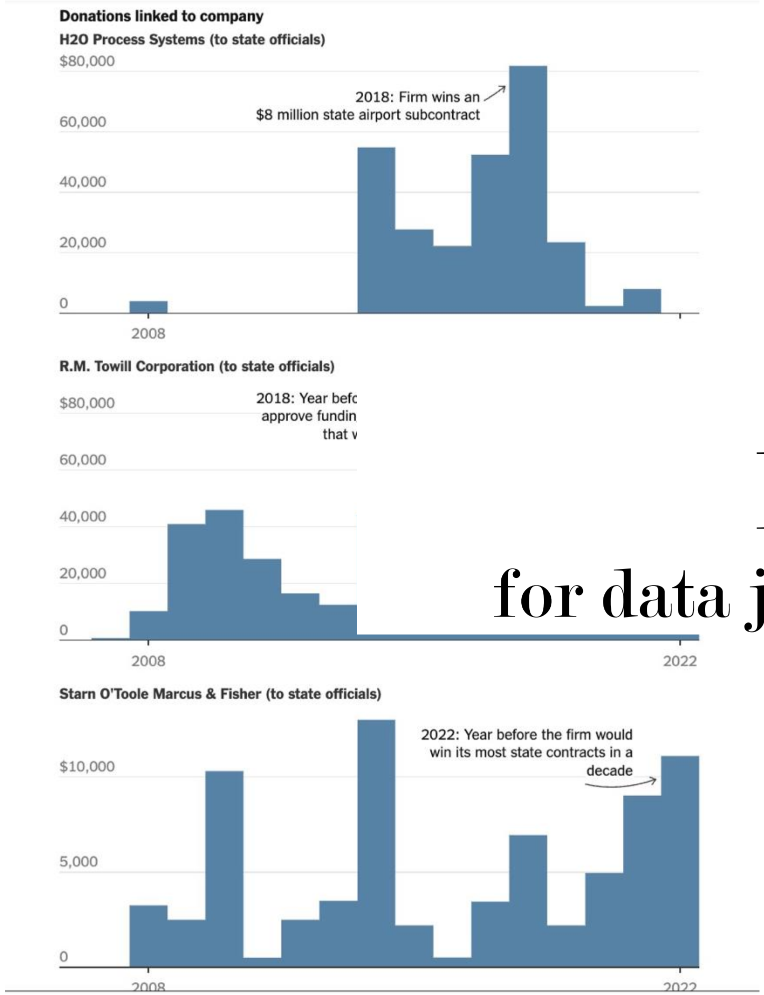
April 17, 2024

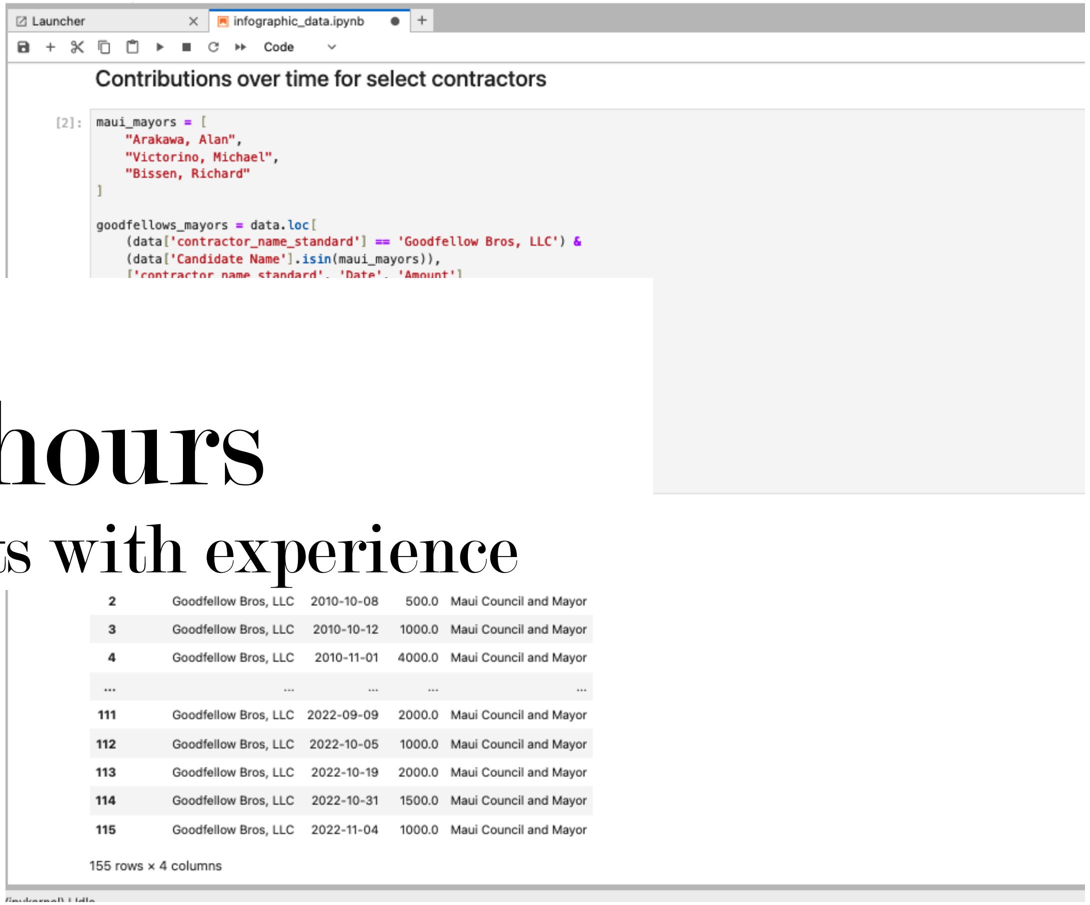
This article was reported in partnership with Big Local News at Stanford University.

https://www.nytimes.com/2024/04/17/us/haw
contractors-campaign-donations.html

The New York Times

Inside the Late-Night Parties Where Hawaii Politicians Raked In Money





100+ hours

for data journalists with experience

https://www.nytimes.com/2024/04/17/us/hawaii-contractors-campaign-donations.html

Increase Truthful Reporting in Journalism

Can Journalists Just *TALK* to Data

Without Needing a Data Scientist?

DataTalk
A Campaign Finance Agent (Beta)

Shicheng Liu[1] Harold Triedman[4] Leah Harrison[3] Eryn Davis[3]
Sajid Omar Farook[1] Alexander Spangher[5] Cheryl Phillips[2] Derek Willis[6]
Serdar Tumgoren[2] Monica S. Lam[1]

[1] Stanford CS [2] Stanford Big Local News [3] Columbia Journalism School
[4] Cornell Tech [5] USC [6] University of Maryland

Get started

✨ Supported by a Magic Grant from the Brown Institute, the Verdant Foundation, and Microsoft Azure Credits

DataTalk

A Campaign Finance Agent (Beta)

Chat with publicly available campaign finance data from the Federal Election Commission (FEC) and PAC classification data from OpenSecrets.org

Get Started →

DataTalk

How much have Kamala Harris and Donald Trump raised and spent so far?

Intermediate thoughts

CAND_ID	CAND_NAME	TTL_RECEIPTS	TTL_DISB
P00009423	HARRIS, KAMALA	678,223,639.47	444,597,617.80
P80001571	TRUMP, DONALD J.	309,218,794.51	177,664,005.95

DataTalk: Campaign Finance

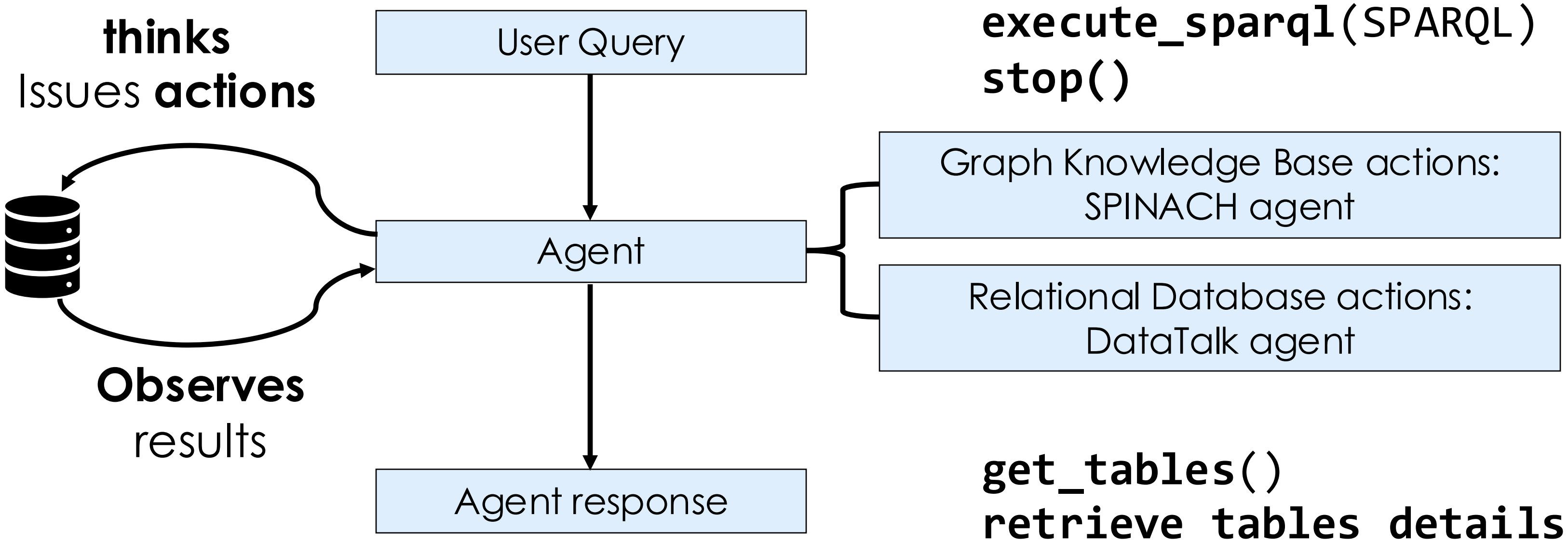
- Chat with publicly available campaign finance data
- Based on
  - Federal Election Commission (FEC)
  - OpenElections.org data
- 36 Large, relational databases

Data Tables

- all_candidates_2019_2020
- all_candidates_2021_2022
- all_candidates_2023_2024
- any_transaction_from_one_committee_to_another_2019_2020
- any_transaction_from_one_committee_to_another_2021_2022
- any_transaction_from_one_committee_to_another_2023_2024
- candidate_committee_linkage_2019_2020
- candidate_committee_linkage_2021_2022
- candidate_committee_linkage_2023_2024
- candidate_master_2019_2020
- candidate_master_2021_2022
- candidate_master_2023_2024
- committee_master_2019_2020
- committee_master_2021_2022
- committee_master_2023_2024
- committee_type_codes
- contributions_by_individuals_2019_2020
- contributions_by_individuals_2021_2022
- contributions_by_individuals_2023_2024
- contributions_from_comm_to_cand_and_ind_expenditures_2019_2020
- contributions_from_comm_to_cand_and_ind_expenditures_2021_2022
- contributions_from_comm_to_cand_and_ind_expenditures_2023_2024
- current_campaigns_for_house_and_senate_2019_2020
- current_campaigns_for_house_and_senate_2021_2022
- current_campaigns_for_house_and_senate_2023_2024
- disbursement_category_codes
- operating_expenditures_2019_2020
- operating_expenditures_2021_2022
- operating_expenditures_2023_2024
- pac_and_party_summary_2019_2020
- pac_and_party_summary_2021_2022
- pac_and_party_summary_2023_2024
- pac_details
- party_codes
- report_type_codes
- transaction_type_codes

Agentic Approach for Knowledge Bases

```
search_wikidata(string)
get_wikidata_entry(QID)
get_property_examples(PID)
execute_sparql(SPARQL)
stop()
```




```
get_tables()
retrieve_tables_details([table1, table2, …])
execute_SQL(SQLquery)
entity_linking(search_str, [column1, column2, …])
location_linking(search_str)
stop()
```

Real-World Experience with Journalists

- **Used by journalists and journalism students**
- **Agentic approach is necessary**
  - Too many tables: need to retrieve knowledge about the tables
- **Interpretation of results are difficult**
  - Lots of caveats on the data
    (e.g. contributions below $200 not included in some tables)
  - Requires experts on the data
- **Need to capture the expertise just like an apprentice**
  - More in the next class

LocalNews Matters
BAY AREA

DONATE TO SUPPORT LOCAL NEWS! Sign in

Local News · Aging & Longevity · Civic Engagement · Arts & Culture · Bay City Books · Equity Ripples · Prison Journalism · Experiments with AI · About Us

LOCAL NEWS

Keeping an eye on the future: Silicon Valley's star fundraiser Khanna may have big dreams

by Linda Liu, Anne Li and Taylor Torres, Bay City News · November 1, 2024

HONOLULU CIVIL BEAT
News That Matters • Support us

Legislature 2025 · Fireworks · Ideas · Special Projects · Topics · More +

Elections 2024

In Hawaiʻi Congressional Fundraising, Tokuda Far Outraised Case

---

THE MAINE MONITOR

About Us · Topics · Statehouse Hub · Projects · Audio & Video · Newslett

Content Type: Analysis

2024 ELECTIONS, FEDERAL GOVERNMENT, FOLLOW THE MONEY · 3 minute read

Who donated to Jared Golden and Austin Theriault?

AJC POLITICS

Politically Georgia · The First 100 Days · Legislature · Legislative Navigator

POLITICS

Harris spent $2.4M on reproductive health ads targeting Georgia voters in campaign's closing month

Focus on abortion and in vitro fertilization followed Georgia Supreme Court decision to reinstate Georgia's six-week abortion law

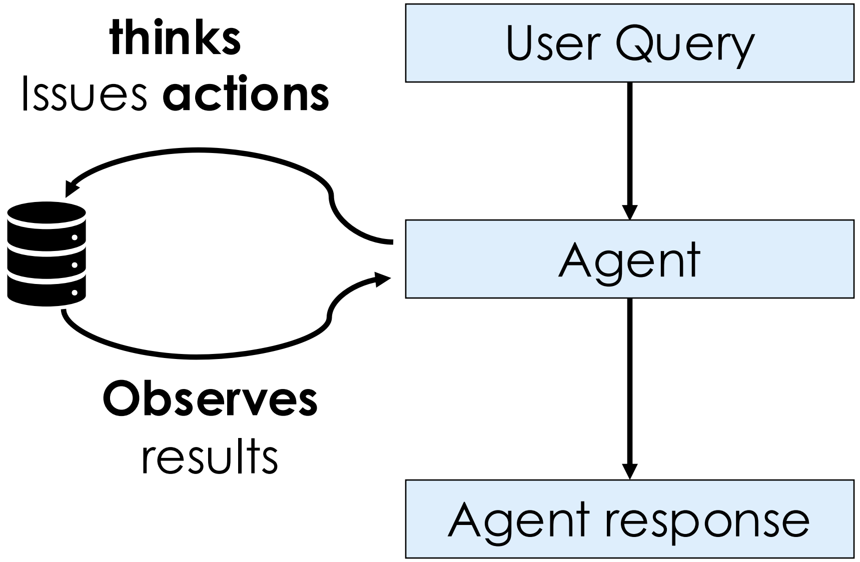
Outline

- Knowledge Graphs – aka Knowledge Bases (KB)
  - Challenges of KBQA (KB Question Answering)
  - Prior Approaches
  - The Agentic Approach for KBs
  - Dataset and Evaluation

- Agentic Approach to SQL Databases

- **Computational Thinking for KB Queries**

Agentic Approach

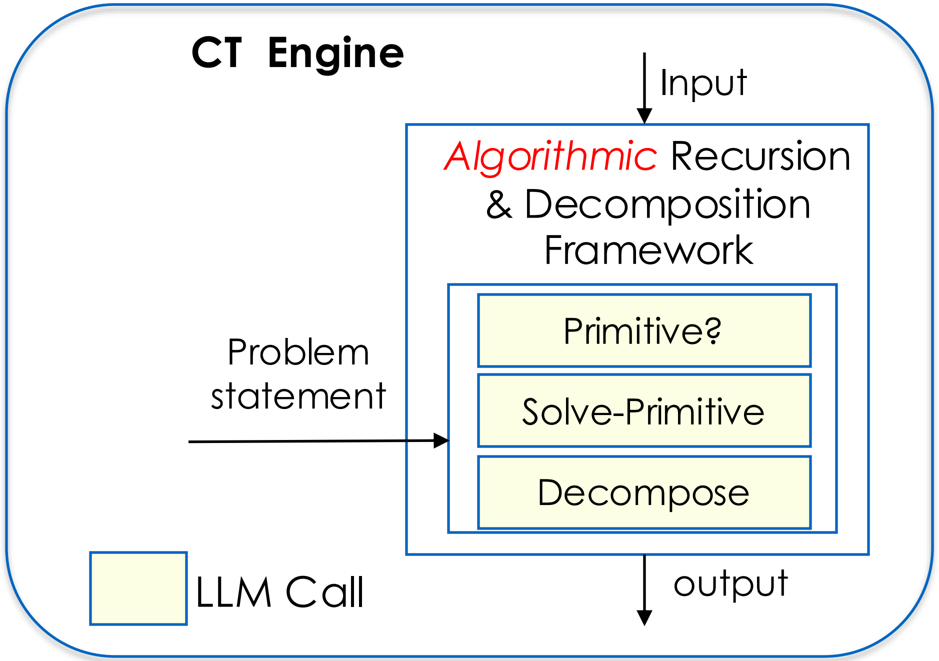


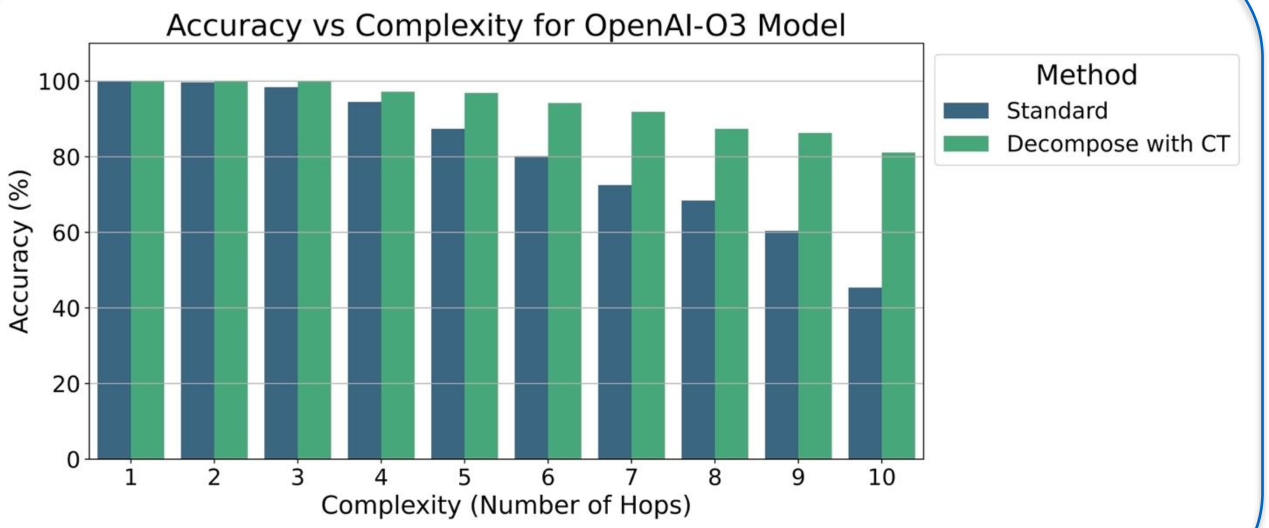
- Incremental decision procedure
  - Decides on the action based on observations
- Problems
  - May loop infinitely without forward progress
  - Does not plan actions involving multiple steps

LLMs Lack Computational Thinking (CT)

Example: Composition

Question	OpenAI-03 Model
Who is the wife of Benjamin Harrison?	Caroline Harrison
Who is the grandfather of Caroline Harrison?	George Scott
Who's the grandfather of the wife of Benjamin Harrison?	Dr. John Witherspoon



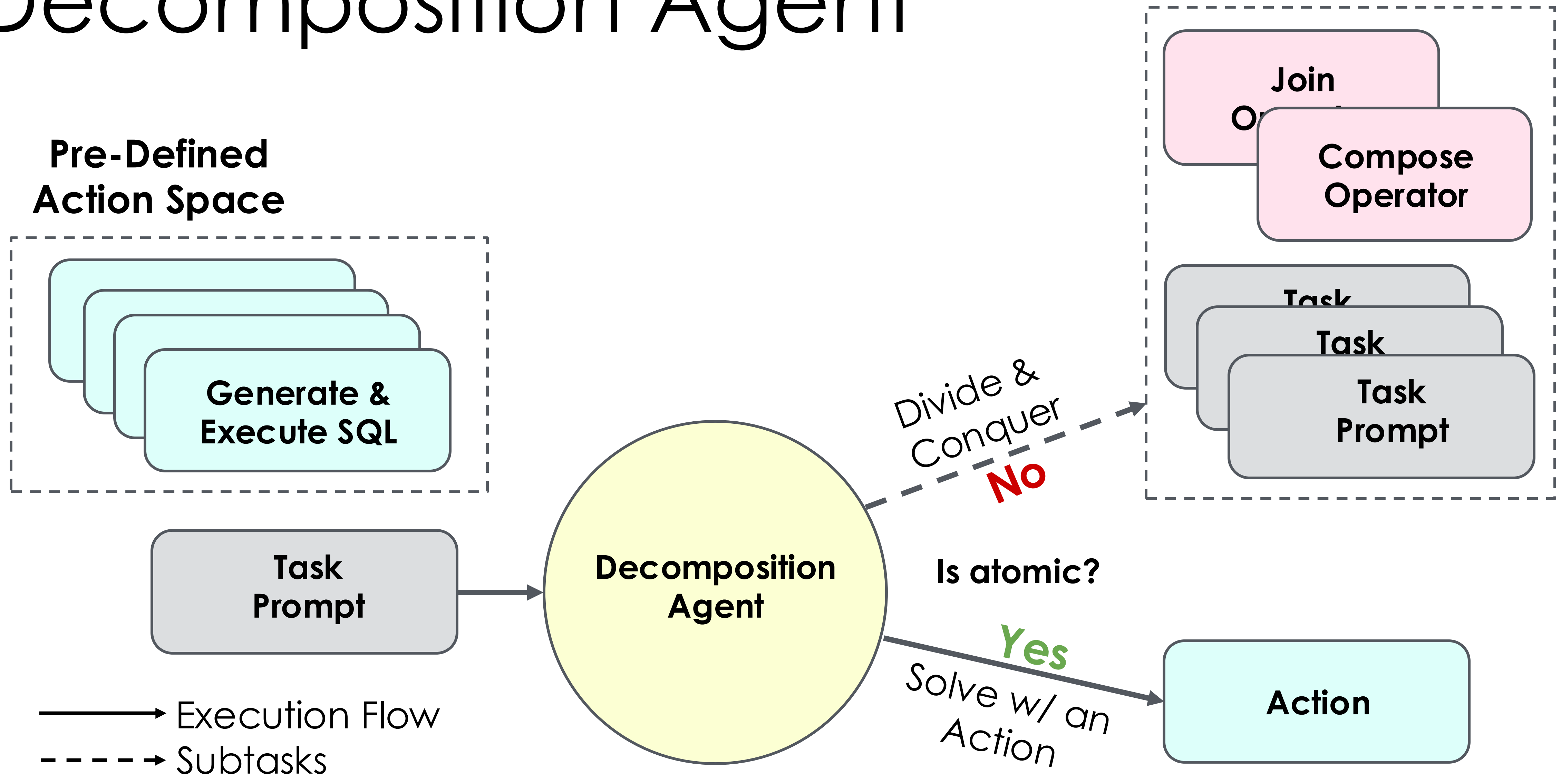


Dataset: LLM knows every hop of the question

A Computational Thinking (CT) Engine

- An algorithmic engine
  - Manages recursion
  - Calls LLM with the relevant information
- LLM functions
  - Decomposition agent
  - Evaluation if the task should be decomposed
  - Actions for the task
  - Operators to combine subtasks

Decomposition Agent



Semantic Parsing Actions for SQL

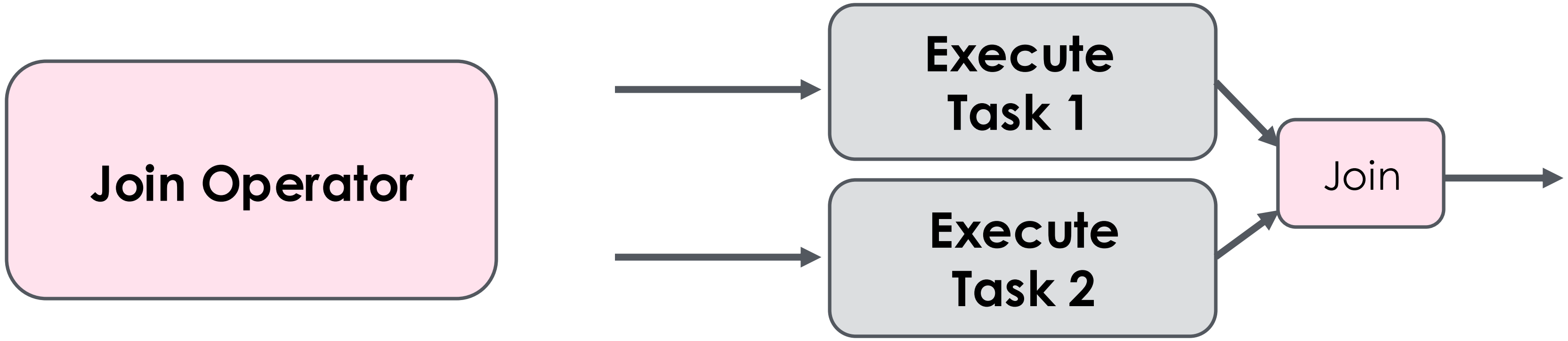
Describe Columns	Query → List of columns and their description Describe columns with all the information including type and its values
Get Background Knowledge	Query → A dictionary of detailed explanation of each related knowledge Using a retrieval-based function call
Sketch the Parse	Query → A natural language sketch Elaborate on the current query, and sketch the skeleton of the query (what columns to use? With what logic?) with reasoning
Generate & Execute SQL	Query → SQL + execution result Generate and execute the SQL query based on all info gathered

Across Different Query Languages

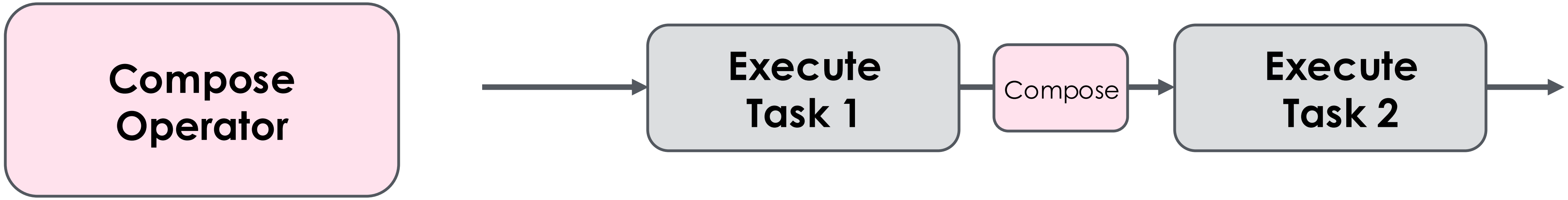
SQL	SPARQL	Cypher
Describe Columns	Search Wikidata	Describe Related Schema
Get Background Knowledge	Get Entity QID	Generate & Execute Cypher
Sketch the Parse	Get Property PID	
Generate & Execute SQL	Generate & Execute SPARQL	

Operators to Combine Subtasks

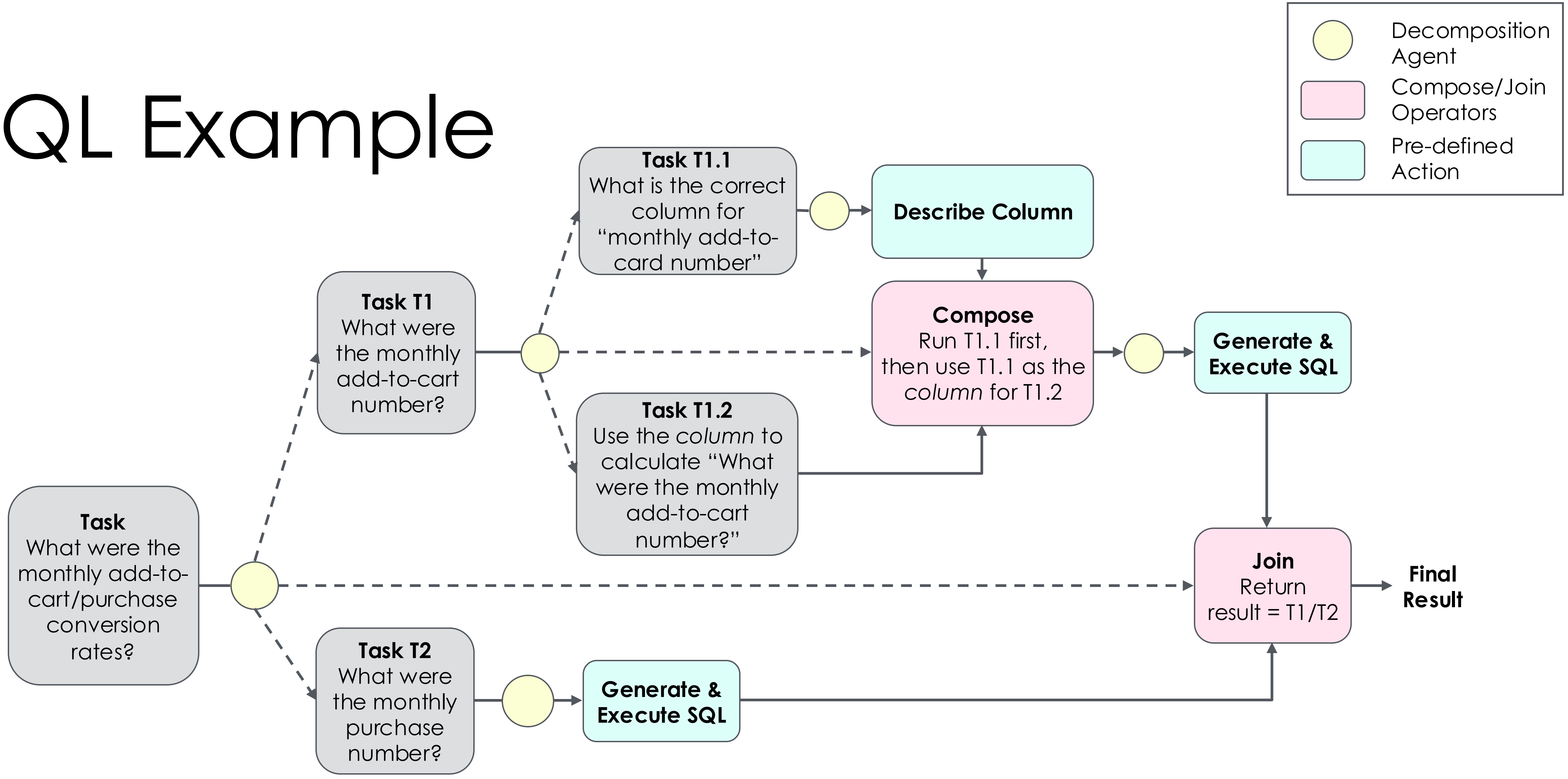
1. Parallel



2. Sequential

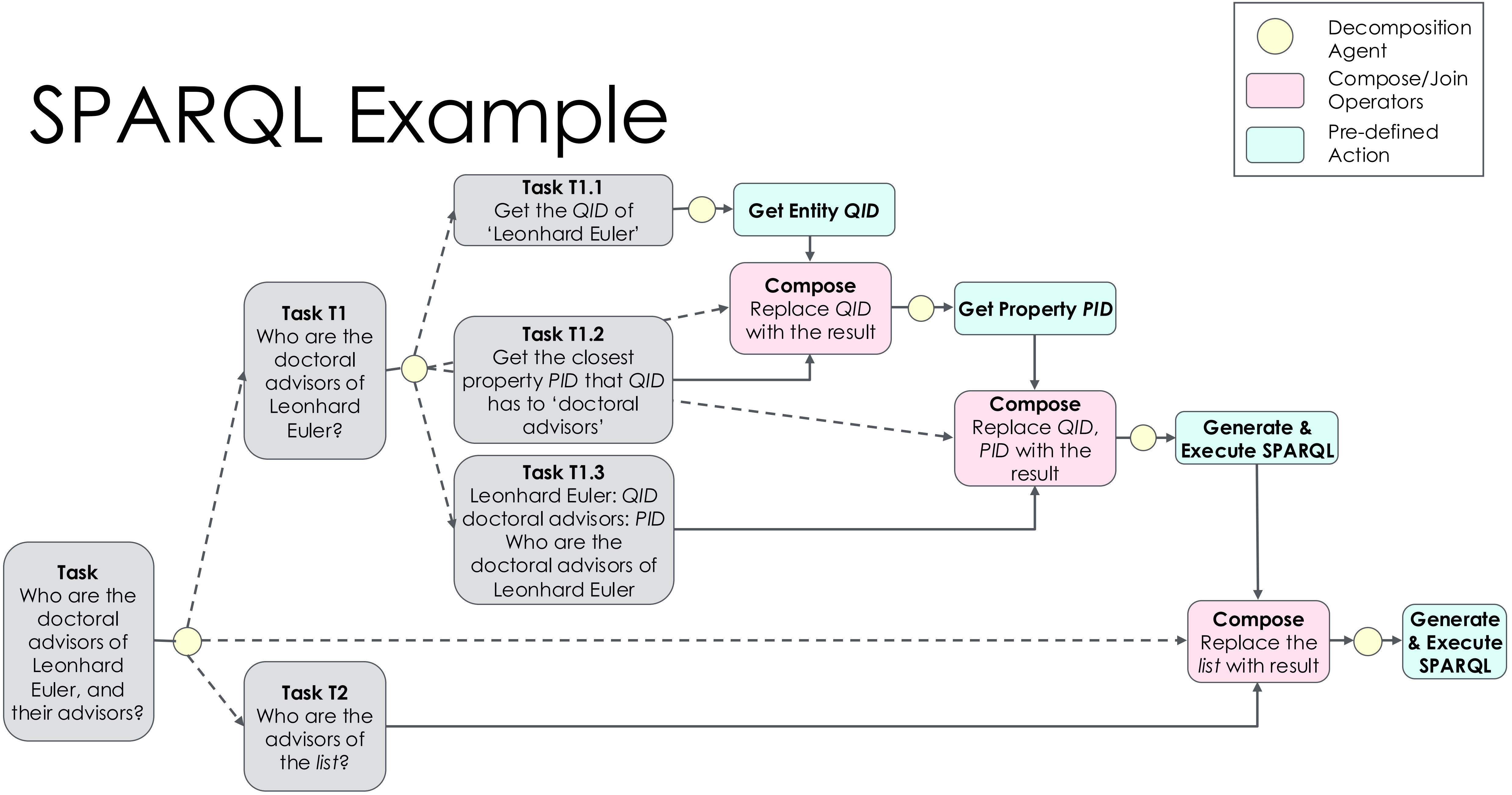


SQL Example



Decomposition Agent
Compose/Join Operators
Pre-defined Action
Task
What were the monthly add-to-cart/purchase conversion rates?
Task T1
What were the monthly add-to-cart number?
Task T1.1
What is the correct column for "monthly add-to-card number"
Describe Column
Compose
Run T1.1 first, then use T1.1 as the *column* for T1.2
Task T1.2
Use the *column* to calculate "What were the monthly add-to-cart number?"
Generate & Execute SQL
Join
Return result = T1/T2
Final Result
Task T2
What were the monthly purchase number?
Generate & Execute SQL

SPARQL Example



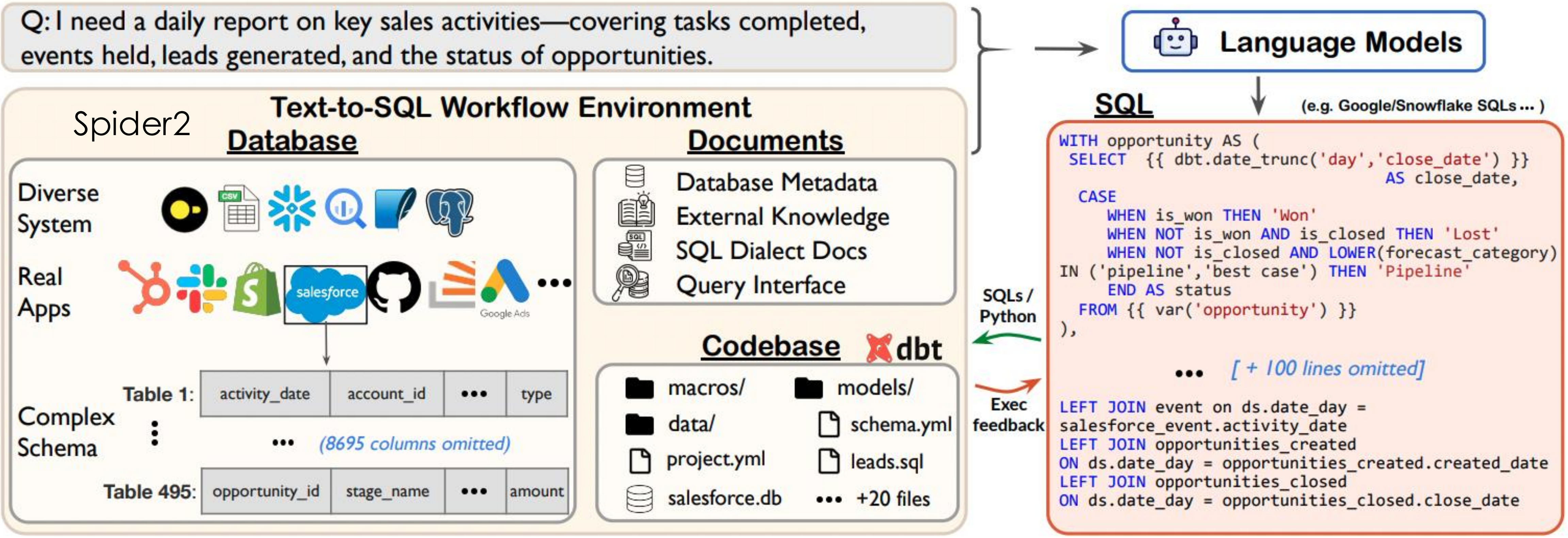
Spider 2.0

Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows

ICLR 2025 Oral🔥

- 632 real-world enterprise text-to-SQL workflow problems
  - Contains over 1,000 columns
- Stored in local or cloud databases (BigQuery and Snowflake)
- Challenges
  - Complex SQL workflow environments
  - Process extremely long contexts
  - Perform intricate reasoning
  - Need multiple SQL queries with diverse operations >=100 lines

Spider 2.0 Workflow



Lei, Fangyu, et al. "Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows." *arXiv preprint arXiv:2411.07763* (2024).

Computational Thinking on an Example in Spider 2.0

Calculate the change in the number of living trees of each fall color in New York City from 1995 to 2015
by computing, for each tree species, the difference between the number of trees not marked as dead in 1995
and the number of trees alive in 2015,
matching species by the uppercase form of their scientific names from the tree_species table.
Then, group the species by their fall color
and sum these differences to determine the total change in the number of trees for each fall color.

From the two census tables, count living trees for each species in 1995 and 2015, where:
- 1995 = rows whose status is NOT "Dead".
- 2015 = rows whose status is "Alive".

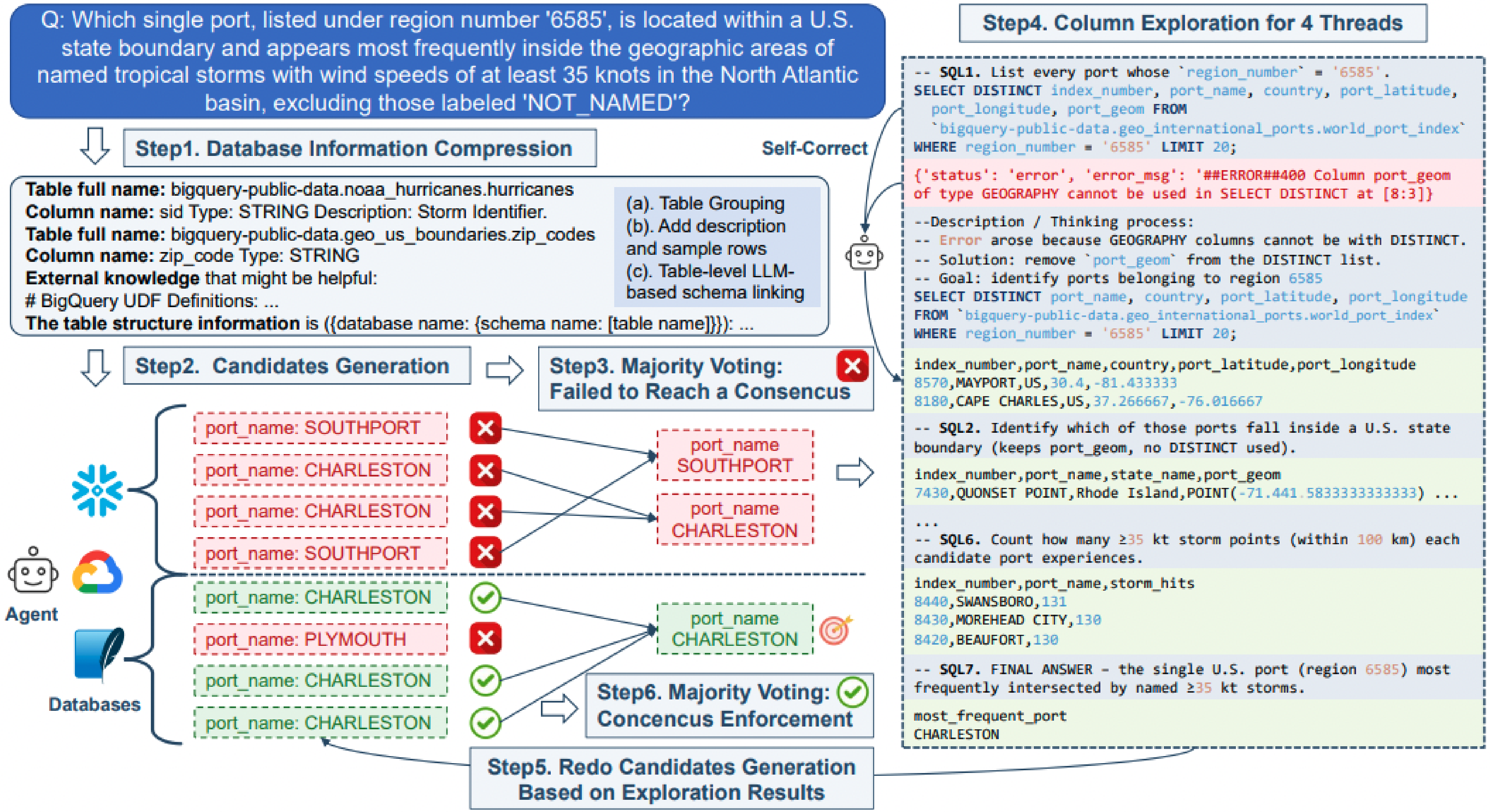
Match the species in both censuses to the tree_species table
    by comparing the upper-case form of spc_latin / species_scientific_name.
For every species that can be matched, produce one row containing:
(a) the upper-case scientific name,
(b) its fall_color taken from tree_species,
(c) cnt_1995 (living trees in 1995),
(d) cnt_2015 (living trees in 2015) and
(e) change = cnt_2015 - cnt_1995.
The result is an intermediate table of per-species changes with the species' fall color already attached.

Using the intermediate table produced above, group the rows by fall_color and sum the change column to obtain the total net change in the number of living trees between 1995 and 2015 for each fall color.

```
SELECT
  fall_color,
  SUM(change) AS total_net_change
FROM `spider-471218.temp_recursive_parser.multitable_subquery_depth_0_20250924_215719_0d4439c4`
GROUP BY
  fall_color
ORDER BY
  total_net_change DESC
```

```
SELECT
    UPPER(spc_latin) AS sci_name,
    COUNT(*) AS cnt_1995
  FROM `bigquery-public-data.new_york.tree_census_1995`
  WHERE spc_latin IS NOT NULL
    AND status <> 'Dead'
  GROUP BY sci_name
),
cnt_2015 AS (
  SELECT
    UPPER(spc_latin) AS sci_name,
    COUNT(*) AS cnt_2015
  FROM `bigquery-public-data.new_york.tree_census_2015`
  WHERE spc_latin IS NOT NULL
    AND status = 'Alive'
  GROUP BY sci_name
),
combined AS (
  SELECT
    COALESCE(c95.sci_name, c15.sci_name) AS sci_name,
    COALESCE(c95.cnt_1995, 0) AS cnt_1995,
    COALESCE(c15.cnt_2015, 0) AS cnt_2015
  FROM cnt_1995 c95
  FULL OUTER JOIN cnt_2015 c15
    ON c95.sci_name = c15.sci_name
)
SELECT
  cmb.sci_name,
  ts.fall_color,
  cmb.cnt_1995,
  cmb.cnt_2015,
  cmb.cnt_2015 - cmb.cnt_1995 AS change
FROM combined cmb
JOIN `bigquery-public-data.new_york.tree_species` ts
  ON cmb.sci_name = UPPER(ts.species_scientific_name)
ORDER BY cmb.sci_name;
```

Baseline1: SOTA of Published Systems: Reforce Agent



Deng et al., ReFoRCE: A Text-to-SQL Agent with Self-Refinement, Format Restriction, and Column Exploration, June 2025, https://arxiv.org/abs/2502.00675

Baseline 2: Computation Thinking (CT) Prompt

- **Without the CT Engine**
- Prompt LLM to
  "decompose the task and generate the SQL accordingly"

  Using the given set of actions

Preliminary Result

Spider2-Snow

Method	Base Model	Execution Accuracy @ 1
ByteBrain-Agent	Unknown	67.6
ReForce	GPT o3	62.9
CT Prompt without an Engine	GPT-5	41.2
CT Engine(ours)	GPT o3	**69.6**

Spider2-Lite

Method	Base Model	Execution Accuracy @ 1
ReForce	GPT o3	55.2
CT Prompt without an Engine	GPT-5	31.9
CT Engine(ours)	GPT o3	**55.7**

Conclusions

- Semantic Parsing for real KB is hard
- Natural language access to KB is important
  - Wikidata users, FEC journalists, enterprises

- Agentic approaches
  - ReAct (Reason + action)
  - Computational thinking

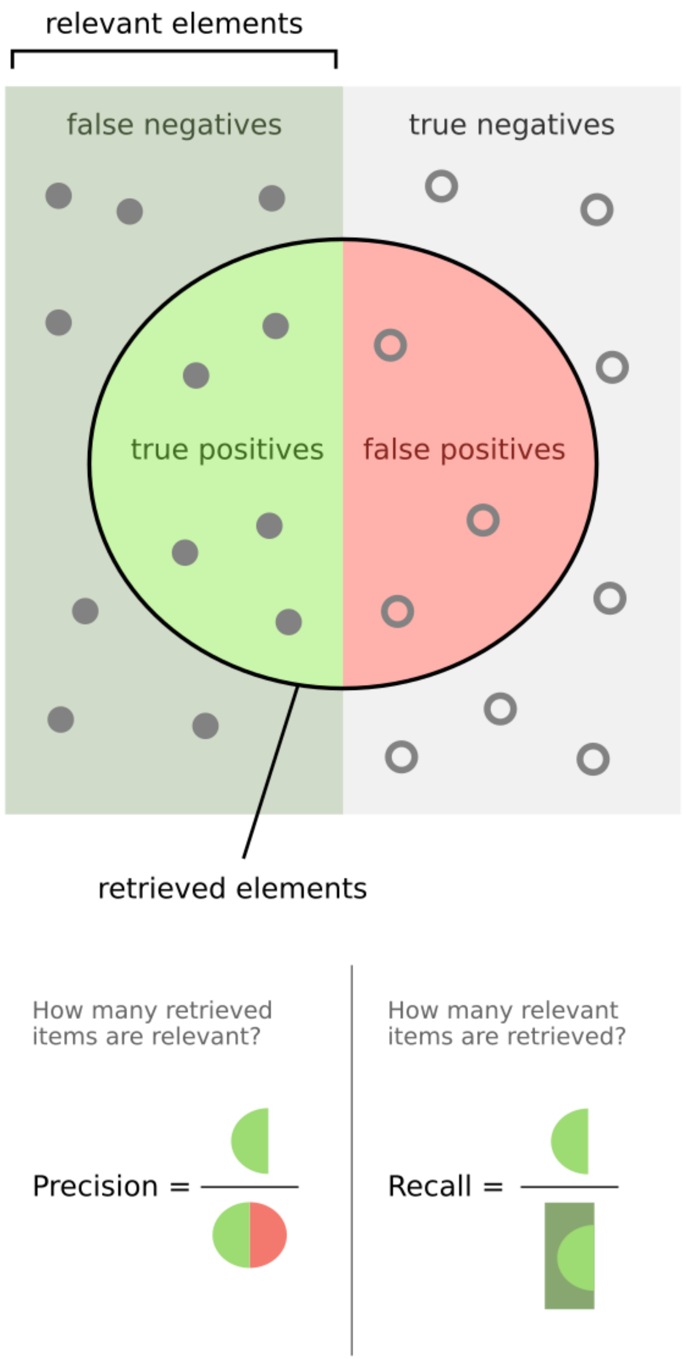
Appendix

Calculating F1 for Tables

The F1 metric

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}.$$

TP: True Positive
FP: False Positive
FN: False Negative



But: traditional EM & F1does not work

what are the top 3 counties with most people in South Dakota?

Predicted

Minnehaha County	**206,930**
Pennington County	**115,903**
Lincoln County	**73,238**

Gold

Minnehaha County
Pennington County
Lincoln County

Question: Do we want to penalize this column?

New Metric: **row-major** EM & F1

1. Assignment of rows

Predicted

Q5994	piano	99
Q8350	trombone	11
Q8338	trumpet	8

Assigned rows

Gold

Q8350	11
Q5994	99
Q1467690	2
Q83509	1

Step 1: Run an **assignment algorithm** to maximize **total overlap** (recall)

Without penalizing extra columns in prediction

y_i a row in gold

y'_i assigned row of y_i in prediction

$$\mathrm{recall}(y_i, y'_j) = \frac{|y_i \cap y'_j|}{|y_i|}$$

Matching rows with 0 recall is not allowed

New Metric: **row-major** EM & F1

2. True positives

Predicted

Q5994	piano	99
Q8350	trombone	11
Q8338	trumpet	8

Assigned rows

Gold

Q8350	11
Q5994	99
Q1467690	2
Q83509	1

Step 2: Calculate **true positives** as sum of recalls in assigned rows

$$tp = \sum_{(i,j) \in A(\mathbf{y}, \mathbf{y}')} \text{recall}(y_i, y'_j) \qquad = 2$$

New Metric: **row-major** EM & F1

3. false positives

Predicted

Q5994	**piano**	99
Q8350	**trombone**	11
Q8338	**trumpet**	**8**

Assigned rows

False Positives

Gold

Q8350	11
Q5994	99
Q1467690	**2**
Q83509	**1**

Step 3: Each unassigned row in prediction counts as a **false positive**

n': number of rows in prediction

r: number of rows in matching

$$fp = n' - r \qquad = 1$$

New Metric: **row-major** EM & F1

4. false negatives

Predicted

Q5994	piano	99
Q8350	trombone	11
Q8338	trumpet	8

Assigned rows

False Positives

Gold

Q8350	11
Q5994	99
Q1467690	2
Q83509	1

False Negatives

Step 4: Each unassigned row in gold counts as a **false negative**

plus sum of 1– recall in the matching (0 in this case)

n: number of rows in gold

r: number of rows in matching

$$fn = n - r + \sum_{(i,j)\in A(\mathbf{y},\mathbf{y}')} 1 - \mathrm{recall}(y_i, y'_j) \quad = 2$$

New Metric: **row-major** EM & F1

5. F1

Predicted

Q5994	piano	99
Q8350	trombone	11
Q8338	trumpet	8

Assigned rows

False Positives

Gold

Q8350	11
Q5994	99
Q1467690	2
Q83509	1

False Negatives

Step 5: Calculate F1 with the usual formula

$$F_1 = \frac{2tp}{2tp+fn+fp} = 0.66$$