Carnegie Mellon University
School of Computer Science

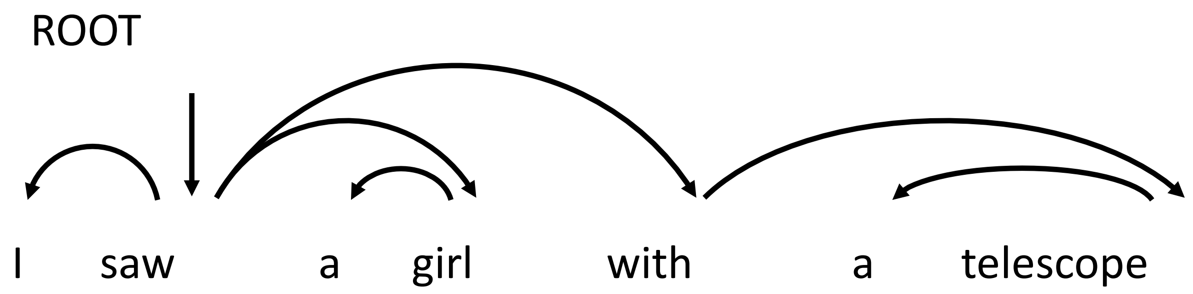
CS11-747 Neural Networks for NLP
Generate Trees Incrementally

Graham Neubig
gneubig@cs.cmu.edu
Language Technologies Institute
Carnegie Mellon University

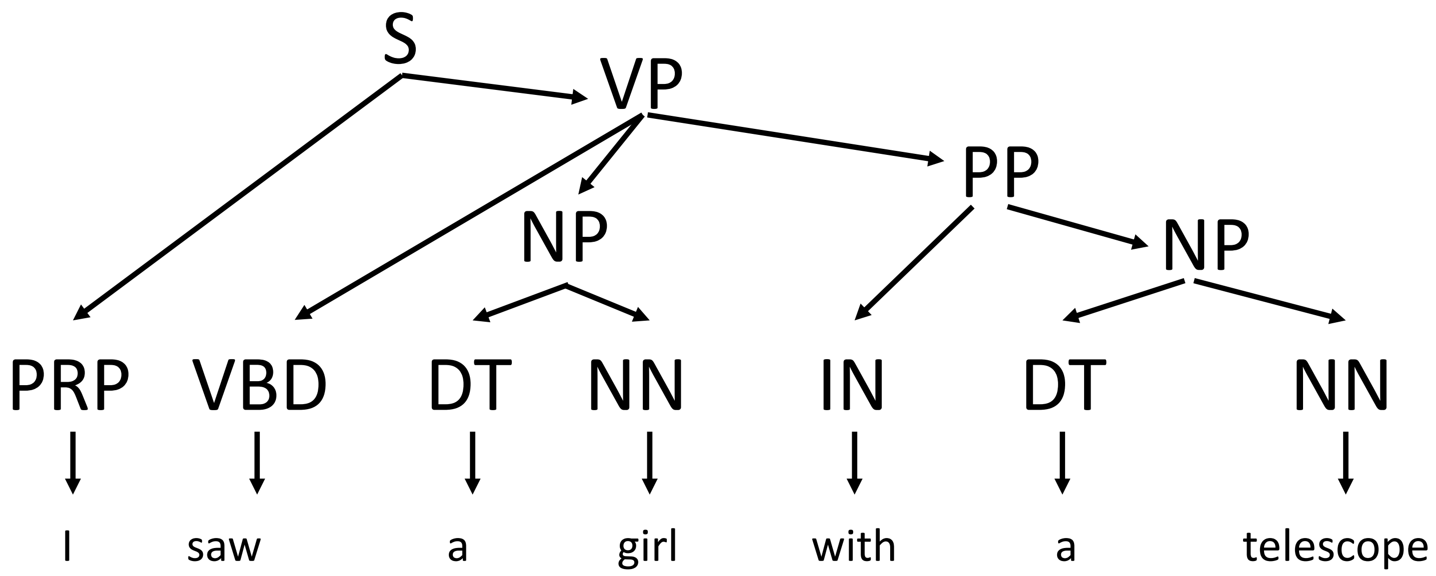
Language
Technologies
Institute

The Two Two Most Common of Linguistic Tree Structures

- **Dependency Trees** focus on relations between words



- **Phrase Structure** models the structure of a sentence



Semantic Parsing: Another Representative Text-to-Structure Task

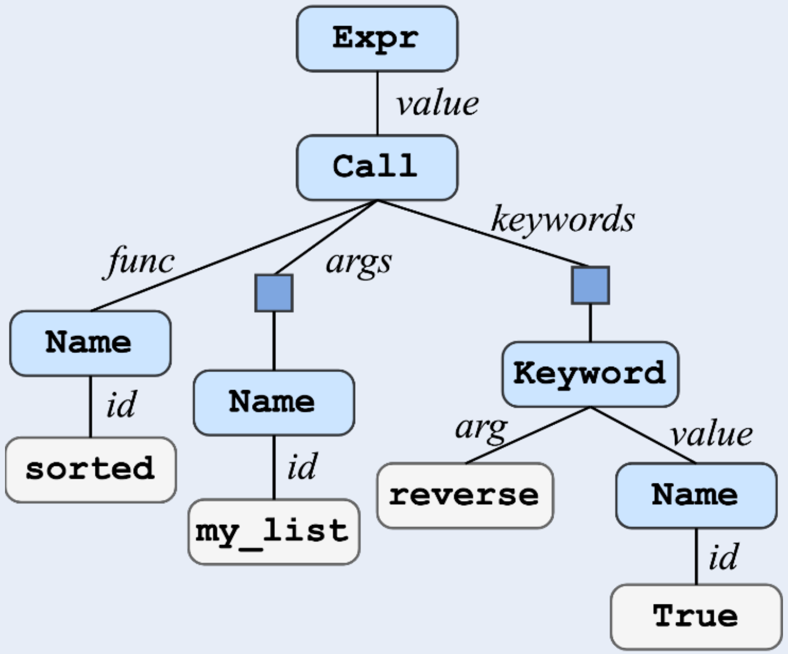
Transform Natural Language Intents to Executable Programs

Sort my_list in descending order

```
sorted(my_list, reverse=True)
```

Example: Python code generation

Structured Meaning Representations



Abstract Syntax Trees

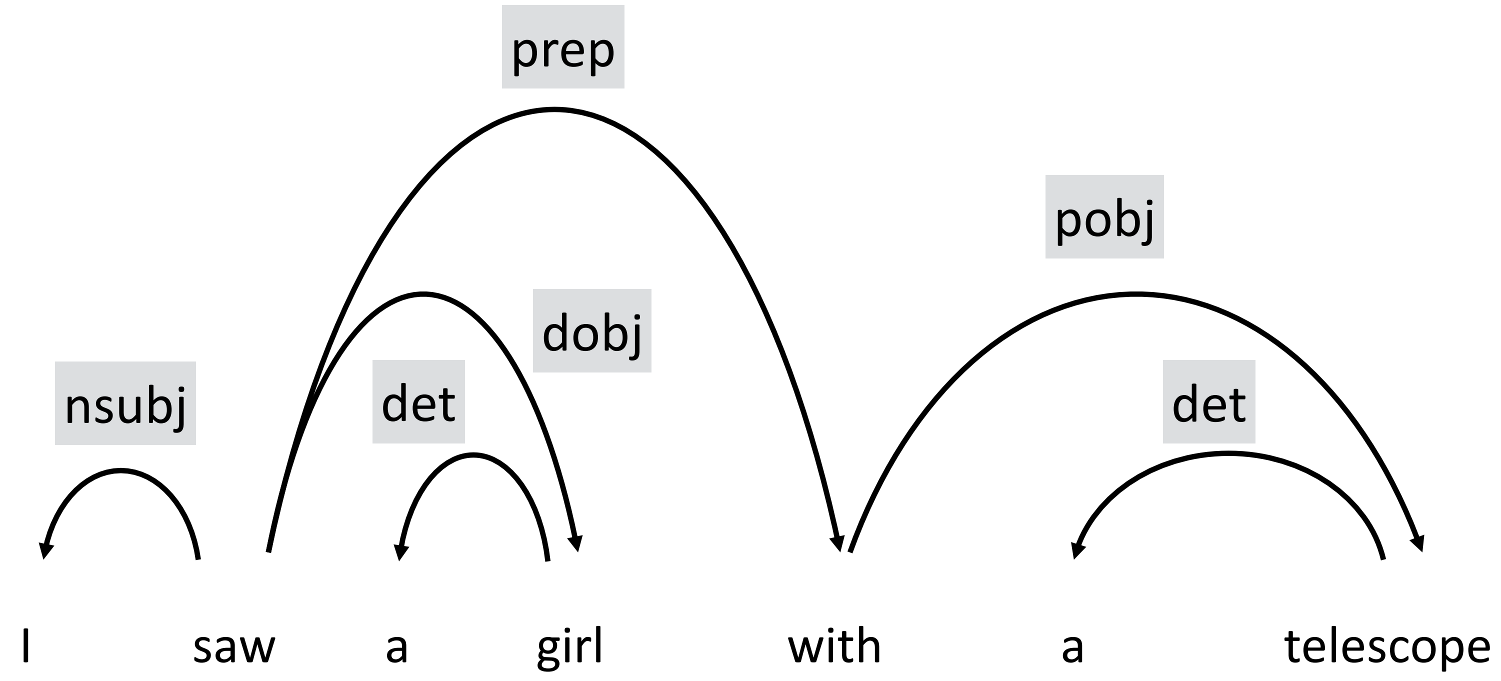
Parsing: Generate Linguistic Structures of Sentences

- Predicting linguistic structure from input sentences
- **Transition-based models**
  - step through actions one-by-one until we have output
  - like history-based model for POS tagging
- **Dynamic Programming-based models**
  - calculate probability of each edge/constituent, and perform some sort of dynamic programming
  - like linear CRF model for POS

Shift-reduce Dependency Parsing

Why Dependencies?

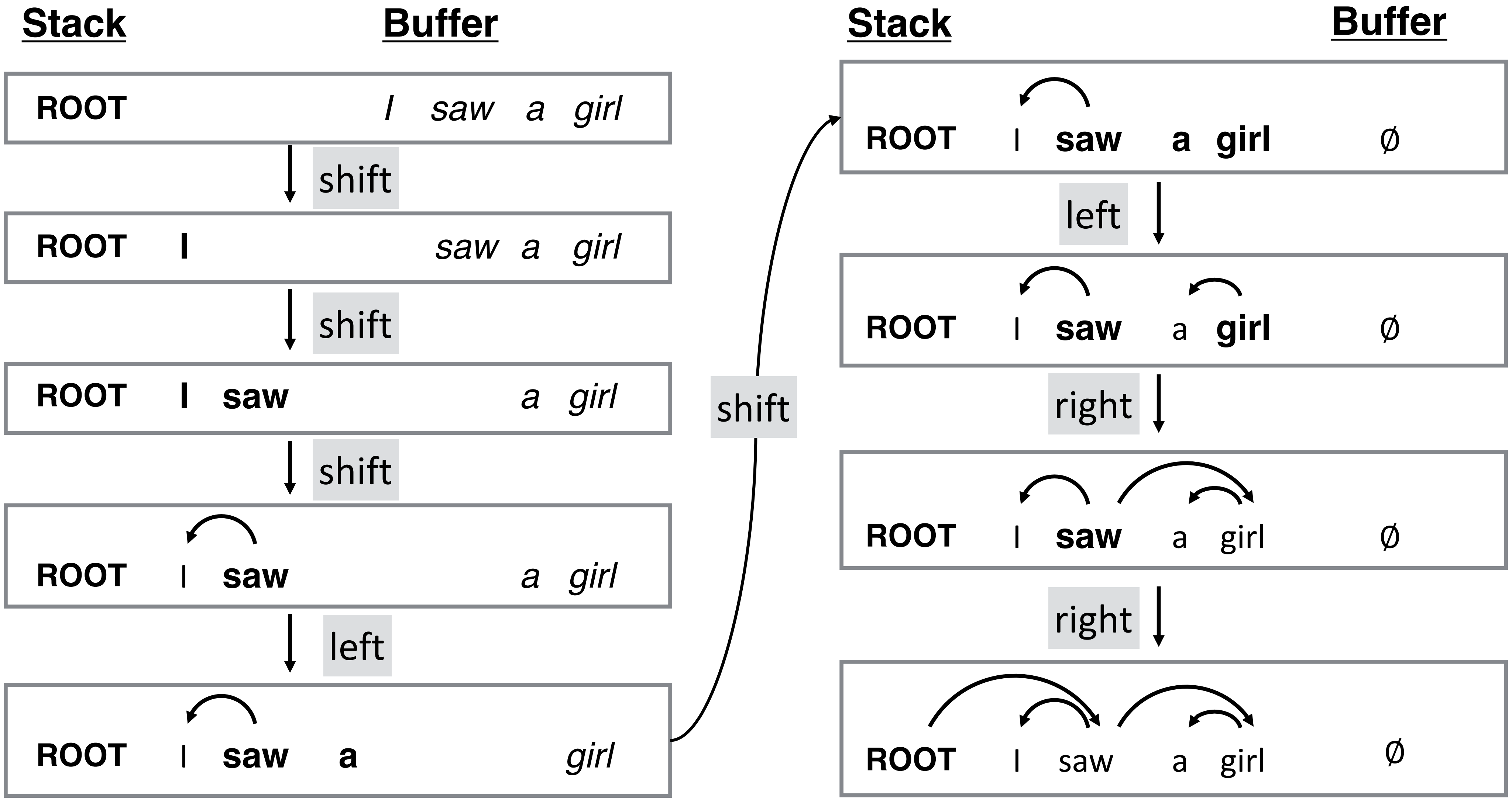
- Dependencies are often good for semantic tasks, as related words are close in the tree
- It is also possible to create labeled dependencies, that explicitly show the relationship between words



Arc Standard Shift-Reduce Parsing (Yamada & Matsumoto 2003, Nivre 2003)

- Process words one-by-one left-to-right
- Two data structures
  - Queue: of unprocessed words
  - Stack: of partially processed words
- At each point choose
  - shift: move one word from queue to stack
  - reduce left: top word on stack is head of second word
  - reduce right: second word on stack is head of top word
- Learn how to choose each action with a classifier

Shift Reduce Example



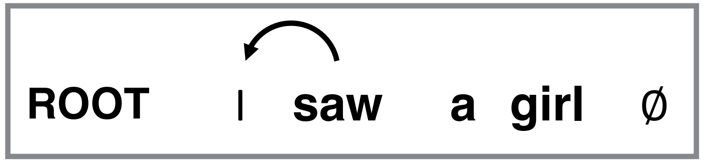
Classification for Shift-reduce

- Given a configuration

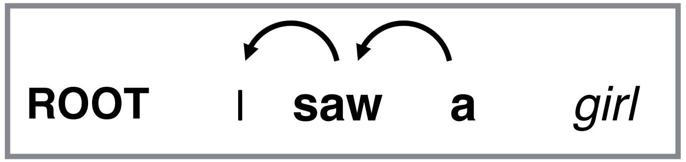
Stack **Buffer**

- Which **action** do we choose?

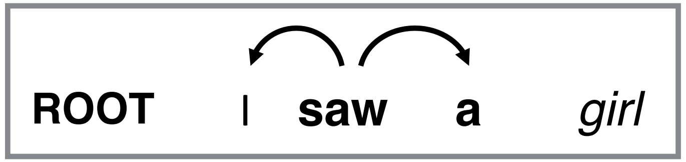
shift



left



right



Making Classification Decisions

- Extract features from the configuration
  - what words are on the stack/buffer?
  - what are their POS tags?
  - what are their children?
- Feature combinations are important!
  - Second word on stack is verb **AND** first is noun: “right” action is likely
- Combination features used to be created manually (e.g. Zhang and Nivre 2011), now we can use neural nets!

Alternative Transition Methods

- All previous methods did left-to-right
- Also possible to do **top-down** -- pick the root first, then descend, e.g. Ma et al. (2018)
- Also can do **easy-first** -- pick the easiest link to make first, then proceed from there, e.g. Kiperwasser and Goldberg (2016)

A Feed-forward Neural Model for Shift-reduce Parsing

(Chen and Manning 2014)

A Feed-forward Neural Model for Shift-reduce Parsing
(Chen and Manning 2014)

- Extract non-combined features (embeddings)
- Let the neural net do the feature combination

Softmax layer:

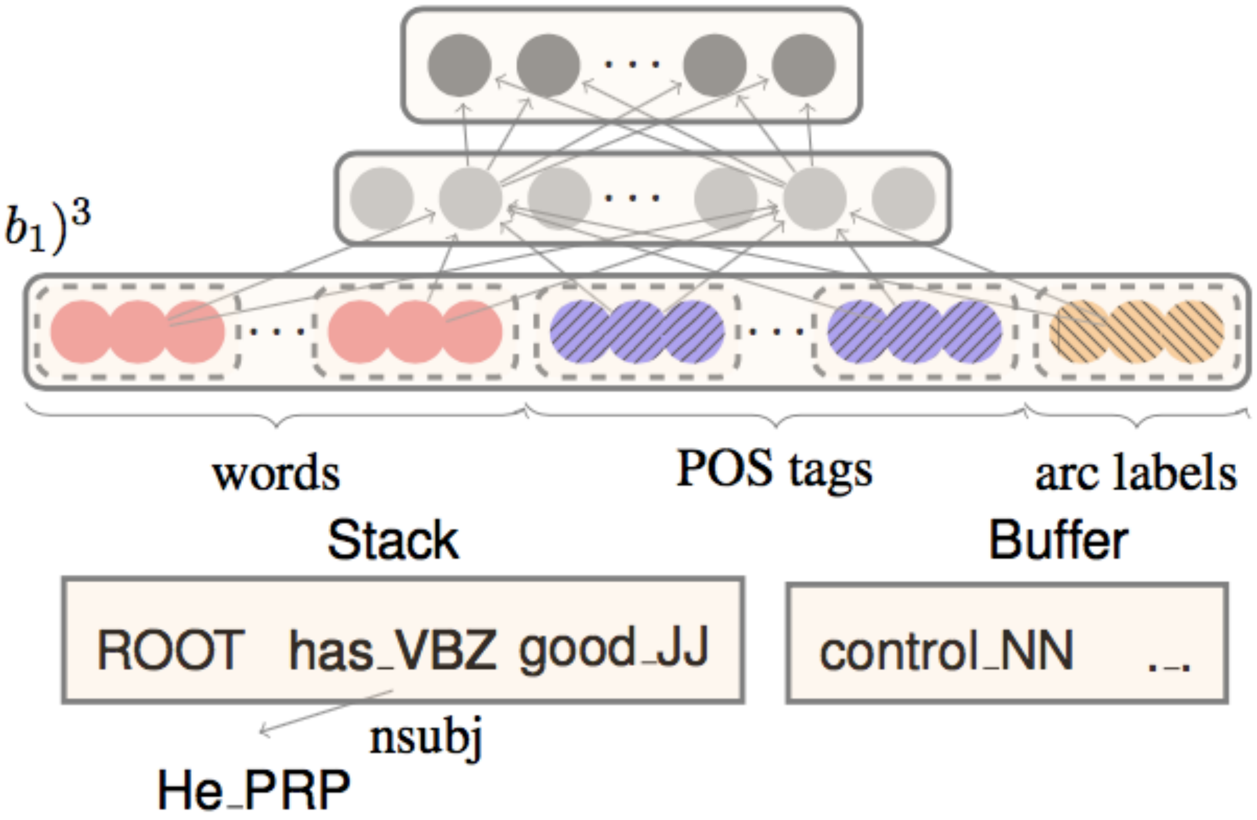
$$p = \mathtt{softmax}(W_2 h)$$

Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Input layer: $[x^w, x^t, x^l]$

Configuration

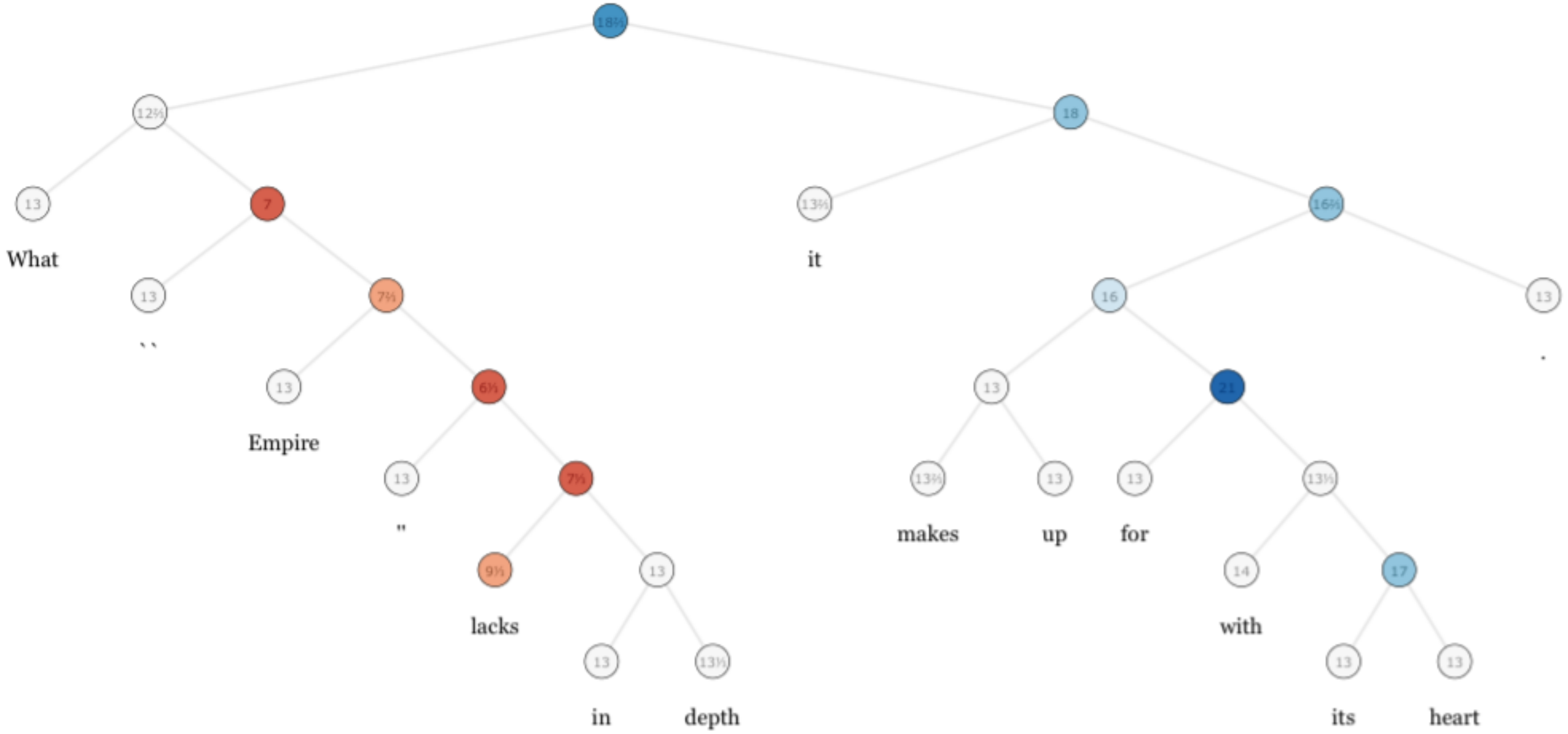


What Features to Extract?

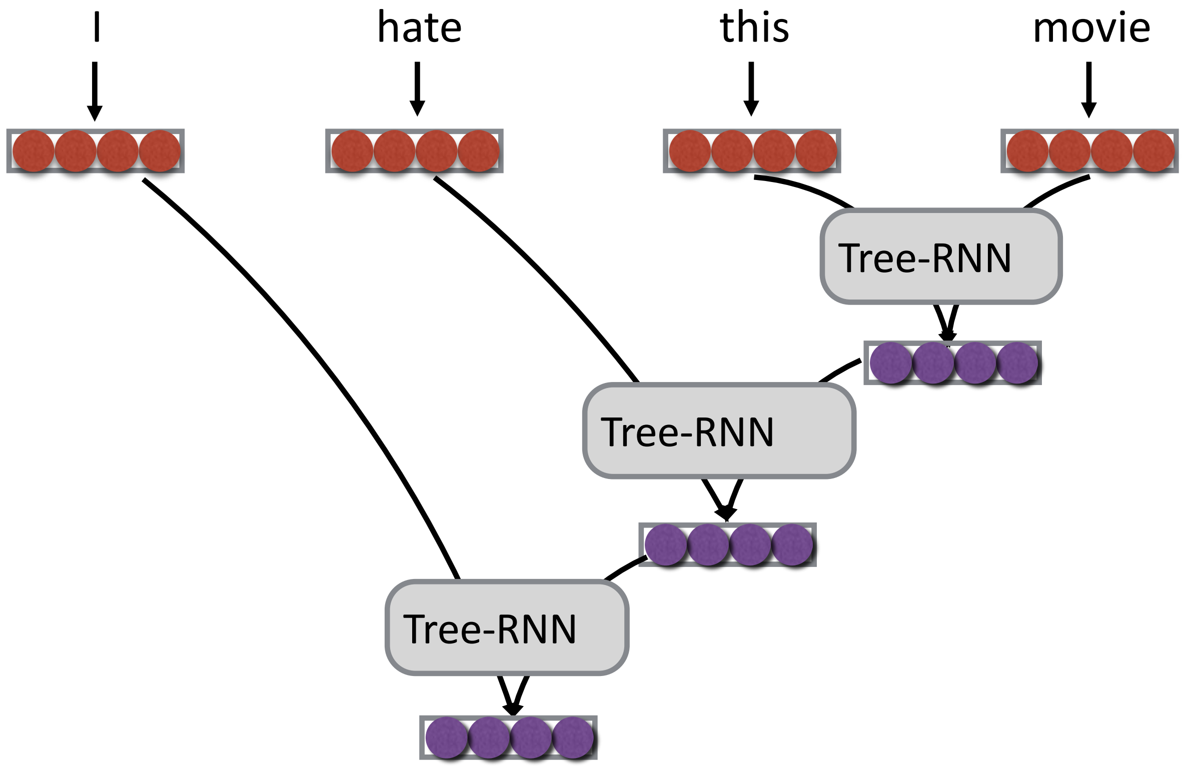
- The top 3 words on the stack and buffer (6 features)
  - `s1, s2, s3, b1, b2, b3`
- The two leftmost/rightmost children of the top two words on the stack (8 features)
  - `lc1(si), lc2(si), rc1(si), rc2(si) i=1,2`
- leftmost and rightmost grandchildren (4 features)
  - `lc1(lc1(si)), rc1(rc1(si)) i=1,2`
- POS tags of all of the above (18 features)
- Arc labels of all children/grandchildren (12 features)

Using Tree Structure in NNs: Syntactic Composition

Why Tree Structure?



Recursive Neural Networks
(Socher et al. 2011)



$$\text{tree-rnn}(\boldsymbol{h}_1, \boldsymbol{h}_2) = \tanh(W[\boldsymbol{h}_1; \boldsymbol{h}_2] + \boldsymbol{b})$$

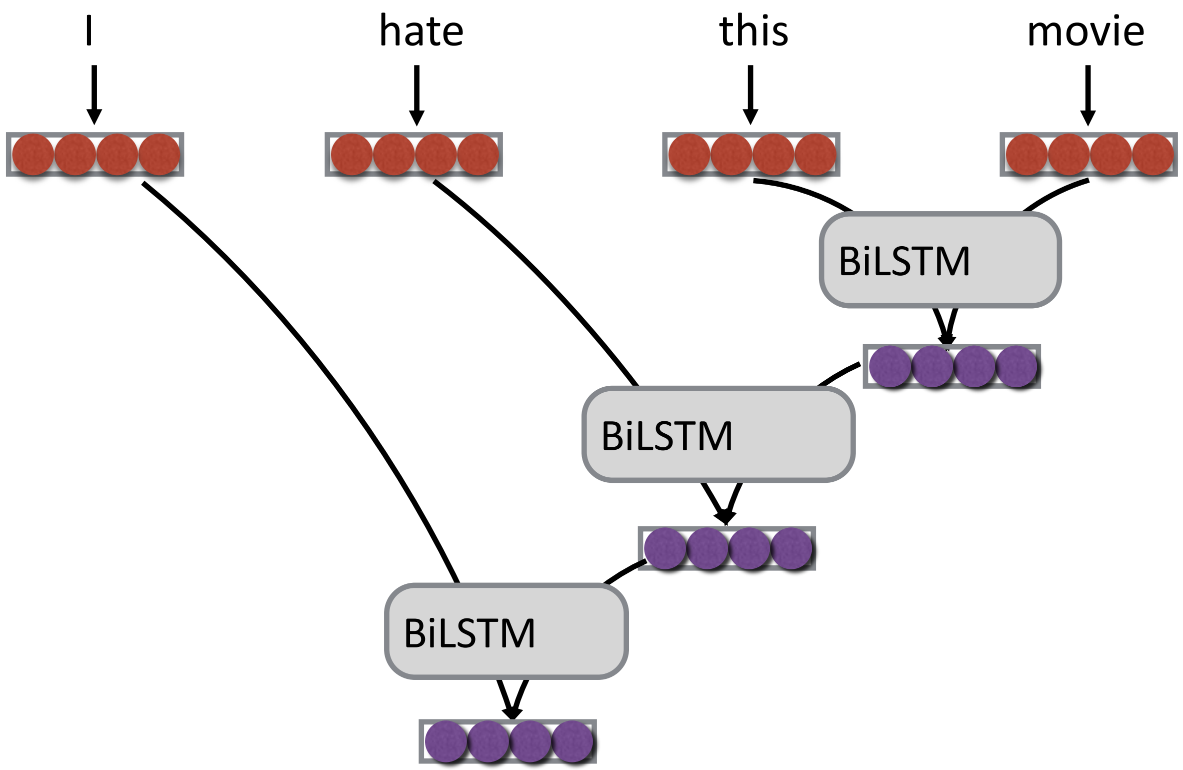
- Can also parameterize by constituent type →
  - different composition behavior for NP, VP, etc.

Tree-structured LSTM (Tai et al. 2015)

- Child Sum Tree-LSTM
  - Parameters shared between all children (possibly based on grammatical label, etc.)
  - Forget gate value is different for each child → the network can learn to "ignore" children (e.g. give less weight to non-head nodes)
- N-ary Tree-LSTM
  - Different parameters for each child, up to N (like the Tree RNN)

Bi-LSTM Composition (Dyer et al. 2015)

- Simply read in the constituents with a BiLSTM
- The model can learn its own composition function!



Let’s Try it Out!

`tree-lstm.py`

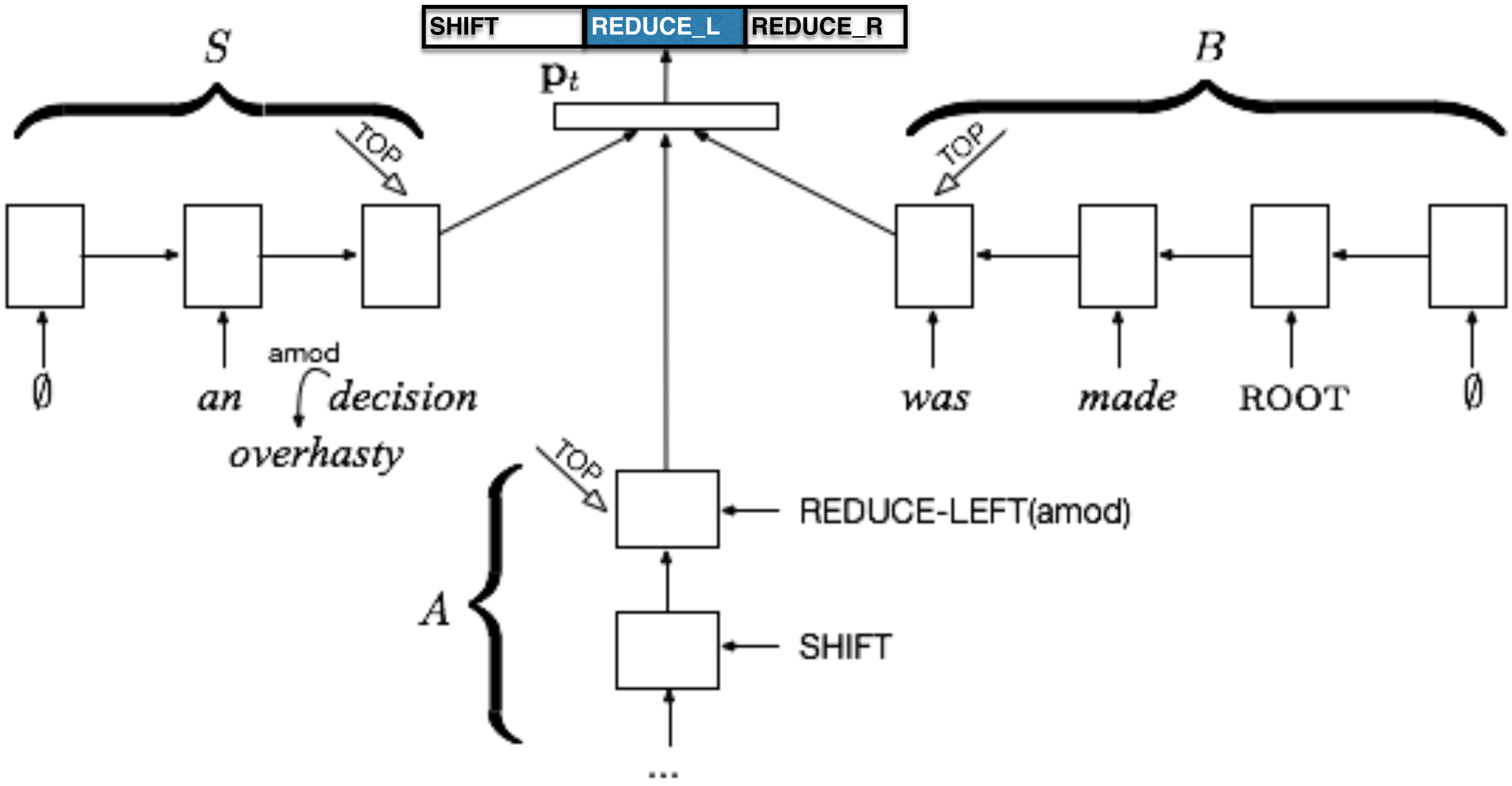
Stack LSTM: Dependency Parsing w/ Less Engineering, Wider Context

(Dyer et al. 2015)

Encoding Parsing Configurations w/ RNNs

- We don’t want to do feature engineering (why leftmost and rightmost grandchildren only?!)
- Can we encode all the information about the parse configuration with an RNN?
- Information we have: stack, buffer, past actions

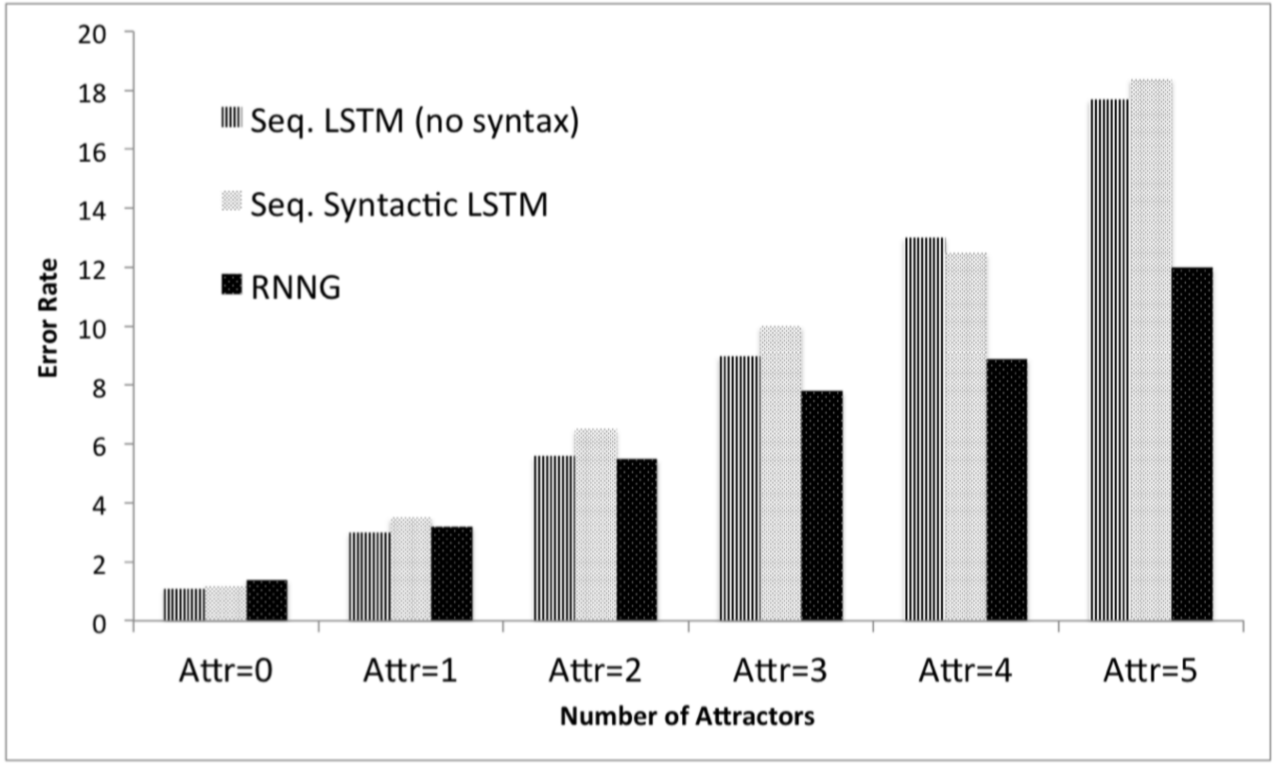
Encoding Stack Configurations w/ RNNs



(Slide credits: Chris Dyer)

Why Linguistic Structure?

- Regular linear language models do quite well
- But they may not capture phenomena that inherently require structure, such as long-distance agreement
- e.g. Kuncoro et al (2018) find agreement with distractors is much better with syntactic model



Carnegie Mellon University
School of Computer Science

CS11-747 Neural Networks for NLP
Neural Semantic Parsing

Pengcheng Yin
pcyin@cs.cmu.edu
Carnegie Mellon University

Language Technologies Institute

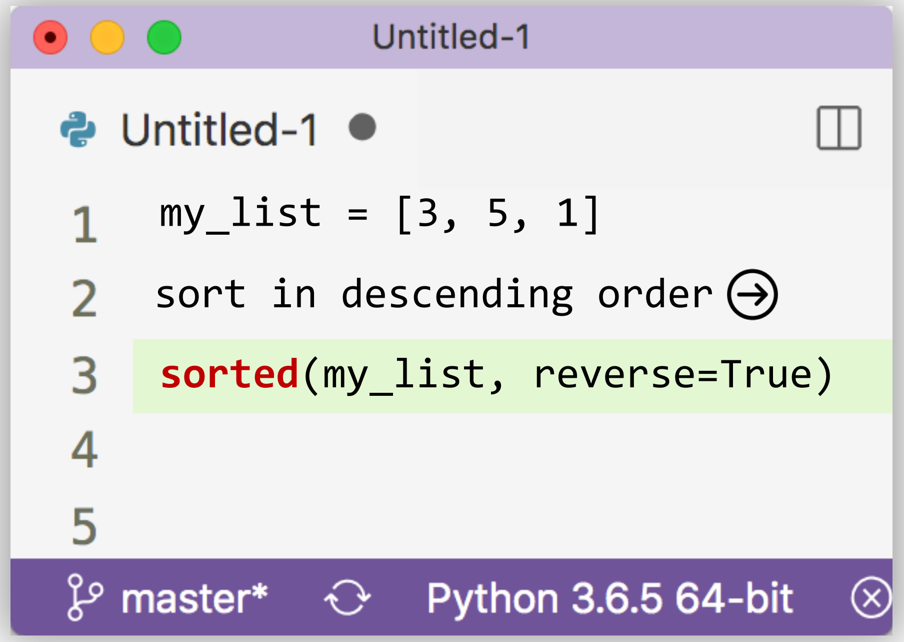
[Some contents are adapted from talks by Graham Neubig]

Semantic Parsers: Natural Language Interfaces to Computers

Cortana

Hey Siri

alexa



Virtual Assistants

- *Set an alarm at 7 AM*
- *Remind me for the meeting at 5pm*
- *Play Jay Chou's latest album*

Natural Language Programming

- *Sort my_list in descending order*
- *Copy my_file to home folder*
- *Dump my_dict as a csv file output.csv*

The Semantic Parsing Task

Parsing natural language utterances into machine-executable meaning representations

Natural Language Utterance

Show me flights from Pittsburgh to Seattle

Meaning Representation

```
lambda $0 e (and (flight $0)
          (from $0 pittsburgh:ci)
          (to $0 seattle:ci))
```

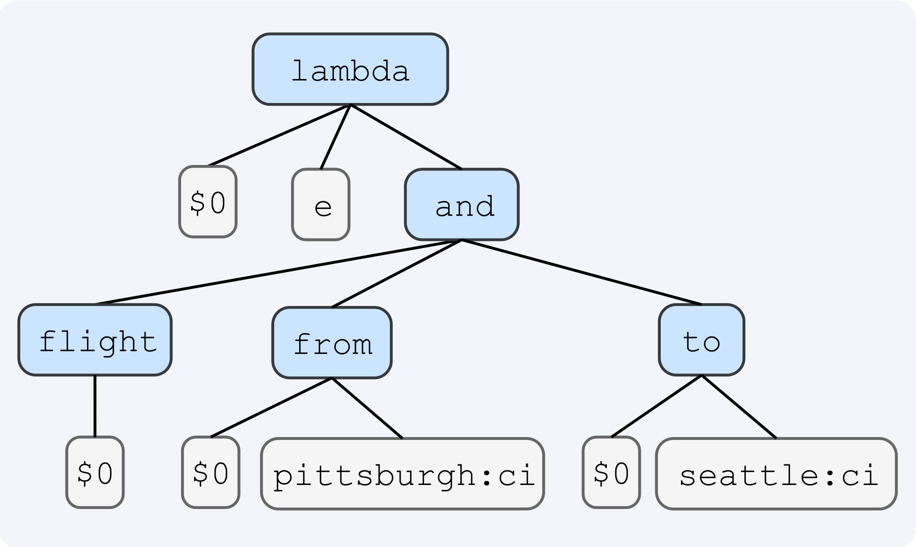
Meaning Representations have Strong Structures

Semantic Parsing

Show me flights from Pittsburgh to Seattle

```
lambda $0 e (and
     (flight $0)
     (from $0 Pittsburgh:ci)
     (to $0 Seattle:ci)
)
```

lambda-calculus logical form



Tree-structured Representation

[Dong and Lapata, 2016]

Machine-executable Meaning Representations

Translating a user's **natural language utterances** (e.g., queries) into machine-executable **formal meaning representations** (e.g., logical form, SQL, Python code)

Domain-Specific, Task-Oriented Languages (DSLs)

Show me flights from Pittsburgh to Seattle

```
lambda $0 e (and (flight $0)
     (from $0 Pittsburgh:ci)
     (to $0 Seattle:ci))
```

`lambda-calculus logical form`

General-Purpose Programming Languages

Sort my_list in descending order

```
sorted(my_list, reverse=True)
```

`Python code generation`

Clarification about Meaning Representations (MRs)

Machine-executable MRs (our focus today) executable programs to accomplish a task

MRs for Semantic Annotation capture the semantics of natural language sentences

Machine-executable Meaning Representations

Show me flights from Pittsburgh to Seattle

```
lambda $0 e (and (flight $0)
     (from $0 pittsburgh:ci)
     (to $0 seattle:ci))
```

Lambda Calculus Logical Form

Lambda Calculus

Python, SQL, …

Meaning Representations For Semantic Annotation

The boy wants to go

```
(want-01
    :arg0 (b / boy)
    :arg1 (g / go-01))
```

Abstract Meaning Representation (AMR)

Abstract Meaning Representation (AMR),

Combinatory Categorical Grammar (CCG)

Workflow of a Semantic Parser

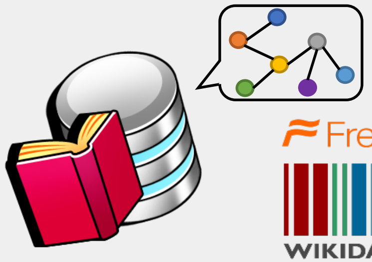
User’s Natural Language Query

Show me flights from Pittsburgh to Seattle

Parsing to Meaning Representation

```
lambda $0 e (and (flight $0)
     (from $0 pittsburgh:ci)
     (to $0 seattle:ci))
```

Execute Programs against KBs



Freebase

WIKIDATA

Execution Results (Answer)

```
1. Alaska Air 119
2. American 3544 -> Alaska 1101
3. …
```

Semantic Parsing Datasets

Domain-Specific, Task-Oriented Languages (DSLs)

Show me flights from Pittsburgh to Seattle

```
lambda $0 e (and (flight $0)
     (from $0 Pittsburgh:ci)
     (to $0 Seattle:ci))
```

`lambda-calculus logical form`

GeoQuery / ATIS / JOBs

WikiSQL / Spider

IFTTT

General-Purpose Programming Languages

Sort my_list in descending order

```
sorted(my_list, reverse=True)
```

`Python code generation`

Django

HearthStone

CONCODE

CoNaLa

JuICe

GEO Query, ATIS, JOBS

- **GEO Query** 880 queries about US geographical information
- **ATIS** 5410 queries about flight booking and airport transportation
- **Jobs** 640 queries to a job database

GEO Query

which state has the most rivers running through it?

```
argmax $0
 (state:t $0)
 (count $1 (and
            (river:t $1)
           (loc:t $1 $0)))
```

Lambda Calculus Logical Form

ATIS

Show me flights from Pittsburgh to Seattle

```
lambda $0 e
  (and (flight $0)
  (from $0 pittsburgh:ci)
  (to $0 seattle:ci))
```

Lambda Calculus Logical Form

JOBS

what Microsoft jobs do not require a bscs?

```
answer(
  company(J,'microsoft'),
  job(J),
  not((req deg(J,'bscs'))))
```

Prolog-style Program

Text-to-SQL Tasks

Natural Language Questions with Database Schema

Input Utterance

Show me flights from Pittsburgh to Seattle

Flight	
FlightNo	UniqueId
Departure	foreign key
Arrival	foreign key

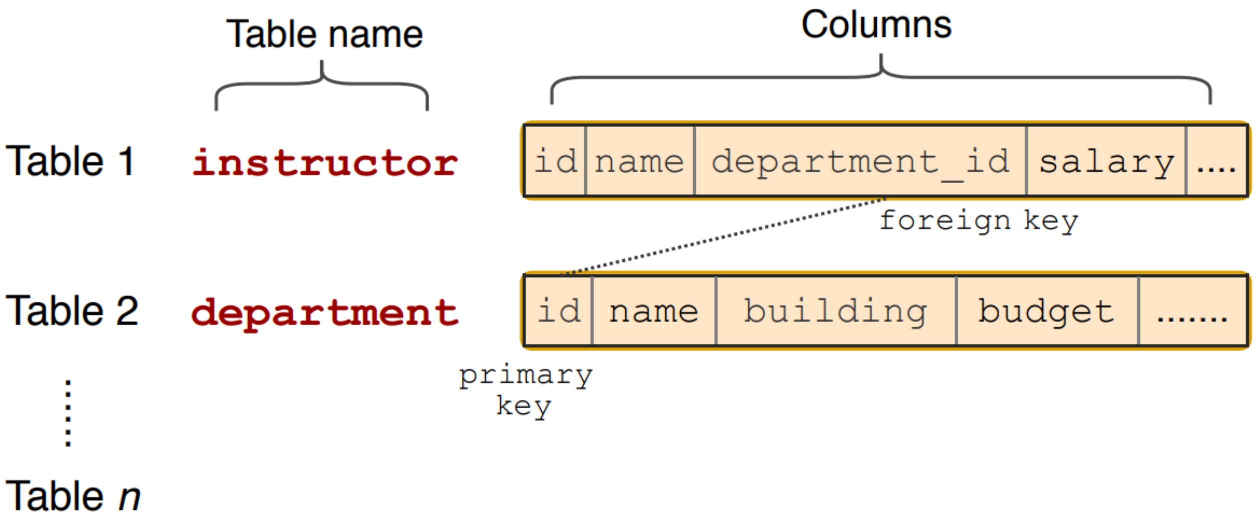
Airport	
Name	UniqueId
CityName	string
PublicTransport	boolean

SQL Query

```
SELECT Flight.FlightNo
FROM Flight
JOIN Airport as DepAirport
ON
   Flight.Departure == DepAirport.Name
JOIN Airport as ArvAirport
ON
   Flight.Arrival == ArvAirport.Name
WHERE
    DepAirport.CityName == Pittsburgh
    AND
    ArvAirport.CityName == Seattle
```

Spider

Annotators check database schema (e.g., database: `college`)



Annotators create:

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

Complex SQL

```
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)
```

- Examples from 200 databases
- Target SQL queries involve joining fields over multiple tables
- Non-trivial Compositionality
  - Nested queries
  - Set Union
  - …

https://yale-lily.github.io

[Yu et al., 2018]

Semantic Parsing Datasets

Domain-Specific, Task-Oriented Languages (DSLs)

Show me flights from Pittsburgh to Berkeley

```
lambda $0 e (and (flight $0)
     (from $0 Pittsburgh:ci)
     (to $0 Berkeley:ci))
```

`lambda-calculus logical form`

GeoQuery / ATIS / JOBs

WikiSQL / Spider

IFTTT

General-Purpose Programming Languages

Sort my_list in descending order

```
sorted(my_list, reverse=True)
```

`Python code generation`

Django

HearthStone

CONCODE

CoNaLa

The CoNaLa Code Generation Dataset

Get a list of words `words` of a file 'myfile'

```
words = open('myfile').read().split()
```

Copy the content of file 'file.txt' to file 'file2.txt'

```
shutil.copy('file.txt’, 'file2.txt')
```

Check if all elements in list `mylist` are the same

```
len(set(mylist)) == 1
```

Create a key `key` if it does not exist in dict `dic` and append element `value` to value

```
dic.setdefault(key, []).append(value)
```

- 2,379 training and 500 test examples
- Natural Language queries collected from StackOverflow
- Manually annotated, high quality natural language queries
- Code is highly expressive and compositional

`conala-corpus.github.io` [Yin *et al.*, 2018]

Supervised Learning of Semantic Parsers

User's Natural Language Query

Show me flights from Pittsburgh to Seattle

Parsing to Meaning Representation

```
lambda $0 e (and (flight $0)
     (from $0 pittsburgh:ci)
     (to $0 seattle:ci))
```

Train a neural semantic parser with source natural language utterances and target programs

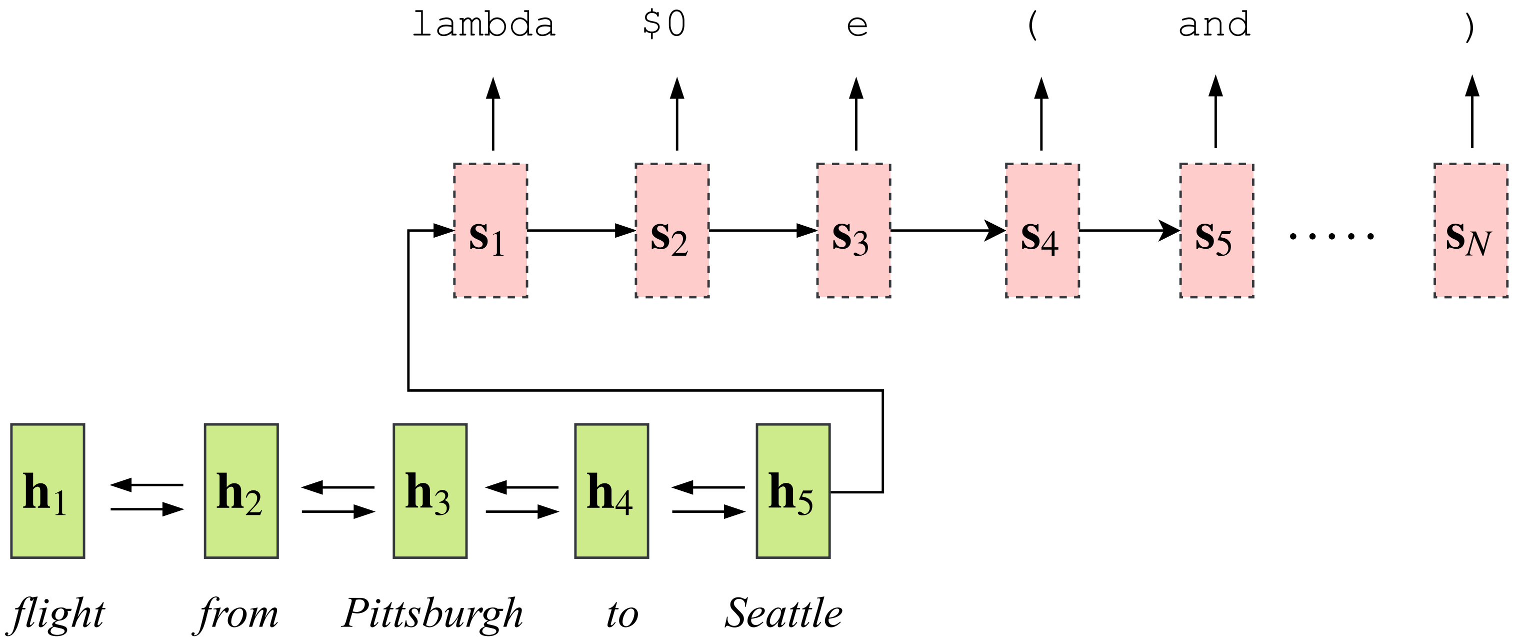
Semantic Parsing as Sequence-to-Sequence Transduction

Task-Specific
Meaning Representations

Show me flights from Pittsburgh to Seattle

```
lambda $0 e (and (flight $0)
      (from $0 pittsburgh:ci)
      (to $0 seattle:ci))
```

Lambda Calculus Logical Form



- Treat the target meaning representation as a sequence of surface tokens
- Reduce the (structured prediction) task as another sequence-to-sequence learning problem

[Dong and Lapata, 2016; Jia and Liang, 2016]

Issues with Predicting Linearized Programs

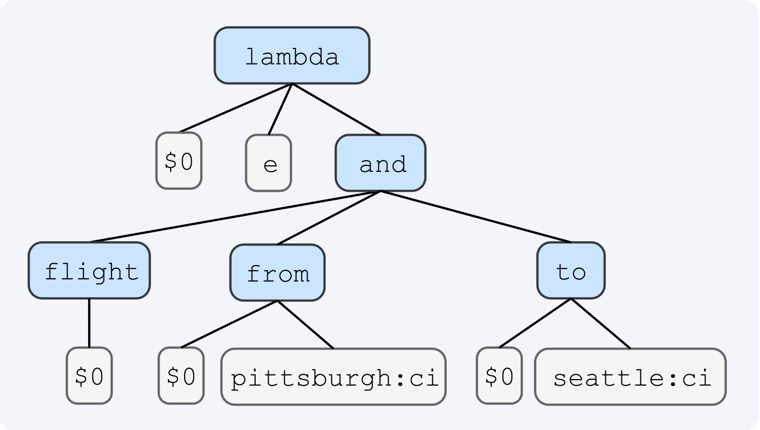
- **Meaning Representations** (e.g., a database query) have strong underlying structures!
- **Issue** Using vanilla seq2seq models ignore the rich structures of meaning representations, and could generate invalid outputs that are not trees

Task-Specific Meaning Representations

Show me flights from Pittsburgh to Seattle

```
lambda $0 e (and (flight $0)
     (from $0 san_Francisco:ci)
     (to $0 seattle:ci))
```

Task specific logical form



Tree-structured Representation

[Jia and Liang, 2016; Dong and Lapata, 2016]

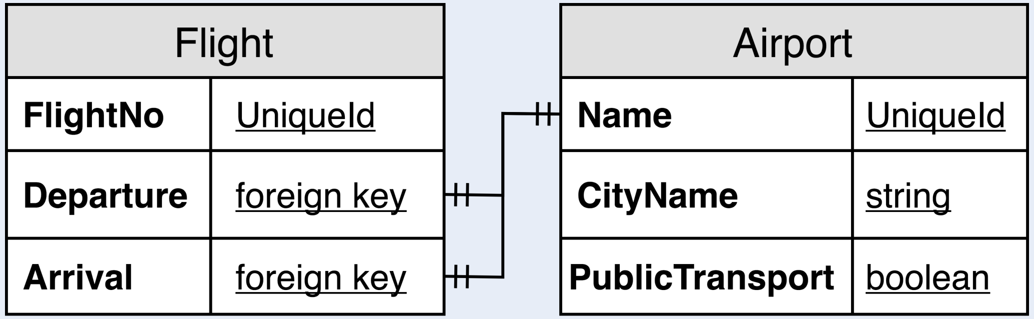
Core Research Question for Better Models

How to add inductive biases to networks a to better capture the structure of programs?

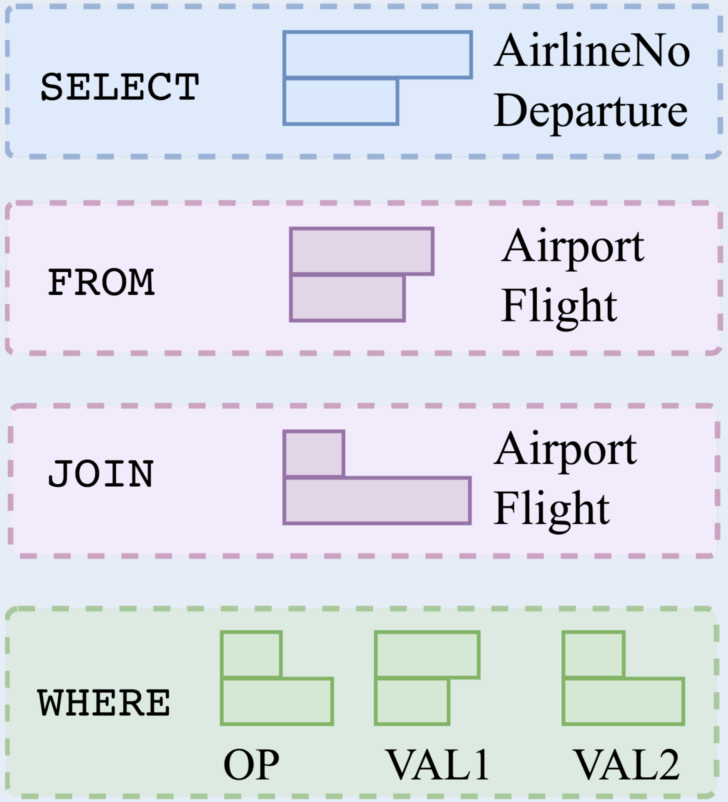
Encode Utterance and In-Domain Knowledge Schema

Input Utterance

Show me flights from Pittsburgh to Berkeley



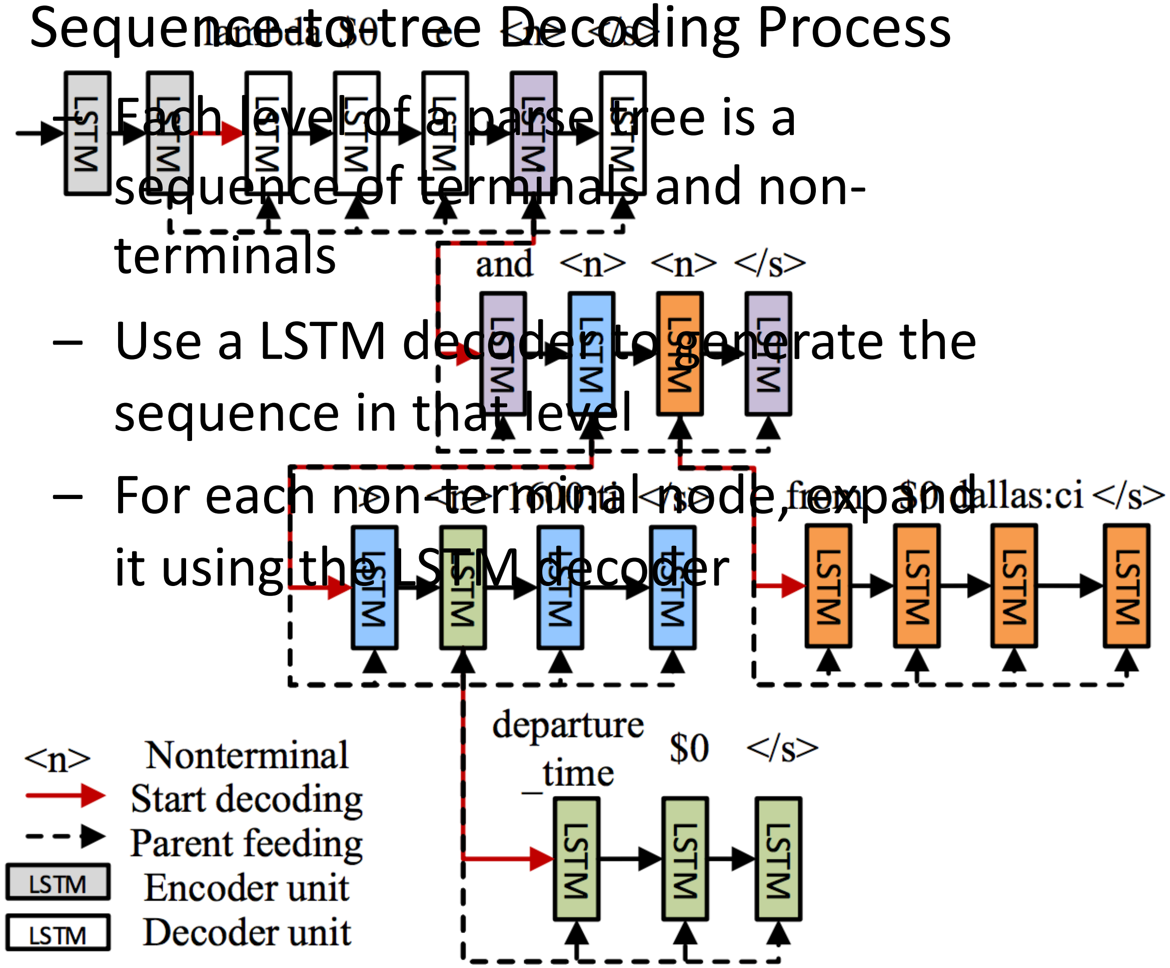
Predict Programs Following Task-Specific Program Structures



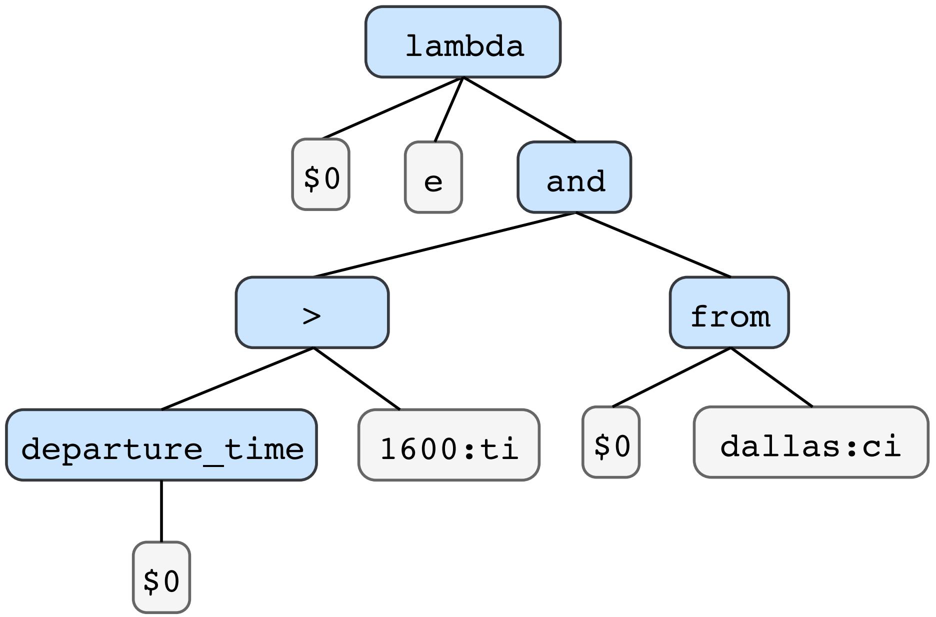
[Xu et al., 2017; Yu et al., 2018]

Structure-aware Decoding for Semantic Parsing

- **Seq2Tree** Generate the parse tree of a program using a hierarchy of recurrent neural decoders following the tree structure
- Sequence-to-tree Decoding Process
  - Each level of a parse tree is a sequence of terminals and non-terminals
  - Use a LSTM decoder to generate the sequence in that level
  - For each non-terminal node, expand it using the LSTM decoder



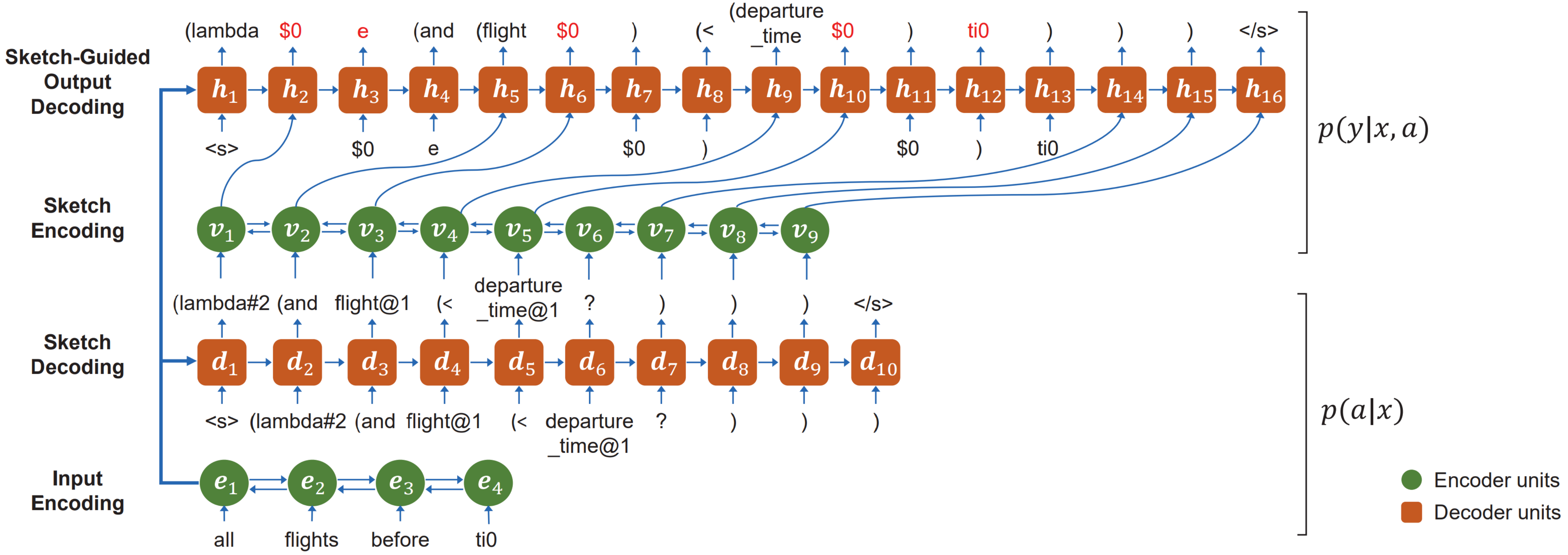
Show me flight from Dallas departing after 16:00



[Dong and Lapata, 2016]

Structure-aware Decoding (Cont’d)

- **Coarse-to-Fine Decoding** decode a coarse **sketch** of the target logical form first and then decode the full logical form conditioned on both the input query and the sketch
- Explicitly model a **coarse global structure** of the logical form, and use it to guide the generation of the **fine-grained structure**



[Dong and Lapata, 2018]

Grammar/Syntax-driven Semantic Parsing

- Previously introduced methods could generate tree-structured representations but cannot guarantee they are gramatically correct.
- Meaning (e.g., Python) have strong underlying grammar/syntax
- How can we explicitly leverage the grammar of programs for better generation?

Python Abstract Grammar

expr $\longmapsto$ Name | Call

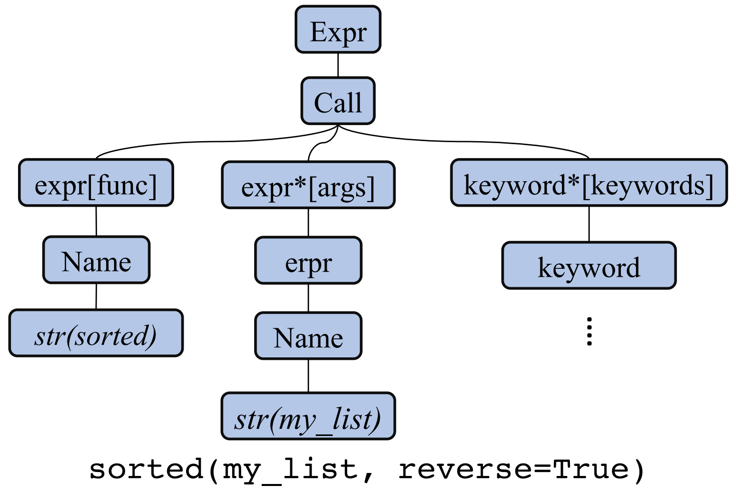
Call $\longmapsto$ expr[*func*] expr*[*args*] keyword*[*keywords*]

If $\longmapsto$ expr[*test*] stmt*[*body*] stmt*[*orelse*]

For $\longmapsto$ expr[*target*] expr*[*iter*] stmt*[*body*] stmt*[*orelse*]

FunctionDef $\longmapsto$ identifier[*name*] expr*[*iter*] stmt*[*body*] stmt*[*orelse*]

Abstract Syntax Tree



[Yin and Neubig, 2017; Rabinovich *et al.*, 2017]

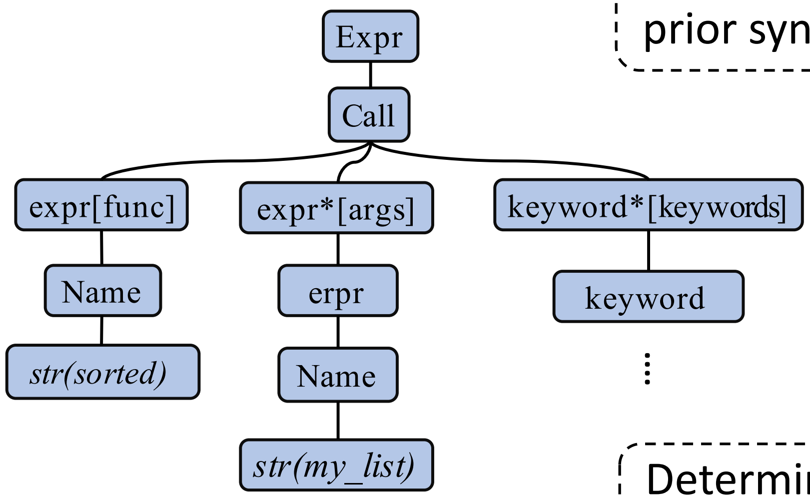
Grammar/Syntax-driven Semantic Parsing

- **Key Idea** use the grammar of the target meaning representation (Python AST) as prior symbolic knowledge in a neural sequence-to-sequence model

Input Intent (x) *sort my_list in descending order*

$p(y|x)$: a seq2seq model with prior syntactic information

Generated AST (y)



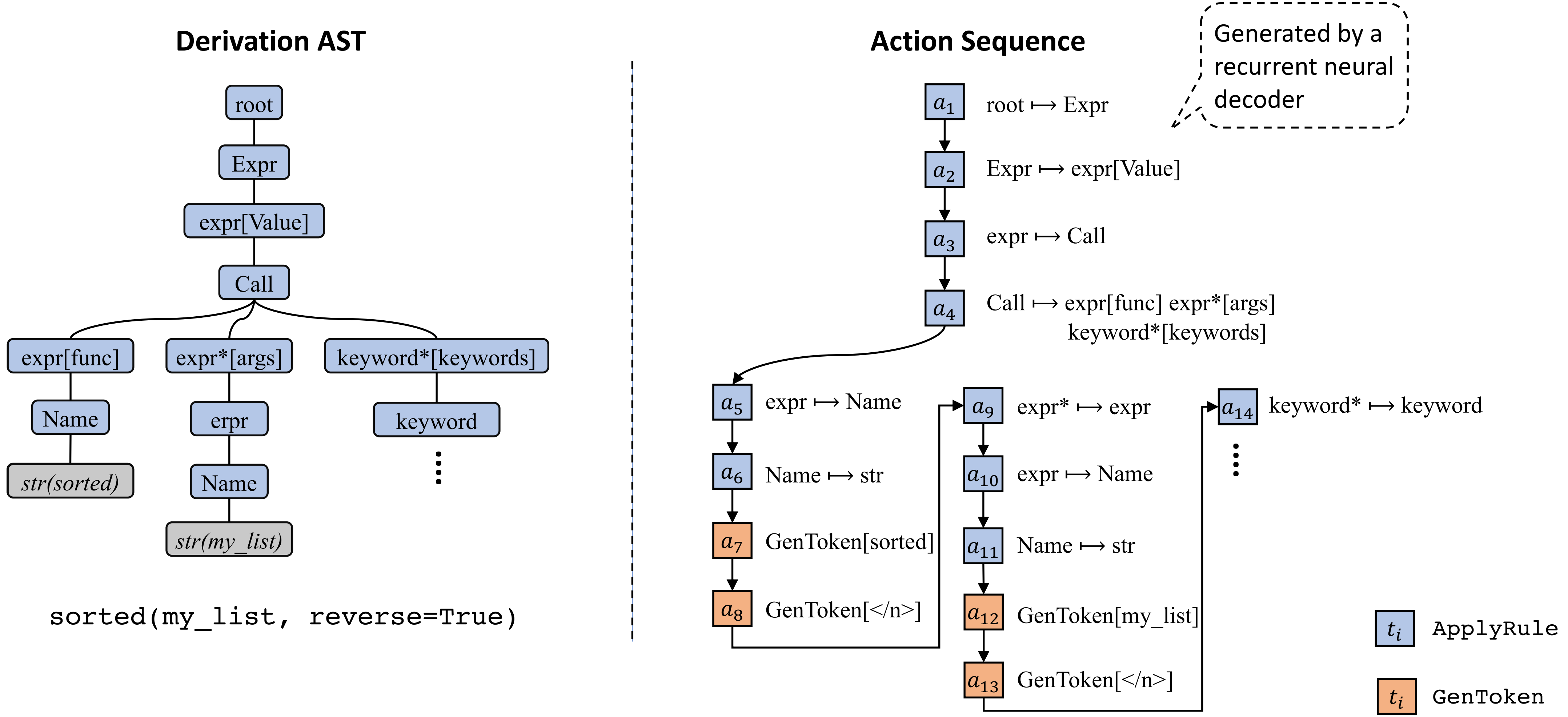
Deterministic transformation (using Python `astor` library)

Surface Code (c) `sorted(my_list, reverse=True)`

[Yin and Neubig, 2017; Rabinovich *et al.*, 2017]

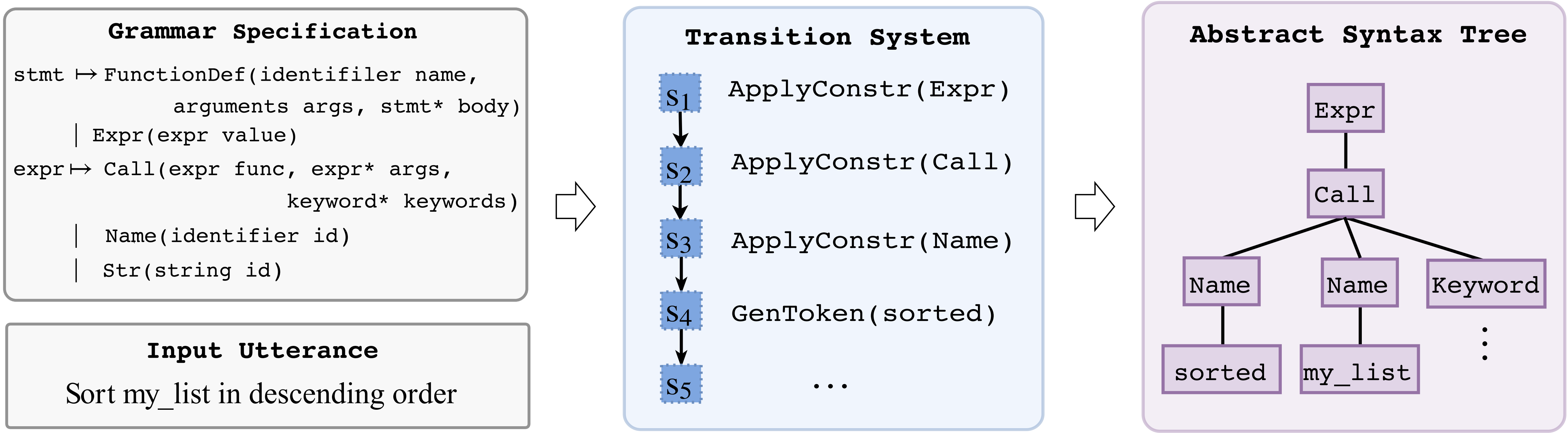
Grammar/Syntax-driven Semantic Parsing

- Factorize the generation story of an AST into sequential application of *actions* $\{a_t\}$:
  - `ApplyRule[r]`: apply a production rule r to the frontier node in the derivation
  - `GenToken[v]`: append a token v (e.g., variable names, string literals) to a terminal



TranX: Transition-based Abstract SyntaX Parser

- Convenient interface to specify task-dependent grammar in plain text
- Customizable conversion from abstract syntax trees to domain-specific programs
- Built-in support for many languages: Python, SQL, Lambda Calculus, Prolog…

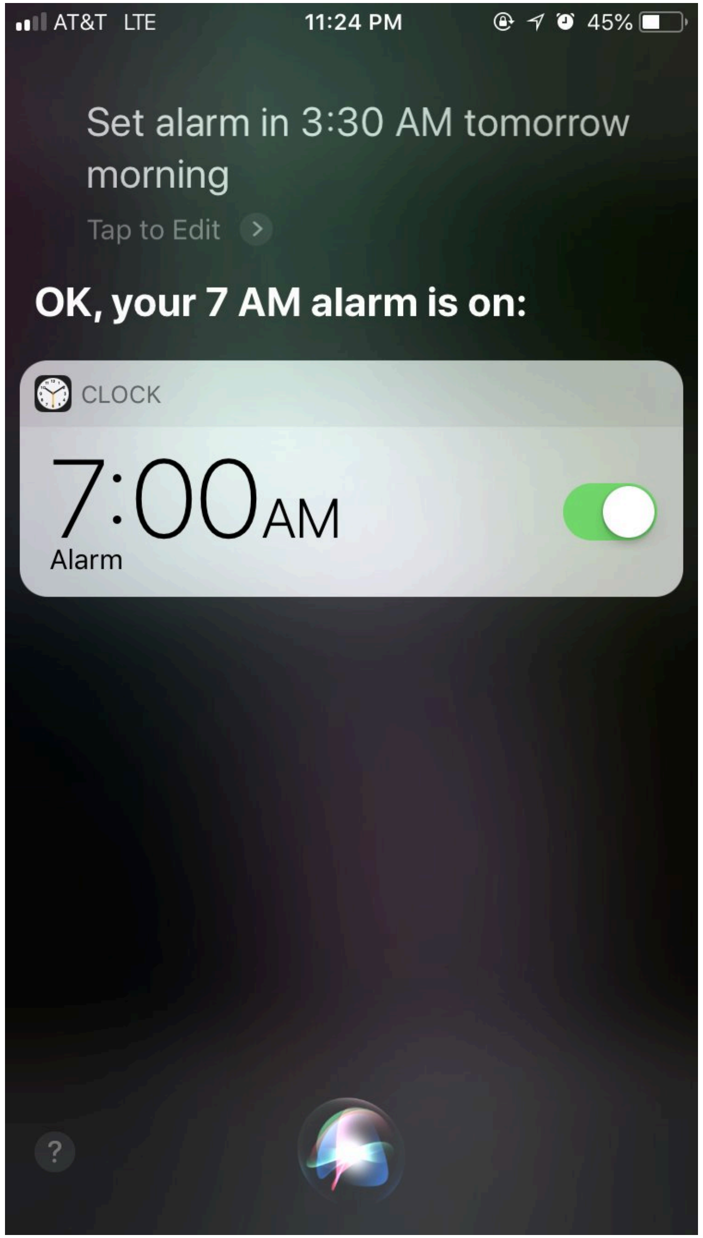


github.com/pcyin/tranX

[Yin and Neubig 2018, Yin and Neubig 2019]

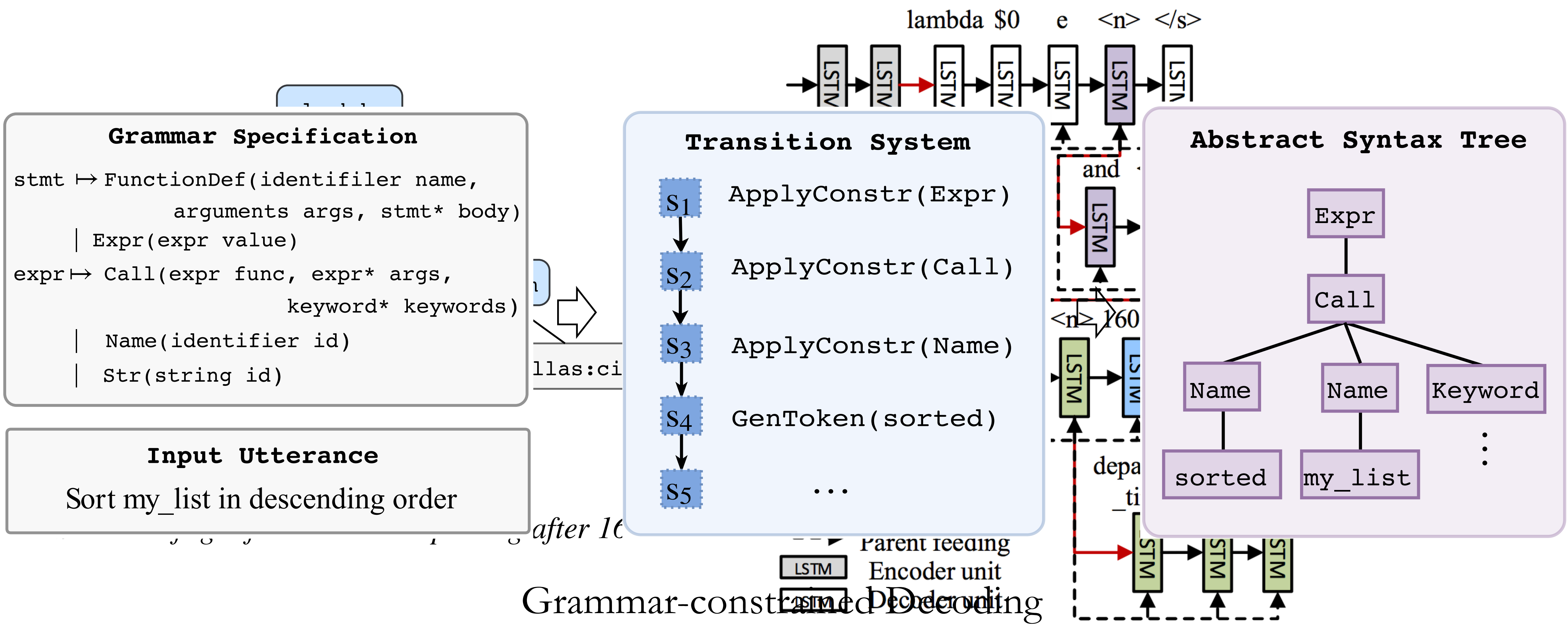
Side Note: Importance of Modeling Copying

- Modeling copying is very important for neural semantic parsers!
- Out-of-vocabulary entities (e.g., city names, date time) often appear in the input utterance
- Neural seq2seq models like to hallucinate entities not in the input utterance ☺



Summary: Supervised Learning of Semantic Parsers

Key Research Question design decoders to capture the structure of programs

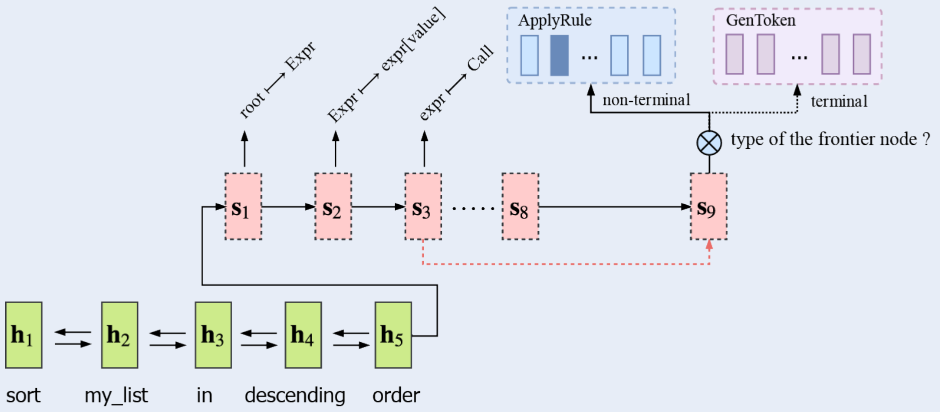


Grammar-constrained Decoding

Structure-aware Decoding

Supervised Learning: the Data Inefficiency Issue

Supervised Parsers are Data Hungry



Purely supervised neural semantic parsing models require large amounts of training data 

Data Collection is Costly

Copy the content of file 'file.txt' to file 'file2.txt'

```
shutil.copy('file.txt','file2.txt')
```

Get a list of words `words` of a file 'myfile'

```
words = open('myfile').read().split()
```

Check if all elements in list `mylist` are the same

```
len(set(mylist)) == 1
```

Collecting parallel training data costs  and 

*Examples from `conala-corpus.github.io` [Yin et al., 2018]

1700 USD for <3K Python code generation examples

Weakly-supervised Learning of Semantic Parsers

User's Natural Language Query

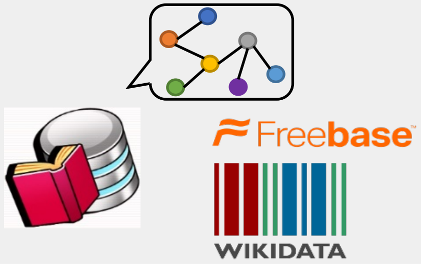
Show me flights from Pittsburgh to Seattle

Parsing to Meaning Representation

```
lambda $0 e (and (flight $0)
     (from $0 pittsburgh:ci)
     (to $0 seattle:ci))
```

As unobserved latent variable

Query Execution



Execution Results (Answer)

```
1. AS 119
2. AA 3544 -> AS 1101
3. …
```

Weak supervision signal

Train a semantic parser using natural language query and the execution results (a.k.a. Semantic Parsing **with Execution**)

[Clarke *et al.*, 2010; Liang *et al.*, 2011]

Weakly-supervised Parsing as Reinforcement Learning

Weakly Supervised Semantic Parsing

What is the most populous city in United States?

City	Country	Population	GDP
New York	USA	8.62M	1275B
Hong Kong	China	7.39M	341.4B
Tokyo	Japan	9.27M	1800B
London	UK	8.78M	650B
Los Angeles	USA	4.00M	941B

`Answer: New York`

Hypothesized Programs

```
City.OrderBy(Population)
    .First() => Result: Tokyo
```

```
City.Filter(Country=='USA')
    .OrderBy(Population)
    .First() => Result: New York
```

```
City.Filter(Country=='USA')
    .OrderBy(GDP)
    .First() => Result: New York
```

Weakly-supervised Learning -- Challenges

Hypothesized Programs

```
City.OrderBy(Population)
    .First() => Result: Tokyo
```

```
City.Filter(Country==‘USA’)
    .OrderBy(Population)
    .First() => Result: New York
```

```
City.Filter(Country==‘USA’)
    .OrderBy(GDP)
    .First() => Result: New York
```

Large Search Space

Exponentially large search space w.r.t. the size of programs

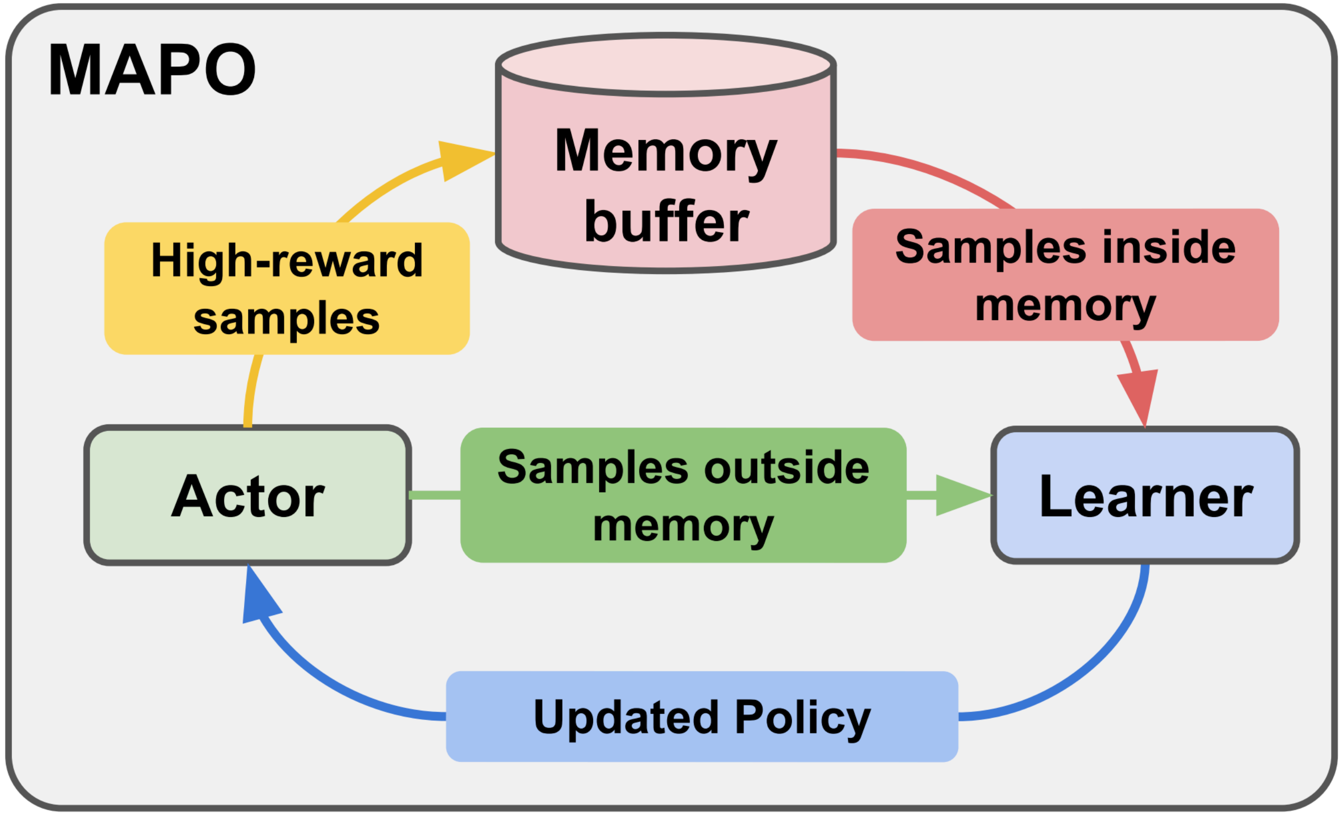
Very Sparse Rewards

Only very few programs are actually correct

Spurious Programs

Spurious programs could also hit the correct answer, leading to noisy reward signals.

Efficient Search: Cache High-reward Programs



- Use a memory buffer to cache high-rewarding logical forms sampled so far
- During training, bias towards high-rewarding queries in the memory buffer

[Liang et al., 2018]

Tackle Spurious Programs using Heuristics

What is the most populous city in United States?

```
City.Filter(Country=='USA')
    .OrderBy(Population)
    .First() => Result: New York
```

```
City.Filter(Country=='USA')
    .OrderBy(GDP)
    .First() => Result: New York
```

Similarity('populous', population)

Back-translation-score $p(\,\cdot\,|\,\cdot\,)$

Similarity('populous', GDP)

Back-translation-score $p(\,\cdot\,|\,\cdot\,)$

[Guu et al., 2017; Misra et al., 2018; Cheng et al., 2018]

Conclusion: Workflow of a Semantic Parser

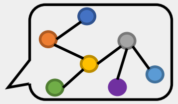
User's Natural Language Query

Show me flights from Pittsburgh to Seattle

Parsing to Meaning Representation

```
lambda $0 e (and (flight $0)
     (from $0 san_Francisco:ci)
     (to $0 seattle:ci))
```

Query Execution



Freebase

WIKIDATA

Execution Results (Answer)

```
1. AS 119
2. AA 3544 -> AS 1101
3. …
```

Conclusion: Two Learning Paradigms

Supervised Semantic Parsing

What is the most populous city in United States?

```
City.Filter(Country==‘USA’)
     .OrderBy(Population)
     .First() => Result: New York
```

Weakly Supervised Semantic Parsing

What is the most populous city in United States?

City	Country	Population	GDP
New York	USA	8.62M	1275B
Hong Kong	China	7.39M	341.4B
Tokyo	Japan	9.27M	1800B

```
Answer: New York
```

Tree-based Decoding

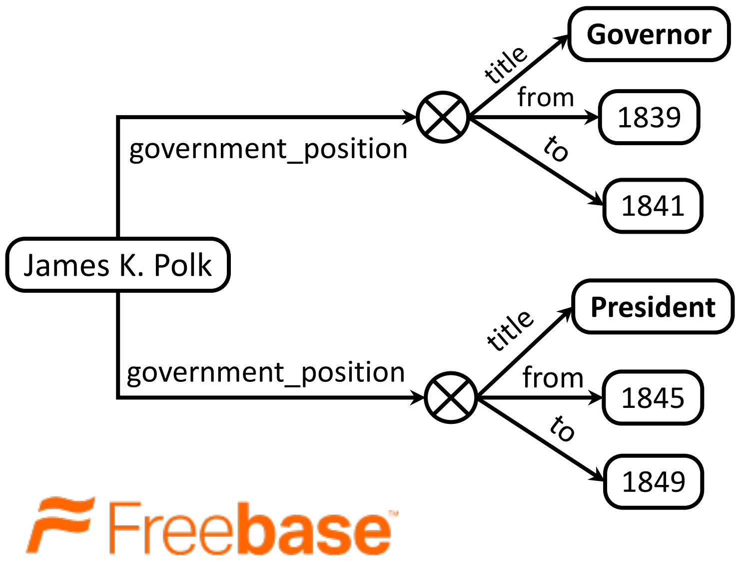
Grammar-constrained Decoding

Efficient Exploration over Large Search Space

Tackle Spurious Programs

Challenge: Natural Language is Highly Compositional

Q: *what was James K. Polk before he was president?*




```
SELECT ?job_title.
FROM Freebase
WHERE{
    James K. Polk  government_position  ?job.
    ?job           title                ?job_title.

    ?job           to                   ?to_date.

    FILTER(?to_date < (
        SELECT ?start_date.
        WHERE{
            James K. Polk government_position ?job1.
            ?job1          title               President.
            ?job1          from                ?start_date.
        }
    ))
}
```

Meaning Representation in SPARQL Query

- Sometimes even a short NL phrase/clause has complex structured grounding

[Yin et al., 2015]

Challenge: Scale to Open-domain Knowledge

- Most existing works focus on parsing natural language to queries to structured, curated knowledge bases
- Most of the world's knowledge has unstructured, textual form!
  - Machine Reading Comprehension tasks (e.g., SQUAD) use textual knowledge

User's Natural Language Query

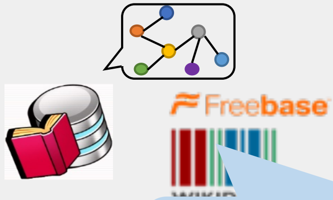
Show me flights from Pittsburgh to Seattle

Parsing to Meaning Representation

```
lambda $0 e (and (flight $0)
     (from $0 san_Franci  o:ci)
     (to $0 seattle:ci))
```

How to design MRs that can be used to query textual knowledge?

Query Execution

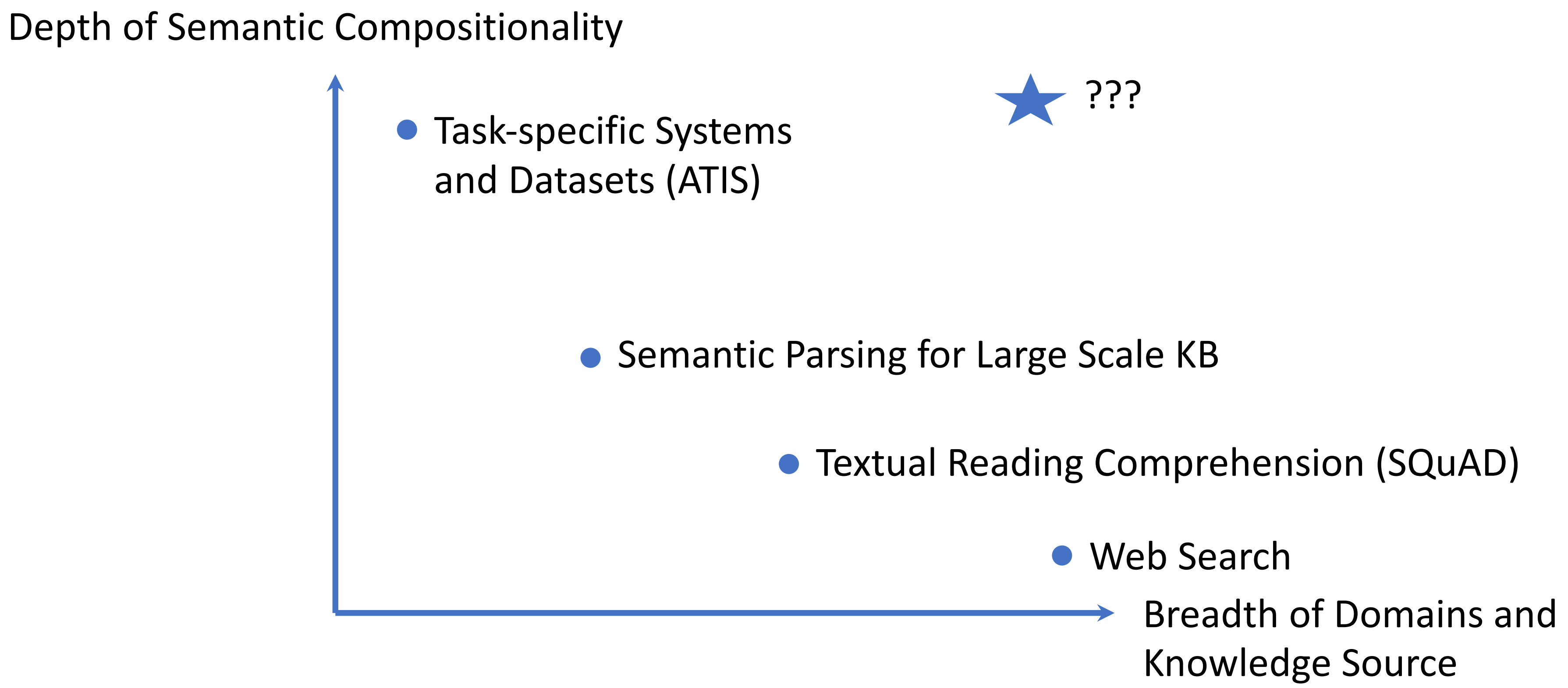


Textual Knowledge (e.g., Wikipedia Articles)

Execution Results (Answer)

```
1. AS 119
2. AA 3544 -> AS 1101
3. …
```

Final Notes: Challenges



(Figure adapted from Pasupat and Liang, 2015)

Supplementary Slides

More Semantic Parsing Datasets

WikiSQL Dataset

Table: CFLDraft

Pick #	CFL Team	Player	Position	College
27	Hamilton Tiger-Cats	Connor Healy	DB	Wilfrid Laurier
28	Calgary Stampeders	Anthony Forgone	OL	York
29	Ottawa Renegades	L.P. Ladouceur	DT	California
30	Toronto Argonauts	Frank Hoffman	DL	York
...	...	...	...	...

Question:

How many CFL teams are from York College?

SQL:

```
SELECT COUNT CFL Team FROM
CFLDraft WHERE College = “York”
```

Result:

2

- 80,654 examples of Table, Question, SQL Query and Answer
- **Context** a small, single database table extracted from a Wikipedia article
- **Target** an SQL query

[Zhong et al., 2017]

HearthStone (HS) Card Dataset

- Description: properties/fields of an HearthStone card
- Target code: implementation as a Python class from HearthBreaker

Utterance (Card Property)

<name> Divine Favor </name>
<cost> 3 </cost>
<desc> Draw cards until you have as many in hand as your opponent </desc>

Target Code (Python class)

```
class DivineFavor(SpellCard):
  def __init__(self):
    super().__init__("Divine Favor", 3, CHARACTER_CLASS.PALADIN,
                     CARD_RARITY.RARE)
  def use(self, player, game):
    super().use(player, game)
    difference = len(game.other_player.hand) - len(player.hand)
    for i in range(0, difference):
      player.draw()
```

[Ling *et al.*, 2016]

IFTTT Dataset

- Over 70K user-generated task completion snippets crawled from ifttt.com
- Wide variety of topics: home automation, productivity, etc.
- Domain-Specific Language: IF-THIS-THEN-THAT structure

https://ifttt.com/applets/1p-autosave-your-instagram-photos-to-dropbox

IFTTT Natural Language Query and Meaning Representation

Autosave your Instagram photos to Dropbox

```
IF Instagram.AnyNewPhotoByYou
THEN Dropbox.AddFileFromURL
```

Domain-Specific Programming Language

[Quirk *et al.*, 2015]

Django Annotation Dataset

- Description: manually annotated descriptions for 10K lines of code
- Target code: one liners
- Covers basic usage of Python like variable definition, function calling, string manipulation and exception handling

Utterance *call the function _generator, join the result into a string, return the result*

Target
```
return ''.join(_generator())
```

[Oda *et al.*, 2015]

Notes for Weakly Supervised Parsing

Weakly-supervised Parsing as Reinforcement Learning

NL question *What is the most populous city in United States?*

Semantic Parsing

Sampled Logical From
(Lambda DCS, Liang 2011)

z_1 **argmax(λx.city(x)∧located(x,US), λx.population(x))** ✔

z_2 argmax(λx.city(x), λx.population(x)) ✘

z_3 argmax(λx.city(x)∧loc(x,US), λx.GDP(x)) ✔

⋮

Query Execution

Answer
(with rewards)

y_1 New York ✔

y_2 Tokyo ✘

y_3 New York ✔

Optimize Objective

Gradient Updates

Probability of Gold Answer

$$p(\boldsymbol{y}^* = \text{New York}) = p(\boldsymbol{z}_1|\boldsymbol{x}) + p(\boldsymbol{z}_3|\boldsymbol{x})$$

Maximum Marginal Likelihood Training Objective

What is the most populous city in United States?

Semantic Parsing | Reward

z_1 **argmax(λx.city(x)∧located(x,US), λx.population(x))** ✅

z_3 argmax(λx.city(x)∧loc(x,US), λx.GDP(x)) ✅

Marginalization over all (sampled) hypotheses

$$\nabla \log p_\theta(\boldsymbol{y}^*|\boldsymbol{x}) = \sum_{\boldsymbol{z}:\text{answer}(\boldsymbol{z})=\boldsymbol{y}^*} w(\boldsymbol{z},\boldsymbol{x}) \cdot \nabla \log p_\theta(\boldsymbol{z}|\boldsymbol{x})$$

Gold Answer

Candidate Logical Form (Latent Variable)

where $$w(\boldsymbol{z},\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{z}|\boldsymbol{x})}{\sum_{\boldsymbol{z}':\text{answer}(\boldsymbol{z}'=\boldsymbol{y}^*)} p_\theta(\boldsymbol{z}'|\boldsymbol{x})}$$

- Intuitively, the gradient from each candidate logical form is weighted by its normalized probability. The more likely the logical form is, the higher the weight of its gradient

Weakly-supervised Learning Issue 1: Spurious Logical Forms

- **Spurious Logical Forms** have the correct execution result, but are semantically wrong

What is the most populous city in United States?

↓ Semantic Parsing

			Reward
Correct	z_1	**`argmax(λx.city(x)∧located(x,US), λx.population(x))`**	✓
Spurious	z_3	`argmax(λx.city(x)∧loc(x,US), λx.GDP(x))`	✓

- Solutions:
  - Encourage diversity in gradient updates by updating different hypotheses with roughly equal gradient weights (Guu *et al.*, 2017)
  - Use prior lexical knowledge to promote promising hypotheses. E.g., *populous* has strong association with `λx.population(x)` (Misra *et al.*, 2018)

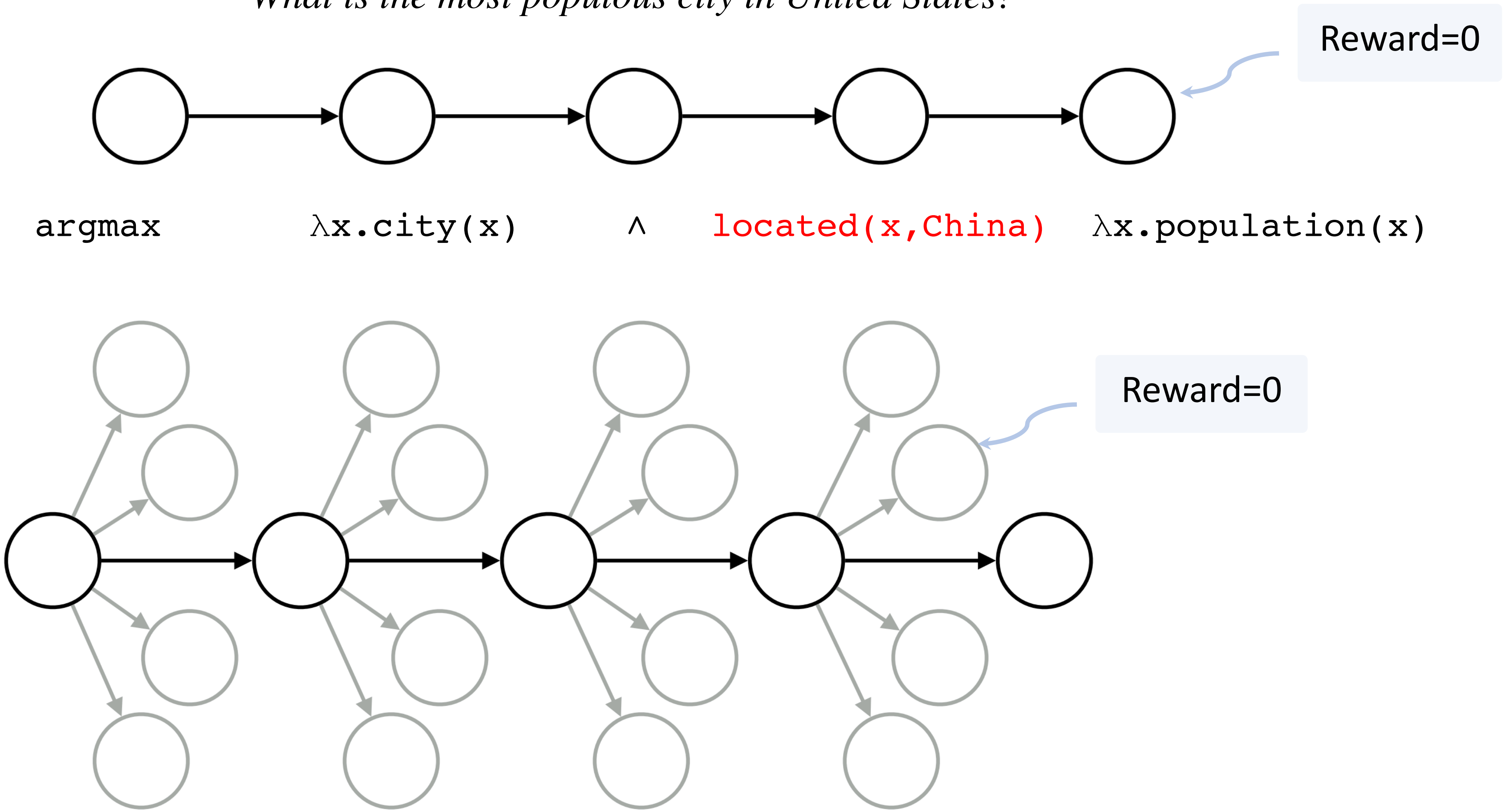
Weakly-supervised Learning Issue 2: Search Space

- The space of possible logical forms with correct answers is exponentially large
- How to search candidate logical forms more efficiently?

$$\nabla \log p_\theta(\boldsymbol{y}^*|\boldsymbol{x}) = \underbrace{\sum_{\boldsymbol{z}:\text{answer}(\boldsymbol{z})=\boldsymbol{y}^*}}_{\text{Prohibitively Large Search Space}} w(\boldsymbol{z}, \boldsymbol{x}) \cdot \nabla \log p_\theta(\boldsymbol{z}|\boldsymbol{x})$$

Efficient Search: Single Step Reward Observation

What is the most populous city in United States?



Factorize the reward into each single time step (a.k.a., reward shaping)

[Suhr and Artzi, 2018]