

Natural Language Interfaces to Complex Databases

Dragomir Radev

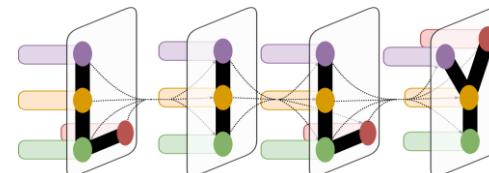
<http://lily.yale.edu>

Yale Institute for Network Science
Department of Computer Science
Yale University

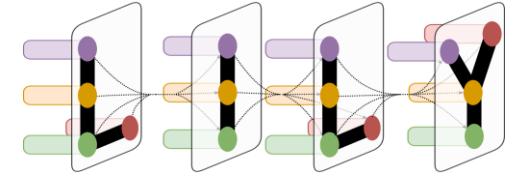
dragomir.ralev@yale.edu

February 1, 2019

Yale University

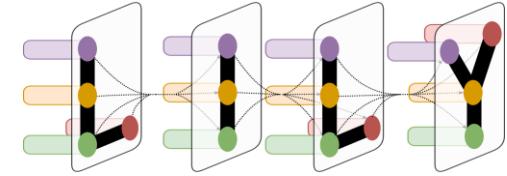


YALE INSTITUTE FOR NETWORK SCIENCE



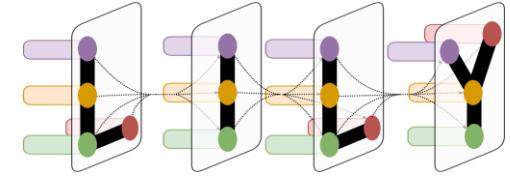
- PhD students
 - **Tao Yu**, Alexander Fabbri, Yavuz Nuzumlali, Irene Li, Rui Zhang
- MS students
 - Suyi Li, Tianwei She
- Undergraduate students
 - **Michihiro Yasunaga**, Caitlin Westerfield, Wei Tai Ting, Seungkyoon Bong, Sungrok Shim, Eren Orbey, David Proctor, Yi Chern Tang, Vinny Costa, Katherine Sun, Garrett Bingham, Jack Koch
- Collaborators
 - **Catherine Finegan-Dollak**, Lucy Jiang, Jonathan Kummerfeld, HV Jadadish, Harlan Krumholz, John Lafferty, Wade Schulz, Walter Lasecki

Current Projects at LILY



- **Multilingual information retrieval** -- We collaborate with researchers from Columbia University, the University of Maryland, the University of Edinburgh, and the University of Cambridge to build search engines for English users to query documents written in other languages including Swahili and Tagalog. This cross-lingual information retrieval system improves our capability of understanding and processing different low-resource languages and it offers users a reliable access to foreign documents. Currently working on Tagalog, Swahili, Somali.
- **Resources for learning NLP and AI** -- We aim to make dynamic research topics more accessible to the public by generating surveys of topics, discovering prerequisite relations among topics and recommending appropriate resources based on a given individual's education background and needs. We host a search engine, AAN (All About NLP) and tool which is available at <http://aan.how>.
- **Medical NLP** --We work with the Center for Outcomes Research & Evaluation (CORE) at the School of Medicine at Yale, investigating the use of NLP on electronic health records. The tasks include abbreviation disambiguation, patient digital pairs, patient history record summarization. We are interested in how to transfer general knowledge to the medical domain and how to take good advantage of the limited high-quality data. Besides, we are also developing an NLP toolkit for the medical domain to perform tasks like named entity recognition and relation extraction.
- **Natural language database interfaces and dialogue systems** -- The goal of this project is to allow users with no SQL background to talk to relational databases directly using human language. This way, anyone can easily query and analyze a vast amount of data. Moreover, we also aim to build conversational interfaces for even more natural information access, where the users participate in a conversation and the system takes the responsibility of choosing data sources and developing queries.

Project Sapphire



- Dialogue System for Advising CSE Students at UM
 - Help students with course selection, career choices, etc., in conjunction with the regular advisers
- Examples:
 - What courses should I take next semester?
 - Am I eligible to declare the CS-LSA major?
 - I am interested in Artificial Intelligence.
- Research questions:
 - Semantic parsing
 - Neural dialogue modeling
 - Next utterance prediction
- Not enough!
 - Step back

Data is important...

Data

- Data explosion
- Computation power is not a big problem anymore
- Make profits and reduce cost for companies by using machine learning models
- **Also, more and more people from different backgrounds are trying to query and use their own data.**
- The demand for data scientists is increasing so quickly
- Even at Yale, Dept. of Stats -> Dept. of Stats and Data Science

Not everyone is a data scientist ...

- The tasks require some technical background
- Writing SQL to query your data can be hard
- Let's make these tasks easier to everyone
- Can we talk to our databases directly to get our data?
 - Today's presentation

Database Schema

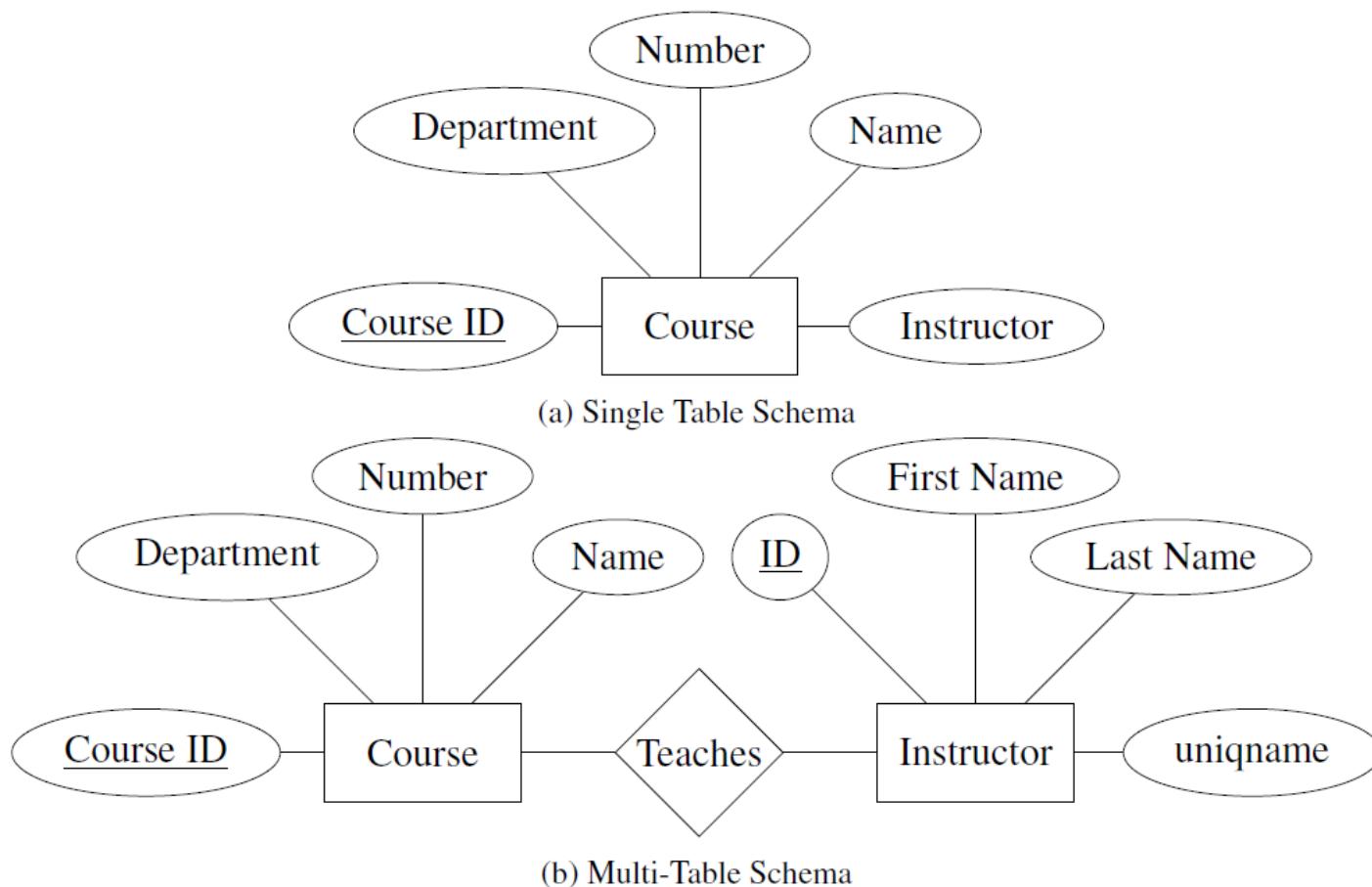
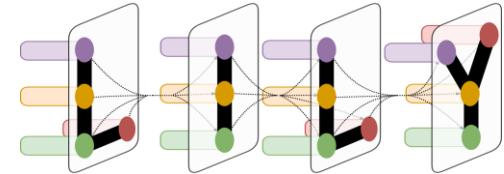
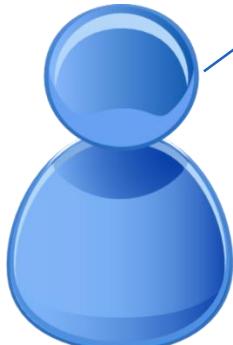


Figure 1: Two possible schemas for a database that could answer, “Who teaches Discrete Mathematics?”

Example: Text-to-SQL



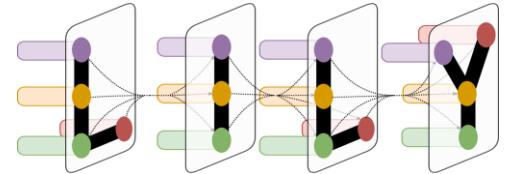
“Who teaches NLP?”



```
SELECT I.NAME  
FROM INSTRUCTOR AS I,  
OFFERING_INSTRUCTOR AS OI,  
COURSE_OFFERING AS O, SEMESTER AS S,  
COURSE AS C  
WHERE OI.INSTRUCTOR_ID=I.INSTRUCTOR_ID  
AND O.OFFERING_ID=OI.OFFERING_ID  
AND O.SEMESTER=S.SEMESTER_ID  
AND O.COURSE_ID=C.COURSE_ID  
AND C.NAME="NLP"  
AND S.YEAR=2016 AND S.SEMESTER="FA"
```

All About SQL

Course				
Course ID	Dept.	Number	Name	Credits
1	EECS	203	Discrete Math	4
2	LING	137	Epic Grammar Fails	3
3	EECS	595	NLP	4



Instructor

Instructor ID	First Name	Last Name
1	Dragomir	Radev
2	Walter	Lasecki
3	Ezra	Keshet
4	Rada	Mihalcea

Course Offering

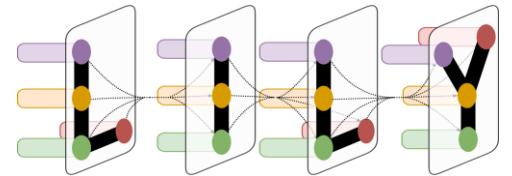
Course ID	Instructor ID	Year	Semester
3	1	2016	Fall
3	4	2017	Fall
2	3	2018	Winter

```
SELECT C.CREDITS  
FROM COURSE AS C  
WHERE C.NAME = "NLP";
```

How many credits is NLP?

All About SQL

Course				
Course ID	Dept.	Number	Name	Credits
1	EECS	203	Discrete Math	4
2	LING	137	Epic Grammar Fails	3
3	EECS	595	NLP	4



Instructor

Instructor ID	First Name	Last Name
1	Dragomir	Radev
2	Walter	Lasecki
3	Ezra	Keshet
4	Rada	Mihalcea

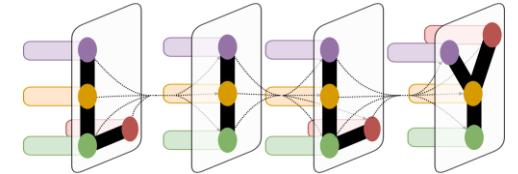
Course Offering

Course ID	Instructor ID	Year	Semester
3	1	2016	Fall
3	4	2017	Fall
2	3	2018	Winter

Who teaches NLP?

```
SELECT I.FIRST_NAME, I.LAST_NAME  
FROM INSTRUCTOR AS I,  
COURSE AS C,  
COURSE_OFFERING AS CO  
WHERE C.NAME = "NLP"  
AND C.COURSE_ID = CO.COURSE_ID  
AND CO.INSTRUCTOR_ID =  
I.INSTRUCTOR_ID;
```

All About SQL



Instructor

Course				
Course ID	Dept.	Number	Name	Credits
1	EECS	203	Discrete Math	4
2	LING	137	Epic Grammar Fails	3
3	EECS	595	NLP	4

Instructor ID	First Name	Last Name
1	Dragomir	Radev
2	Walter	Lasecki
3	Ezra	Keshet
4	Rada	Mihalcea

Course Offering

Course ID	Instructor ID	Year	Semester
3	1	2016	Fall
3	4	2017	Fall
2	3	2018	Winter

What course is worth the most credits?

```
SELECT C1.NAME  
FROM COURSE AS C1  
WHERE C1.CREDITS =  
(SELECT MAX C2.CREDITS  
FROM COURSE AS C2);
```

More Complicated SQL

Which countries in Europe have at least 3 car manufacturers?

```
SELECT T1.country_name  
FROM countries AS T1 JOIN continents  
AS T2 ON T1.continent = T2.cont_id  
JOIN car_makers AS T3 ON  
T1.country_id = T3.country  
WHERE T2.continent = 'Europe'  
GROUP BY T1.country_name  
HAVING COUNT(*) >= 3
```

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")
```

An English Language Question Answering System for a Large Relational Database

)> How many flights did plane 3 make in Jan 73?

(interim processing results and messages ordinarily are printed, but can be suppressed, as they are here)

I have interpreted your request as follows:

```
(FIND 'ALL
  '((V O))
  '((SUM (V TOTFLTS)))
  '(AND (EQU (V ACTDATEMON) 1.)
        (EQU (V ACTDATEYR) 3.)
        (EQU (V BUSER) 3.)))
  ` NIL)
```

A paraphrase of this request is: PLANES searches the MONTHLY FLIGHT AND MAINTENANCE SUMMARIES and returns: The value of TOTAL FLIGHTS for plane SERIAL #3 during January 1973.

Should I evaluate the query?

Type y or n >> y

Evaluating

(SUM TOTFLTS) = 17.

David L. Waltz

University of Illinois at Urbana-Champaign

)> How many flights did the A7 with tail number 003 make in January, 1973?

(SUM TOTFLTS) = 17.

)> How many flights did it make in Feb. 73?

(SUM TOTFLTS) = 1.

)> During April?

(SUM TOTFLTS) = 8.

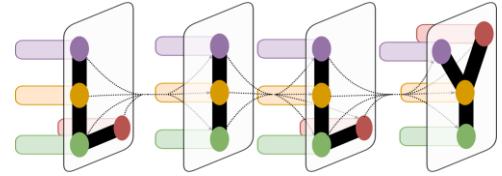
)> March?

(SUM TOTFLTS) = 13.

)> All of 1973?

(SUM TOTFLTS) = 39.

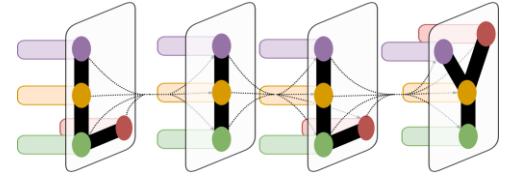
NLIDB Early Work



- Very early work hand-engineered QA systems for specific databases (e.g., LUNAR, Woods et al. 1972)
- PRECISE (Popescu et al., 2004): Syntax-based, high precision, only a limited class of questions
- The DB community has tried pattern matching and grammar-based methods; (e.g., Li & Jagadish 2014—grammar based + user feedback)
- Transfer of techniques from logical semantic parsing (e.g., Poon 2013; Iyer et al. 2017)
- Skip SQL with end-to-end systems (Yin et al. 2016, Neelakantan et al. 2017)

Limitations of Early Work

- Very limited vocabulary
- No machine learning
 - Lots of rules
- Syntactic parsing
- Overfitting
- No distinction between training and testing data
- The output was not SQL
 - introduced 1974, standardized 1986





Text2SQL is hard

- **Natural language understanding**
 - The system has to understand users' questions, which could be ambiguous, random, and diverse
- **Database schema representation**
 - The schema can be very complex with >100 attributes, >10 tables, and >20 foreign keys
- **Complex SQL decoding/generation**
 - SQL queries can be very complex, e.g., nested queries, multiple conditions

Task Comparison

- **Compared to other NLU (e.g., chatbots), seq2SQL is**
 - more task-oriented
 - less ambiguous
 - more streamlined
- **Compared to other semantic parsing tasks such as code generation, seq2SQL has**
 - A much smaller decoding search space - SQL keywords and column names
 - Smaller vocabulary - SELECT, WHERE, ORDERBY, NESTED...

A Natural Language Processing approach

- Text Understanding
- Training Examples
- Annotated Corpora

What is NLP?

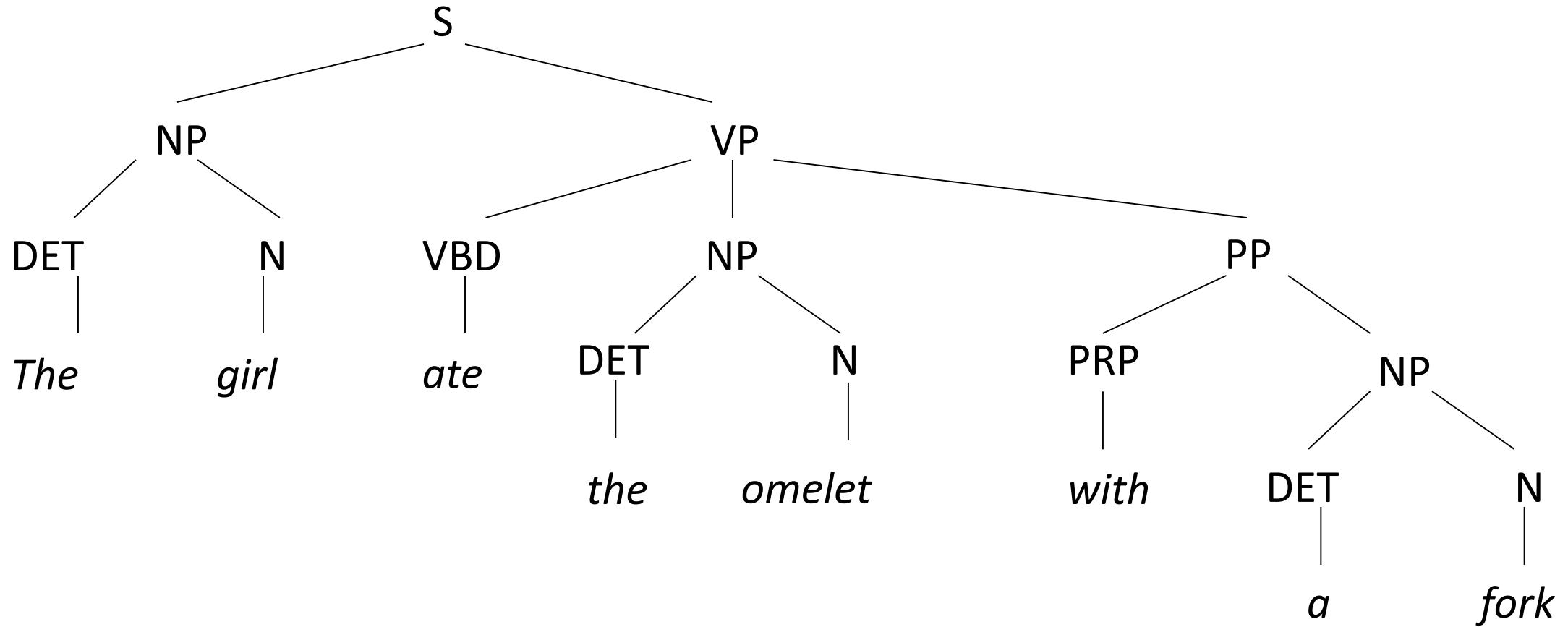
- Natural Language Processing (NLP)
 - The study of the computational treatment of natural (human) language.
 - Teaching computers how to understand (and generate) human language.

The NLP Pipeline

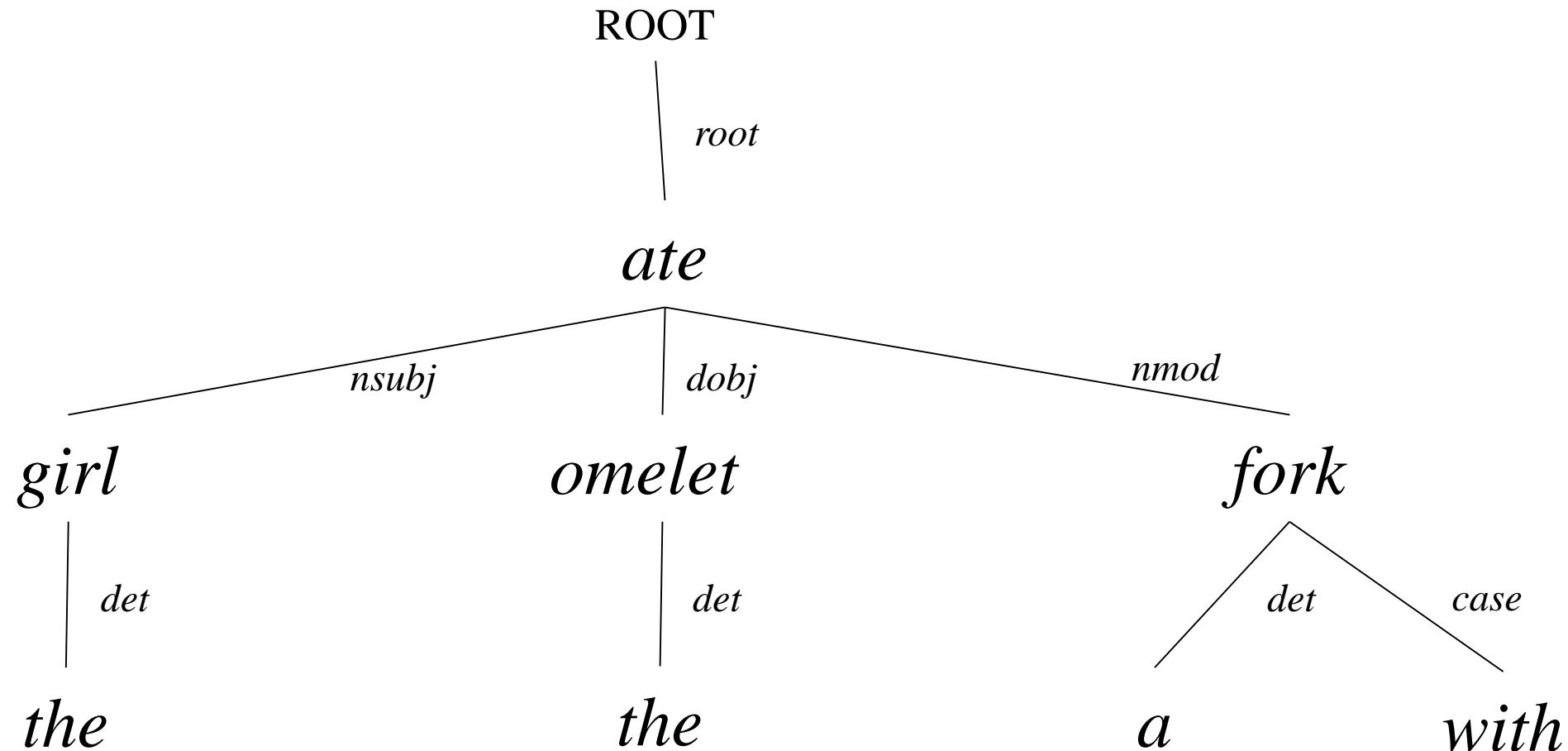
DET N VBD DET N PRP DET N

The girl ate the omelet with a fork.

Constituent Parsing



Dependency Parsing



Language Understanding

- Semantic Analysis

Girl (g_1)

Omelet (o_1)

Fork (f_1)

Eating (e_1) \wedge Eater (e_1, g_1) \wedge Eaten (e_1, o_1) \wedge Instrument (e_1, f_1)

- World Knowledge

Omelet (X) \Rightarrow Food (X)

- Inference

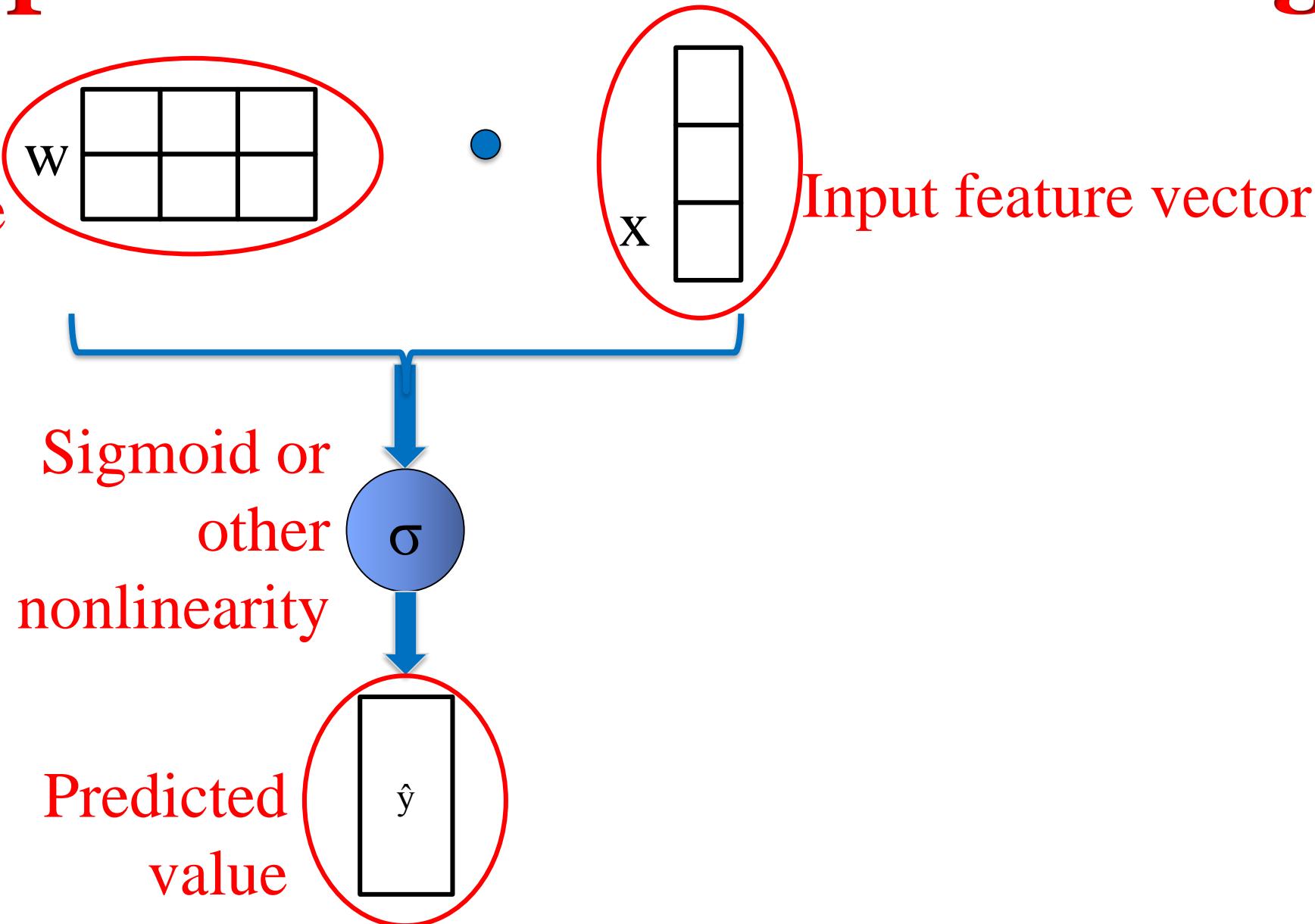
Hungry (Z, t_0) \wedge Eater (e_1, Z) \wedge Eaten (e_1, Y) \wedge Time (e_1, t_1) \wedge Food (Y) \wedge Precedes (t_0, t_1) \Rightarrow \neg Hungry (Z, t_1)

- Conclusion

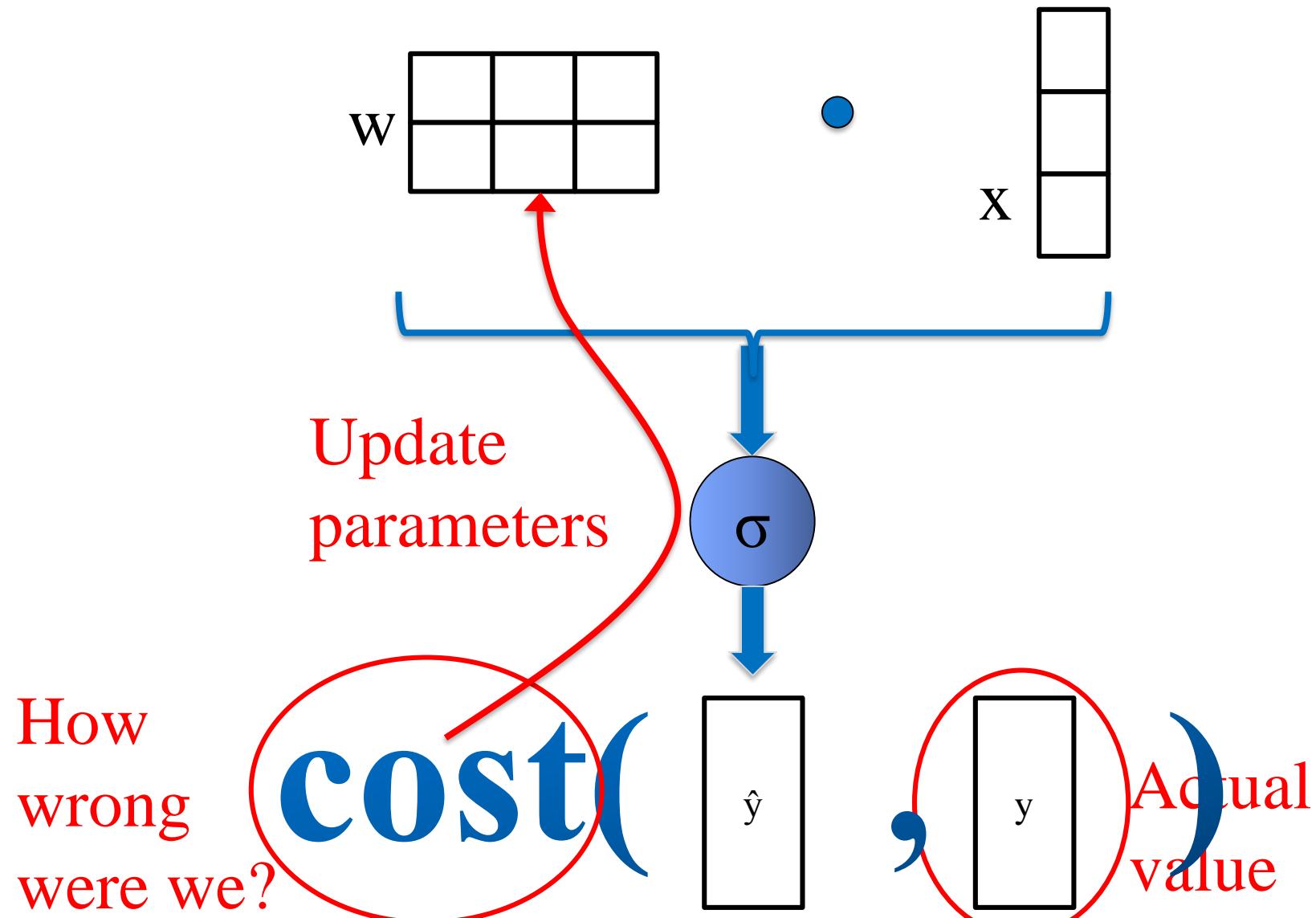
\neg Hungry (g_1, t_1)

Supervised Machine Learning

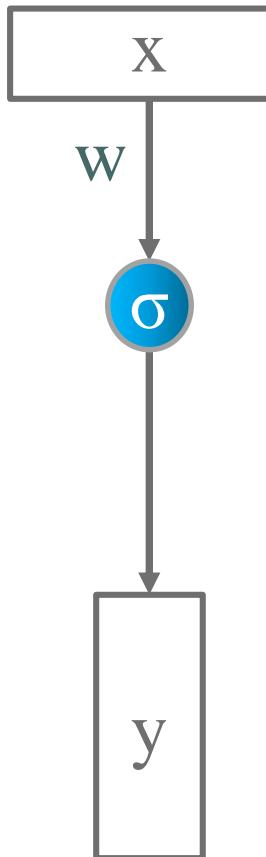
Parameters
(things we're
learning)



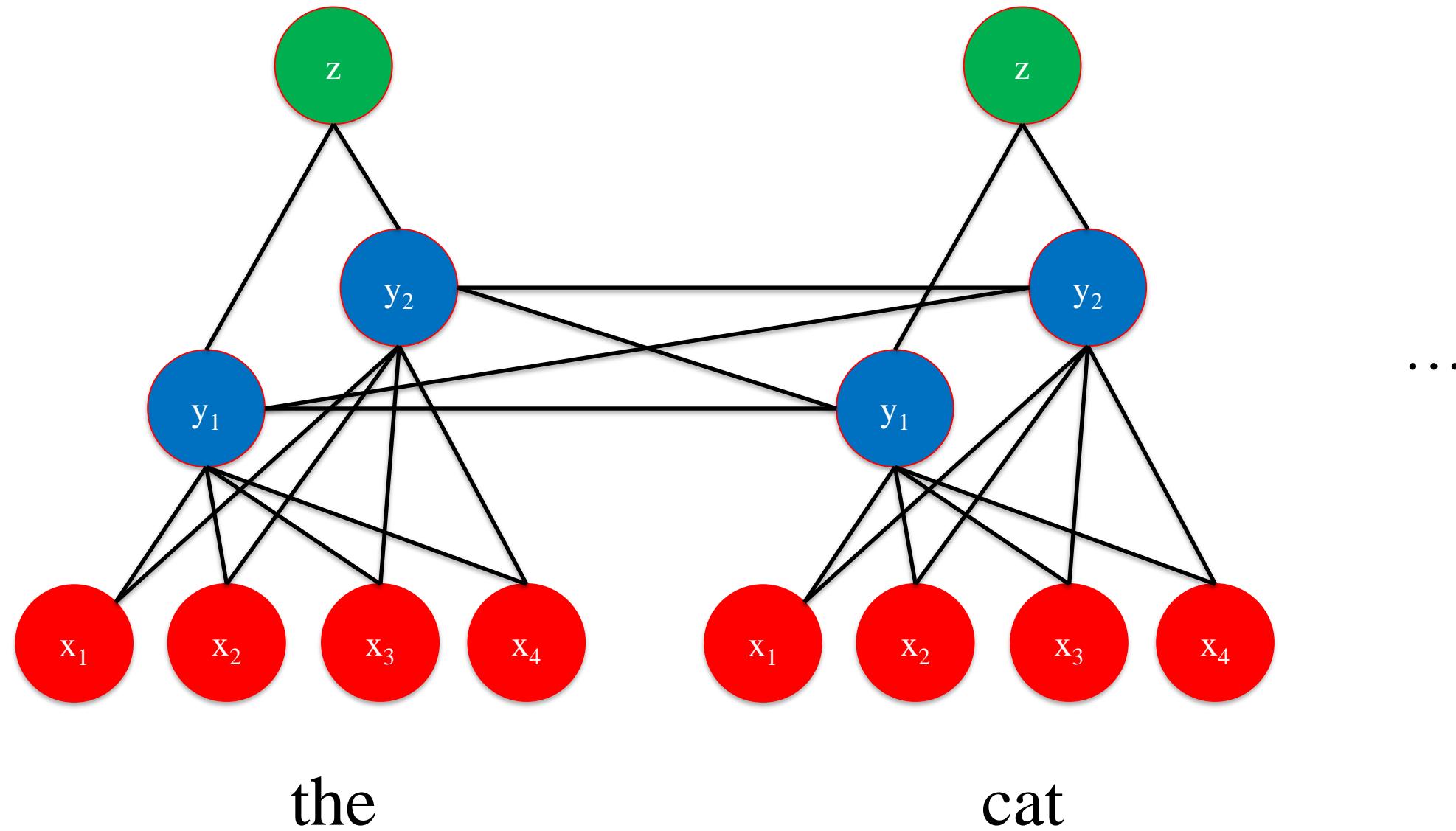
Supervised Machine Learning



A Simplified Diagram

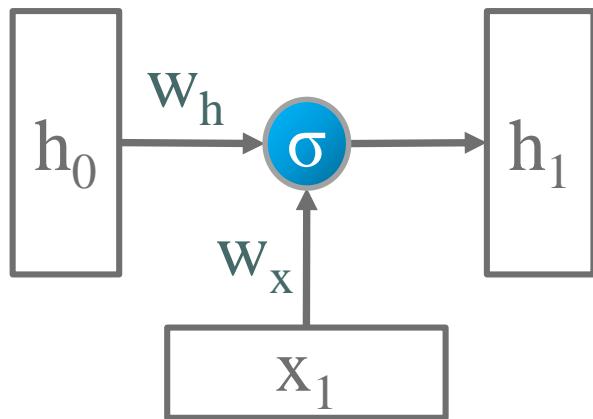


Recurrent Neural Networks

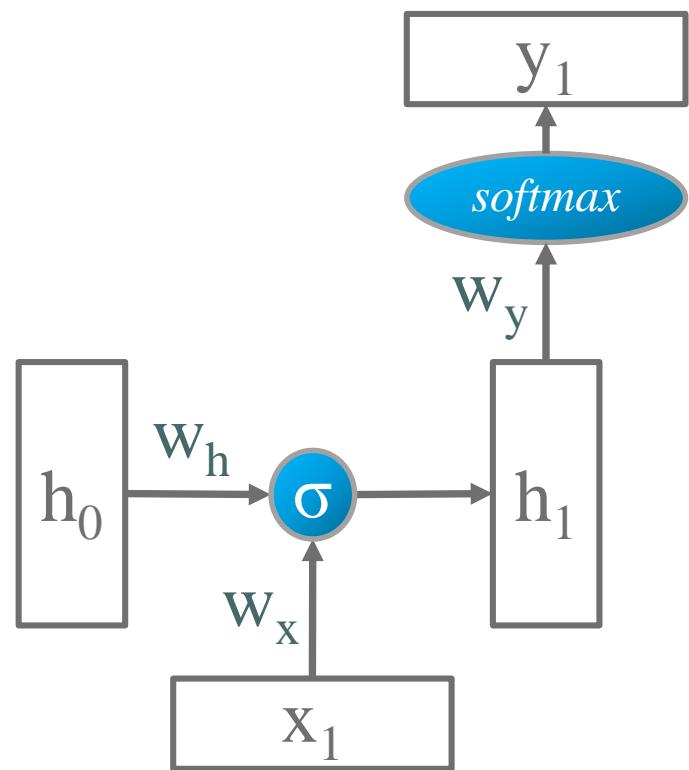


Recurrent Neural Networks

$$h_t = \sigma(W_h h_{t-1} + W_x x_t)$$

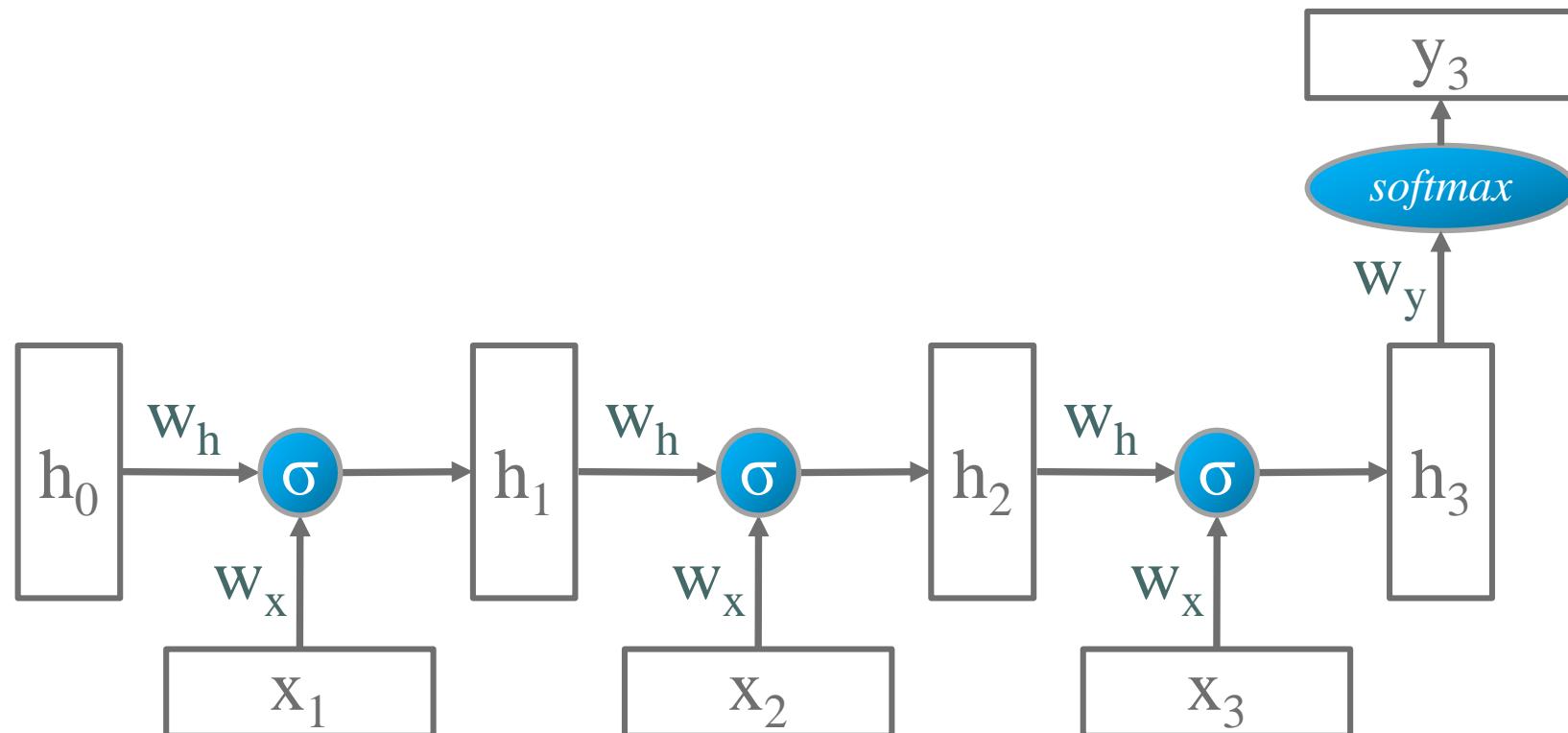


RNN



$$h_t = \sigma(W_h h_{t-1} + W_x x_t)$$
$$y_t = \text{softmax}(W_y h_t)$$

RNN

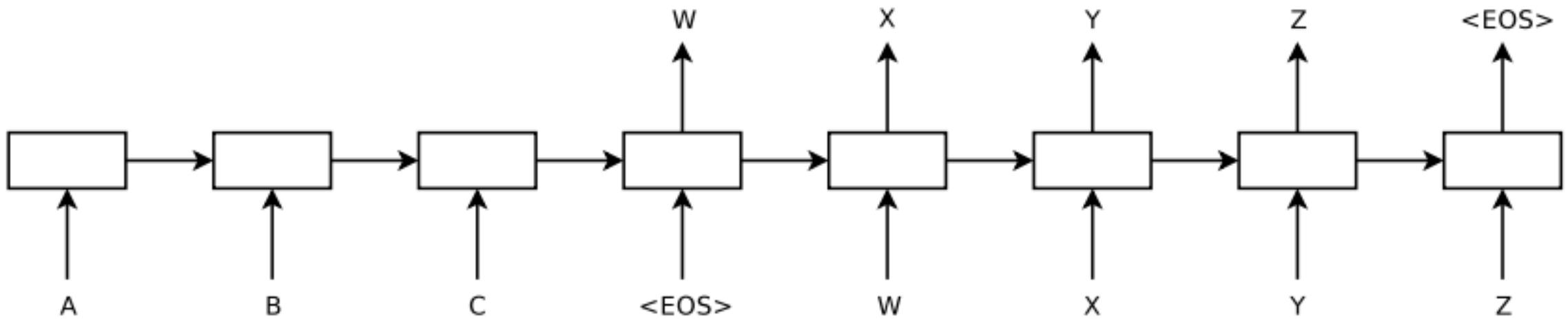


The

cat

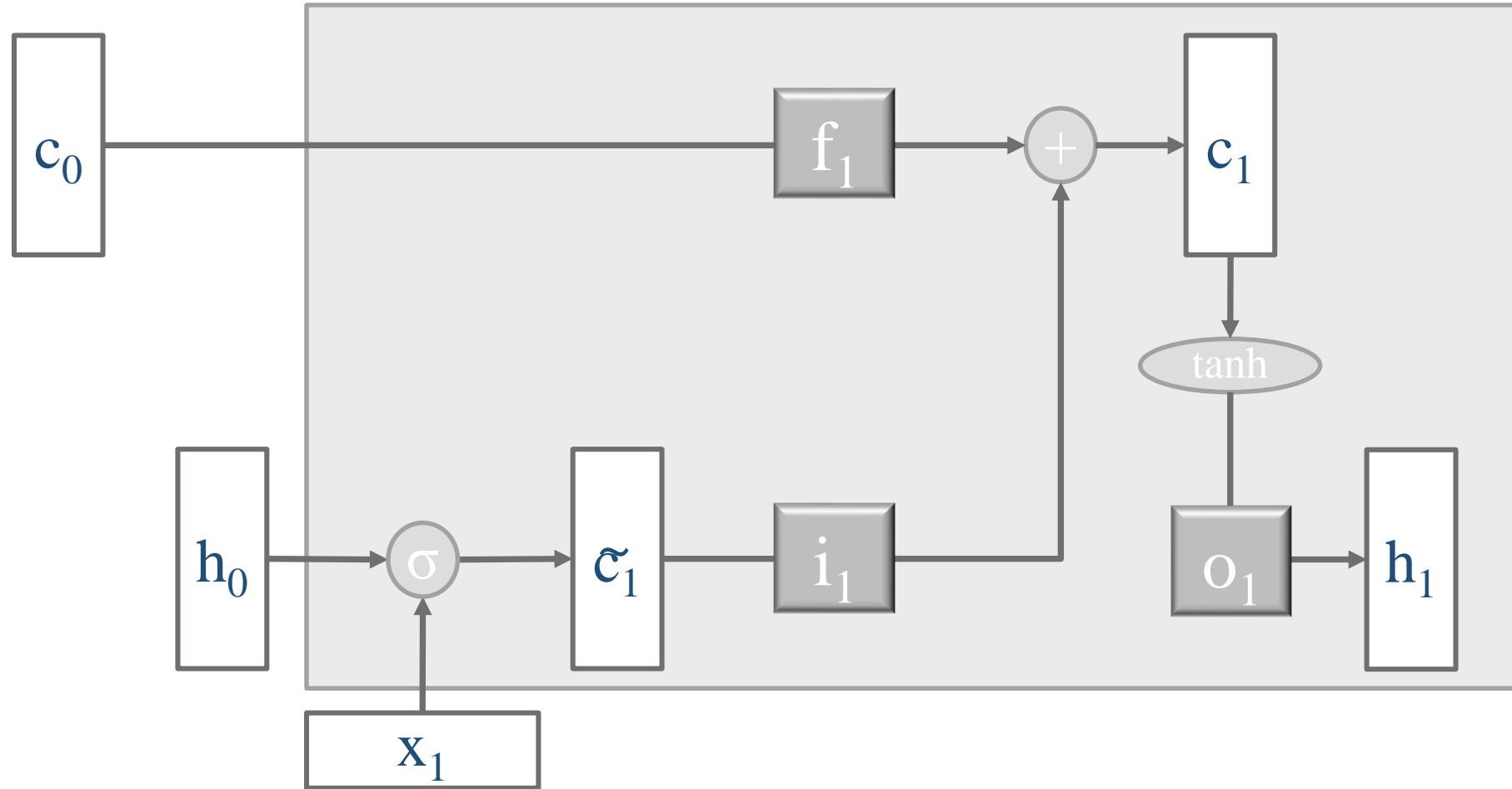
sat

Example Application: Machine Translation

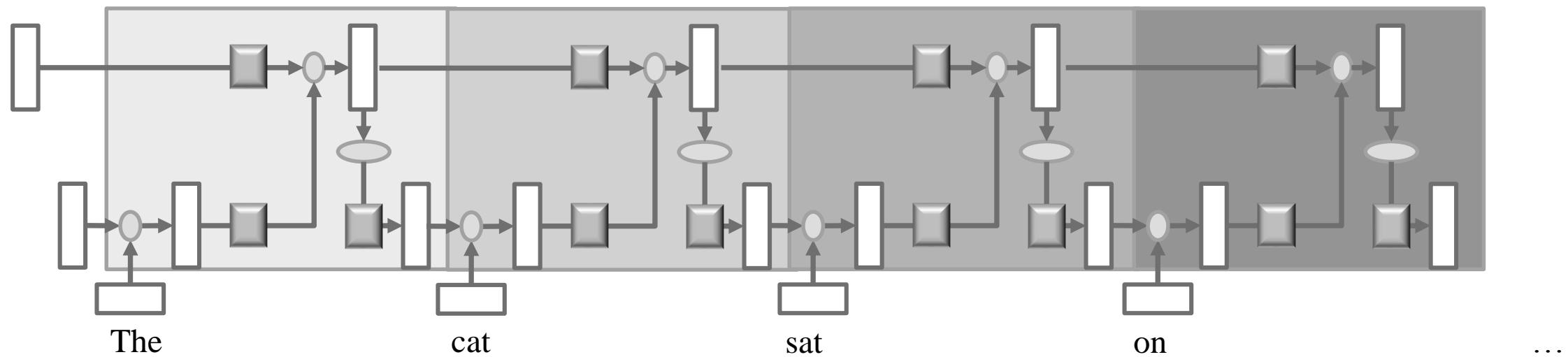


Sequence-to-sequence learning (Sutskever, Vinyals, and Le, 2014)

LSTMs: One Time Step



LSTMs: Building a Sequence



Recursive Neural Tensor Networks

Socher et al. (2013)
Recursive Deep Models for
Semantic Compositionality
Over a Sentiment Treebank

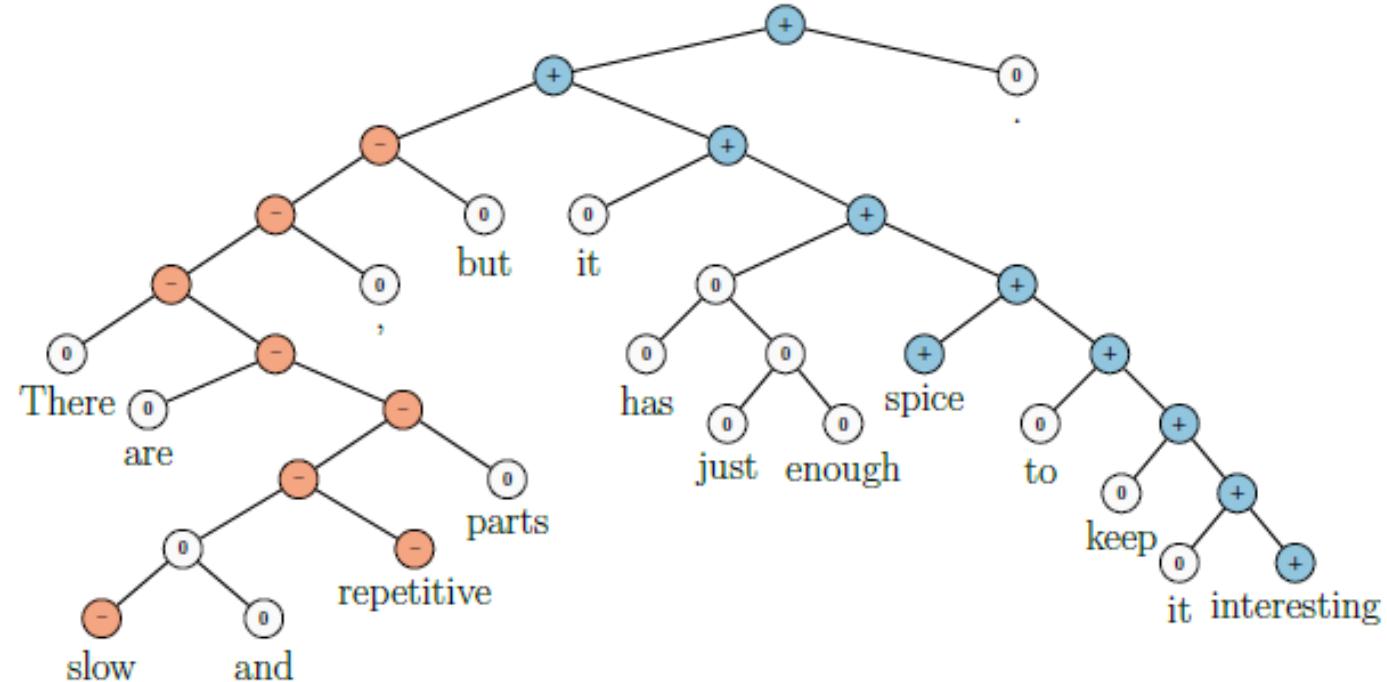
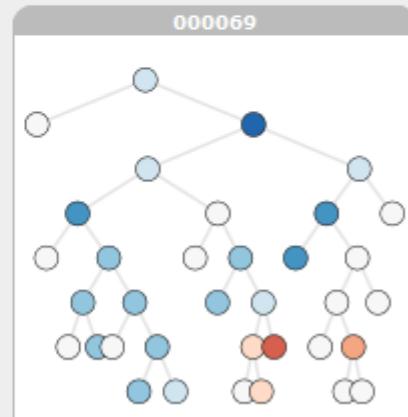
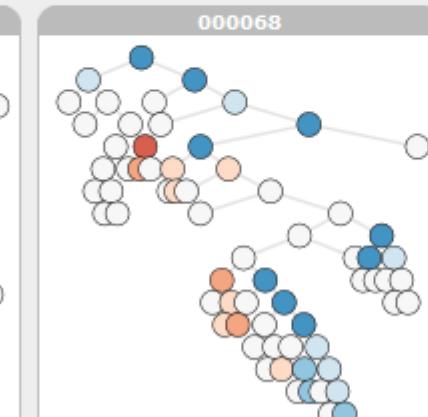
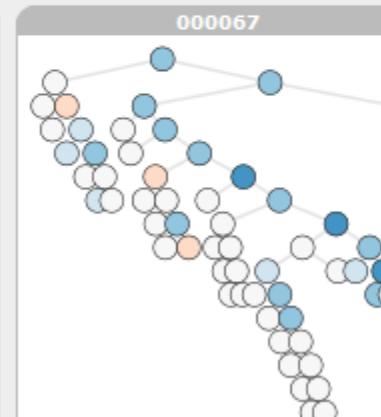
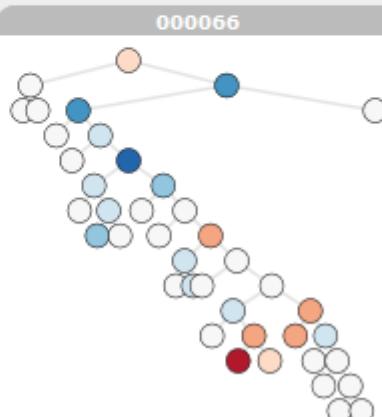
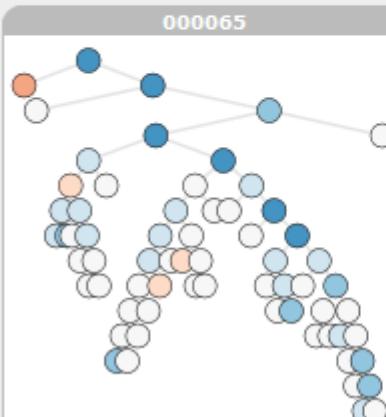
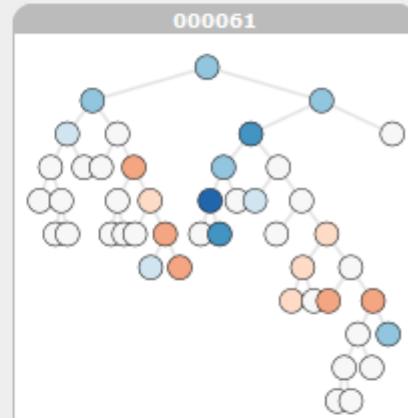
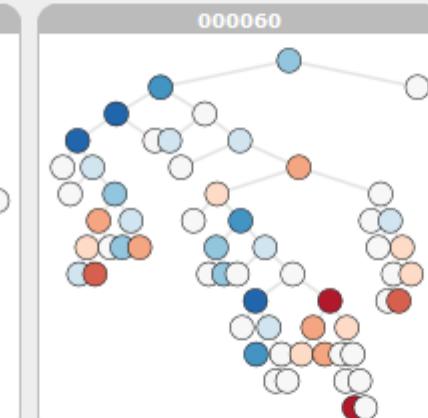
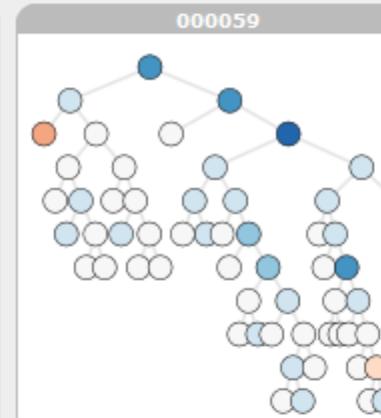
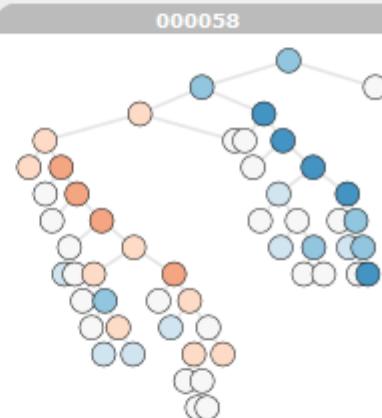
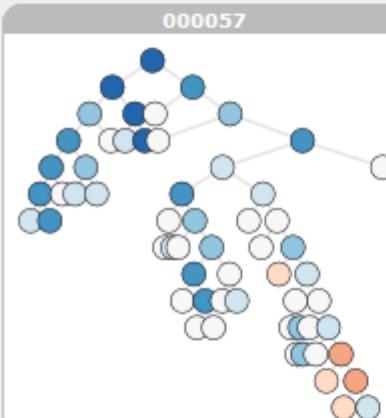
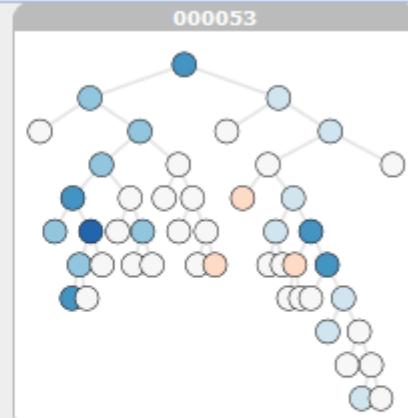
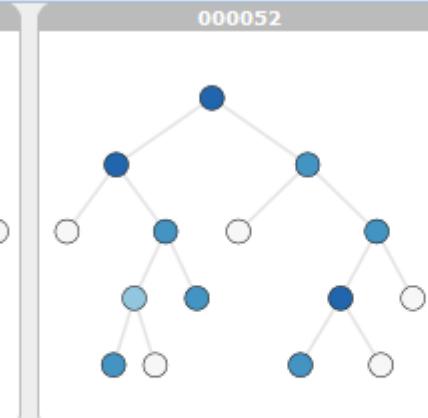
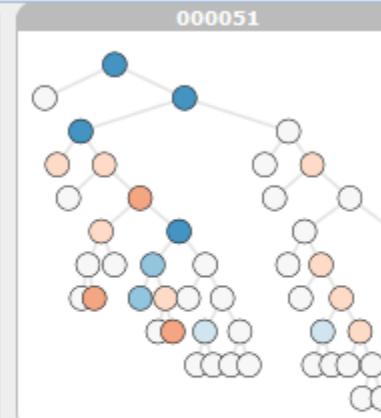
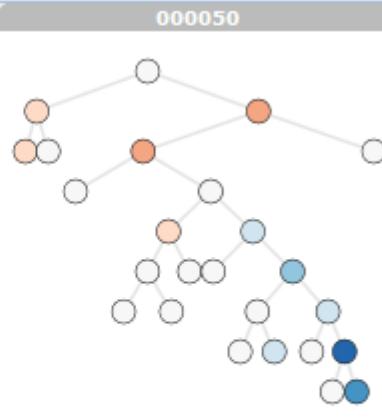
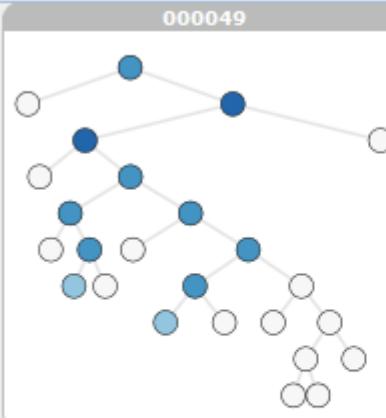


Figure 7: Example of correct prediction for contrastive conjunction X *but* Y .



Neural Semantic Parsing

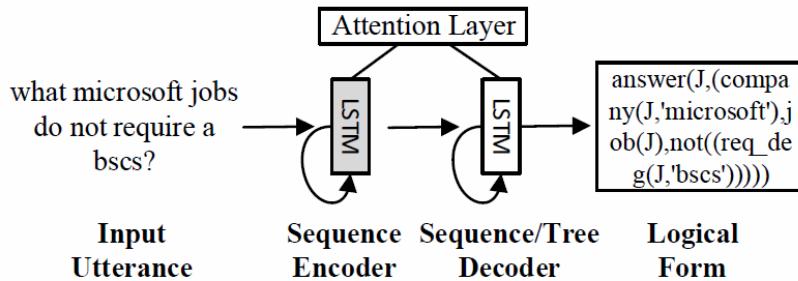


Figure 1: Input utterances and their logical forms are encoded and decoded with neural networks. An attention layer is used to learn soft alignments.

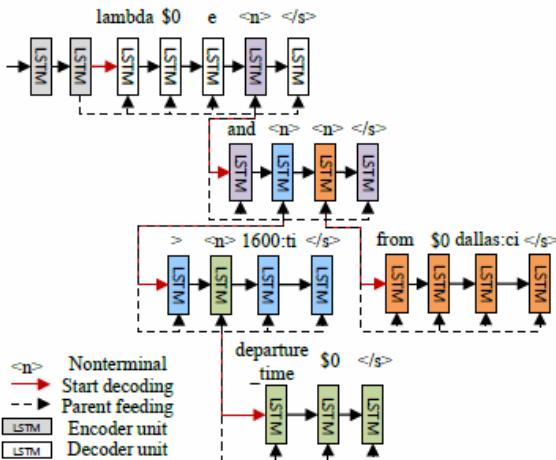


Figure 3: Sequence-to-tree (SEQ2TREE) model with a hierarchical tree decoder.

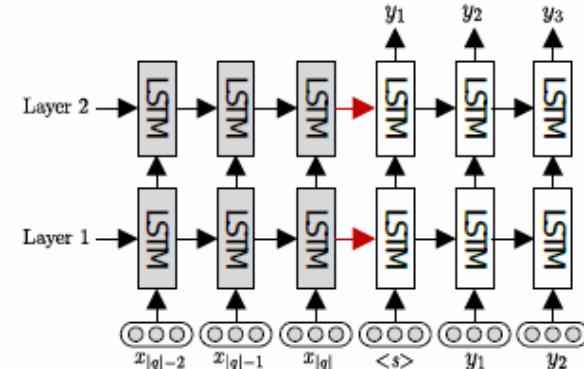


Figure 2: Sequence-to-sequence (SEQ2SEQ) model with two-layer recurrent neural networks.

Dataset	Length	Example
JOBS	9.80	<i>what microsoft jobs do not require a bscs?</i>
	22.90	answer(company(J,'microsoft').job(J),not((req_deg(J,'bscs'))))
GEO	7.60	<i>what is the population of the state with the largest area?</i>
	19.10	(population:i (argmax \$0 (state:t \$0) (area:i \$0)))
ATIS	11.10	<i>dallas to san francisco leaving after 4 in the afternoon please</i>
	28.10	(lambda \$0 e (and (>(departure_time \$0) 1600:ti) (from \$0 dallas:ci) (to \$0 san_francisco:ci)))
IFTTT	6.95	<i>Turn on heater when temperature drops below 58 degree</i>
	21.80	TRIGGER: Weather - Current_temperature_drops_below - ((Temperature (58)) (Degrees_in (f))) ACTION: WeMo_Insight_Switch - Turn_on - ((Which_switch? ("")))

Table 1: Examples of natural language descriptions and their meaning representations from four datasets. The average length of input and output sequences is shown in the second column.

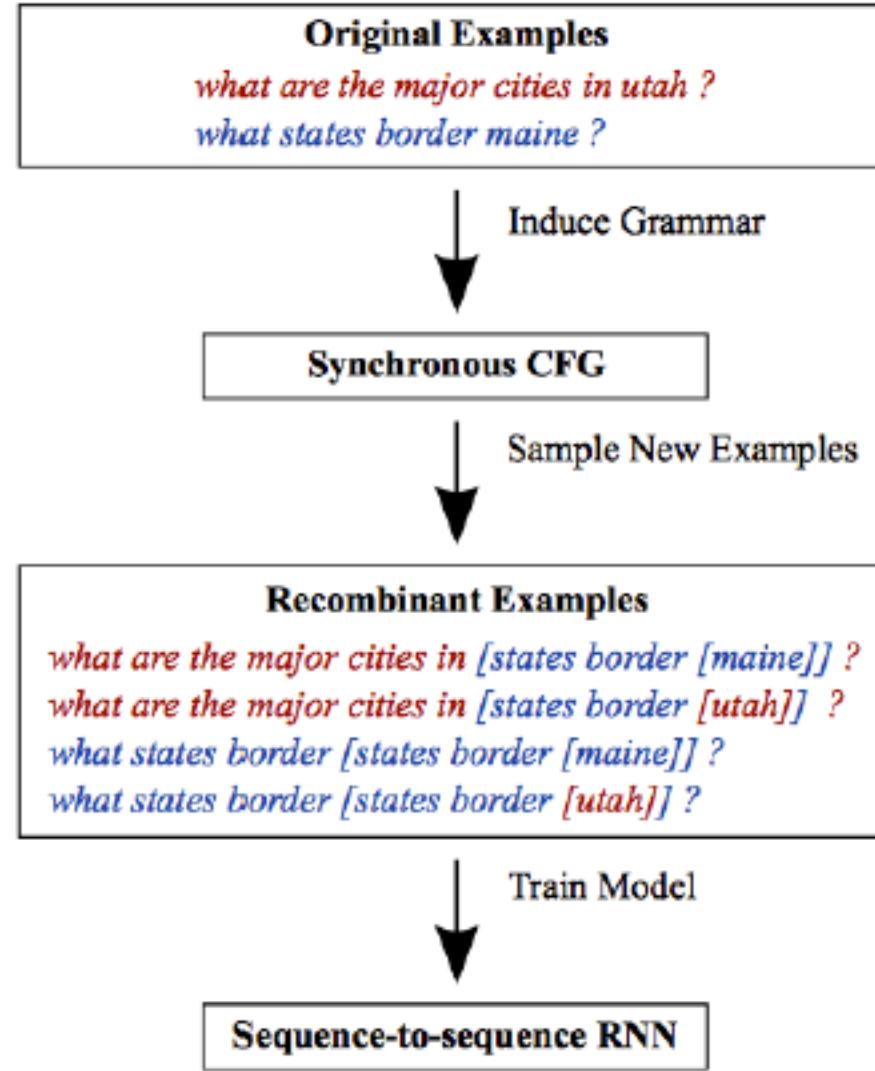
[Dong and Lapata 2016]

- **Geoquery:** Parsing to Prolog queries over small database (Zelle and Mooney 1996)

```
x: "what is the population of iowa ?"  
y: _answer ( NV ,  
             _population ( NV , V1 ) , _const ( V0 ,  
               _stateid ( iowa ) ) ) )
```

- **Free917:** Parsing to Freebase query language (Cai and Yates 2013)
 1. What are the neighborhoods in New York City?
 $\lambda x . \text{neighborhoods}(\text{new_york}, x)$
 2. How many countries use the rupee?
 $\text{count}(x) . \text{countries_used}(\text{rupee}, x)$
- Many others: WebQuestions, WikiTables, etc.

- Simple string-based sequence-to-sequence model
- Doesn't work well as-is, so generate extra synthetic data from a CFG



Neural Semantic Parsing

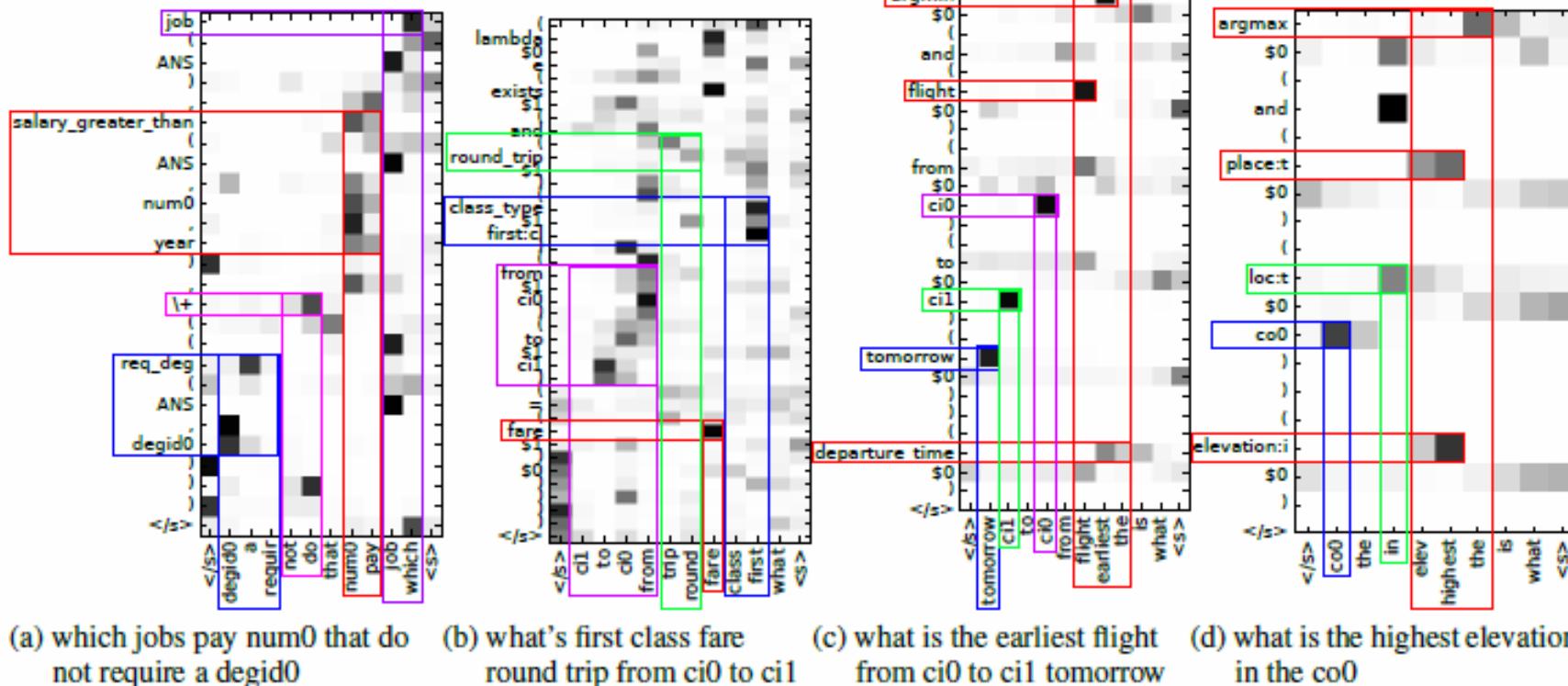
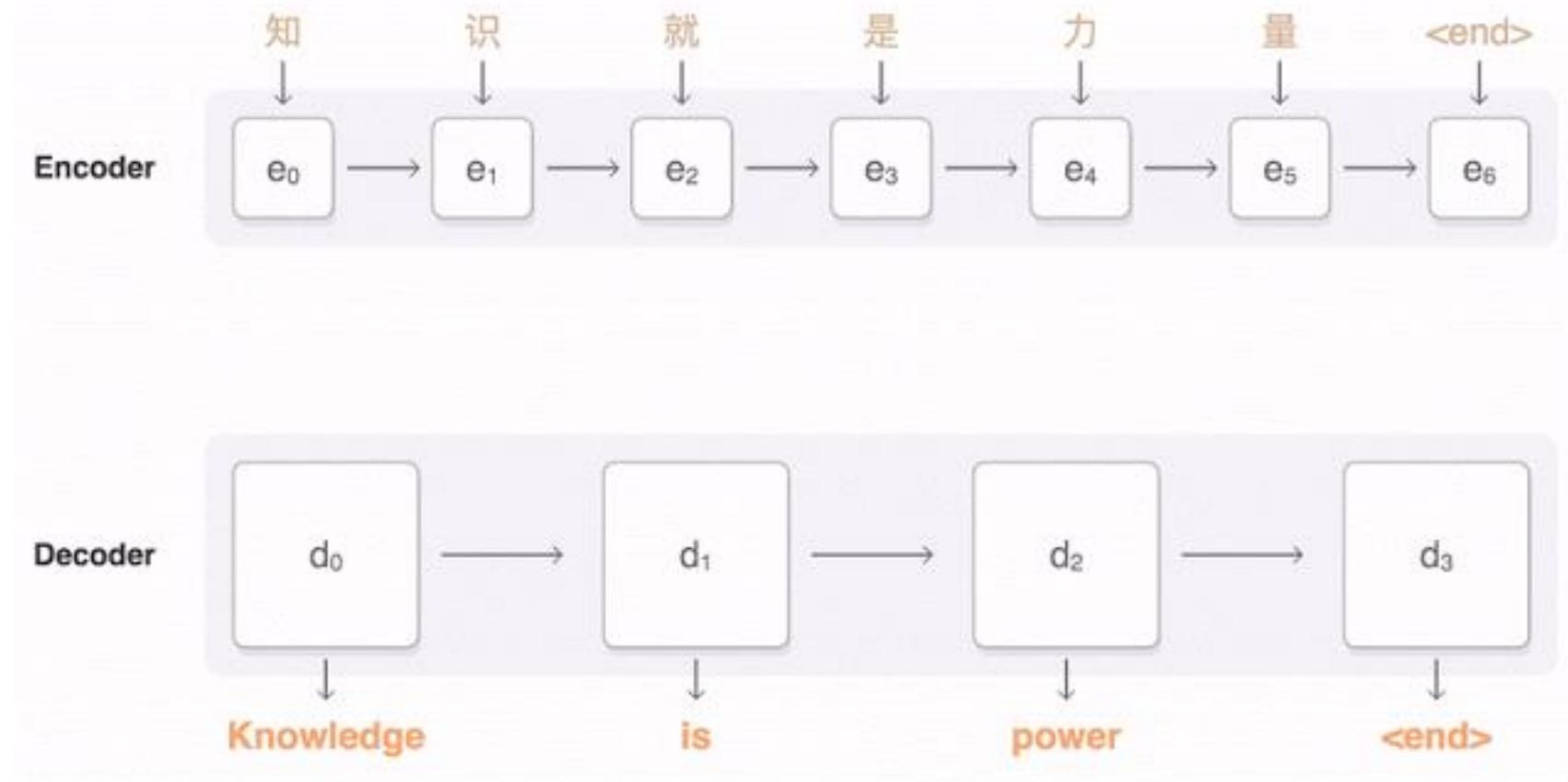


Figure 6: Alignments (same color rectangles) produced by the attention mechanism (darker color represents higher attention score). Input sentences are reversed and stemmed. Model output is shown for SEQ2SEQ (a, b) and SEQ2TREE (c, d).

Google Neural Translation System



Seq2SQL datasets are scarce!

- Compared to other large datasets such as ImageNet for object recognition, building a decent seq2SQL dataset is even more time-consuming
 - Hard to find many databases with multiple tables online
 - Annotation requires very specific knowledge in databases

Traditional Seq2SQL datasets

Traditional 9 seq2SQL datasets: [ATIS](#), [Geo](#), [Scholar](#), etc. + Advising

- Pros
 - SQL queries cover **complex** SQL structures and components
- Cons
 - The number of labeled queries is **small** (< 500)
 - **Paraphrase** about 4-10 natural language questions for each SQL query.
 - The total # of question-SQL pairs: ~500 -> ~5,000
 - Each of datasets contains SQL queries only to a **single** database

ATIS (Price, 1990; Dahl et al., 1994) User questions for a flight-booking task, manually annotated. We use the modified SQL from Iyer et al. (2017), which follows the data split from the logical form version (Zettlemoyer and Collins, 2007).

GeoQuery (Zelle and Mooney, 1996) User questions about US geography, manually annotated with Prolog. We use the SQL version (Popescu et al., 2003; Giordani and Moschitti, 2012; Iyer et al., 2017), which follows the logical form data split (Zettlemoyer and Collins, 2005).

Restaurants (Tang and Mooney, 2000; Popescu et al., 2003) User questions about restaurants, their food types, and locations.

Scholar (Iyer et al., 2017) User questions about academic publications, with automatically generated SQL that was checked by asking the user if the output was correct.

Academic (Li and Jagadish, 2014) Questions about the Microsoft Academic Search (MAS) database, derived by enumerating every logical query that could be expressed using the search page of the MAS website and writing sentences to match them. The domain is similar to that of Scholar, but their schemas differ.

Yelp and IMDB (Yaghmazadeh et al., 2017)

Questions about the Yelp website and the Internet Movie Database, collected from colleagues of the authors who knew the type of information in each database, but not their schemas.

WikiSQL (Zhong et al., 2017) A large collection of automatically generated questions about individual tables from Wikipedia, paraphrased by crowd workers to be fluent English.

Advising (This Work) Our dataset of questions over a database of course information at the University of Michigan, but with fictional student records. Some questions were collected from the EECS department Facebook page and others were written by CS students with knowledge of the database who were instructed to write questions they might ask in an academic advising appointment.

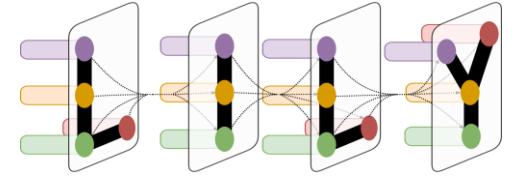
Dialog2SQL Data Creation

Our Complex and Cross-Domain Text-to-SQL Dataset: Spider

Dataset	# Q	# SQL	# DB	# Table /DB	ORDER BY	GROUP BY	NESTED	HAVING	LIMIT
ATIS	5,280	947	1	32	0	5	315	0	0
GeoQuery	877	247	1	6	20	46	167	9	20
Scholar	817	193	1	7	75	100	7	20	1
Academic	196	185	1	15	23	40	7	18	23
IMDB	131	89	1	16	10	6	1	0	10
Yelp	128	110	1	7	18	21	0	4	18
Advising	3,898	208	1	10	15	9	22	0	11
Restaurants	378	378	1	3	0	0	4	0	0
WikiSQL	80,654	77,840	26,521	1	0	0	0	0	0
Spider (original)	10,181	5,693	200	5.1	1335	1491	844	388	903

Figure: Comparisons of text-to-SQL datasets

Standard Practice in ML

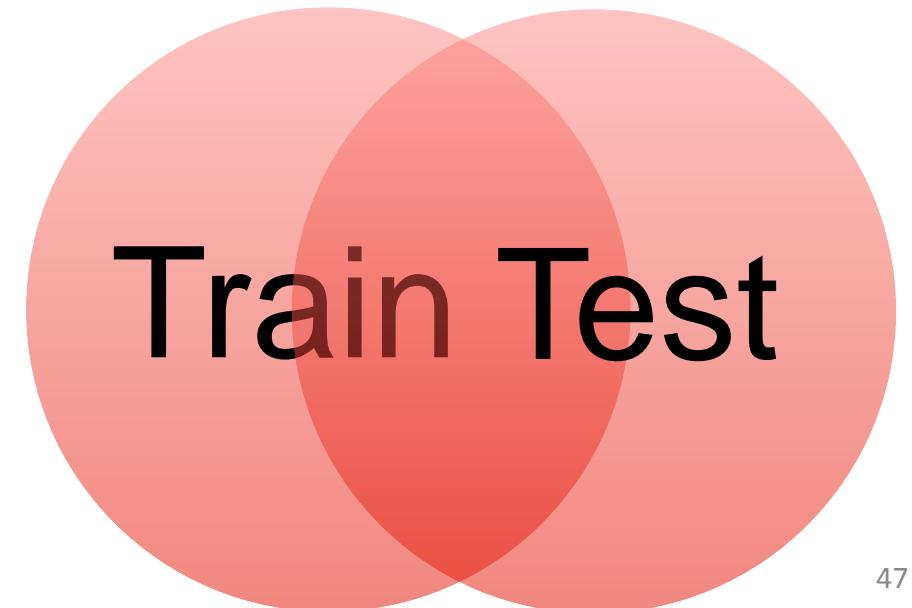


- Split dataset into train set, test set, optional development (dev) set.
- No training example can also appear in test set.

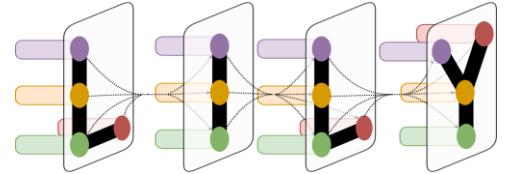
Good Split



Bad Split



How Do We Define an Example?



how many people are there in iowa?

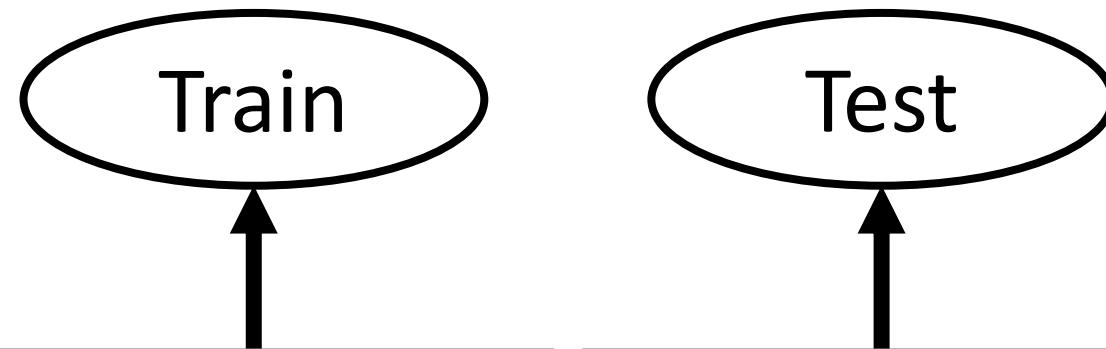
```
select population  
from state  
where state_name = "iowa"
```

how many people live in utah?

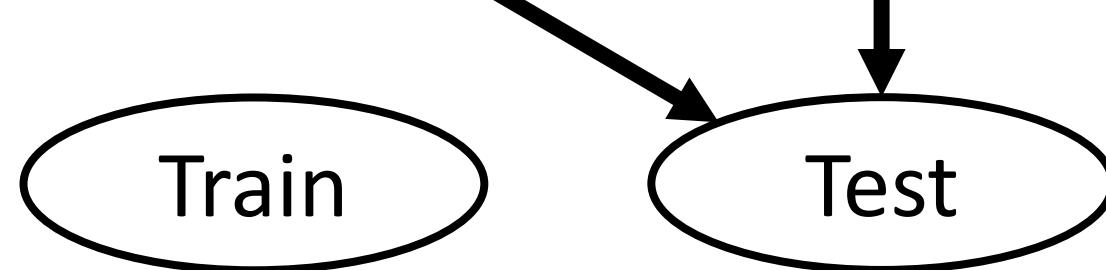
```
select population  
from state  
where state_name = "utah"
```

Question-Based Split

how many people are there in iowa?
select population
from state
where state_name = "iowa"



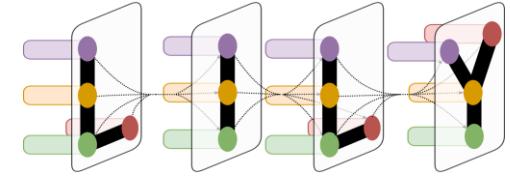
Query-Based Split



Prior Work: Improving Text-to-SQL Evaluation Methodology

- Why Question-based split -> query-based split matters?
 - The text-to-SQL **translation problem** becomes a **simpler classification problem** if you have exactly the same SQL labels in the train and test splits.
 - **Simpler classification problem:**
 - input is natural language questions,
 - **output classes** are all unique SQL queries appearing in your train/test
 - Query-based split avoids this kind of cheating

Experiments



- State of the art models:
 - Dong & Lapata 2016, seq2tree
 - Iyer et al. 2017, seq2seq with dataset expansion
 - Seq2seq with attention-based copying (similar to Jia & Liang 2016)
- Baseline:
 - Classifier + slot-filler
 - Oracle: What is the best a perfect classifier + slot-filler could do?

Model	Advising		ATIS		GeoQuery		Restaurants		Scholar		Academic		IMDB		Yelp	
	?	Q	?	Q	?	Q	?	Q	?	Q	?	Q	?	Q	?	Q
No Variable Anonymization																
Baseline	80	0	46	0	57	0	95	0	52	0	0	0	0	0	1	0
seq2seq	6	0	8	0	27	7	47	0	19	0	6	7	1	0	0	0
+ Attention	29	0	46	18	63	21	100	2	33	0	71	64	7	3	2	2
+ Copying	70	0	51	32	71	20	100	4	59	5	81	74	26	9	12	4
D&L seq2tree	46	2	46	23	62	31	100	11	44	6	63	54	6	2	1	2
Iyer et al.	41	1	45	17	66	40	100	8	44	3	76	70	10	4	6	6
With Oracle Entities																
Baseline	89	0	56	0	56	0	95	0	66	0	0	0	7	0	8	0
seq2seq	21	0	14	0	49	14	71	6	23	0	10	9	6	0	12	9
+ Attention	88	0	57	23	73	31	100	32	71	4	77	74	44	17	33	28
D&L seq2tree	88	8	56	34	68	23	100	21	68	6	65	61	36	10	26	23
Iyer et al.	88	6	58	32	71	49	100	33	71	1	77	75	52	24	44	32
Baseline-Oracle	100	0	69	0	78	0	100	0	84	0	11	0	47	0	25	0

Table 3: Accuracy of neural text-to-SQL systems on English question splits ('?' columns) and SQL query splits ('Q' columns). The vertical line separates datasets from the NLP (left) and DB (right) communities. Results for Iyer et al. (2017) are slightly lower here than in the original paper because we evaluate on SQL output, not the database response.

Seq2SQL data – WikiSQL (salesforce)

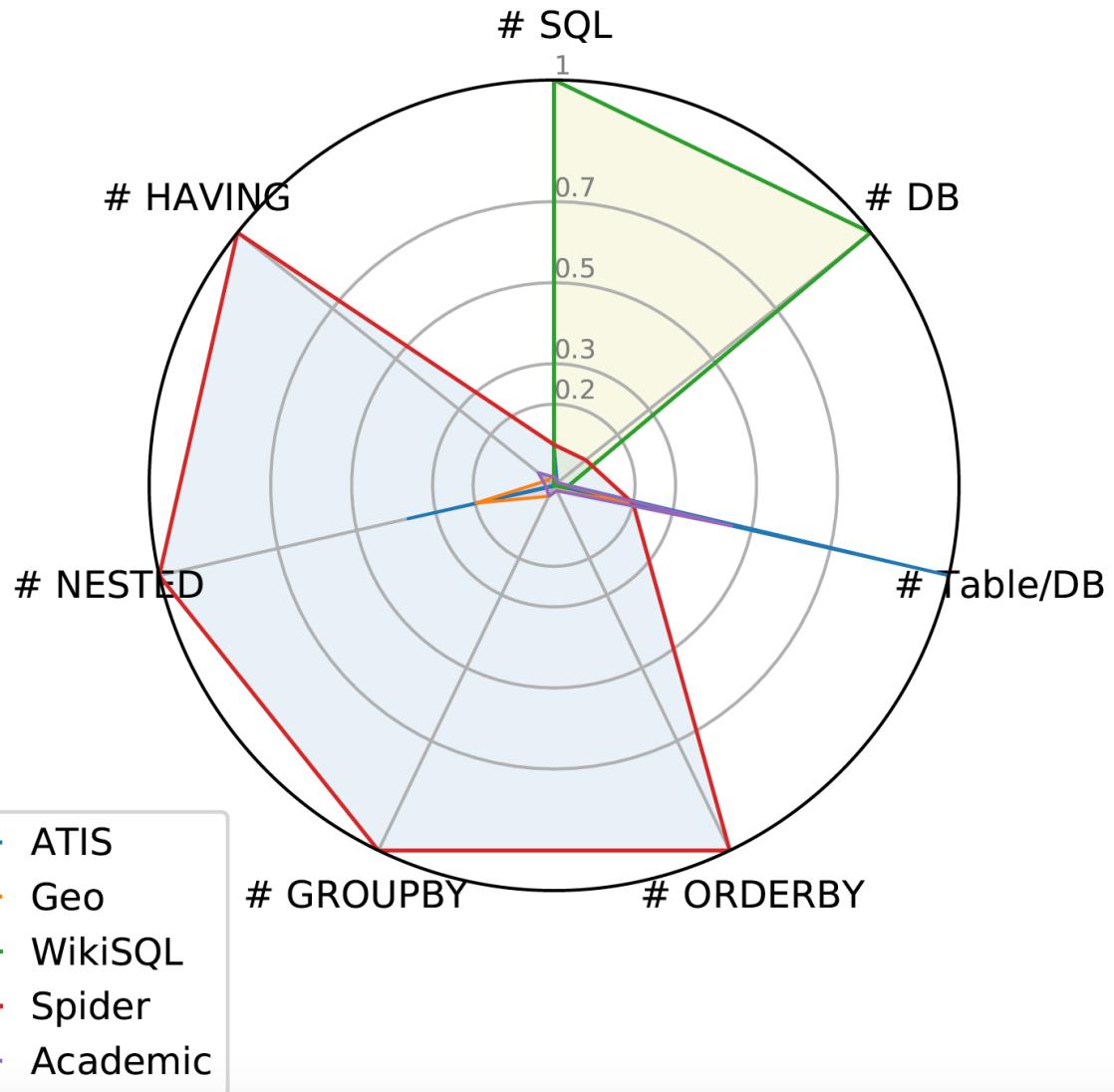
- The **first realistic seq2SQL task definition** on the top of WikiSQL makes it the most popular seq2SQL dataset
 - Databases in the test set **do not appear** in the train/dev set, which requires model to generalize to new databases
 - <https://github.com/salesforce/WikiSQL>

Seq2SQL data – WikiSQL (Salesforce paper)

- **WikiSQL - Pros**
 - The number of SQL queries and databases is huge (>20,000)
 - Databases in the test set do **not** appear in the train/dev set, which requires model to generalize to new databases
- **WikiSQL - Cons**
 - SQL queries are generated by templates and paraphrased by Turkers
 - All databases have **only one** table - not a full relational database
 - SQL **only contains** SELECT and WHERE. No GROUP BY/Nested queries etc.

Seq2SQL data - Spider

- WikiSQL is great. But it has limited **SQL coverage and a very simple schema**, which makes the task simple and less interesting



Seq2SQL data - Yale Spider

- SQL labels cover **almost all** important SQL components
- Each database **has multiple** tables and several foreign keys
- It is currently **the only** large-scale ***complex and cross-domain*** semantic parsing and text-to-SQL dataset!
- **Check it out!!!**
 - [Our Blog](#)
 - [Project Page](#): <https://yale-lily.github.io/spider>
 - [Github Page](#): <https://github.com/taoyds/spider>



Spider 1.0



Yale Semantic Parsing and Text-to-SQL Challenge

What is Spider?

Spider is a large-scale *complex and cross-domain* semantic parsing and text-to-SQL dataset annotated by 11 Yale students. The goal of the Spider challenge is to develop natural language interfaces to cross-domain databases. It consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains. In Spider 1.0, different complex SQL queries and databases appear in train and test sets. To do well on it, systems must *generalize well to not only new SQL queries but also new database schemas*.

Why we call it "Spider"? It is because our dataset is complex and cross-domain like a spider crawling across multiple complex(with many foreign keys) nests(databases).

[Spider Paper](#)

News

- **1/14/2019** The submission tutorial is out!
- **12/17/2018** We updated 7 sqlite database files ([issue 14](#)). Please download the Spider dataset from this page again.
- **10/25/2018** The evaluation script and results were updated ([issue 5](#)). Please download the latest versions of the script and papers. Also, please follow instructions in [issue 3](#) to generate the latest SQL parsing results (fixed a bug).

Leaderboard - Exact Set Match without Values

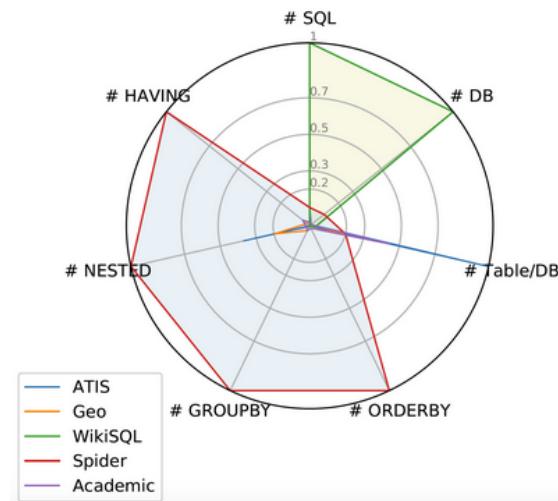
For exact matching evaluation, instead of simply conducting string comparison between the predicted and gold SQL queries, we decompose each SQL into several clauses, and conduct set comparison in each SQL clause. Please refer to the paper and the [Github page](#) for more details.

Rank	Model	Dev	Test
1 <small>Sep 20, 2018</small>	SyntaxSQLNet <i>Yale University</i> (Yu et al. EMNLP '18)	18.9	19.7
2 <small>Sep 20, 2018</small>	SQLNet <i>Shanghai Jiao Tong University</i> (Xu et al. '18)	10.9	12.4
3 <small>Sep 20, 2018</small>	TypeSQL <i>Yale University</i> (Yu et al. NAACL '18)	8.0	8.2
4 <small>Sep 20, 2018</small>	Seq2Seq + attention <i>University of Edinburgh</i> (Dong and Lapata, 2016)	1.8	4.8

Leaderboard - Execution with Value Selection



Why Spider?



As the above spider chart shows, Spider 1.0 is distinct from most of the previous semantic parsing tasks because:

- *ATIS, Geo, Academic*: Each of them contains only a single database with a limited number of SQL queries, and has exact same SQL queries in train and test splits.
- *WikiSQL*: The numbers of SQL queries and tables are significantly large. But all SQL queries are simple, and each database is only a simple table without any foreign key.

Spider 1.0 spans the largest area in the chart, making it the first complex and cross-domain semantic parsing and text-to-SQL dataset! Read more on the [blog post](#).

Getting Started

The data is split into training, development, and unreleased test sets. Download a copy of the dataset (distributed under the [CC BY-SA 4.0](#) license):

[Spider Dataset](#)

Details of baseline models and evaluation script can be found on the following GitHub site:

[Spider GitHub Page](#)

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, you can assume that a list of gold values for each question is given. Your model has to fill them into the right slots in the SQL. Is your system going to the first one showing up on this leaderboard?

Rank	Model	Dev	Test

Example Split Results

For comparison, the models achieve much higher results if we split the dataset based on data examples instead of databases since the systems don't need to generalize to new database schemas.

Rank	Model	Exact Set Match
1	TypeSQL <i>Yale University</i> (Yu et al. NAACL '18)	33.0
2	SQLNet <i>Shanghai Jiao Tong University</i> (Xu et al. '18)	18.3
3	Seq2Seq + attention <i>University of Edinburgh</i> (Dong and Lapata, 2016)	15.9

Easy

What is the number of cars with more than 4 cylinders?

```
SELECT COUNT(*)  
FROM cars_data  
WHERE cylinders > 4
```

Medium

For each stadium, how many concerts are there?

```
SELECT T2.name, COUNT(*)  
FROM concert AS T1 JOIN stadium AS T2  
ON T1.stadium_id = T2.stadium_id  
GROUP BY T1.stadium_id
```

Hard

Which countries in Europe have at least 3 car manufacturers?

```
SELECT T1.country_name  
FROM countries AS T1 JOIN continents  
AS T2 ON T1.continent = T2.cont_id  
JOIN car_makers AS T3 ON  
T1.country_id = T3.country  
WHERE T2.continent = 'Europe'  
GROUP BY T1.country_name  
HAVING COUNT(*) >= 3
```

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")
```

Click here to enter commands (e.g., help, run '<bash command>', rm <bundle>, kill <bundle>, etc.).
CodaLab>

Yale Semantic Parsing and Text-to-SQL Challenge (Spider) 1.0 Submission Guideline

[? Keyboard Shortcuts](#)[View](#) [View source](#)

In this tutorial, we'll cover the process of submitting your model and results for official evaluation. Once your model has been evaluated officially, your scores will be added to the leaderboard.

Before we begin, please create a CodaLab account, and complete the [Codalab tutorial](#). We'll begin by detailing how you can evaluate on the development set by showing how we evaluate one of our baseline model, and then delve into evaluation on our hidden test set.

Evaluating on the development set

Here we use one of our pretrained [SQLNet](#) model to predict on the development set to show you how Codalab works. Please make sure your model generates prediction with same format.

Glove dataset

First we uploaded the pretrained glove_42B_300d wordvec downloaded from [here](#). It is needed for the SQLNet model.

uuid[0:8]	name	summary	data_size	state	description
0xd1ffde	glove	[uploaded]	4.7g	ready	glove_42b_300d

Spider dataset

Next we uploaded spider dataset and put them in the Data/ which contains dev.json and tables.json. Those were directly downloaded from [Spider](#) which has the updated version.

uuid[0:8]	name	summary	data_size	state	description
0xb357bf	data	[uploaded]	4.4m	ready	contains spider dataset. Specificall tables.json and dev.json
0xd3bf68	dev_gold.sql	[uploaded]	121k	ready	
0x853385	database	[uploaded]	839m	ready	

Pretrained baseline models

Here we uploaded SQLNet model as our pretrained baseline models. Can be found [here](#).

Why complex and cross-domain?

For building a natural language interface to relational databases, we want it to automatically process:

- **The full SQL syntax - Complex**
 - Includes different SQL structures (e.g. nested) and unseen SQL in training
 - Covers all SQL clauses including SELECT, WHERE, GROUP BY, ORDER BY
- **Queries scaling to unseen databases**
 - You do not want to relabel data and retrain your model specific to a new database

Seq2SQL dataset summary

- **Traditional seq2SQL datasets and tasks:** Data size is small with a little problematic task setting
- **(Finegan-Dollak et al., 2018) introduces the idea of resplitting the datasets based on SQL queries**
- **WikiSQL makes the task more realistic by requiring generalization to new databases but with some limitations on complexity**
- **Spider moves WikiSQL limitations, and becomes the only large complex and cross-domain seq2SQL task**

Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task

Tao Yu Rui Zhang Kai Yang Michihiro Yasunaga

Dongxu Wang Zifan Li James Ma Irene Li

Qingning Yao Shanelle Roman Zilin Zhang Dragomir R. Radev

Department of Computer Science, Yale University

{tao.yu, r.zhang, k.yang, michihiro.yasunaga, dragomir.radev}@yale.edu

Need for New Semantic Parsing Data

Introduction: Semantic parsing (SP) is one of the most important tasks in natural language processing (NLP). It has been studied for decades. However, the lack of large size SP data seriously limits the continuous progress in this field!

A large, high quality dataset is essential for new progress in SP

Why is Data Scarce? High quality SP data creation is hard.

- Need to understand the database schema, ask questions and then write SQL query answers.
- Qualified annotators are hard to find, cannot be Turkers.
- There are only few public databases with multiple tables available

Problems in Existing Datasets: Previous SP data can be grouped into the two following groups:

- **ATIS, Geo, Academic:** Each of them contains only a single database with a small number of SQL queries (< 500), and has exact same SQL queries in train and test splits.
more like semantic matching, limited to a specific database
- **WikiSQL:** The numbers of SQL queries and tables are significantly larger. But all SQL queries are simple, and each database is only a simple table without any foreign key.
Only simple semantic cases (simple SQL and DB schema)

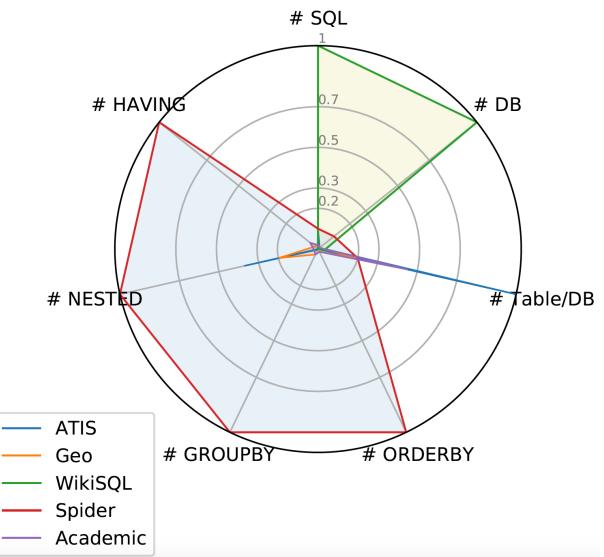
Meet the Need: Spider

Spider: is a large *complex and cross-domain* SP/text-to-SQL dataset.

- consists of 10,181 questions, 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains.
- different complex SQL queries and databases appear in the training and test sets. To do well on it, systems must generalize well to not only new SQL queries but also new database schemas.

Why Spider?

Data Comparison: In the following spider chart, *Spider* spans the largest area, making it the ***first complex and cross-domain*** text-to-SQL dataset!

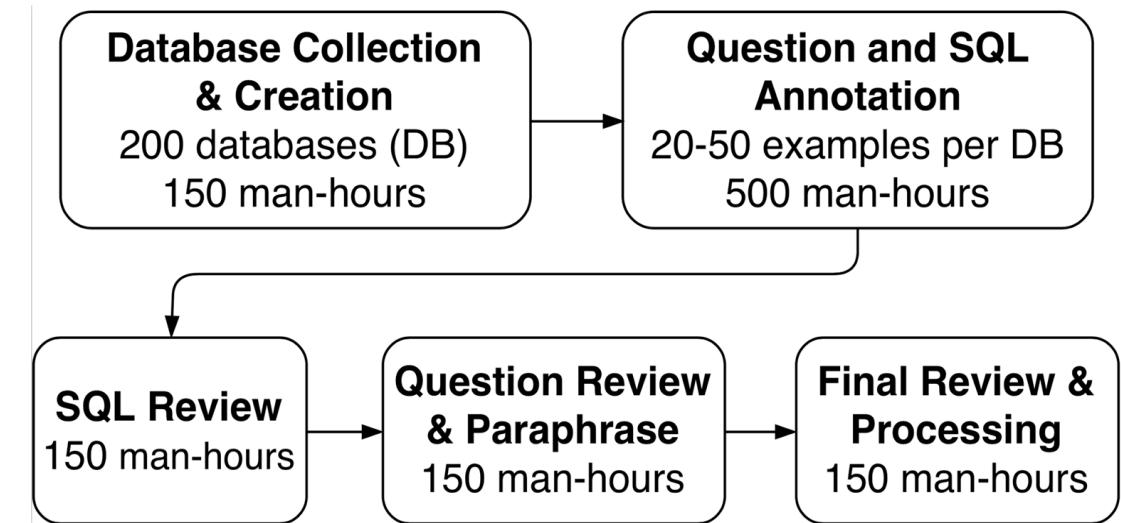
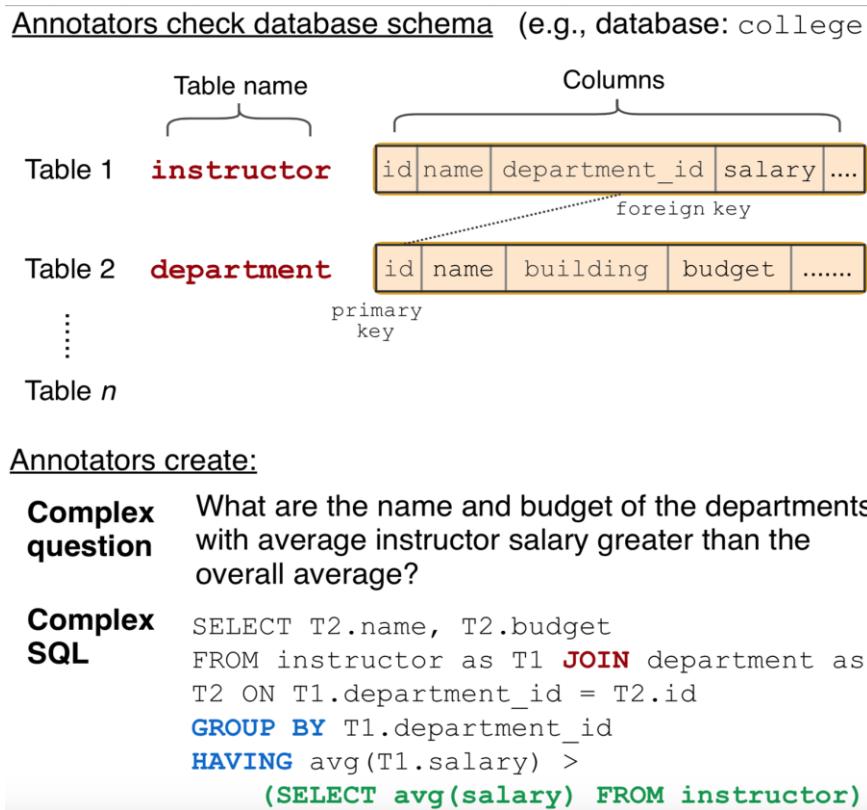


Why it is a large complex and cross-domain dataset?

- **Large:** Over 10,000 questions, and about 6,000, unique SQL queries
- **Complex:** Our SQL queries cover almost all important SQL components. All databases have multiple tables with foreign keys.
- **Cross-domain:** Split 200 DBs into train/dev/test based on database. Thus, *Spider* tests model's performance on unseen databases.

Corpus Construction

All questions and SQL queries were written and reviewed by 11 Yale CS students. As illustrated in the following figures, we developed our dataset in five steps, spending around 1,000 hours of human labor in total.



New Challenging SP/Text-to-SQL Task

A New SP/Text-to-SQL Task: On top of the proposed dataset, we define *the most realistic and challenging* SP/text-to-SQL task so far. Unlike most previous SP tasks, in the Spider task

models will be tested on both different complex SQL queries and databases in different domains

- models can only make right predictions by truly understanding the semantic meaning of the questions, rather than by memorization.
- ⇒ tests model's ability to generalize to new databases.

		Test					Dev All
		Easy	Medium	Hard	Extra Hard	All	
Example 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	
Database Split							
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.

Method	SELECT	WHERE	GROUP BY	ORDER BY	KEYWORDS
Example Split					
Seq2Seq	23.3	4.9	15.3	9.2	17.9
Seq2Seq+Attention	31.1	9.1	28.2	20.8	21.4
Seq2Seq+Copying	28.2	8.3	25.5	21.3	19.0
SQLNet	59.8	32.9	35.9	65.5	76.1
TypeSQL	77.3	52.4	47.0	67.5	78.4
Database Split					
Seq2Seq	13.0	1.5	3.3	5.3	8.7
Seq2Seq+Attention	13.6	3.1	3.6	9.9	9.9
Seq2Seq+Copying	12.0	3.1	5.3	5.8	7.3
SQLNet	44.5	19.8	29.5	48.8	64.0
TypeSQL	36.4	16.0	17.2	47.7	66.2

Table 3: F1 scores of Component Matching on all SQL queries on Test set.

Experiments

Evaluation Metric: Measure whether the predicted query as a whole matches the gold query using set comparison for each SQL clause.

Performances of Previous SP Models: All prior models achieve low performances, indicating that our task is challenging and there is still a large room for improvement. There is a big gap in models' results under DB or SQL splits.

	Example Split	Test					Dev All
		Easy	Medium	Hard	Extra Hard	All	
SQL Generation Performance (%)							
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	
Database Split							
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	

Conclusion

We introduced *Spider*, a large, complex and cross-domain SP and text-to-SQL dataset. Based on Spider, we define a new challenging and realistic semantic parsing task. Experimental results on several SP models on this task suggest a large room for improvement.

Acknowledgements: We thank Graham Neubig, Tianze Shi, Catherine Finegan-Dollak, and three anonymous reviewers for their discussion and feedback. We also thank Barry Williams for providing part of our database schemas from the DatabaseAnswers.

SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task

Tao Yu Michihiro Yasunaga Kai Yang Rui Zhang

Dongxu Wang Zifan Li Dragomir R. Radev

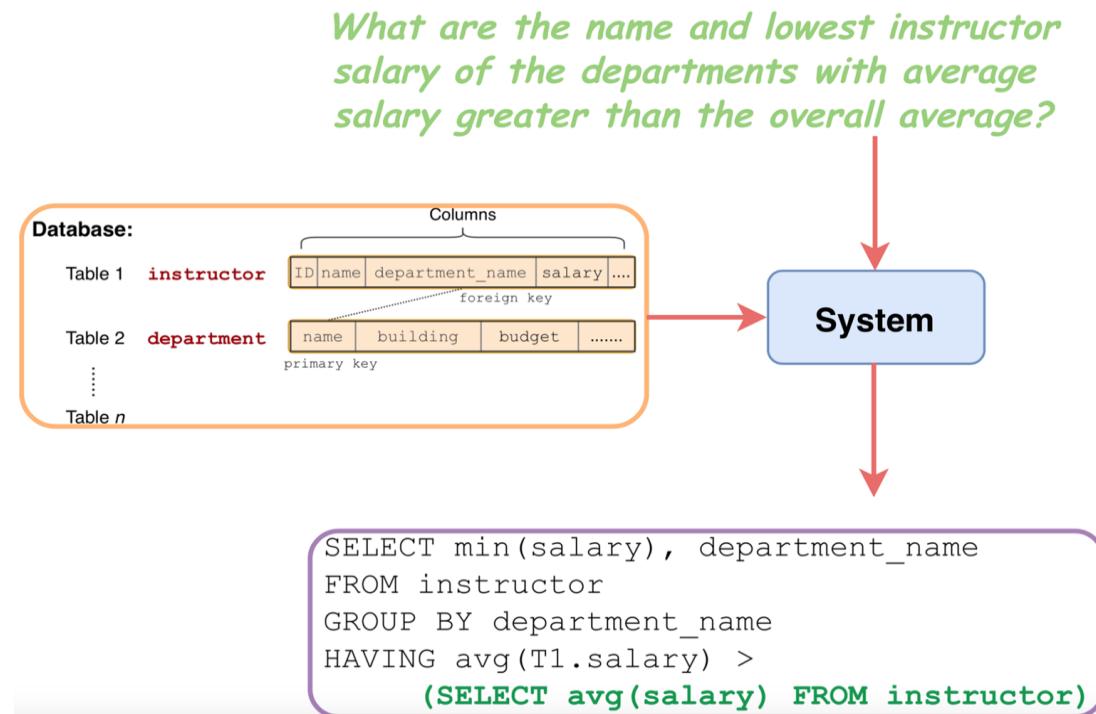
Department of Computer Science, Yale University

{tao.yu, michihiro.yasunaga, k.yang, r.zhang, dragomir.ralev}@yale.edu

Overview

Goal: build a text-to-SQL system that can

- generate **complex** SQL queries with multiple clauses or sub-queries given natural language questions
- also work on **unseen/cross-domain** databases



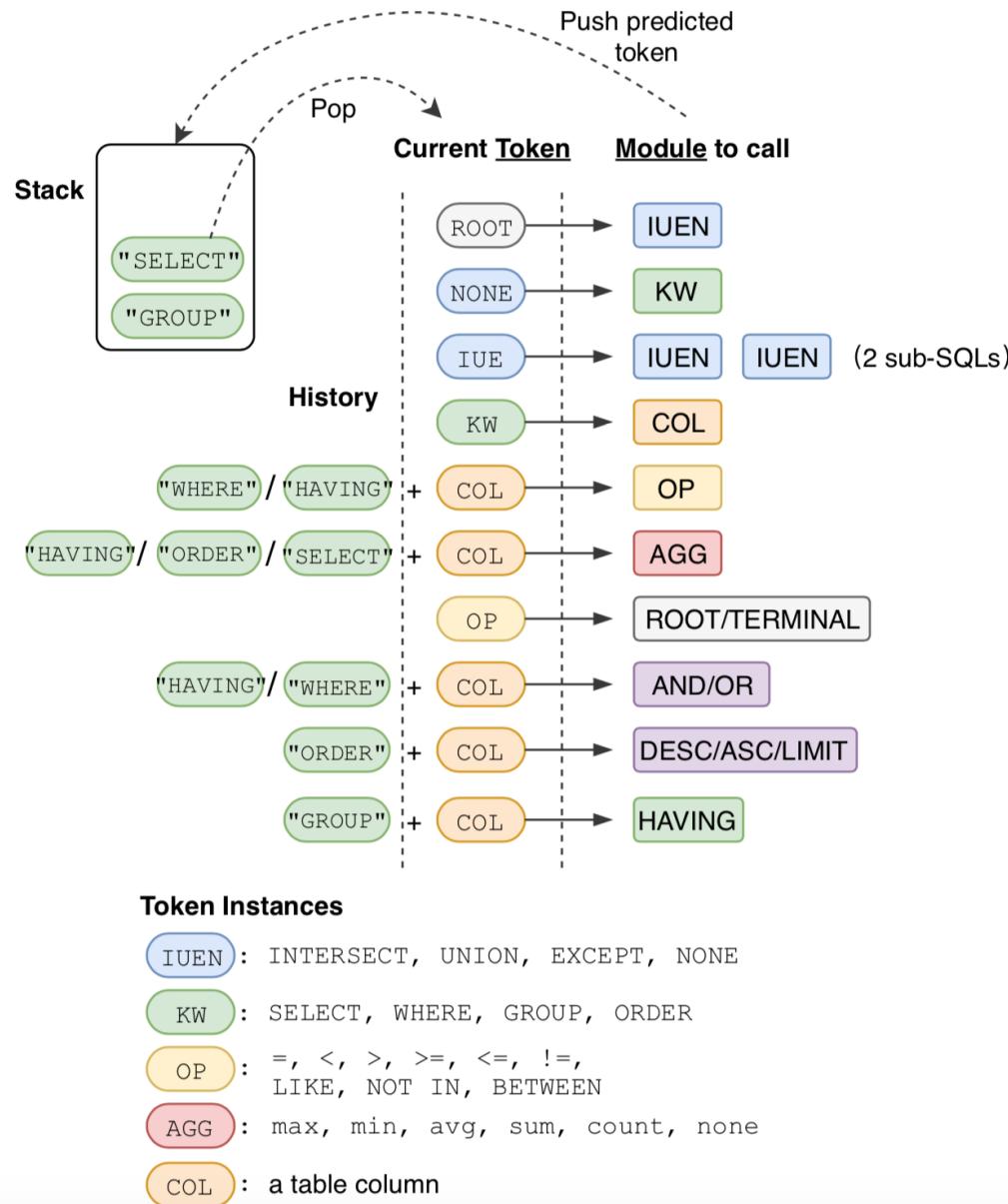
Approach

Main Ideas:

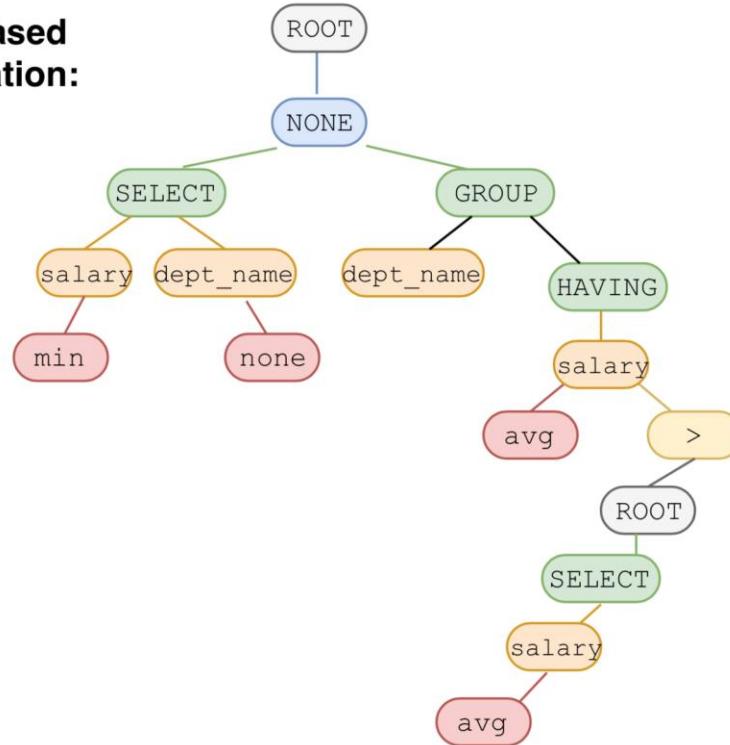
- make use of a SQL specific syntax grammar to structure the decoding process.
- introduce a SQL specific syntax tree-based decoder with SQL path history and table-aware column attention encoders.
- develop a cross-domain data augmentation method to generate more diverse training examples across databases.

Syntax Tree Networks

- decomposes the SQL decoding process into 9 modules to handle the prediction of different SQL components such as keywords, operators, and columns.
- SQL path history is pass through all modules during decoding
- in order to structure our decoder to generate complex queries, we consider a SQL grammar that determines which module to be invoked at each recursive decoding step.



Our tree-based SQL generation:



Current Directions

- **Task-Oriented Dialogue-to-SQL Task**
 - Mapping natural language questions against cross-domain databases to corresponding SQL states as a dialog progress between users and systems
 - conversational dialog interactions between users and systems, which is more challenging and interactive.

Task-Oriented Dialog-to-SQL

Good morning, how can I help?

I want to know the average salary of all professors.

```
SELECT avg(salary) FROM instructor
```

Ok, wait a moment.

The average salary is 200k.

Great, how about the numbers in different departments?

```
SELECT T1.department_id  
FROM instructor as T1 JOIN department as T2 ON T1.department_id = T2.id  
GROUP BY T1.department_id
```

Did you mean the average salary by the departments?

Exactly!

```
SELECT avg(T1.salary), T1.department_id  
FROM instructor as T1 JOIN department as T2 ON T1.department_id = T2.id  
GROUP BY T1.department_id
```

Here is the result table that shows the average salary in each department.

Hmm, could you please also find the departments with average instructor salary greater than the overall average?

```
SELECT *  
FROM instructor as T1 JOIN department as T2 ON T1.department_id = T2.id  
GROUP BY T1.department_id HAVING avg(T1.salary) > (SELECT avg(salary) FROM instructor)
```

The department table has the following columns, Which ones do you want to return?

Please just show the department name and budget.

```
SELECT T2.name, T2.budget  
FROM instructor as T1 JOIN department as T2 ON T1.department_id = T2.id  
GROUP BY T1.department_id HAVING avg(T1.salary) > (SELECT avg(salary) FROM instructor)
```

How about their average salary?

No, that is fine.

```
SELECT T2.name, T2.budget  
FROM instructor as T1 JOIN department as T2 ON T1.department_id = T2.id  
GROUP BY T1.department_id HAVING avg(T1.salary) > (SELECT avg(salary) FROM instructor)
```

Ok, what else can I help?

No, thanks for your help, bye!

Exist

Thank you. goodbye!

Complex question What are the name and budget of the departments with average instructor salary greater than the overall average?

User

User utterance

I want to know the average salary of all professors.

System

Turn goals and requests

SELECT avg(salary) FROM instructor

System actions

CONFIRM(SQL, result table)

System response

The average salary of all instructors is 200k.

Great, how about the numbers in different departments?

SELECT avg(T1.salary), T1.department_id
FROM instructor as T1 JOIN department as T2
ON T1.department_id = T2.id
GROUP BY T1.department_id

CLARIFY(SQL)

Did you mean the average salary by the departments?

Exactly!

SELECT avg(T1.salary), T1.department_id
FROM instructor as T1 JOIN department as T2
ON T1.department_id = T2.id
GROUP BY T1.department_id

CONFIRM(SQL, result table)

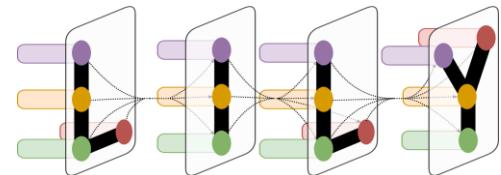
Here is the result table that shows the average salary in each department. For example, the average of CS professors is 250k.

Hmm, could you please also find the departments with average instructor salary less than the overall average?

SELECT *
FROM instructor as T1 JOIN department as T2
ON T1.department_id = T2.id
GROUP BY T1.department_id
HAVING avg(T1.salary) <
(SELECT avg(salary)
FROM instructor)

CONFIRM(SQL, result table)

The result table gives all info of the departments where their instructors' average salary is lower than the average salary of all instructors.



Please just show the department name and budget.

SELECT T2.name, T2.budget
FROM instructor as T1 JOIN department as T2
ON T1.department_id = T2.id
GROUP BY T1.department_id
HAVING avg(T1.salary) <
(SELECT avg(salary)
FROM instructor)

CONFIRM(SQL, result table)

These are the names and budgets of the departments where instructors' average salary is lower than the average salary of all instructors. For example, the LING department has 800k budget.

Ok, how about those above the overall average?

SELECT T2.name, T2.budget
FROM instructor as T1 JOIN department as T2
ON T1.department_id = T2.id
GROUP BY T1.department_id
HAVING avg(T1.salary) >
(SELECT avg(salary)
FROM instructor)

CONFIRM(SQL, result table)

These are the names and budgets of the departments where instructors' average salary is greater than the average salary of all instructors. For example, the CS department has 2,000k budget.

Great! Thank you very much.

THANK YOU()

GOODBYE()

Thank you. goodbye!

Text-to-SQL Next Steps

- **Human-in-the-loop interactions**
 - via a **dialogue**
 - double check and get feedback from users
 - multi-turns interactions b/w systems and users via human language
 - via **interactive visualization systems**
 - gain feedback and hints from the user interactive interface
 - incorporate users in some subtasks that are ambiguous or challenging for the model

Dialog2SQL: Motivation

- **Motivation**
 - most of users don't know how to ask complex questions in a single turn
 - need multiple turns of interactions b/w users and the system to clarify some questions
 - no need to conduct text-to-SQL in one shot
 - more natural way is to involve dialogs or interactive interfaces
 - users' data exploration experience is also very important
 - combine Dialog2SQL & Interact2SQL ⇒ a real-world NLIDB application

Dialog2SQL: General Ideas

- **Dialog2SQL**
 - **novel work: task-oriented Dialog + text-to-SQL task**
 - **Goal:** mapping natural language questions against cross-domain databases to corresponding SQL states as a dialog progress between users and systems.
 - **Why no previous work?** No dataset! Will built upon the Spider
 - **Can be used for multiple tasks:**
 - Dialog state tracking, dialogue act
 - Dialog response generation
 - Semantic parsing
 - Structured data (table results/SQL) to text

Dialog2SQL: General Ideas

- Consider the complex question in Spider:
What are the name and budget of the departments with average instructor salary greater than the overall average?
- Use the complex question as the query goal, and decompose it into a dialog b/w the user and the system.
- Task: given multi-turns natural language interactions and DB schema -> map each interaction to SQL/actions



Dialog2SQL: Task Assumptions

- **Who are the most reasonable users/usage scenarios?**
 - Users without any SQL/DB background
 - don't know what questions to ask, how to guide/interact with them in order to ask “good”/SQL related questions
 - not like Alexa/Siri, the usage scenarios of our system are more professional working environment
 - Users with some SQL/DB background
 - might ask some unrelated questions and ones that cannot be converted to SQL
 - What different task settings compared to the above users?

Dialog2SQL: Data Collection

- **Dialog2SQL**
 - two annotators: one plays the user, another plays the assistant
 - quality control: two other annotators double check if the dialog meets some requirements
 - does the assistant have to know SQL/DB? Not usually
 - group questions based on SQL types, and create an assistant-user interaction example (show to turkers in training) and a list of must-to-check things (show to turkers during dialog creation and double check)
 - should the assistant be given the complex user's query goal? even for users, how to show the goal?
 - Each interaction - **one SQL label or an act type**

Dialog2SQL: Data Collection

- **Wizard-of-Oz (WOZ): the “standard” way of creating dialog data**
 - Annotator enters response for one turn, repeat until dialog ends
 - use a checklist to track if the user’s intent is clear and dialog should be ended.
 - more natural/diverse and wide coverage (but long-tail). how to control the dialog flow?
- **Combining automation & human intelligence**
 - First automatically generate some unnatural dialog interactions, then ask annotators to rewrite

Dialog2SQL: User Side

Help Desk: Hello, welcome to the TownInfo centre. I can help you find a restaurant or hotel, look for tourist information, book a train or taxi. How may I help you ?

Customer : I want a place to stay in the east.

Help Desk : I have 6 guesthouses and 1 hotel on the east side. What's your price range?

Customer : Doesn't matter too much. I'd like a 4 star property, though, and would prefer one of the guesthouses.

Help Desk : I'd recommend 517a coldham lane. Would you like me to book a room?

Customer : Could you give me their phone number? I would like to verify that they have free parking.

Help Desk : Allenbell does have parking and the phone is 01223210353. Can I help with anything else?

Next turn

Customer : (Your response)

you need to go through the dialogue first by clicking the 'next turn' button

30

What topics were mentioned in this turn:

General: Booking: Restaurant: Hotels: Attraction:

Hospital: Police: Train: Taxi: Bus:

Submit the HIT

Please try to chat about the following topic:

Task MUL0002:

- You are traveling to and looking forward to try local restaurants.
- You are looking for a **place to stay**. The hotel should be in the **east** and should **include free parking**.
- The hotel should have **a star of 4** and should be in the type of **guesthouse**.
- Make sure you get **address** and **phone number**.
- You are also looking for a **place to dine**. The restaurant should be in the **moderate** price range and should serve **australian** food.
- If there is no such restaurant, how about one that serves **turkish** food.
- Once you find the **restaurant** you want to book a table for **4 people** at **17:45 on friday**.
- Make sure you get the **reference number**

--- The End ---

Figure A1: Interface from the User side

Dialog2SQL: User Side

User Request:

Show the organizer and name for churches that opened between 1830 and 1840.

- The dialog needs to be at least 10 lines.
- Every time you receive an answer, you can use the table shown below to briefly check if it's correct.
- When you receive ALL the information asked in the request, you can finish the dialog in a natural way (e.g., Thank you, goodbye), and click the "the dialog is finished" button to end the task.

Dialog with The Assistant

Assistant: XXX

You: XXX

Enter what you want to say here

The dialog is finished

DB Table for Checking

Dialog2SQL: System Side

- **Task Settings**

- System mainly response with query results, summarize returned results, sometimes question clarification
- No need to request info from user much
- System need to access to the database content
- Group q-SQL into different groups, **how to find results?**
 - Create a checklist to help system prevent as a professional DB worker
 - How to create this checklist for different SQL types?
 - What if users ask different questions that require different SQL
 - Data interactive visualization interface to find results by clicks
 - Time-consuming
 - How to make sure results are right
 - Some complex questions can't be answered by clicks?
- What if users ask some unexpected SQL questions?

System Side: SQL query build process for cloud workers

- **Take advantages of given info**
 - We already have complex SQL label for users' query goals
 - We can group all question-SQL pairs based on SQL components into 20~30 groups. Ask workers to label only several groups. Structures would be very similar.
 - We can join tables in the given SQL into a single big table, just show this table. No need to join tables and consider complexity of DB (maybe not join together since sometimes on join at the beginning)

System Side: SQL query build process for cloud workers

- **At the beginning**
 - Provide the final SQL answer skeleton for this group (without columns and values to avoid generated dialogs overfit, but give assistant some prior knowledge of this kind of SQL answer)
 - Provide possible intermediate SQL & question examples (from other DB)
 - Build an **easy-to-understand/operate interface** to write SQL indirectly.
- **At the end**
 - Show the gold SQL let them double check and correct SQL labels

Dialog2SQL: System Side

- Your job is to play the role of an assistant, help the user find the information by getting information from the given database.
- When the user indicates that all the information they want have been found, you can finish the dialog in a natural way, and click the “the dialog is finished” button to end the task.

Dialog with The Assistant

Assistant: XXX

You: XXX

Enter what you want to say here

The dialog is finished

Query Generate Template

1. Which information column do you want to select from the table below?
(id, name, location, year...allow workers to click and highlight as selection)

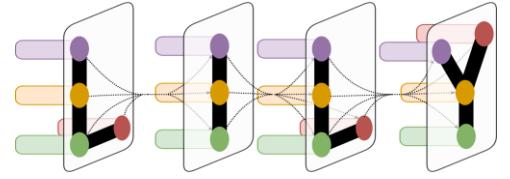
2. Want to set up limitations to narrow down answer range?

Location (a drop down menu for >, <, =, include, ..., operations) <value>

Year (a drop down menu for >, <, =, include, ..., operations) <value>

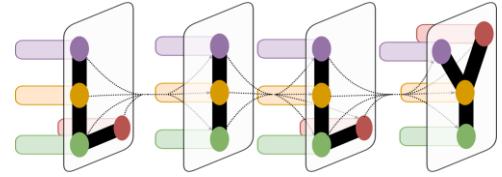
DB Table (content changes based on the assistant's input)

Current Status



- **Dialog Goals:** 6000-8000 dialogs, 200 databases, to be done by 2/15

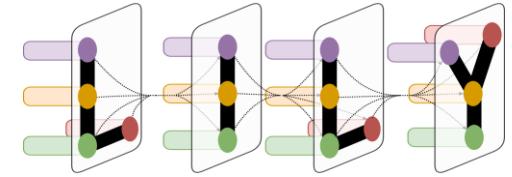
Takeaways



- Fully domain independent, scaleable, NL interfaces to DB are not happening soon
- Traditional methods are brittle and overfit to the training data
- The methodologies for evaluation need to be rethought
- Large annotated data sets are needed for training
 - WikiSQL, Spider
- Dialogue can be used to construct queries interactively

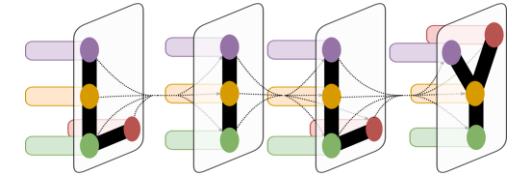
Relevant Papers

- *SyntaxSQLNet*: [Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task](#)
- *Zero-shot Parser*: [Decoupling Structure and Lexicon for Zero-Shot Semantic Parsing](#)
- *A Detailed Analysis of WikiSQL*: [What It Takes to Achieve 100% Condition Accuracy on WikiSQL](#)
- *Coarse2fine*: [Coarse-to-Fine Decoding for Neural Semantic Parsing](#)
- *SQL Evaluation Methodology*: [Improving Text-to-SQL Evaluation Methodology](#)
- *User Feedback via Dialog*: [DialSQL: Dialogue Based Structured Query Generation](#)
- *Involving Context in the Task*: [Learning to Map Context-Dependent Sentences to Executable Formal Queries](#)
- *TypeSQL*: [TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation](#)
- *SQLNet*: [SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning](#)
- *Seq2SQL*: [Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning](#)
- *Syntactic Neural Models*: [A Syntactic Neural Model for General-Purpose Code Generation](#), [Abstract Syntax Networks for Code Generation and Semantic Parsing](#)
- *NL2SQL*: [Learning a Neural Semantic Parser from User Feedback](#)
- *Seq2Tree*: [Learning a Neural Semantic Parser from User Feedback](#)
- *NaLIR*: [Constructing an Interactive Natural Language Interface for Relational Databases](#)
- *Paraphrase*: [Cross-domain Semantic Parsing via Paraphrasing](#)
<https://medium.com/@tao.yu/awesome-sequence-to-sql-and-semantic-parsing-1d7656861679>
- *NL2WebAPI*: [Building Natural Language Interfaces to Web APIs](#)



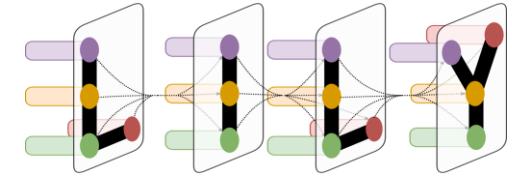
Other Materials

- [Spider: One More Step Towards Natural Language Interfaces to Databases](#)
- [How to Talk to Your Database](#)
- [ACL 2018 Tutorial on Neural Semantic Parsing](#)
- [Natural Language Data Management and Interfaces](#)
- [A Syntactic Neural Model for General-Purpose Code Generation](#)
- [Learning to Map Context-Dependent Sentences to Executable Formal Queries](#)



Datasets and Code

- [Spider: a large-scale *complex and cross-domain* semantic parsing and text-to-SQL dataset](#)
- [WikiSQL](#)
- [8 traditional single datasets: ATIS, GeoQuery, Academic, Advising, Scholar etc.](#)
- [More on NLP progress page](#)
- [Coarse2fine](#)
- [SyntaxSQL](#)
- [TypeSQL](#)
- [SQLNet](#)
- [NL2Code](#)



NLseq2SQL



<http://lily.yale.edu>