

Evolution of Semantic Parsing (& modern testing)

CS395T - Communicative and Grounded AI Agents
UT Austin, Fall 2025

Presented by Rishabh Mediratta

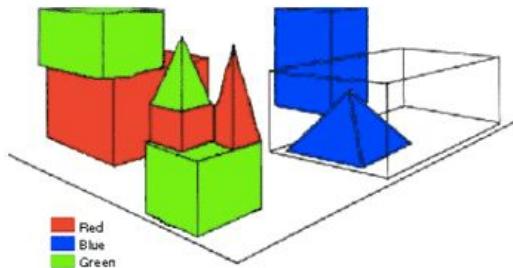
The Big Picture before we begin

Keep this in mind as we go through this presentation - evolution of the task of natural language to a formal machine-understandable representation moving from symbolic rule-based to modern data-driven systems

The pioneer - SHRDLU

The idea - talk to a machine about a world it understands

In this case, the world was a simulated micro block world with a hand and limited set of actions available



Person: Pick up a big red block.

Computer: OK.

Person: Grasp the pyramid.

Computer: I don't understand which pyramid you mean.

SHRDLU capabilities

- Commands to actions
 - "Pick up a red block" -> MOVETO, GRASP, UNGRASP
 - Used planner to plan
- Resolve ambiguity with context
 - "Grasp the pyramid" -> if unclear, ask for clarification
- Remember dialogue history
 - Dialogue manager
- Explain actions
 - "Why did you pick up the pyramid?" -> "SO THAT THE GREEN BLOCK WOULD BE CLEAR"

Moving to learning

SHRDLU played a part in popularizing natural language interfaces for machines

Key limitation - hand-programmed - every rule, action, etc all by Winograd; limited context - block world

New question - instead of defining rules manually - can a machine learn how to do this?

Learning to Parse Database Queries Using Inductive Logic Programming

- Shift to learning instead of creating
- Natural Language interface for DB queries with a system called CHILL
- Given question-query pairs, learn to create a parser automatically

Inductive Logic Programming primer

Given a set of facts, find a general rule that explains the facts

For question-query pairs, learn rules that translate english sentences to DB queries

Given Program

Examples of Target concept

C1 : Child (Mike, Joe)

C2 : Child (Mike, Anne)

Background knowledge

C3 : Father (Joe, Mike)

C4 : Father (x, y) → Parent (x, y)

C5 : Mother (Anne, Mike)

C6 : Mother (x, y) → Parent (x, y)

Generated Program

C7 : Parent (x, y) → Child (y, x)

CHILL

The domain: US geography dataset called Geobase with the query language being Geoquery

Data: question-query pairs

```
What are the major cities in Kansas?  
answer(C, (major(C), city(C), loc(C,S),  
equal(S,stateid(kansas)))).
```

Goal: Learn a parser capable of taking in a new question about geography and producing the correct query

How: ILP

Results: 225-25 train-test split. Outperforms existing system on 175 or more queries. Also produced fewer spurious parses.

The next frontier

CHILL proved that data can be used for this parsing problem to produce effective solutions for generalized problems

Previous bottleneck - Handwritten rules. New bottleneck - Data.

Next problem - data synthesis.

The next frontier

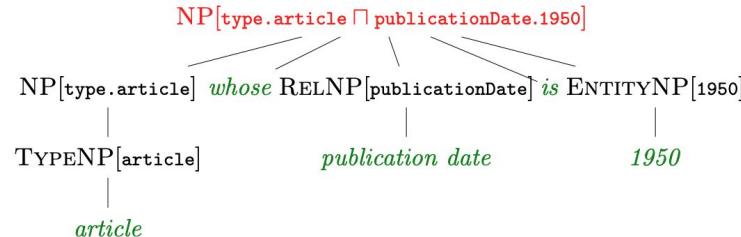
How do you build a parser for a new domain with no training examples?

Traditional data collection is slow and expensive

“Building a semantic parser overnight” introduces a 4-step method to generate the necessary data

The 2 core ideas you need - Idea 1 - Canonical Compositionality

Compositionality - Complex expressions are defined by the meanings of its parts and the rules to combine them. The paper assumes for any complex logical expression, it is possible to get a (somewhat awkward) English phrase based on some rules (this English phrase is called canonical utterance)



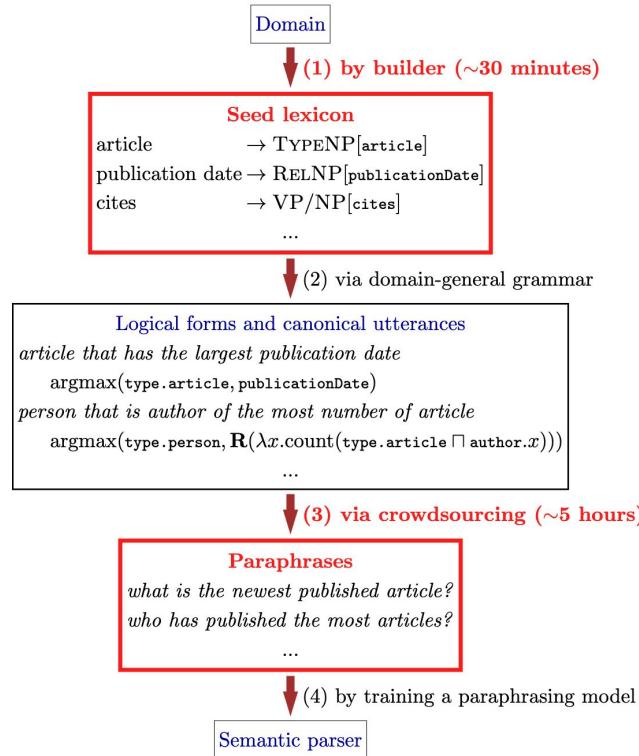
The 2 core ideas you need - Idea 2 - Bounded Non-Compositionality

Canonical utterances correspond to logical structure but natural language often does not

For eg, “co-author of X” is the natural language version of “person that is author of paper whose author is X” (this is why I said awkward)

The “Bounded” assumption - these non-compositional chunks are bounded (limited). A limited number of examples will let a model learn to map from canonical utterances to natural language.

4-step process



4-step process

1. Create a seed lexicon - eg DB property publicationDate is mapped to publication date
2. Generate logical forms and canonical utterances - using the domain general grammar and seed lexicon, logical forms and canonical utterances are produced. Eg - NP[x] that has the largest RELNP[r] → NP[arg max(x, r)]. substitute article for NP and publicationDate for RENLP
3. Crowdsource canonical utterance -> natural language (authors use Amazon MT), create (natural_language, canonical_utterance, loacl_form) triplets
4. Train a log-linear model to find correct logical form given natural language (generate candidates with beam search and keep based on similarity between input query and canonical utterances)

Results

Avg accuracy of 58.8% across all domains

Overnight experiment - parser built within 24 hours got to 70.8%

An experiment on the CALENDAR domain confirms bounded non-compositionality hypothesis

Different ablations prove that implemented steps of the system are useful.
Important ablation - removing domain specific features drop occurrence significantly

Yet another frontier

Now that the model and data problems have been explored, how do we push systems?

The next step is establishing a benchmark to measure the performance of modern semantic parsers (spoiler - a lot of these modern systems are LLM-based and the task is to parse natural language to SQL)

Late 2010s

- Models were performing very well on tasks like GeoQuery (> 80%)
- Problems with existing datasets
 - Single DB for both training and testing
 - Same queries appeared in both train and test (with different questions)
 - Tasks like WikiSQL were large but simple queries

The need for a large scale and complex task hence arose



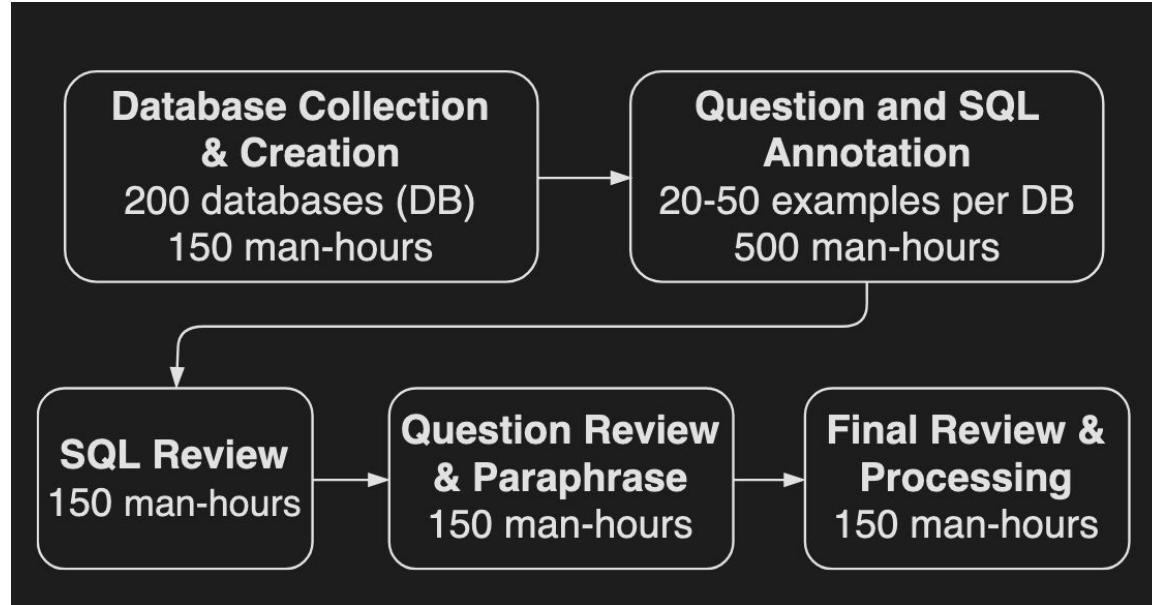
Spider - Yale Semantic Parsing and Text-to-SQL Challenge

Large scale, complex, cross-domain human annotated text-toSQL dataset

200 databases with multiple tables and foreign key relations within tables, 138 domains, 10181 natural language questions, 5693 unique complex SQL queries labeled by 11 CS students investing >1000 man hours

Corpus Creation

WikiSQL +
online,
multiple
domains



SQL with complex
structure (LIKE, JOIN,
nested queries, UNION,
etc) + no ambiguous or
out of knowledge-base
questions allowed

Spider as compared to other datasets

Dataset	# Q	# SQL	# DB	# Domain	# Table /DB	ORDER BY	GROUP BY	NESTED	HAVING
ATIS	5,280	947	1	1	32	0	5	315	0
GeoQuery	877	247	1	1	6	20	46	167	9
Scholar	817	193	1	1	7	75	100	7	20
Academic	196	185	1	1	15	23	40	7	18
IMDB	131	89	1	1	16	10	6	1	0
Yelp	128	110	1	1	7	18	21	0	4
Advising	3,898	208	1	1	10	15	9	22	0
Restaurants	378	378	1	1	3	0	0	4	0
WikiSQL	80,654	77,840	26,521	-	1	0	0	0	0
Spider	10,181	5,693	200	138	5.1	1335	1491	844	388

Databases in the test set do not appear in the training set

Evaluation Metrics

- Correctness
 - Component Matching - whether the generated query has the right clauses (SELECT, WHERE) with the right features (student_id, room_number) - follows a set based comparison, different order not penalized
 - Exact Match - every component must match in the generated and gold query. Here, the specific values usually required in a query is not required, the structure alone suffices.
 - Execution accuracy - to test whether does the query actually runs. Model is given the correct values to fill in the right placeholders.
- SQL Hardness
 - 4 levels: easy, medium, hard, extra hard based on the number of SQL components, selections, and conditions

Extra Hard

What is the average life expectancy in the countries where English is not the official language?

```
SELECT AVG(life_expectancy)
FROM country
WHERE name NOT IN
(SELECT T1.name
FROM country AS T1 JOIN
country_language AS T2
ON T1.code = T2.country_code
WHERE T2.language = "English"
AND T2.is_official = "T")
```

Results

	Example Split	Test					Dev All
		Easy	Medium	Hard	Extra Hard	All	
Database Split							
Seq2Seq	22.0	7.8	5.5	1.3	9.4	10.3	
Seq2Seq+Attention (Dong and Lapata, 2016)	32.3	15.6	10.3	2.3	15.9	16.0	
Seq2Seq+Copying	29.3	13.1	8.8	3.0	14.1	15.3	
SQLNet (Xu et al., 2017)	34.1	19.6	11.7	3.3	18.3	18.4	
TypeSQL (Yu et al., 2018)	47.5	38.4	24.1	14.4	33.0	34.4	
Seq2Seq	11.9	1.9	1.3	0.5	3.7	1.9	
Seq2Seq+Attention (Dong and Lapata, 2016)	14.9	2.5	2.0	1.1	4.8	1.8	
Seq2Seq+Copying	15.4	3.4	2.0	1.1	5.3	4.1	
SQLNet (Xu et al., 2017)	26.2	12.6	6.6	1.3	12.4	10.9	
TypeSQL (Yu et al., 2018)	19.6	7.6	3.8	0.8	8.2	8.0	

Table 2: Accuracy of Exact Matching on SQL queries with different hardness levels.

Modern-ish Results

Leaderboard - Execution with Values

Our current models do not predict any value in SQL conditions so that we do not provide execution accuracies. However, we encourage you to provide it in the future submissions. For value prediction, your model should be able to 1) copy from the question inputs, 2) retrieve from the database content (database content is available), or 3) generate numbers (e.g. 3 in "LIMIT 3"). *Notice:* Test results after May 02, 2020 are reported on the new release (collected some annotation errors).

Rank	Model	Test
1 Nov 2, 2023	MiniSeek <i>Anonymous</i> Code and paper coming soon	91.2
1 Aug 20, 2023	DAIL-SQL + GPT-4 + Self-Consistency <i>Alibaba Group</i> (Gao and Wang et al.,'2023) code	86.6
2 Aug 9, 2023	DAIL-SQL + GPT-4 <i>Alibaba Group</i> (Gao and Wang et al.,'2023) code	86.2

Leaderboard - Exact Set Match without Values

For exact matching evaluation, instead of simply conducting string comparison between the predicted and gold SQL queries, we decompose each SQL into several clauses, and conduct set comparison in each SQL clause. Please refer to the paper and [the Github page](#) for more details. *Notice:* Test results after May 02, 2020 are reported on the new release (collected some annotation errors).

Rank	Model	Dev	Test
1 Nov 2, 2023	MiniSeek <i>Anonymous</i> Code and paper coming soon	80.3	81.5
1 Sep 13, 2022	Graphix-3B + PICARD (DB content used) <i>Alibaba DAMO & HKU STAR & SIAT</i> (Li et al., AAAI'2023) code	77.1	74.0
2 Sep 14, 2022	CatSQL + GraPPa (DB content used) <i>Anonymous</i>	78.6	73.9

Actual modern benchmark

Spider 2.0 - move to real-world enterprise-level workflows

Larger and more complex tables with metadata - multiple TBs

More comprehensive task - read documentations, work with retrieved data, transformation, analysis, understand codebase

More data storage systems - Snowflake, BigQuery and more SQL dialects (Postgres, DuckDB)

Not solved

Spider 2.0-Snow

Spider 2.0-lite

Spider 2.0-DBT

Spider 2.0-Lite is a self-contained text-to-SQL task that includes well-prepared [database metadata](#) and [documentation](#). This setup enables a text-in, text-out approach, facilitating faster development and evaluation. Spider 2.0-lite, which has 547 examples, is designed to handle queries for [BigQuery](#), [Snowflake](#), and [SQLite](#) databases.

Rank	Method	Score
1 Aug 7, 2025	AgenticData + Qwen3 <i>Tsinghua University</i> [Sun et al. '25]	44.5
2 May 22, 2025	ReFoRCE + o3 <i>Hao AI Lab x Snowflake</i> [Deng et al. '25]	37.84
3 Aug 7, 2025	ReFoRCE + Qwen3 <i>Tsinghua University</i> [Sun et al. '25]	35.6

Spider 2.0-Snow

Spider 2.0-lite

Spider 2.0-DBT

Spider 2.0-Snow is a self-contained text-to-SQL task that includes well-prepared database metadata and documentation, includes 547 examples, all hosted on [Snowflake](#), which offers participants free quotas.

Methods with -* use special settings (ground-truth tables) and are not included in the ranking.

Rank	Method	Score
1 Aug 14, 2025	PAI-DataSurfer Agent <i>Alibaba Cloud Computing Platform</i>	59.78
2 Aug 8, 2025	WindAgent + Claude-4-Sonnet <i>MeiTuan AI For FinData</i>	59.05
3 Aug 6, 2025	Ask Data with Relational Knowledge Graph <i>AT&T CDO & RelationalAI</i>	57.77

Spider 2.0-Snow

Spider 2.0-lite

Spider 2.0-DBT

Spider 2.0-DBT is a comprehensive code generation agent task that includes 68 examples. Solving these tasks requires models to understand project code, navigating complex SQL environments and handling long contexts, surpassing traditional text-to-SQL challenges.

Rank	Method	Score
1 Jul 26, 2025	Shadowfax-DBT-Agent + GPT-5 <i>Shadowfax Data</i>	41.18
2 Aug 9, 2025	Spider-Agent-Extended + GPT-5 <i>Woods, A. '25.</i>	39.71
3 Jun 25, 2025	DAQUV_QUVI_Agent <i>DAQUV</i>	17.65

Closing Thoughts

Problems arose in Spider with models not generating the correct SQL - how to fix that?

Is translating directly to SQL the best approach? Or will some intermediate representation help?

How to tackle the challenges posed by Spider 2.0 and allow models to use multiple tools to tackle the challenges it presents?

Spoiler - all these will be addressed next

References

<https://dspace.mit.edu/handle/1721.1/7095>

<https://www.cs.utep.edu/nigel/papers/shrdlu.pdf>

<https://cdn.aaai.org/AAAI/1996/AAAI96-156.pdf>

<https://aclanthology.org/P15-1129.pdf>

<https://aclanthology.org/D18-1425/>

<https://arxiv.org/abs/2411.07763>

<https://yale-lily.github.io/spider>

<https://spider2-sql.github.io/>

Evolution of Semantic Parsing - Part II

Lunyu Nie (lynie@utexas.edu)

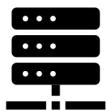
3rd year CS PhD student working on Agent Orchestration

Quick Recap: Semantic Parsing



Definition:

Mapping **natural language utterance** to
executable logical forms.



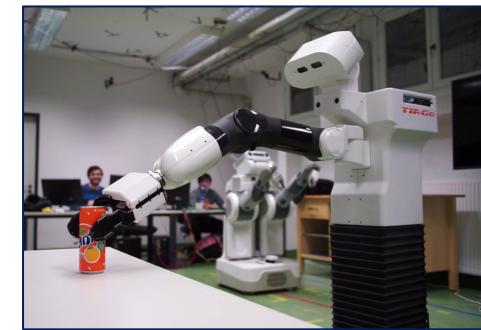
Applications:



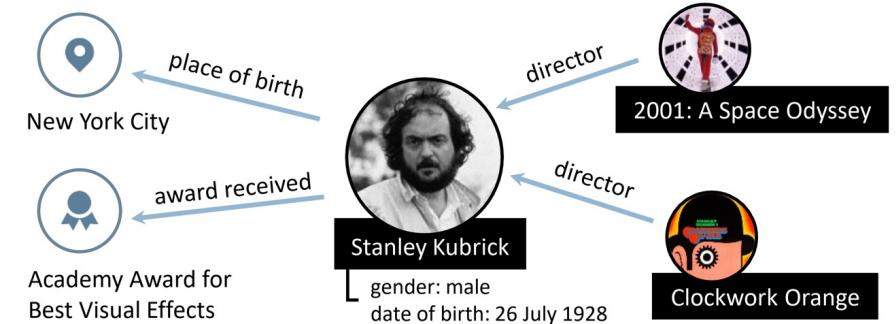
DB/KB Query



Virtual Assistant



Robot Control



Where was the director of film "2001 space odyssey" born?

Semantic Parsing

```
SELECT DISTINCT ?e_2 WHERE { ?e <instance_of> ?c .  
?c <name> "film". ?e <name> "2001: A Space Odyssey" .  
?e <director> ?e_1 . ?e_1 <place_of_birth> ?e_2 }
```

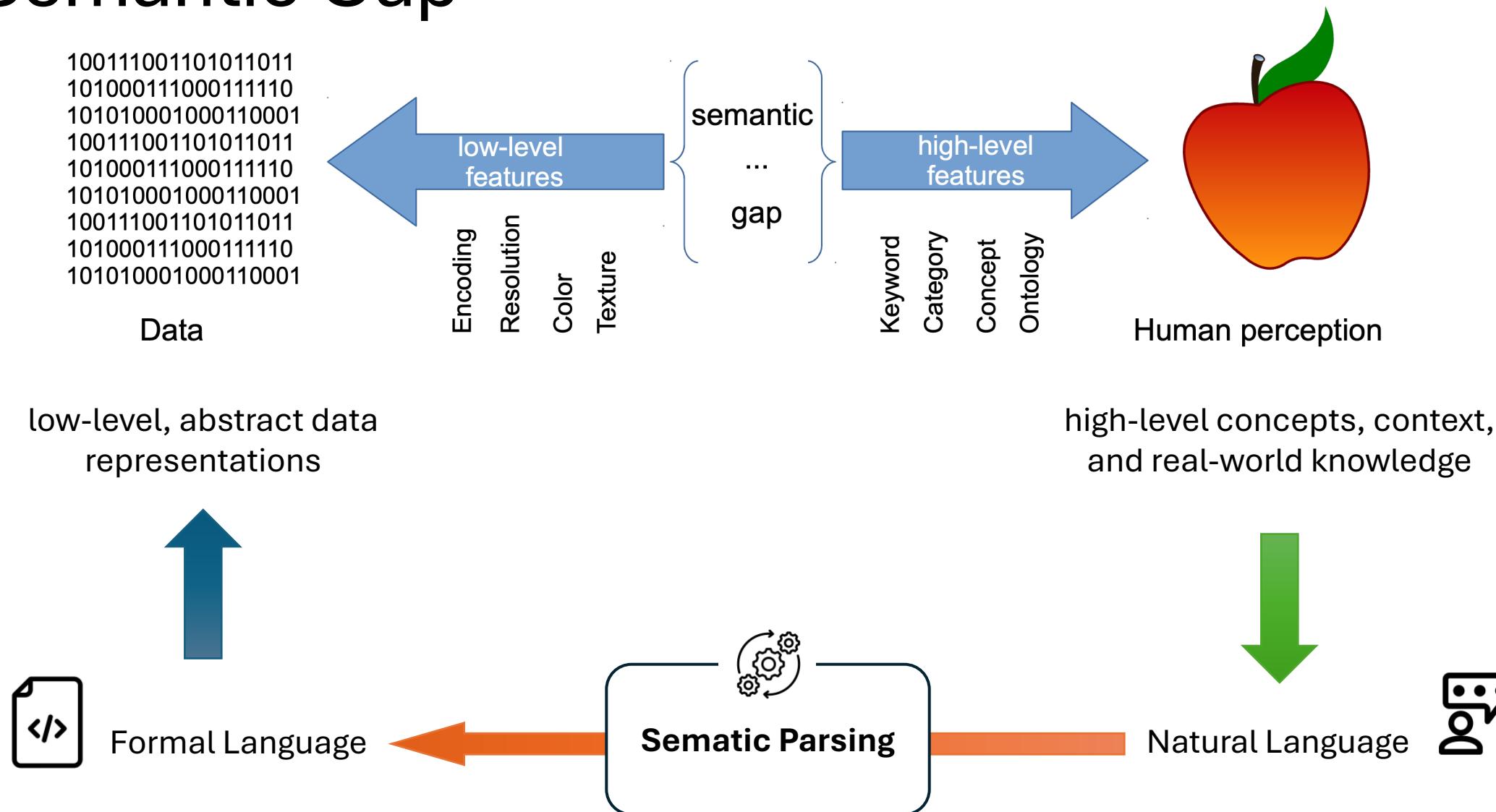
Execution

New York City.



Code Generation

Semantic Gap



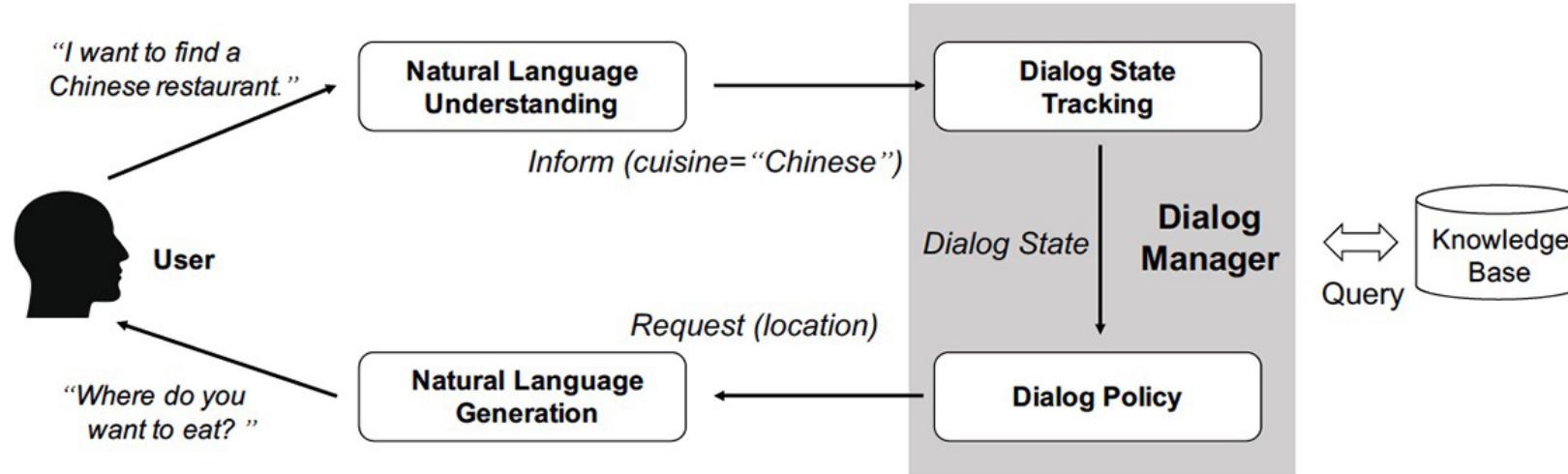
Methods to Bridge the Gap

- Symbolic methods
 - Interpretable
 - Verifiable
 - Structured Domain Knowledge
 - Data efficient
- Neural methods
 - Scalable
 - Flexible
 - Handles messy data
 - Easy to get started (need data!)
- Symbolic heuristics
 - Execution-guided
 - Grammar Constrained-decoding
- Better neural networks
 - Advanced model architecture
 - Scaled Pretraining
- Neurosymbolic combinations
 - Data Augmentation with SCFG
 - Intermediate Representations

Task-oriented Dialogue as Dataflow Synthesis

Microsoft Semantic Machines*

TACL 2020

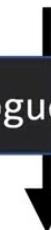


- Problem: how to represent & track user goals and requests?
 - users sometimes request **complex** new values, **refer** to old values, and **revise** previous plans
- Proposed solution: Dialogue as **Dataflow Graph**

Answering Simple Query

Slot filling: e.g. Bobrow et al. 1977, Liu & Lane 2016

Hey, what's on my calendar today?



Dialogue Model

```
showCalendar(date = TODAY , person = USER )
```

Answering Complex User Requests

A specialized function?

freeBetweenNowAndEndOfEvent

Answering **complex** user requests

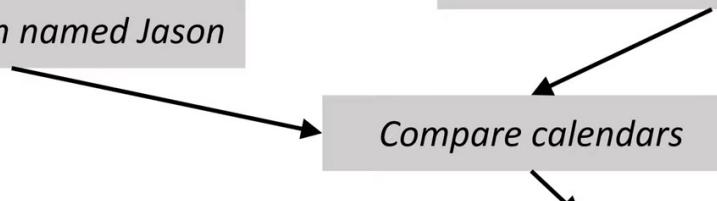
Is Jason free to help between now
and when my TACL talk ends?

Find a person named Jason

Find an event named TACL talk

Compare calendars

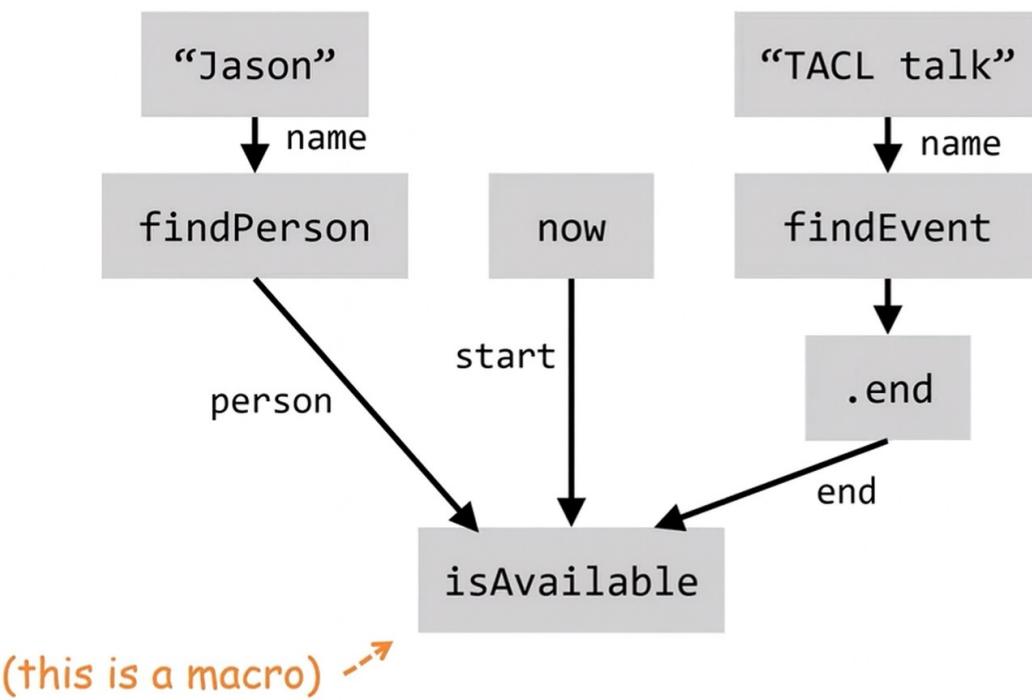
Language generation



Decomposition into a Dataflow Graph

Is Jason free to help between now and when my TACL talk ends?

```
isAvailable(  
    person=findPerson(  
        name="Jason"),  
    start=now(),  
    end=findEvent(  
        name="TACL talk"  
    ).end  
)
```



Dialogue as Interactive Programming

What do I have on my calendar
today?

Your TACL talk started eight
seconds ago.

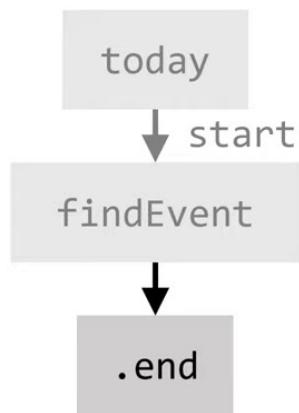
today
↓ start
findEvent

Dialogue as Interactive Programming

What do I have on my calendar today?

Your TACL talk started eight seconds ago.

Oh no, when does the talk end?



Dialogue as Interactive Programming

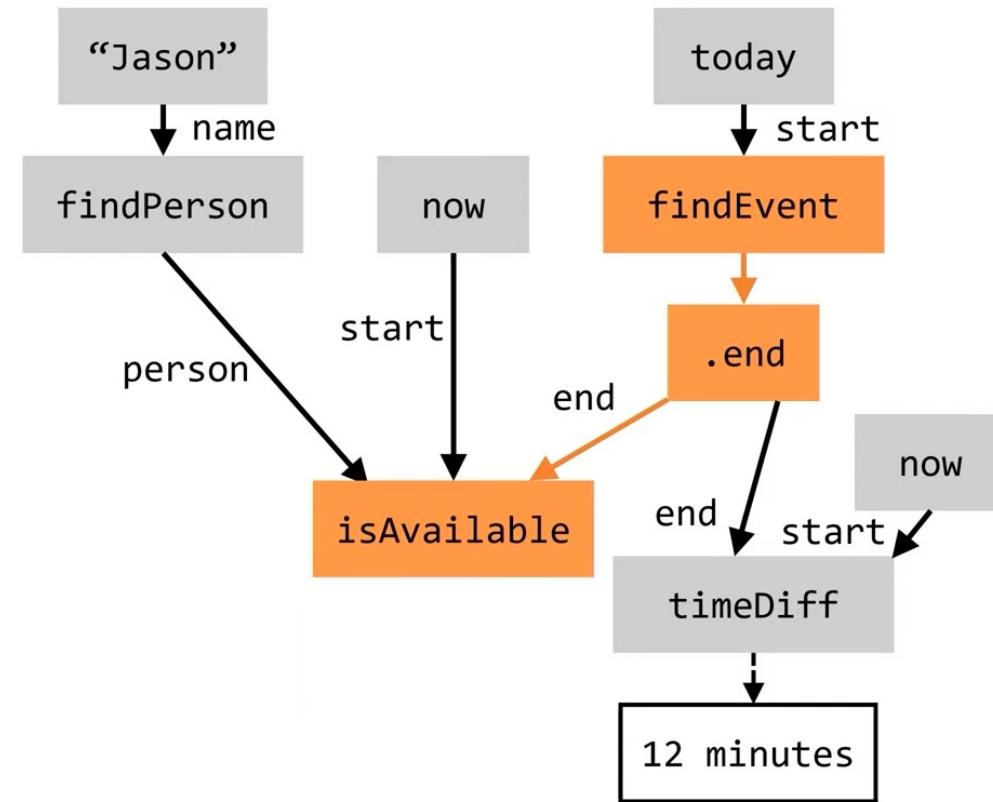
What do I have on my calendar today?

Your TACL talk started eight seconds ago.

Oh no, when does the talk end?

It ends in twelve minutes.

Is Jason free between now and then?



Complex actions are built by **composing** simpler ones rather than defining a large set of top-level behaviors

Graph Transformation I: Reference

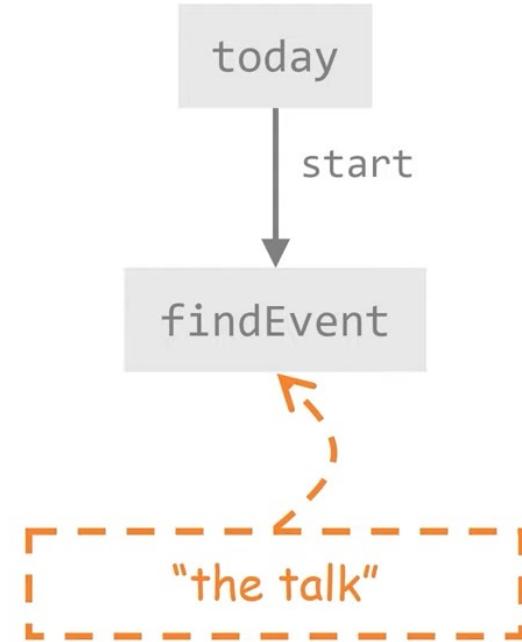
What do I have on my calendar today?

Your TACL talk started eight seconds ago.

Oh no, when does the talk end?

(refer to "the talk").end

Metacomputation Operator 1: Refer

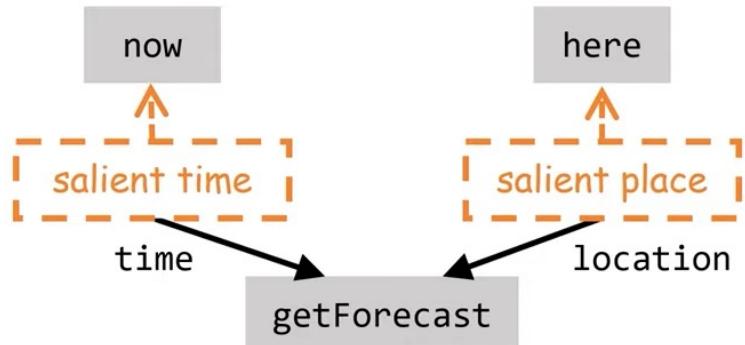


Reference Resolution: Salience

Good morning, Jacob!

What's the weather going to be like?

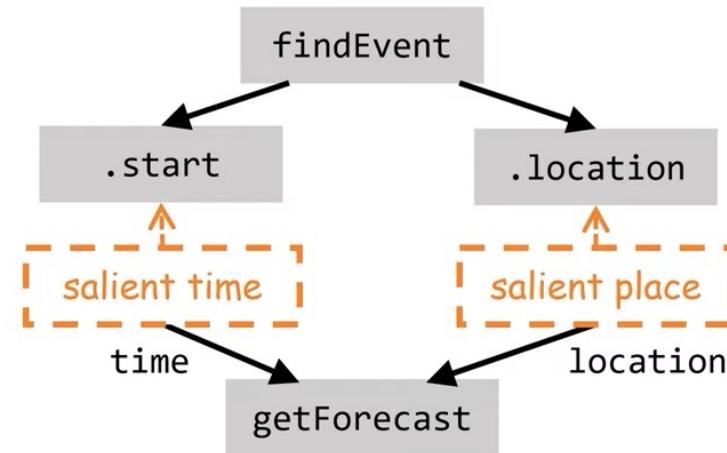
The forecast in Cambridge this afternoon is sunny with a high of 76.



Your dinner is at Café Sushi.

What's the weather going to be like?

The forecast there at 7 pm is cloudy with a high of 55.



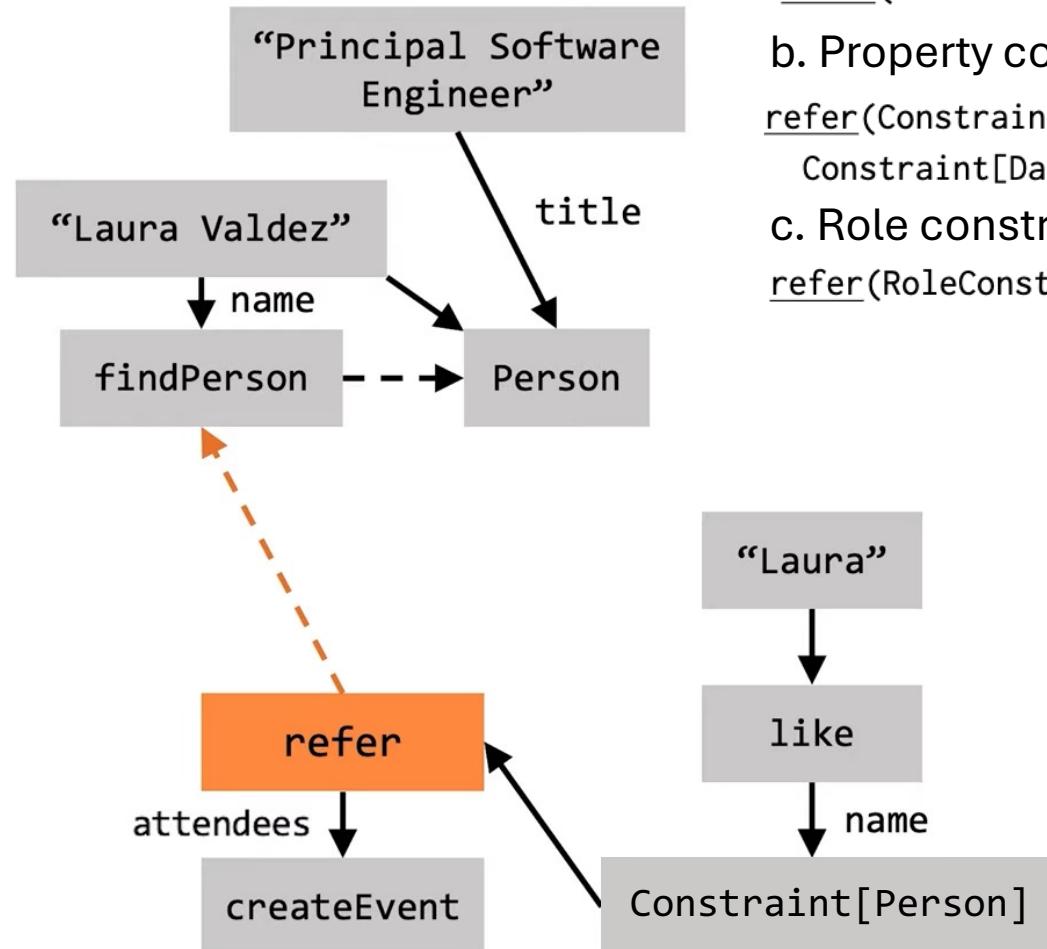
Reference Resolution: Constraints

Most cases are easy: type constraints +
recency + executability
→ 84–98% accuracy!

Is Laura Valdez free today?

Schedule a meeting with Laura

```
createEvent(attendees=  
    refer(Constraint[Person](  
        name=like("Laura"))))
```



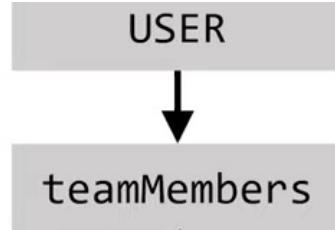
- a. Type constraint
`refer(Constraint[Event]())`
- b. Property constraint
`refer(Constraint[Event](date=Constraint[DateTime](weekday=thurs)))`
- c. Role constraint
`refer(RoleConstraint([date, weekday]))`

Graph Transformation II: Revision

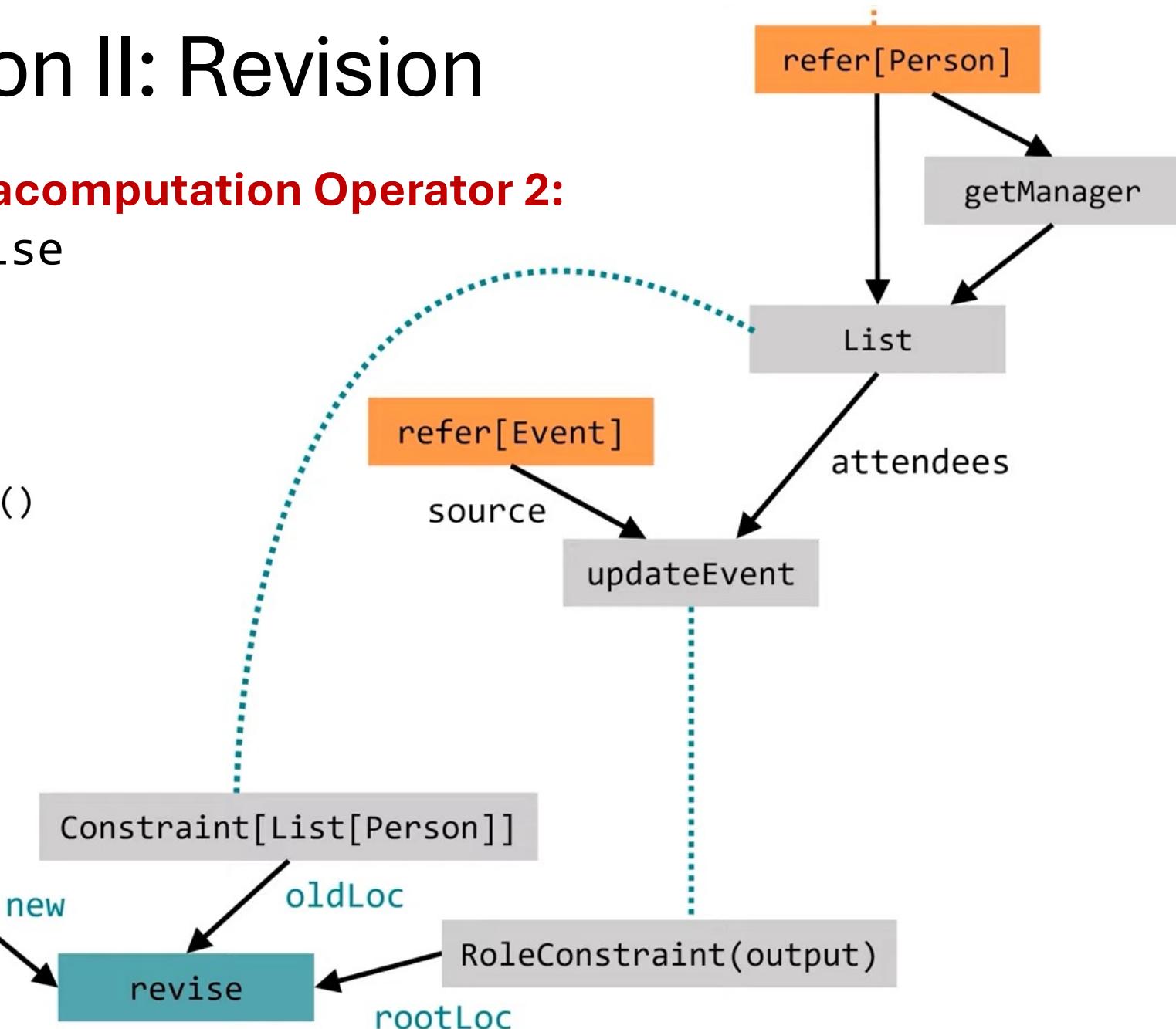
~~Invite him and his manager.~~

Actually, the whole team.

```
revise(  
    rootLoc=RoleConstraint(output),  
    oldLoc=Constraint[List[Person]]()  
    new=teamMembers(USER)  
)
```



Metacomputation Operator 2: Revise



Graph Transformation III: Recovery

User: *Book a meeting for me.*

Agent: *When should the meeting start?*

=> UnderconstrainedException

User: *Who is coming to the planning meeting?*

Agent: *Susan Chu and Susan Brown.*

User: *What is Susan's email?*

=> MultipleMatchesFoundException

User: *When is my first meeting on February 30?*

=> InvalidValueException

Things can go wrong.

User: *Create a meeting.*

`createEvent()`

--> UnderconstrainedException!(name)

Agent: *What should it be called?*

User: *Planning meeting.*

`revise(rootLoc=RoleConstraint(output),
oldLoc=RoleConstraint(name),
new='Planning meeting')`

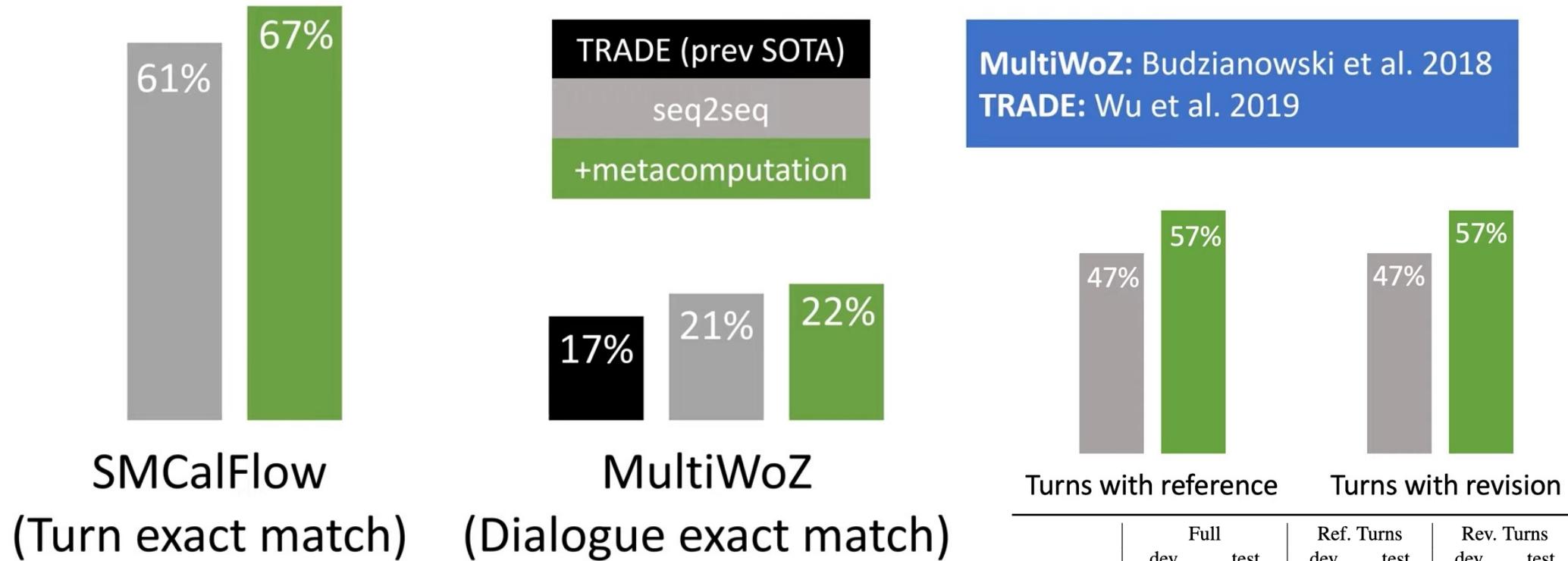
--> UnderconstrainedException!(start)

Agent: *When should it start?*

Revision on dataflow graph!

The SMCalFlow dataset: evaluation

microsoft.github.io/task_oriented_dialogue_as_dataflow_synthesis



	Full		Ref. Turns		Rev. Turns	
	dev	test	dev	test	dev	test
# of Turns	13,499	21,224	3,554	8,965	1,052	3,315
Dataflow	.729	.665	.642	.574	.697	.565
inline	.696	.606	.533	.465	.631	.474

Key Takeaways

- Alignment with High-level User Intent
 - Break complex queries into decomposed operations
 - Simplified Meta-operators: Refer & Revise
- Heuristics from Symbolic Execution
 - Constraint-based Reference & Revision
 - Runtime-exception-guided Recovery

Graphs Meet AI Agents: Taxonomy, Progress, and Future Opportunities

Yuanchen Bei, Weizhi Zhang, Siwen Wang, Weizhi Chen, Sheng Zhou, Hao Chen, Yong Li, Jiajun Bu,
Shirui Pan, *Senior Member, IEEE*, Yizhou Yu, *Fellow, IEEE*, Irwin King, *Fellow, IEEE*,
Fakhri Karray, *Fellow, IEEE*, Philip S. Yu, *Fellow, IEEE*

What if the neural models fail in generating
correct syntax/schema?

Constrained Decoding in Language Models

Constrained Language Models Yield Few-Shot Semantic Parsers

**Richard Shin, Christopher H. Lin, Sam Thomson, Charles Chen,
Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls,
Dan Klein, Jason Eisner, Benjamin Van Durme**

Microsoft Semantic Machines

sminfo@microsoft.com

EMNLP 2021

- Language Models in 2020: GPT-2, GPT-3, T5, BART, etc.
- Better at generating natural language, but not good at formal language

Paraphrasing is easier than Semantic Parsing

- Natural Language Utterance u
who has published the most articles?
- Canonical Utterance c
person that is author of the most number of article
- Meaning Representation (Formal Language) m
 $\text{argmax}(\text{type}. \text{person}, \mathbf{R}(\lambda x. \text{count}(\text{type}. \text{article} \sqcap \text{author}.x)))$

$$u \mapsto c \mapsto m$$

Pretrained Equiv.
LM SCFG

Synchronous Context-Free Grammar (SCFG)

Rules in SCFG apply to two languages at the same time, capturing grammatical structures that are each other's translations.

	[glue]	
(G1)	ENTITYNP[x]	$\rightarrow \text{NP}[x]$
(G2)	TYPENP[x]	$\rightarrow \text{NP}[\text{type}.x]$
(G3)	NP[x] CP[f] (and CP[g])*	$\rightarrow \text{NP}[x \sqcap f \sqcap g]$
	[simple]	
(R0)	that VP[x]	$\rightarrow \text{CP}[x]$
(R1)	whose RELNP[r] CMP[c] NP[y] is is not is smaller than is larger than is at least is at most	$\rightarrow \text{CP}[r.c.y]$ $\rightarrow \text{CMP}[= \neq < > \leq \geq]$
(R2)	that (not)? VP/NP[r] NP[y]	$\rightarrow \text{CP}[(\neg)r.y]$
(R3)	that is (not)? RELNP[r] of NP[y]	$\rightarrow \text{CP}[(\neg)\mathbf{R}(r).y]$
(R4)	that NP[y] (not)? VP/NP[r]	$\rightarrow \text{CP}[(\neg)(\mathbf{R}(r).y)]$
	[counting]	
(C1)	that has CNT[c] RELNP[r]	$\rightarrow \text{CP}[\mathbf{R}(\lambda x.\text{count}(\mathbf{R}(r).x)).c]$
(C2)	that VP/NP[r] CNT[c] NP[y]	$\rightarrow \text{CP}[\mathbf{R}(\lambda x.\text{count}(y \sqcap \mathbf{R}(r).x)).c]$
(C3)	that is RELNP[r] of CNT[c] NP[y]	$\rightarrow \text{CP}[\mathbf{R}(\lambda x.\text{count}(y \sqcap r.x)).c]$
(C4)	that CNT[c] NP[y] VP/NP[r] (less than more than) NUM[n]	$\rightarrow \text{CP}[\mathbf{R}(\lambda x.\text{count}(y \sqcap r.x)).c]$ $\rightarrow \text{CNT}[(< . >.)n]$
	[superlatives]	
(S0)	NP[x] that has the largest RELNP[r]	$\rightarrow \text{NP}[\arg\max(x, r)]$
(S1)	NP[x] that has the most number of RELNP[r]	$\rightarrow \text{NP}[\arg\max(x, \mathbf{R}(\lambda y.\text{count}(\mathbf{R}(r).y)))]$
(S2)	NP[x] that VP/NP[r] the most number of NP[y]	$\rightarrow \text{NP}[\arg\max(x, \mathbf{R}(\lambda y.\text{count}(\mathbf{R}(r).y)))]$
(S3)	NP[x] that is RELNP[r] of the most number of NP[y]	$\rightarrow \text{NP}[\arg\max(x, \mathbf{R}(\lambda z.\text{count}(y \sqcap r.z)))]$
(S4)	NP[x] that the most number of NP[y] VP/NP[r]	$\rightarrow \text{NP}[\arg\max(x, \mathbf{R}(\lambda z.\text{count}(y \sqcap r.z)))]$
	[transformation]	
(T1)	RELNP[r] of NP[y]	$\rightarrow \text{NP}[\mathbf{R}(r).y]$
(T2)	RELNP ₀ [h] CP[f] (and CP[g])*	$\rightarrow \text{NP}[\mathbf{R}(h).(f \sqcap g)]$
(T3)	RELNP[r] of RELNP ₀ [h] NP[x] CP[f] (and CP[g])*	$\rightarrow \text{NP}[\mathbf{R}(r).(h.x \sqcap f \sqcap g)]$
(T4)	NP[x] or NP[y]	$\rightarrow \text{NP}[x \sqcup y]$
	[aggregation]	
(A1)	number of NP[x]	$\rightarrow \text{NP}[\text{count}(x)]$
(A2)	total average RELNP[r] of NP[x]	$\rightarrow \text{NP}[\text{sum} \text{average}(x, r)]$

Example

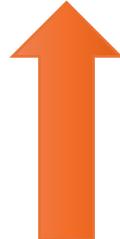
x: "Show me meetings after the weekly standup day"	
c: "meeting whose date is at least date of weekly standup"	
z: type.meeting \sqcap date. $>$ $\mathbf{R}(\text{date}).\text{weeklyStandup}$	
x: "Select the brick that is to the furthest left."	
c: "block that the most number of block is right of"	
z: argmax(type.block, $\mathbf{R}(\lambda x.\text{count}(\mathbf{R}(\text{right}).x)))$	
x: "Housing that is 800 square feet or bigger?"	
c: "housing unit whose size is at least 800 square feet"	
z: type.housingUnit \sqcap area. $> .800$	
x: "What restaurant can you eat lunch outside at?"	
c: "restaurant that has outdoor seating and that serves lunch"	
z: type.restaurant \sqcap hasOutdoorSeating \sqcap serveslunch	
x: "Who has co-authored articles with Efron?"	
c: "person that is author of article whose author is efron"	
z: type.person \sqcap $\mathbf{R}(\text{author}).(\text{type.article} \sqcap \text{author.efron})$	
x: "When did alice start attending brown university?"	
c: "start date of student alice whose university is brown university"	
z: $\mathbf{R}(\text{date}).(\text{student.Alice} \sqcap \text{university.Brown})$	
x: "How many fouls were played by Kobe Bryant in 2004?"	
c: "number of fouls (over a season) of player kobe bryant whose season is 2004"	
z: count($\mathbf{R}(\text{fouls}).(\text{player.KobeBryant} \sqcap \text{season.2004})$)	

Canonical Utterance \Leftrightarrow Formal Logical Form

Paraphrasing is easier than Semantic Parsing

By Prompting LMs:

Few-shot
Examples



Let's translate what a human user says into what a computer might say.

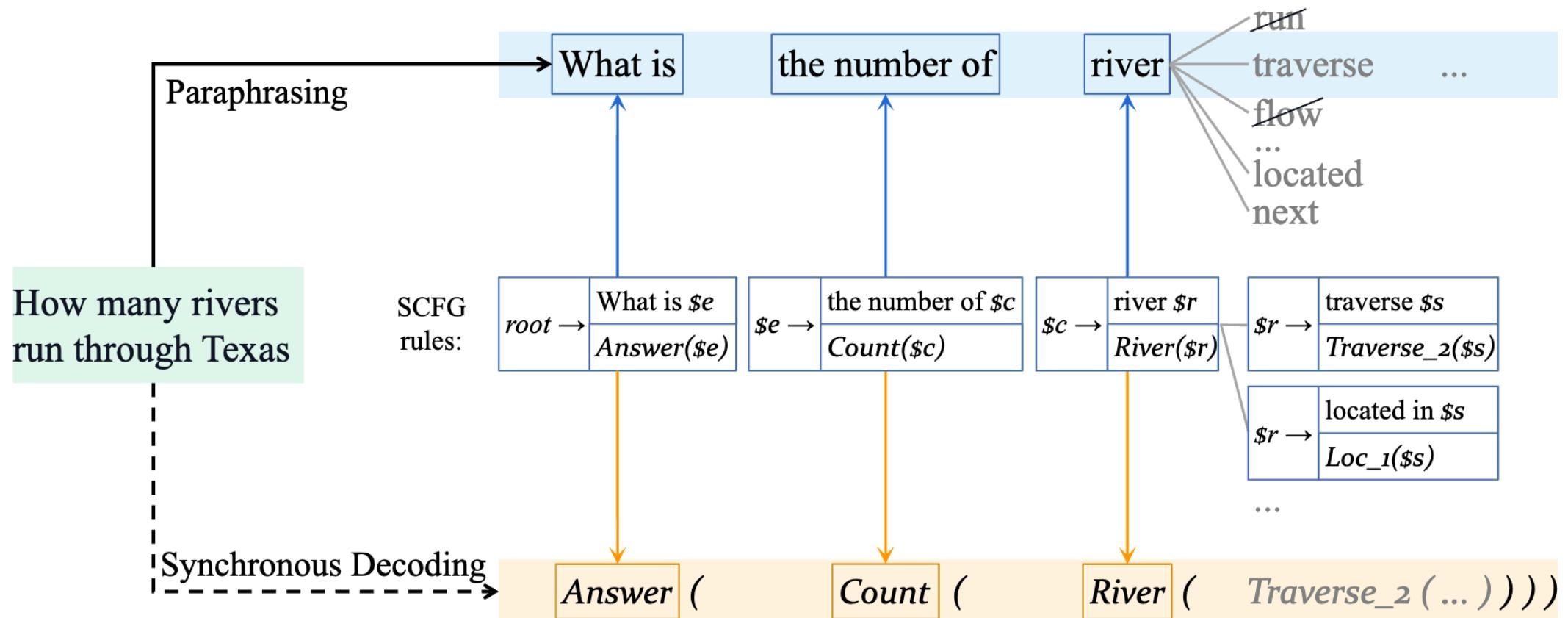
- Human:** when is the weekly standup
- Computer:** start time of weekly standup
- Human:** what date is the weekly standup
- Computer:** date of weekly standup
- ...
- Human:** how long is the weekly standup
- Computer:**

$$u \mapsto c \mapsto m$$

Pretrained
LM

Equiv.
SCFG

Grammar-constrained Decoding



Results

Overnight:

Ablation:

- a) Canonical > Meaning
- b) Constrained >> Unconstrained

Model	Train n	Basketball	Blocks	Calendar	Housing	Publications	Recipes	Restaurants	Social
GPT-3 Constrained Canonical	200	0.80*	0.62*	0.82*	0.71*	0.79*	0.84*	0.89*	0.72*
GPT-3 Constrained Meaning	200	0.68*	0.53*	0.68*	0.58*	0.63*	0.75*	0.78*	0.63*
GPT-3 Unconstrained Canonical	200	0.76*	0.46*	0.68*	0.56*	0.58*	0.74*	0.74*	0.55*
GPT-3 Unconstrained Meaning	200	0.56*	0.39*	0.50*	0.42*	0.46*	0.66*	0.58*	0.48*
GPT-3 Constrained Canonical	20	0.80*	0.55*	0.67*	0.68*	0.81*	0.60*	0.76*	0.67*
BART ^f Constrained Canonical	200	0.85	0.58	0.85	0.73	0.76	0.77	0.83	0.73
BART ^f Constrained Meaning	200	0.83	0.56	0.77	0.75	0.79	0.76	0.81	0.69
BART ^f Unconstrained Canonical	200	0.83	0.56	0.80	0.67	0.72	0.75	0.81	0.65
BART ^f Unconstrained Meaning	200	0.82	0.55	0.76	0.71	0.77	0.73	0.80	0.63
Cao et al. (2019)	640–3535	0.880	0.652	0.807	0.767	0.807	0.824	0.840	0.838
BERT-LSTM (Xu et al., 2020)	640–3535	0.875	0.624	0.798	0.704	0.764	0.759	0.828	0.819

Key Takeaways

- Alignment with Neural Model Strengths
 - Language models are better at paraphrasing than code generation
 - Canonical utterance plays as the intermediate representation (IR) to bridge the Semantic Gap
- Symbolic Heuristics

Type-Constrained Code Generation with Language Models

Who *Niels Mündler, Jingxuan He, Hao Wang, Koushik Sen, Dawn Song, Martin Vechev*

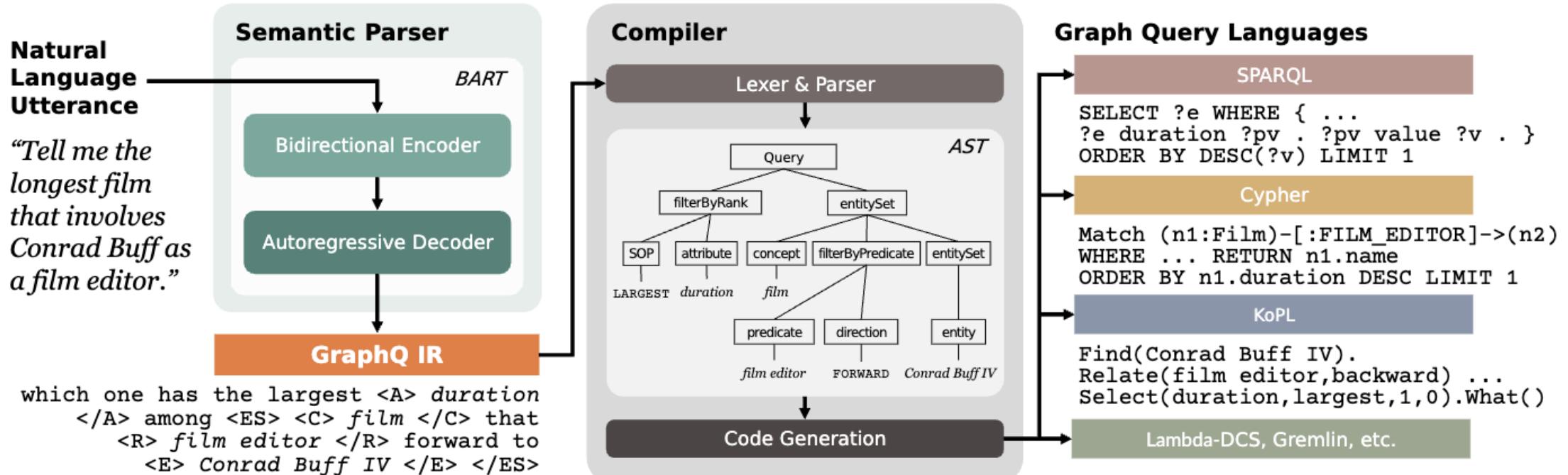
Track [PLDI 2025 PLDI Research Papers](#)

Can we utilize the **Intermediate Representation** to improve
knowledge transferability and **low-resource** semantic parsing?

GraphQ IR: Unifying the Semantic Parsing of Graph Query Languages with One Intermediate Representation

EMNLP 2022

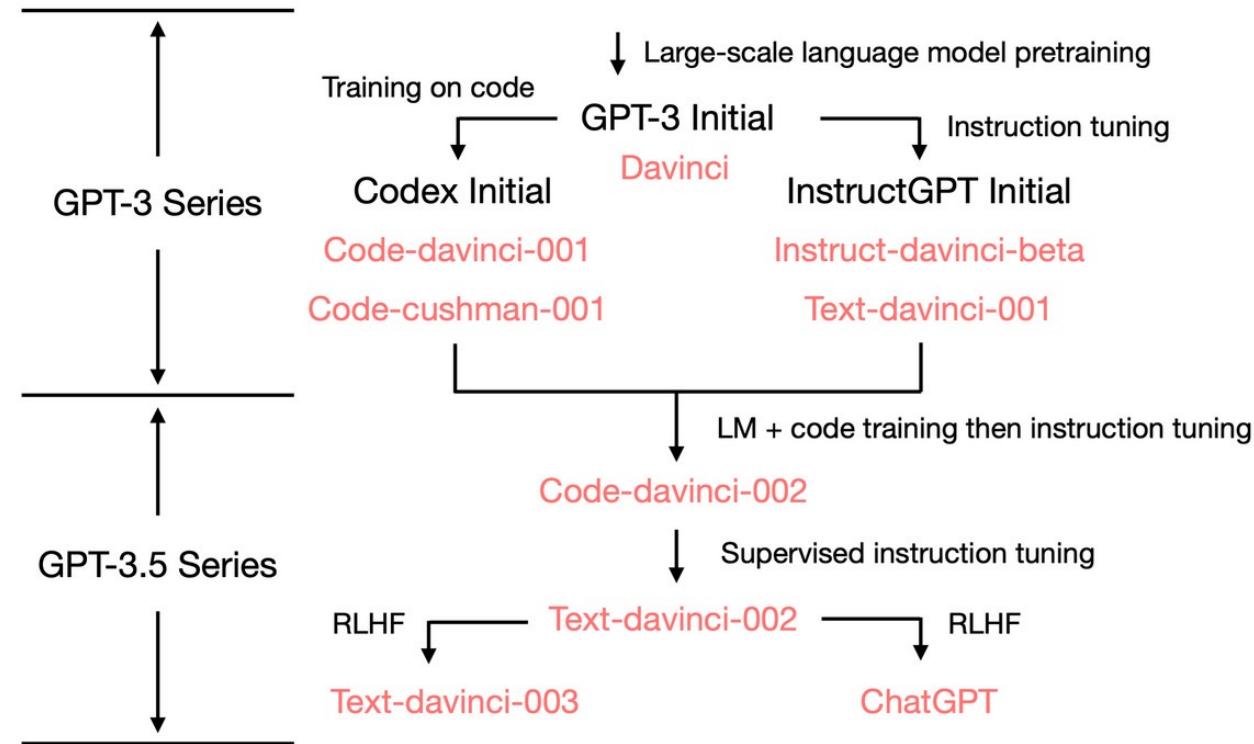
Lunyu Nie¹, Shulin Cao¹, Jiaxin Shi², Jiuding Sun¹,
Qi Tian², Lei Hou¹, Juanzi Li¹, Jidong Zhai¹



Cross-domain training
on SPARQL annotations

	Bas.	Blo.	Cal.	Hou.	Pub.	Rec.	Res.	Soc.	Overall
Baselines									
SPO (Wang et al., 2015)	46.3	41.9	74.4	54.0	59.0	70.8	75.9	48.2	58.8
CrossDomain* (Su and Yan, 2017)	88.2	62.2	82.1	78.8	80.1	86.1	83.7	83.1	80.6
Seq2Action (Chen et al., 2018a)	88.2	61.4	81.5	74.1	80.7	82.9	80.7	82.1	79.0
DUAL (Cao et al., 2019)	84.9	61.2	78.6	67.2	78.3	80.6	78.9	81.3	76.4
2-stage DUAL* (Cao et al., 2020)	87.2	65.7	80.4	75.7	80.1	86.1	82.8	82.7	80.1
Our Approach									
GraphQ IR	88.2	64.7	78.6	72.0	77.6	83.3	84.9	81.6	79.5
GraphQ IR*	88.2	65.4	81.6	81.5	82.6	92.9	89.8	84.1	82.1

From Semantic Parsing to Agent Tool Use



As pretraining scales up & on code corpus, today's LLMs can handle code generation very well

From Semantic Parsing to Agent Tool Use

Toolformer: Language Models Can Teach Themselves to Use Tools

Timo Schick Jane Dwivedi-Yu Roberto Dessì[†] Roberta Raileanu

Maria Lomeli Luke Zettlemoyer Nicola Cancedda Thomas Scialom

Meta AI Research [†]Universitat Pompeu Fabra

NeurIPS 2023
(Post-ChatGPT)

ChatGPT Limitations

Searching the web 

The current President of the United States is Donald J. Trump,

Who is the current President of the United States?



As an AI language model, I don't have real-time data or internet access, and my knowledge was last updated in September 2021. [...]

What day of the week is it today?

Today is Monday, September 1, 2025. 



As an AI language model, I do not have the capability to access current time or date information. [...]

What is the result of $3435 * 235 / 9$?

Let's carefully calculate step by step:



This is approximately 89,937.22.

True answer: 89691.66

.....

 Final Answer:
89,691 remainder 6
or $\approx 89,691.67$ in decimal.

Toolformer Tools/APIs

What other name
is Pittsburgh
known by?



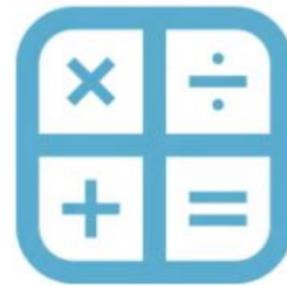
The Steel City

War memorial
Flodden



[...] was created in
memory of the
Battle of Flodden.

$$3435 * \\ 235 / 9$$



89691.67

∅



Thursday,
March 10,
2019

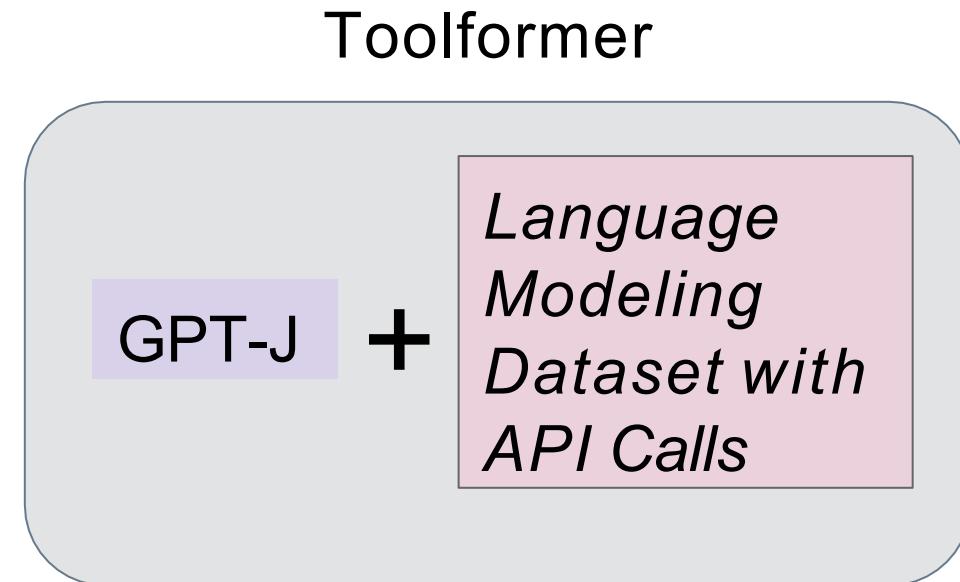
Os Melhores
Escolas em
Jersey



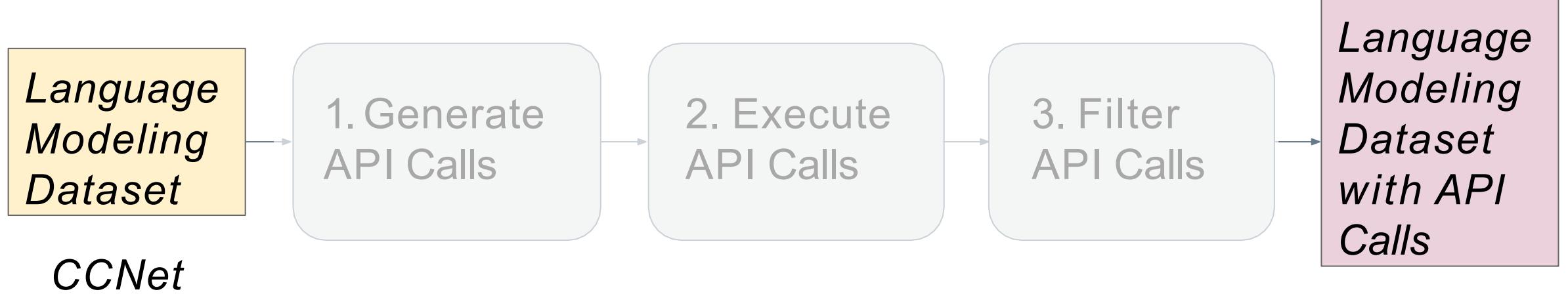
The Best
Schools in
Jersey

Steps to Create Toolformer

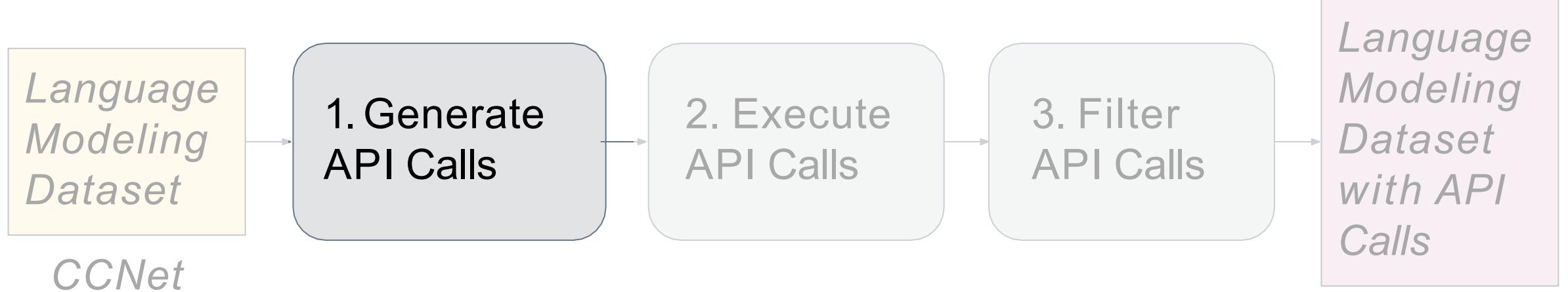
1. Creating a new training dataset augmented with API calls
2. Finetune GPT-J using this new dataset



Creating the Training Dataset



Creating the Training Dataset



Prompting the Model to Generate API Calls

Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text.

You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

Input: Joe Biden was born in Scranton, Pennsylvania.

Output: Joe Biden was born in [QA("Where was Joe Biden born?")]
Scranton, [QA("In which state is Scranton?")] Pennsylvania.

Input: \${input}

Output:

Examples of Generated API Calls

Your task is to add calls to a QA API to a piece of text [...]

Input: Pittsburgh is known as the Steel City.

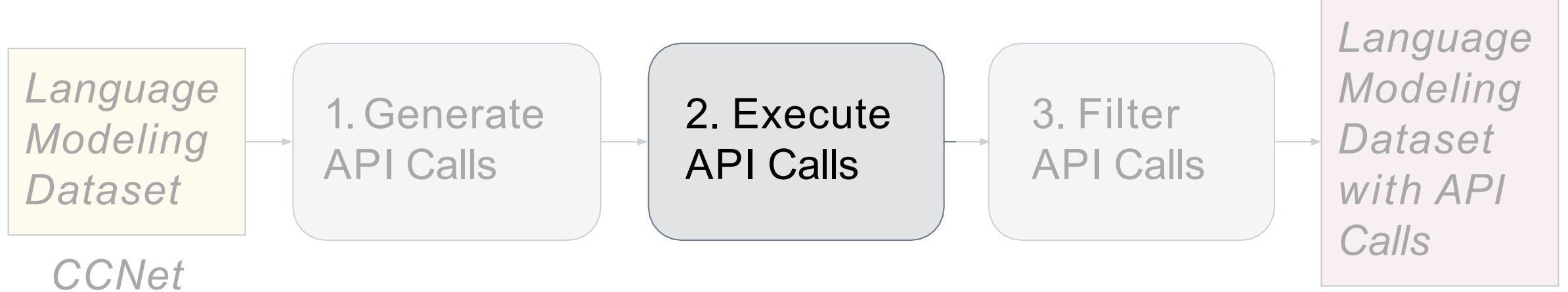
Output:

Pittsburgh is known as [QA("In which state is Pittsburgh?")] the Steel City.

Pittsburgh is known as [QA("What other name is Pittsburgh known by?")]
the Steel City.

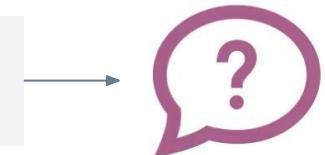
Pittsburgh is known as [QA("What is the second city in Pennsylvania?")]
the Steel City.

Creating the Training Dataset



Execute the API Calls

In which state is Pittsburgh?



Pennsylvania

What other name is Pittsburgh known by?



The Steel City

What is the second city in Pennsylvania?



Pittsburgh

Execute the API Calls

In which state is Pittsburgh?



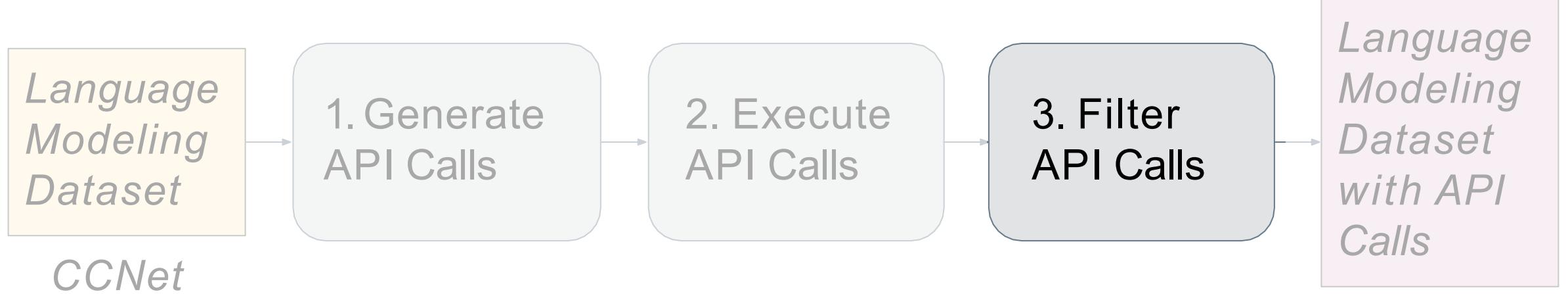
Pennsylvania

Pittsburgh is known as [QA("In which state is Pittsburgh?")]
the Steel City.



Pittsburgh is known as [QA("In which state is Pittsburgh?") → Pennsylvania]
the Steel City.

Creating the Training Dataset



Filter the API Calls using Model-based Perplexity

$$L_{\bullet}(PREFIX) = -\log p(\text{the Steel City.} \mid PREFIX)$$

A. No API Call

$$L_A(Pittsburgh \text{ is known as}) = 2.5$$

B. Non-executed
API Call

$$L_B(Pittsburgh \text{ is known as } [QA("What other name is Pittsburgh known by?") \rightarrow ?]) = 2.1$$

C. Executed
API Call

$$L_C(Pittsburgh \text{ is known as } [QA("What other name is Pittsburgh known by?") \rightarrow Steel \text{ City}]) = 0.8$$

$$\text{Usefulness} = \min(L_A, L_B) - L_C = \min(2.5, 2.1) - 0.8 = 1.3$$

Usefulness Examples

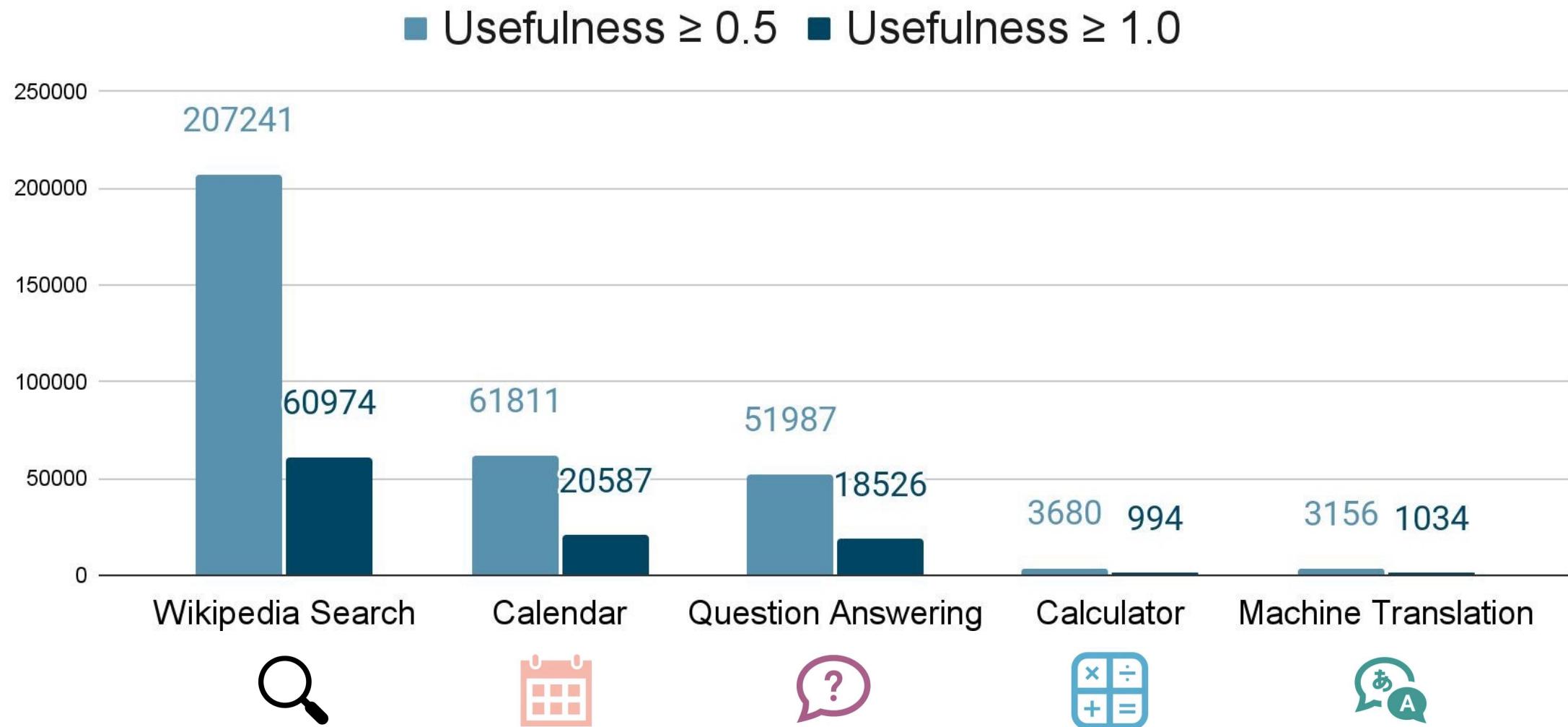
2.11

The WL will be open on Friday, [Calendar() → Today is Thursday, March 9, 2017.] March 10, and Sunday, March 19 for regular hours.

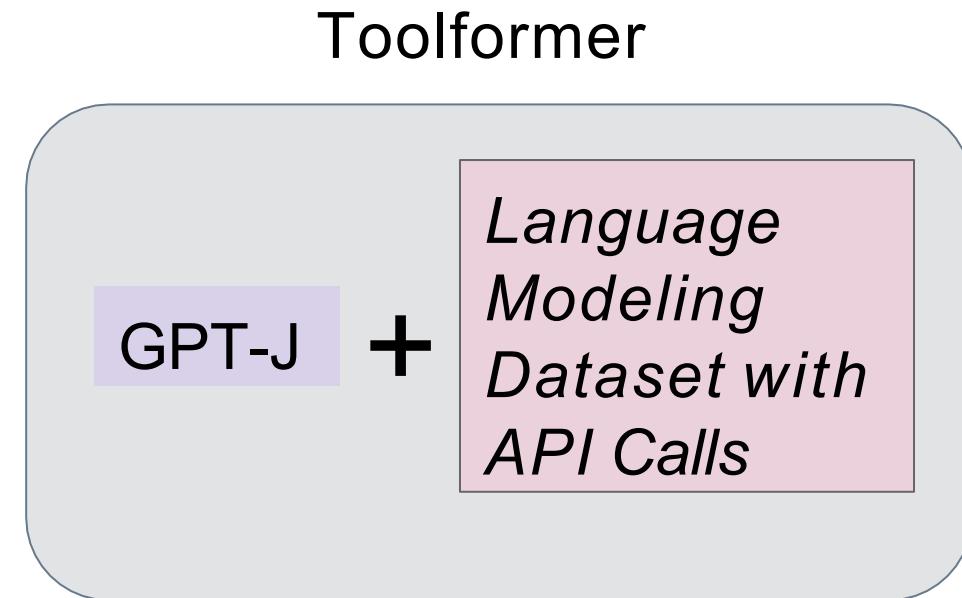
-0.02

85 patients (23%) were hospitalised alive and admitted to a hospital ward. Of them, [Calculator($85 / 23 \rightarrow 3.70$) 65% had a cardiac aetiology.

Number of Tool-Usage Samples After Filtering



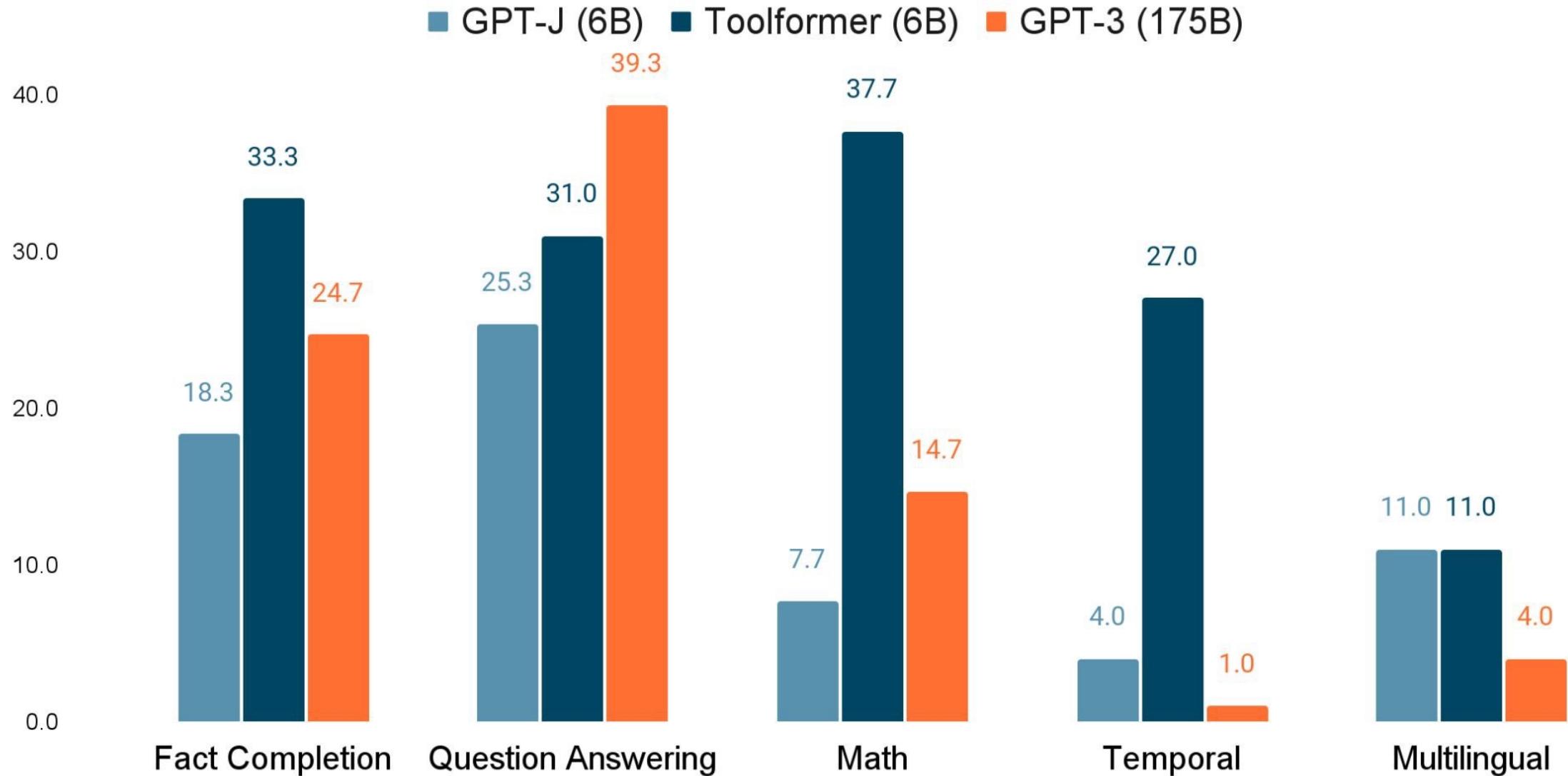
Fine-tuning Toolformer



Evaluation Tasks

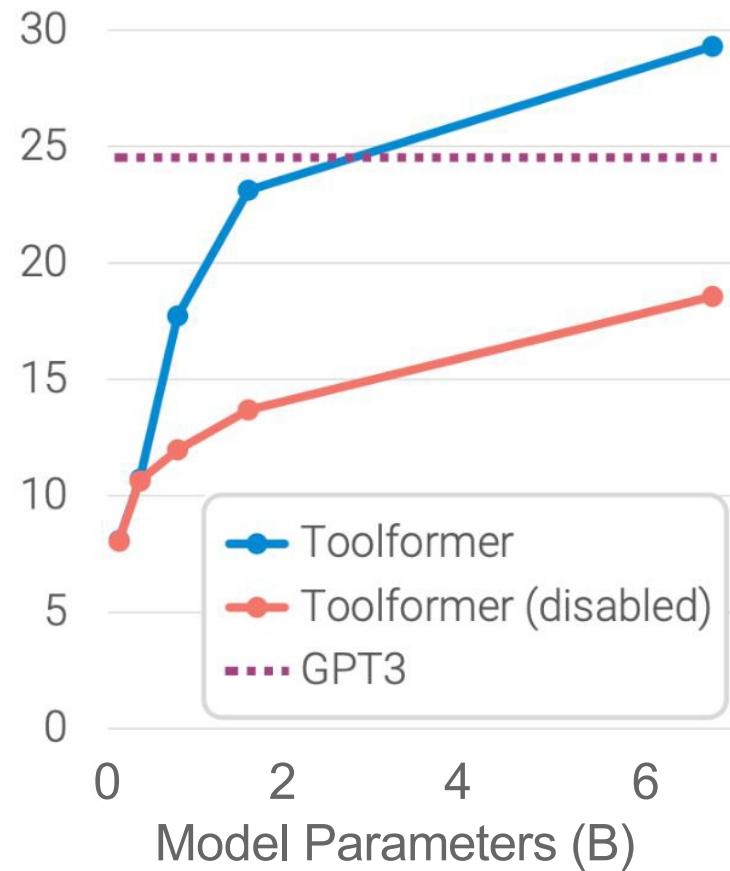
1. Fact Completion/Question Answering
 - a. “The theory of relativity was developed by _____”
 - b. “In Greek Mythology, who is the goddess of spring growth?”
2. Math Computations
3. Multilingual Questions
 - a. Context is given in English, question is multilingual.
4. Temporal Questions
 - a. How many days is it until Christmas?

Results

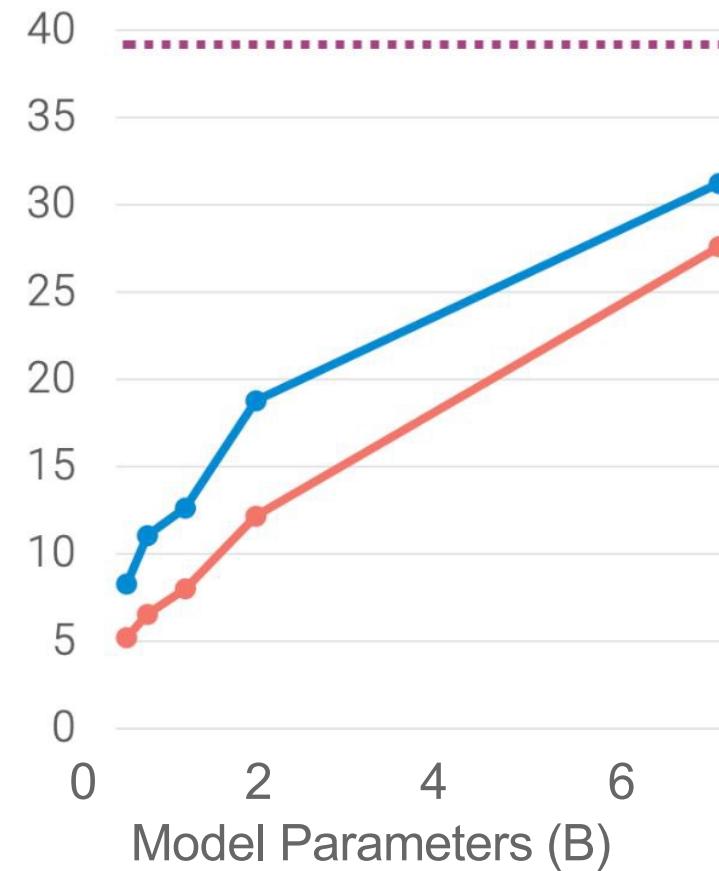


Can small models effectively use tools?

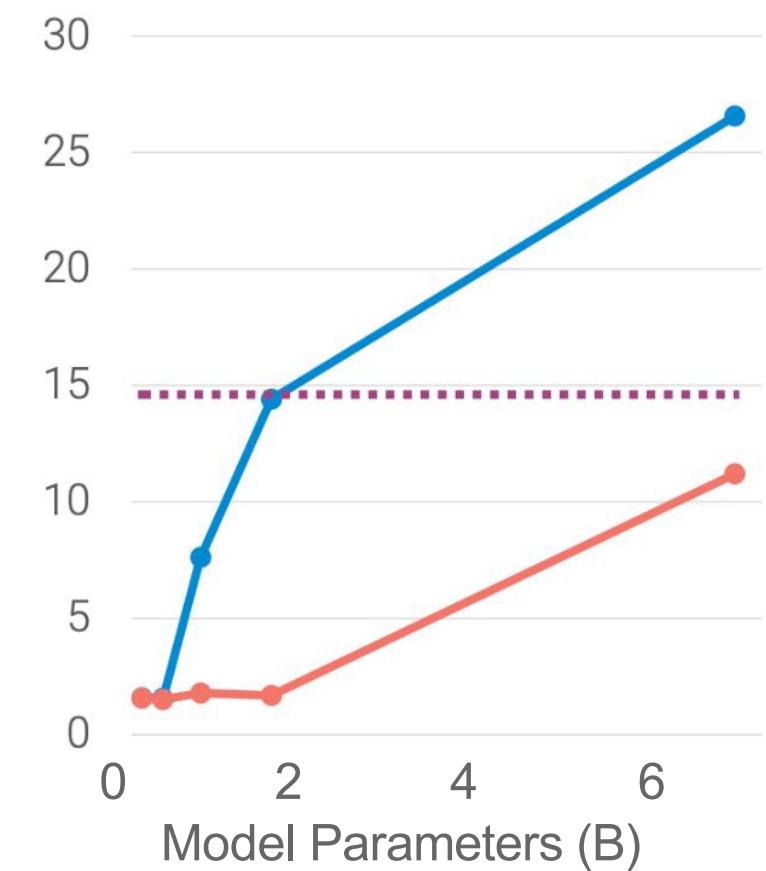
Fact Completion Benchmarks



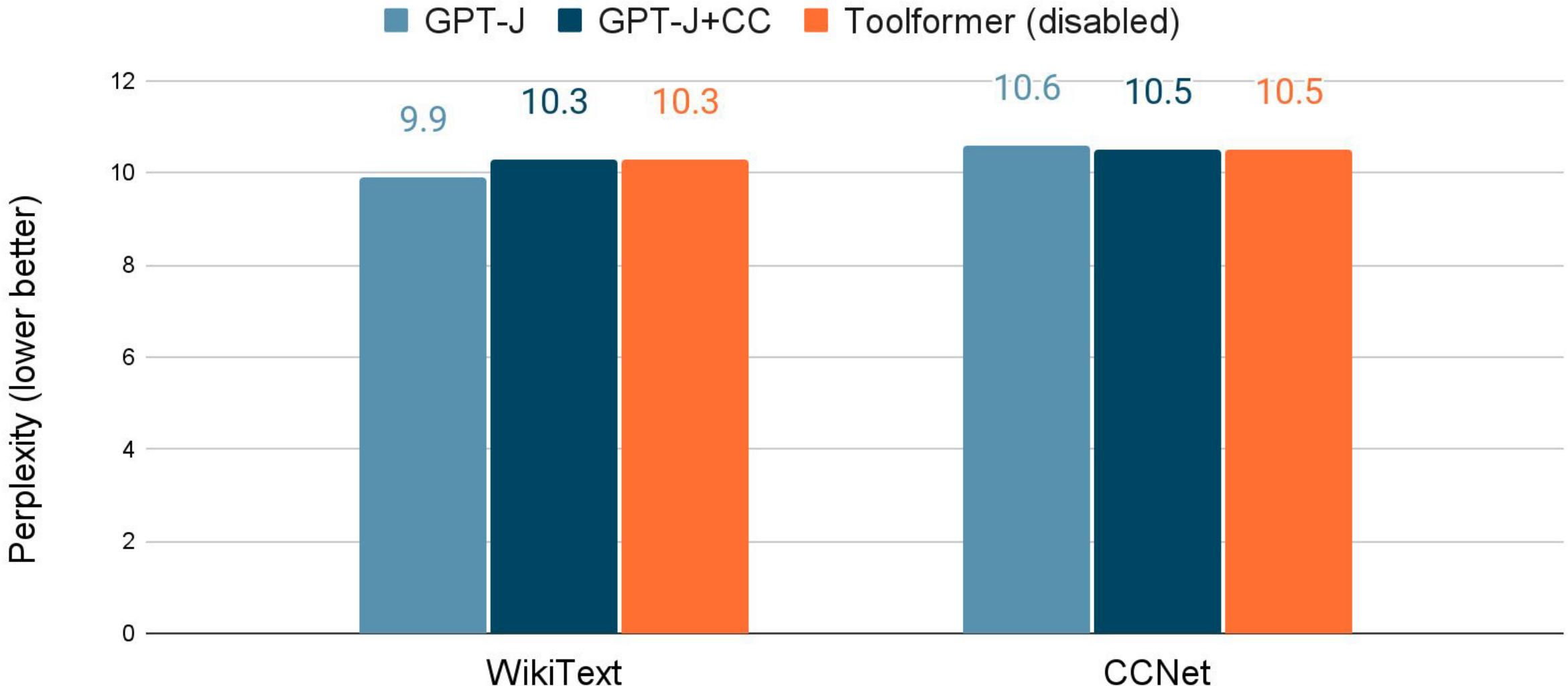
QA Benchmarks



Math Benchmarks

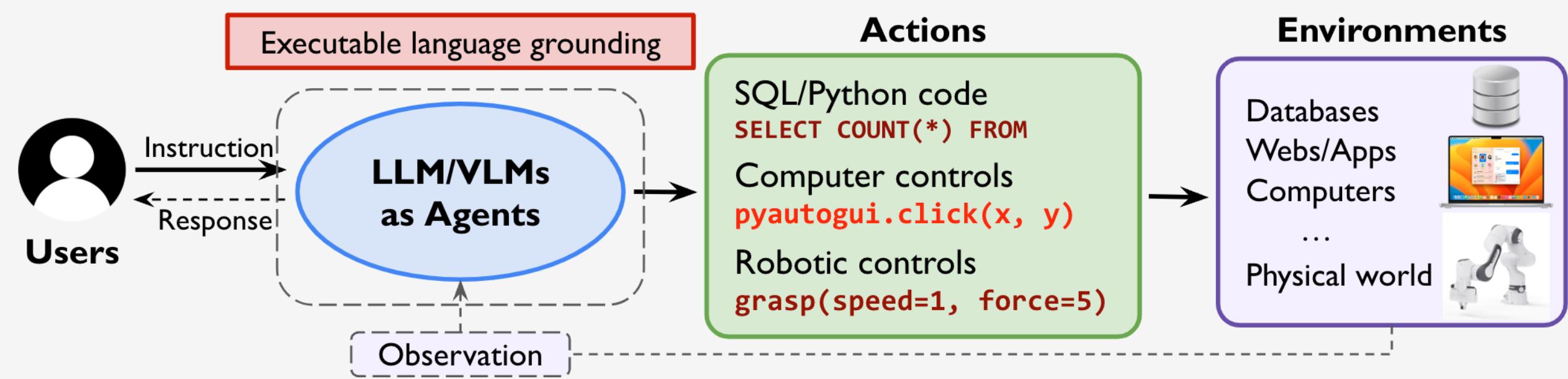


Is Toolformer still a good language model?



What Old-School Semantic Parsing can teach us in the Age of LLMs

- Symbolic methods
 - Interpretable
 - Verifiable
 - Structured Domain Knowledge
 - Data efficient
- Neural methods
 - Scalable
 - Flexible
 - Handles messy data
 - Easy to get started (need data!)
- Symbolic heuristics
 - Execution-guided
 - Grammar Constrained-decoding
- Better neural networks
 - Advanced model architecture
 - Pretraining
- Neurosymbolic combinations
 - Data Augmentation with SCFG
 - Intermediate Representation



Discussion

- What key challenges from early semantic parsing (ambiguity, compositionality, grounding, etc.) still matter for LLMs today?
- How can structured meaning representations complement LLMs?
- Could semantic parsing ideas help reduce hallucinations or improve reasoning?
- How might symbolic constraints or grammar-driven parsing improve the reliability of LLM outputs?
- What role could hybrid models (symbolic + neural) play in bridging old and new paradigms?

Reference

- Andreas, Jacob, et al. "Task-oriented dialogue as dataflow synthesis." *Transactions of the Association for Computational Linguistics* 8 (2020): 556-571.
- <https://www.microsoft.com/en-us/research/publication/task-oriented-dialogue-as-dataflow-synthesis/>
- Bei, Yuanchen, et al. "Graphs Meet AI Agents: Taxonomy, Progress, and Future Opportunities." *arXiv preprint arXiv:2506.18019* (2025).
- Shin, Richard, et al. "Constrained language models yield few-shot semantic parsers." *arXiv preprint arXiv:2104.08768* (2021).
- Wu, Shan, et al. "From paraphrasing to semantic parsing: Unsupervised semantic parsing via synchronous semantic decoding." *arXiv preprint arXiv:2106.06228* (2021).
- Mündler, Niels, et al. "Type-Constrained Code Generation with Language Models." *Proceedings of the ACM on Programming Languages* 9.PLDI (2025): 601-626.
- Nie, Lunyu, et al. "GraphQ IR: Unifying the semantic parsing of graph query languages with one intermediate representation." *arXiv preprint arXiv:2205.12078* (2022).
- Fu, Yao, Hao Peng, and Tushar Khot. "How does gpt obtain its ability? tracing emergent abilities of language models to their sources." *Yao Fu's Notion* (2022).
- Schick, Timo, et al. "Toolformer: Language models can teach themselves to use tools." *Advances in Neural Information Processing Systems* 36 (2023): 68539-68551.
- <https://neurips.cc/media/neurips-2023/Slides/71288.pdf>