

Stanford CS224v Course

Conversational Virtual Assistants with Deep Learning

Lecture 3

Knowledge Assistants Grounded on Structured Data on Small Domains

Monica Lam & Shicheng Liu

Let's See What Chat-GPT can do

WE

Hey! Can you suggest a vegan place in Palo Alto that is good for brunch?



Of course! One popular vegan-friendly place in Palo Alto for brunch is "Veggie Grill." It's a fast-casual restaurant that offers a variety of plant-based options, including breakfast items. Keep in mind that restaurant information can change, so I recommend checking their current menu and reviews before heading over. Enjoy your brunch!



Permanently closed

Veggie Grill

4.3 ★★★★★ 875 Google reviews

\$\$ · Vegan restaurant

This Lecture

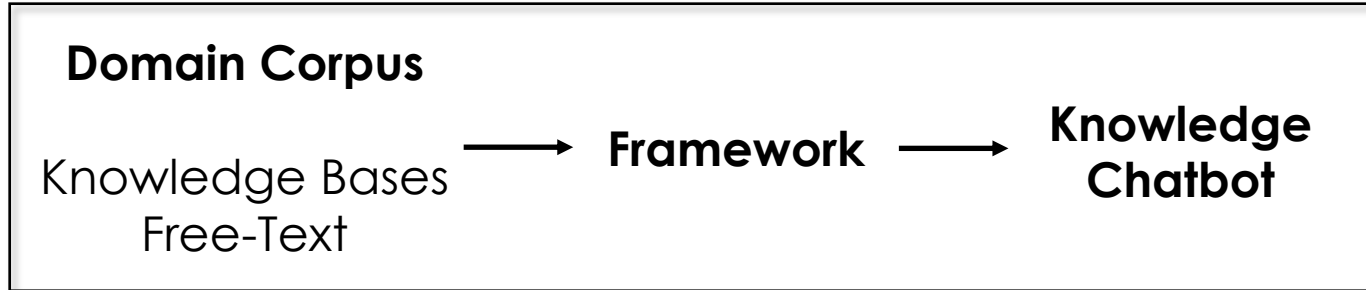
Based on an ongoing project: Yelpbot

- Goal: A factual conversational chatbot for consumers
- Drove the development of a general framework
 - Testing on Amazon products
- Status: *Preliminary* evaluation is promising

Why we choose this topic for Lecture 2?

- Experience / general framework: useful for your project

Lecture Goal



- Why knowledge bases?
- Why formal representation for NL (SQL)?
- Semantic parsing: Translates natural into formal language
- Combining databases and free-text
 - A new formal language:
SUQL (Structured & Unstructured Query Language)

Public Live Knowledge Bases



- A community-driven knowledge graph
 - 100M items: each Wikipedia page has an item in Wikidata
 - 10K properties (3K relevant), 12B facts, 22K contributors
 - Represented in a semantic web, retrieved with SPARQL (Property, ID, value)

Public Knowledge Bases: Schema.org

- The web has a schema: **Schema.org**
 - Structured data to mark up web pages
 - Mainly used by search engines
 - It covers many domains, including ebay, restaurants, hotels, people, recipes, products, news ...
- Core: 792 types, 1447 properties
- + extensions

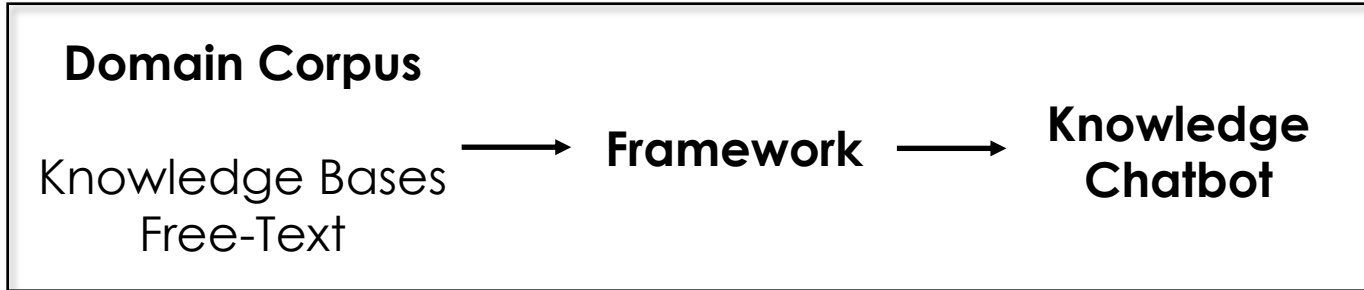
```
<script type="application/ld+json">
{
  @type: "restaurant",
  name: "The French Laundry",
  servesCuisine: "French",
  aggregateRating: {
    @type: "AggregateRating",
    reviewCount: 2527,
    ratingValue: 4.5
  }
  ...
}
```

Schema.org markup on Yelp

Private Knowledge Bases

- Every company is a data company [Forbes 2018]
 - Products: Retail (all of ebay), flights, restaurants (menu, opening hours)
songs, music, books
 - Finance: stocks
 - Every organization: People (employees, customers, students, patients)
Finance
- Personal information: Calendar, emails
- Challenges in data acquisition
 - Proprietary or personal information
 - User privacy
 - Recent dataset: BIRD dataset of many SQL databases [Li et al.]

Lecture Goal



- Why knowledge bases?
- **Why formal representation for NL (SQL)?**
- Semantic parsing: Translates natural into formal language
- Combining databases and free-text
 - A new formal language:
SUQL (Structured & Unstructured Query Language)

Query Languages



- A **FEW** common query languages:
 - Databases (SQL): Tables with a fixed schema – this lecture
 - Knowledge bases (SPARQL, New4j):
Graphs with nodes (entities), edges (properties) Future Topic
- Domains are defined by database **schemas**

Power of Query Languages

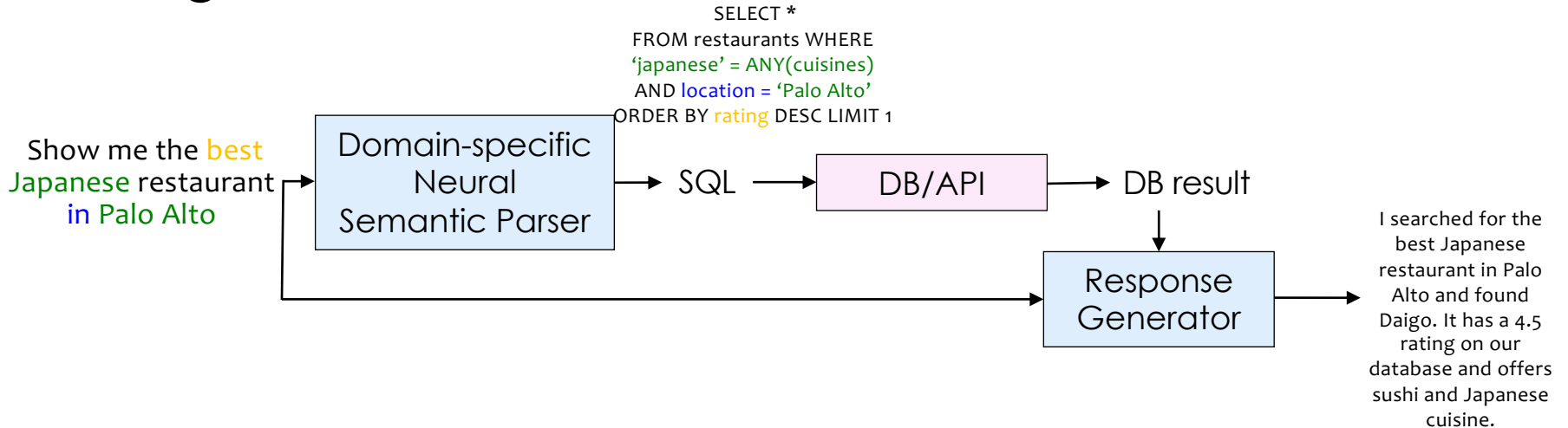


- Domain agnostic
- All queries of any domain are compositions of a FEW relational algebra operations
 - Basic: Selection, Projection, Cartesian product (Join), Union, Set Difference
 - Extended: Sort, Aggregate Operators (Sum, Max, Avg, ...)
- Expressive, succinct, well-defined

Amazing
CS Idea

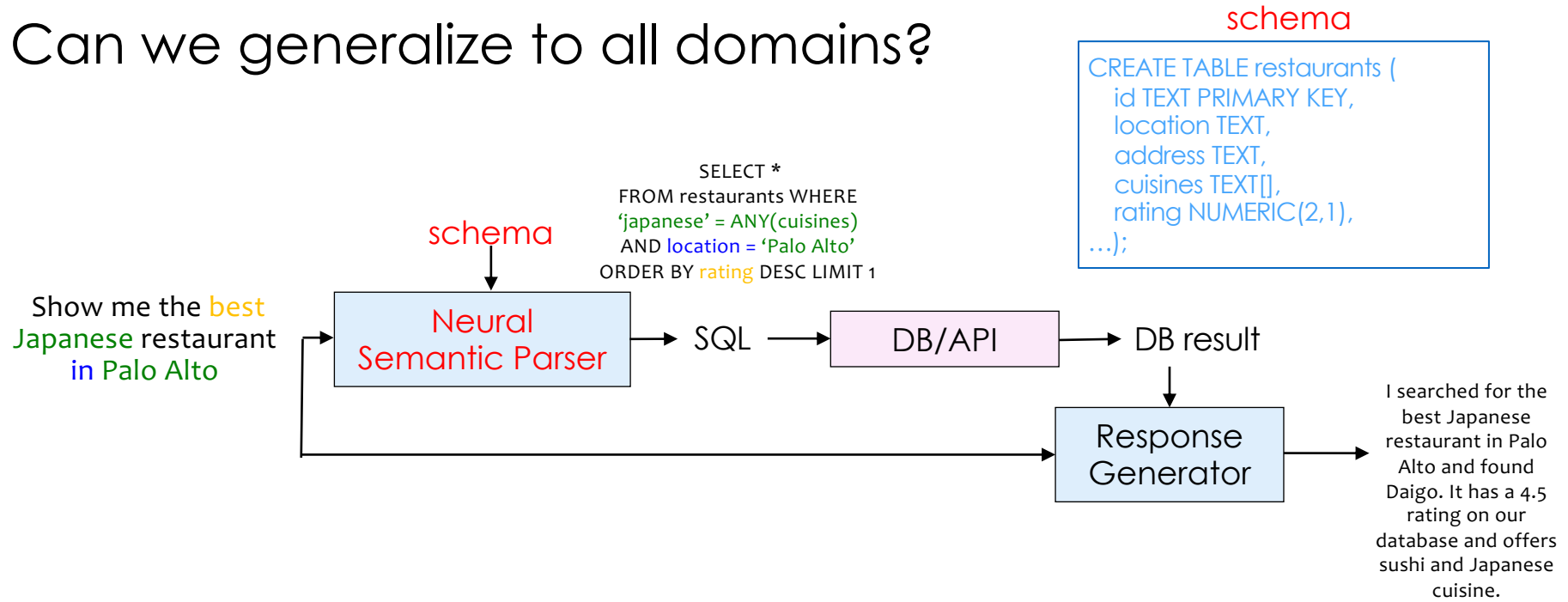
Semantic Parsing: NL \rightarrow Formal Query

For a given domain:

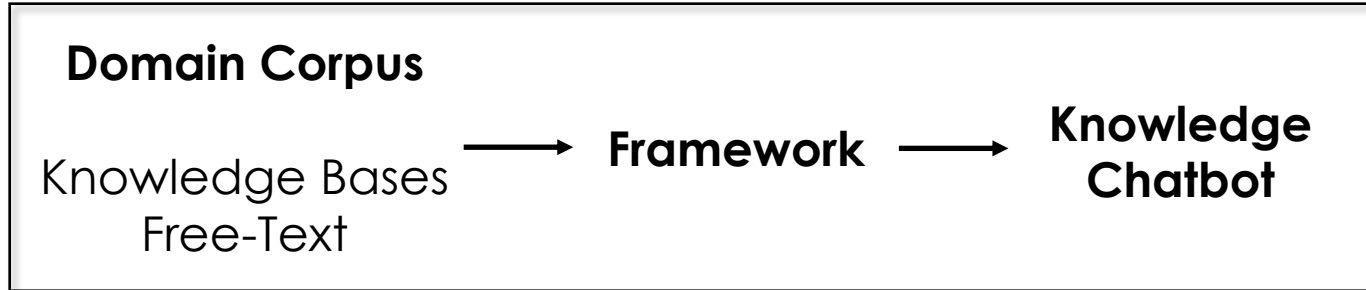


Semantic Parsing: NL \rightarrow Formal Query

Can we generalize to all domains?



Lecture Goal



- Why knowledge bases?
- **Why formal representation for NL (SQL)?**
- Semantic parsing: Translates natural into formal language
- Combining databases and free-text
 - Need a new formal language: SUQL

SQL Queries Factor Out Domain Knowledge

- Databases in many domains
 - Public knowledge: Wikidata, Schema.org
 - Products: Retail, flights, restaurants, songs, music, books
 - Finance: stocks
 - Every organization:
People (employees, customers, students, patients), finance
 - Personal information: Calendar, emails
- Queries: info retrieve, search, compute (tallest, average, ...)
- **Natural conversations: almost every user turn is a query!**

WHERE clause: filter

```
CREATE TABLE restaurants (  
  id TEXT PRIMARY KEY,  
  location TEXT,  
  address TEXT,  
  cuisines TEXT[],  
  rating NUMERIC(2,1),  
  ...);
```

```
SELECT * FROM  
table [WHERE filter];
```

```
SELECT * FROM restaurants  
WHERE location == "Palo Alto"  
AND 'chinese' = ANY (cuisines)
```

Show me Chinese restaurants in Palo Alto

WHERE clause: filter

```
CREATE TABLE restaurants (  
  id TEXT PRIMARY KEY,  
  location TEXT,  
  address TEXT,  
  cuisines TEXT[],  
  rating NUMERIC(2,1),  
  ...);
```

```
SELECT * FROM  
table [WHERE filter]?
```

```
SELECT * FROM restaurants  
WHERE location = "Palo Alto"  
AND "Chinese" = ANY(cuisines)  
AND rating >= 4.5
```

Show me Chinese restaurants in Palo Alto
with at least 4.5 stars

SELECT clause: Projection

```
CREATE TABLE restaurants (  
  id TEXT PRIMARY KEY,  
  location TEXT,  
  address TEXT,  
  cuisines TEXT[],  
  rating NUMERIC(2,1),  
  ...);
```

```
SELECT field+ FROM  
table [WHERE filter];
```

```
SELECT address FROM restaurants  
WHERE location = "Palo Alto"  
AND "Chinese" = ANY(cuisines)  
AND rating >= 4.5
```

Show me **the address of Chinese** restaurants **in Palo Alto**
with at least 4.5 stars

Subquery

```
CREATE TABLE restaurants (  
  id TEXT PRIMARY KEY,  
  location TEXT,  
  address TEXT,  
  cuisines TEXT[],  
  rating NUMERIC(2,1),  
  ...);
```

```
CREATE TABLE reviews (  
  id TEXT PRIMARY KEY,  
  restaurant_id TEXT  
  REFERENCES restaurants(id),  
  author TEXT,  
  ...);
```

```
SELECT field+  
FROM table [WHERE filter];
```



Atom filter:
param op value

Subquery filter:
param op
*(SELECT * FROM ...)*

```
SELECT address FROM restaurants  
WHERE location = "Palo Alto"  
AND "Chinese" = ANY(cuisines)  
AND id IN
```

```
(SELECT restaurant_id FROM reviews WHERE author =~ "Bob")
```

Show me the address of Chinese restaurants in Palo Alto
reviewed by Bob

Sorting

```
CREATE TABLE restaurants (  
  id TEXT PRIMARY KEY,  
  location TEXT,  
  address TEXT,  
  cuisines TEXT[],  
  rating NUMERIC(2,1),  
  ...);
```

```
CREATE TABLE reviews (  
  id TEXT PRIMARY KEY,  
  restaurant_id TEXT  
  REFERENCES restaurants(id),  
  author TEXT,  
  ...);
```

```
SELECT field+ FROM table [WHERE filter];  
ORDER BY field DESC/ASC
```

```
SELECT address FROM restaurants  
  WHERE location = "Palo Alto"  
  AND "Chinese" = ANY(cuisines)  
  AND id IN  
  (SELECT restaurant_id FROM reviews WHERE author =~ "Bob")  
ORDER BY rating DESC
```

Show me the address of top-rated Chinese restaurants in Palo Alto
reviewed by Bob

Joins

```
CREATE TABLE restaurants (  
  id TEXT PRIMARY KEY,  
  location TEXT,  
  address TEXT,  
  cuisines TEXT[],  
  rating NUMERIC(2,1),  
  ...);
```

```
CREATE TABLE reviews (  
  id TEXT PRIMARY KEY,  
  restaurant_id TEXT  
  REFERENCES restaurants(id),  
  author TEXT,  
  ...);
```

```
SELECT * FROM  
table join table [WHERE filter]? [on filter]?
```

```
SELECT * FROM restaurants JOIN reviews ON  
  restaurants.id = reviews.restaurant_id  
  WHERE location = "Palo Alto"  
  AND "Chinese" = ANY(cuisines)  
  ORDER BY rating DESC
```

Show me top-rated Chinese restaurants in Palo Alto
with their reviews

Why is Semantic Parsing Hard?

CREATE TABLE Hospital (
 doctor TEXT,
 patient TEXT);

Consider just 1 fact in *Hospital*: “Dr. Smith is Ann’s doctor”

SELECT patient FROM Hospital
 WHERE doctor = ‘Dr. Smith’

SELECT doctor FROM Hospital
 WHERE patient = ‘Ann’

Relation	Part-of-Speech (POS)	Unknown: Ann	Unknown: Dr. Smith
Doctor	Has-a	Who has Dr. Smith as a doctor?	Who does Ann have as a doctor?
	Is-a	Who is Dr. Smith a doctor of ?	Who is a doctor of Ann?
	Active Verb	Whom does Dr. Smith treat?	Who treats Ann?
	Passive Verb	Who is treated by Dr. Smith?	By whom is Ann treated?
Patient	Has-a	Who does Dr. Smith have as a patient?	Who has Ann as a patient?
	Is-a	Who is a patient of Dr. Smith?	Who is Ann a patient of?
	Active Verb	Who consults with Dr. Smith?	With whom does Ann consult?
	Passive Verb	By whom is Dr. Smith consulted?	Who is consulted by Ann?

Many alternatives to say the same thing!

Summary

- The power of composition of SQL primitives
- Compound SQL can be very simple and natural in English
- Many equivalent NL sentences → same query in SQL
- Quiz: How hard is reading an SQL query in English?
a query in SQL → some equivalent NL query
 - Useful for interpretability

Canonical English Templates (Grammar): Covers All Queries

Operation	SQL	English Template	Example
Selection	SELECT * FROM table WHERE fname = value	table with fname equal to value	restaurants with rating equal to 3
Projection	SELECT fname FROM table	the fname of table	the cuisine of restaurants
Subquery	SELECT * FROM table1 WHERE id = ANY(SELECT table1_id FROM table2))	the table1 of table2	reviews of restaurant X
Join	SELECT * FROM table1 JOIN table2	table1 with their table2	restaurants with their reviews
Aggregate	SELECT count(*) FROM table	the number of table	The number of restaurants
	SELECT op(fname) FROM table	the op fname in table	The average rating of restaurants
Aggregate & Group by	SELECT fname , count(*) FROM table GROUP BY fname	the number of table in each fname	The number of restaurants by rating
	SELECT fname2 , op(fname1) FROM table GROUP BY fname2	the op fname1 in table in each fname2	The average rating of restaurants by cuisine
Ranking	SELECT * FROM table ORDER BY fname LIMIT n	the n table with the min fname	the 3 restaurants with the min rating
Row-wise function	SELECT count(fname) of table	the number of fname in table	The number of reviews in restaurants

From SQL to English

- It is possible using grammar rules, but the result is clunky
 - Because it misses domain terminology
 - Use language models to paraphrase clunky sentence
- Open question:
 - Can LLMs translate SQL query into a smooth sentence?
- Why do we care?
 - Reading the query back to the user is useful for interpretability
 - We can synthesize training data:
Sample all SQL queries using grammar, generate NL sentences

Can Existing Assistants Understand Compound Queries?

Examples of Compound Questions	Alexa	Google	Siri	Genie
Show me restaurants rated at least 4 stars with at least 100 reviews				✓
Show restaurants in San Francisco rated higher than 4.5		✓		✓
What is the highest rated Chinese restaurant near Stanford?			✓	✓
How far is the closest 4 star restaurant?				✓
Find a W3C employee that went to Oxford				✓
Who worked for Google and lives in Palo Alto?				✓
Who graduated from Stanford and won a Nobel prize?	✓	✓		✓
Who worked for at least 3 companies?				✓
Show me hotels with checkout time later than 12PM				✓
Which hotel has a pool in this area?		✓	✓	✓

Genie is trained on synthesized data + 100s manual data

Restaurant Assistant



```
CREATE TABLE restaurants (  
  name          TEXT,  
  address       TEXT,  
  location      TEXT,  
  phone_number  TEXT,  
  opening_hours TEXT,  
  cuisines      TEXT[],  
  price         ENUM ('cheap', 'moderate', 'expensive', 'luxury'),  
  rating        NUMERIC(2,1),  
  num_reviews   NUMBER,  
  reviews       FREE_TEXT,  
  popular_dishes FREE_TEXT,  
);
```

NUMERIC(2,1) means 2-digit precision, 1 digit after decimal

Queries in a Conversation (Yelp)

My dad is visiting me in Mountain View. What do you think about dinner choices?

I found Cascal, a moderate Spanish restaurant with great tapas platters, appetizers, and drinks. It has a great ambience with live music on the weekends and both indoor and outdoor seating with heat lamps to keep guests warm.

Generated SQL:

*SELECT * FROM restaurants
WHERE location = "Mountain View"*

How about Italian instead? I really like the food there when I visited Florence last year.

I found Vaso Azzurro Ristorante, a moderate Italian restaurant near Mountain View. It has a lively atmosphere and offers excellent customer service. Reviews say their pasta is authentic and perfectly seasoned.

Generated SQL:

*SELECT * FROM restaurants
WHERE location = "Mountain View"
AND cuisines = ANY("Italian")*

Oh, what is the ham called in Italian?

Prosciutto is the Italian word for ham.

GPT responds directly

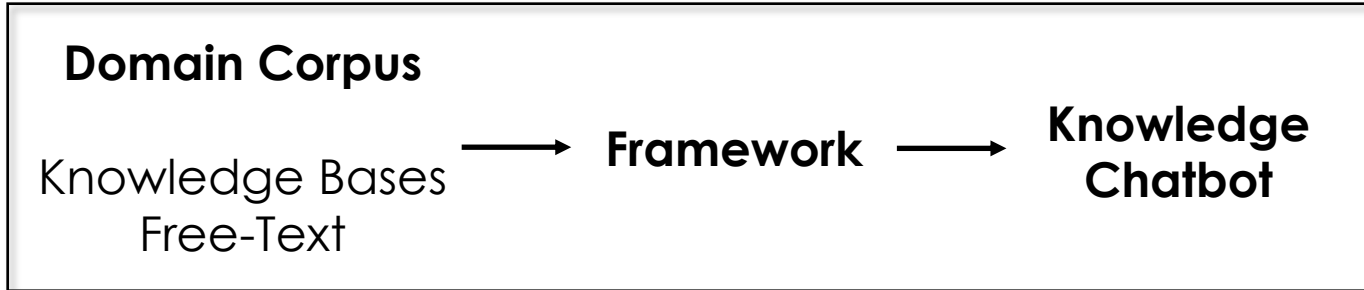
Thanks. We'll be visiting Stanford. How about in Palo Alto?

I found Terún, a 4 star Italian and Pizza restaurant near Stanford in Palo Alto. Reviews say the pizza is great and they can seat customers outside without a reservation.

Generated SQL:

*SELECT * FROM restaurants
WHERE location = "Palo Alto"
AND cuisines = ANY("Italian")*

Lecture Goal



- Why knowledge bases?
- Why formal representation for NL (SQL)?
- **Semantic parsing: Translates natural into formal language**
- Combining databases and free-text
 - Need a new formal language: SUQL

LLM: Zero-Shot Prompt

You are a semantic parser. Generate a query for a restaurant database with the following signature:

```
CREATE TABLE restaurants (  
  name TEXT,  
  address TEXT,  
  location TEXT,  
  phone_number TEXT,  
  opening hours TEXT,  
  cuisines TEXT[],  
  price ENUM ('cheap', 'moderate', 'expensive', 'luxury'),  
  rating NUMERIC(2,1),  
  num_review NUMBER,  
);
```

Issue: Enumerated Types

```
1 CREATE TABLE restaurants ( ...
2     price ENUM ('cheap', 'moderate', '
3         expensive', 'luxury'),
4     ...)
```

- Small # values (e.g. 10): included in schema description
- LLMs know how to use these enumerated values

What are some good **inexpensive** chinese restaurants in Palo Alto?

↓
SELECT * FROM restaurants WHERE
'chinese' = ANY (cuisines) AND location = 'Palo Alto'
AND **price** = 'cheap' AND rating >= 4.0
ORDER BY num_reviews DESC LIMIT 3;

Large Enumerated Types

- Example: Yelp cuisines (text) has over 200 values, e.g.
 - "brazilian", "southern", "soul food", "modern european", "champagne bars", "latin american", "health & medical", "singaporean", ...
- Problem: Not feasible to put all choices in the schema
- Database search expects an exact match

<p>show me a <i>café</i></p> <p>↓</p> <pre>SELECT * FROM restaurants WHERE 'coffee' = ANY (cuisines)</pre>	<p><i>Yelp: no coffee cuisine</i></p> <p><i>Available cuisines:</i></p> <p><i>coffee & tea,</i></p> <p><i>cafe</i></p>
--	--

Large Enumerated Types

show me a *café*

↓ *Semantic parser*

```
SELECT * FROM restaurants WHERE 'coffee' = ANY (cuisines)
```

↓ *Rewrite = operator*

```
SELECT * FROM restaurants WHERE  
'coffee & tea' = ANY (cuisines) OR 'cafe' = ANY (cuisines)
```

*Yelp: no coffee cuisine
Available cuisines:
coffee & tea,
cafe*

Solution

- Change '=' operator for a text field into a value classification operation
- $\text{classify}(x, V)$ finds the closest values of x in V ,
returns $\{\}$ if no close values found
- Given a text field f , collect all possible values V

$x = \text{ANY}(f) \Rightarrow c_1 = \text{ANY}(f) \text{ OR } c_2 = \text{ANY}(f) \dots,$
where $c_i \in \text{classify}(x, V)$

Quiz: How to implement Classify?

Evaluation

- Measure query accuracy of our annotated data
 - Poor accuracy
- Quiz: what should we do next?

Example

What's some great Mexican food around Bernal Heights?

- Gold Target:
 - SELECT FROM restaurants
WHERE “mexican” = ANY (cuisines) AND location = “bernal heights”;
- Predicted Target:
 - SELECT FROM restaurants
WHERE “mexican” = ANY (cuisines) AND location = “bernal heights”
AND rating >= 4;
- Quiz: Which is correct?

Evaluation example (SQL, BIRD dataset)

Name all cards with 2015 frame style ranking below 100 on EDHRec.

- Gold Target:
 - `SELECT id FROM cards WHERE edhrecRank < 100 AND frameVersion = 2015`
- Predicted Target:
 - `SELECT name FROM cards WHERE edhrecRank < 100 AND frameVersion = 2015;`
- Quiz: which is correct?

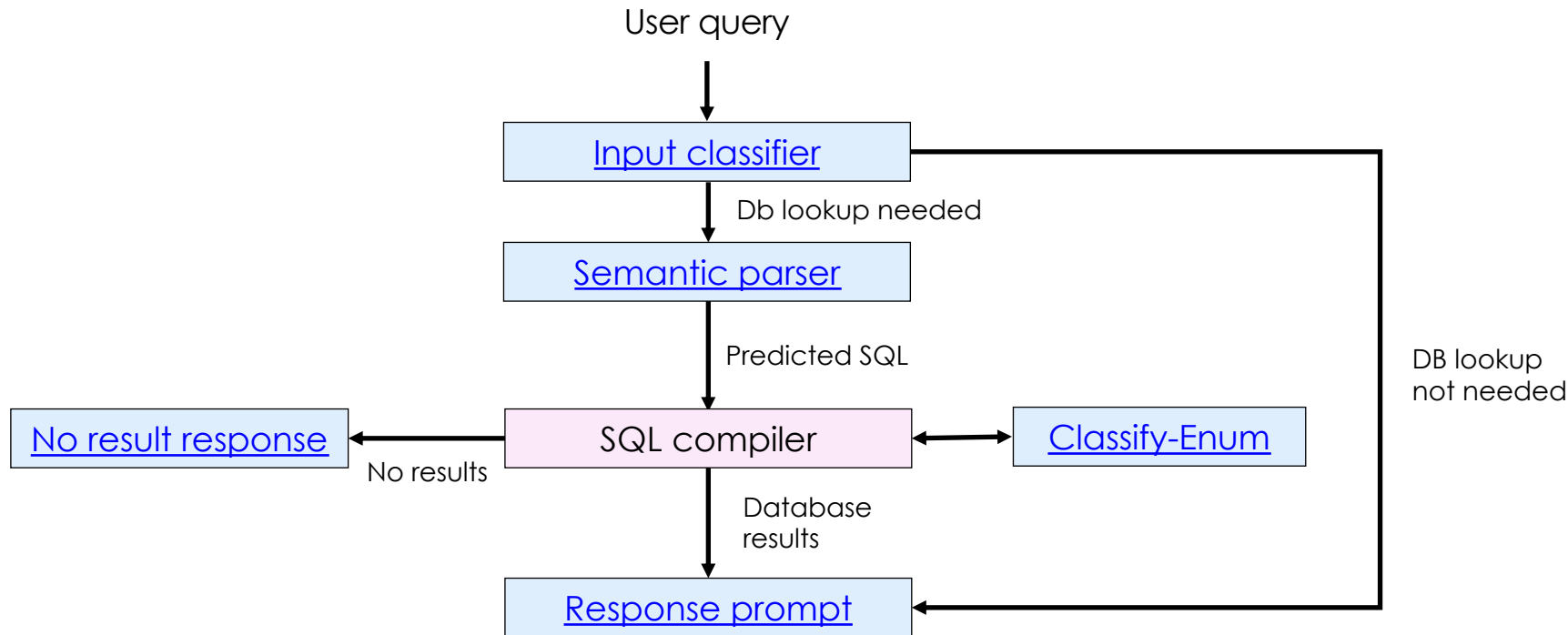
Quiz: What Should We Do Next?

- Tried few-shot ChatGPT to make it adhere to our convention
- Tried to fix the annotations in the benchmark
- ChatGPT is better than annotations by PhD students!
- Observation: Zero-shot ChatGPT works well enough for Yelp!
- Rigorous evaluation deferred till we handle hybrid data

The Full Agent Design

Some turns are not queries, let LLMs answer those

Agent Design



Click the links to see the prompts (written in [jinja syntax](#))

Summary

- Few-shot Chat-GPT parses SQL queries for Yelp
 - Restaurants: well-known domain to ChatGPT
 - Small table: 11 fields (incl. 2 Free-text, 1 small, 1 large ENUM)
 - Well-understood field names
- Open questions
 - BIRD: Can LLM serve as a DB interface? SOTA: 40%
 - HW2: Few-shot prompting of a single domain in BIRD
 - Students get experience and insight into an open question

BIRD: inyang Li et al. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. <https://arxiv.org/pdf/2305.03111.pdf>
HybridQA: <https://aclanthology.org/2020.findings-emnlp.91/>

Conclusion

- Databases are an important source of knowledge
 - Query languages: formal, expressive, compositional
 - Natural conversations on DB content: series of queries
- Open questions for open, large data and knowledge bases
 - Possible solution: data synthesis

Future
Topic