Carnegie Mellon University
School of Computer Science

11-711 Advanced NLP
Executable Semantic Parsing, and Beyond

Frank Xu

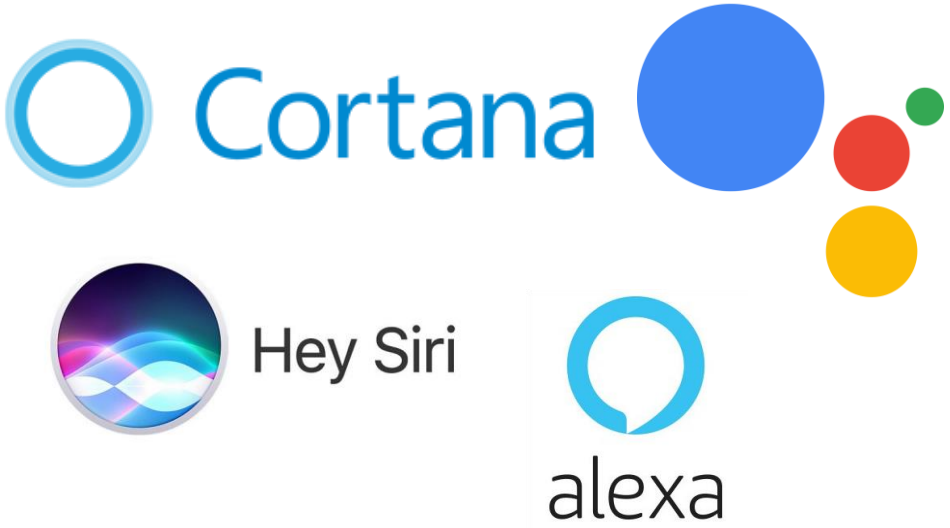
fangzhex@cs.cmu.edu

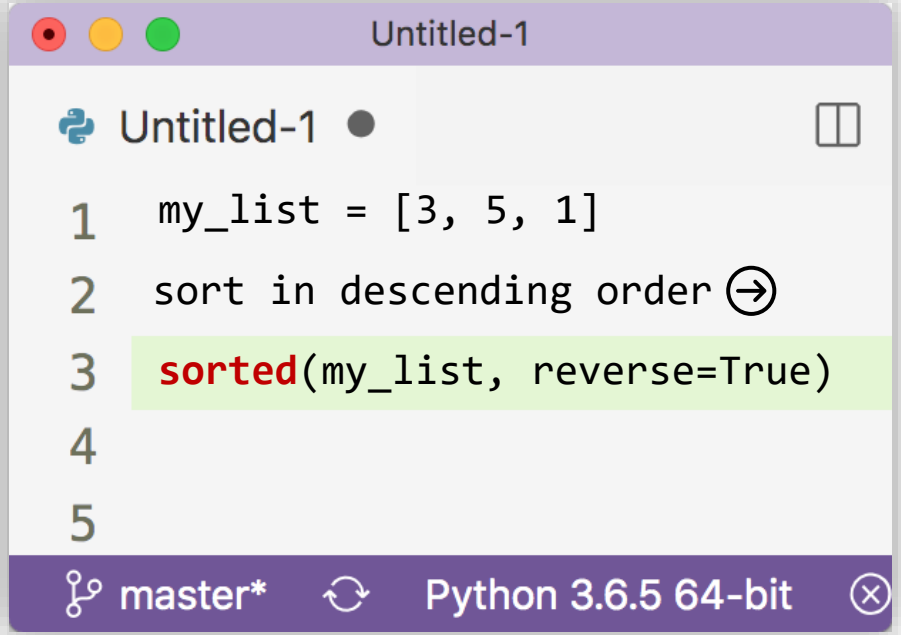
Carnegie Mellon University

Language
Technologies
Institute

[Some contents are adapted from talks by Graham Neubig and Pengcheng Yin]

Semantic Parsers: Natural Language Interfaces to Computers





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))
```

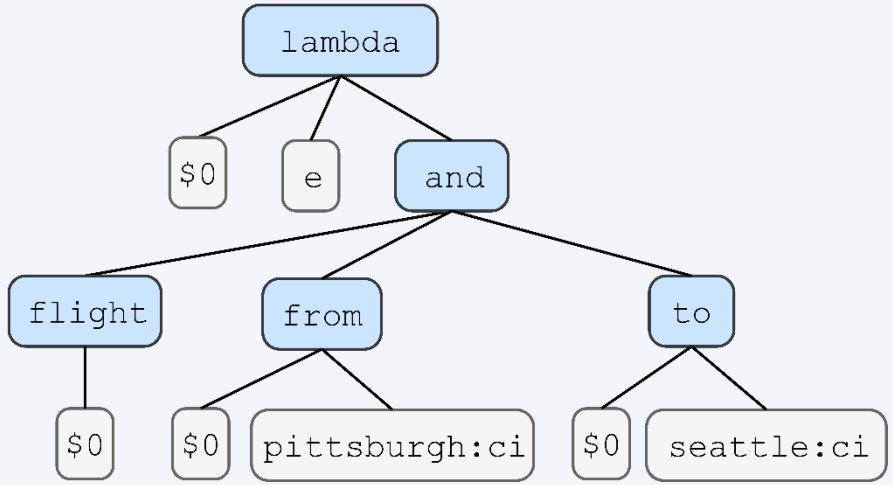
Meaning Representations have Strong Structures

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

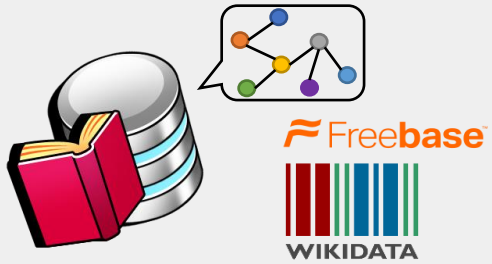
User's Natural Language Query

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



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'))))
```

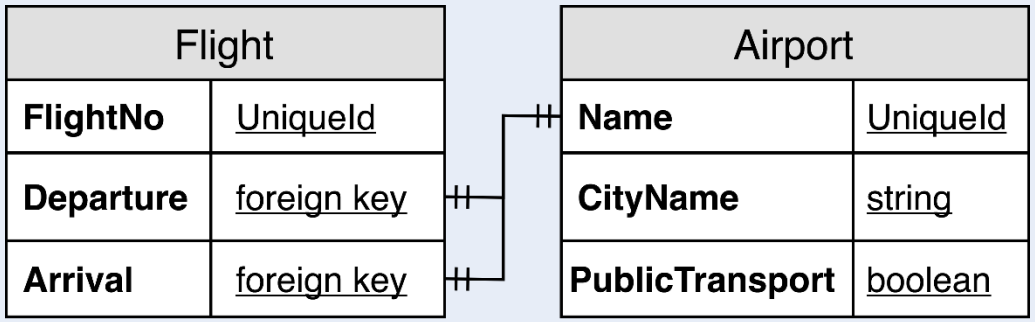
Prolog-style Program

Text-to-SQL Tasks

Natural Language Questions with Database Schema

Input Utterance

Show me flights from Pittsburgh to Seattle

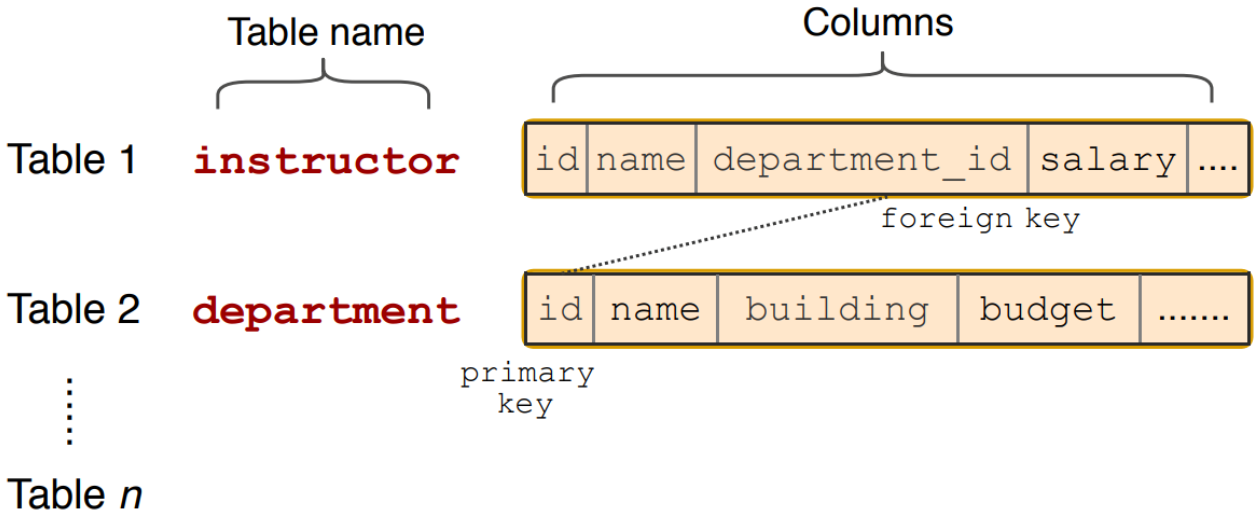


SQL Query

```
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`)



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

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)
```

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

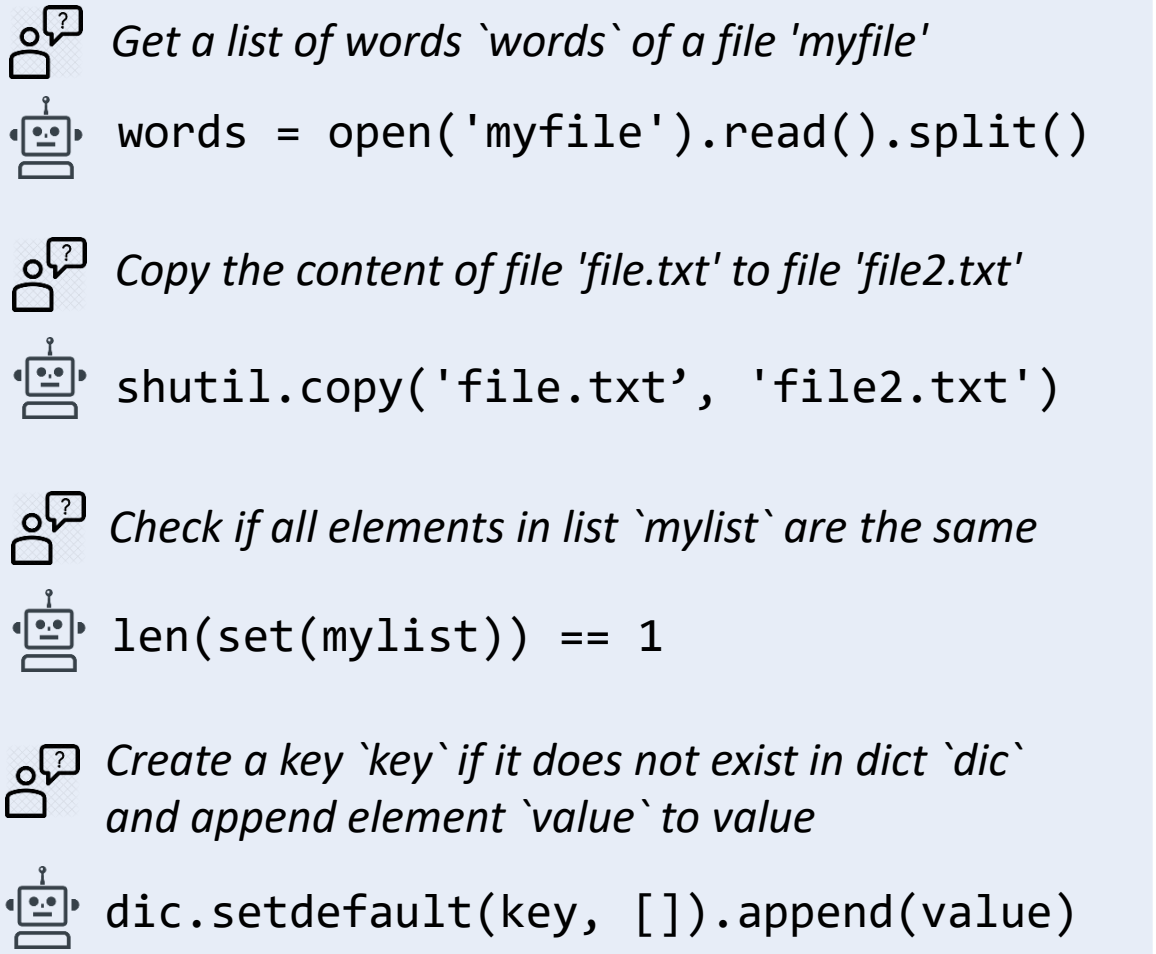
Django

HearthStone

CONCODE

CoNaLa

The CoNaLa Code Generation Dataset



- 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

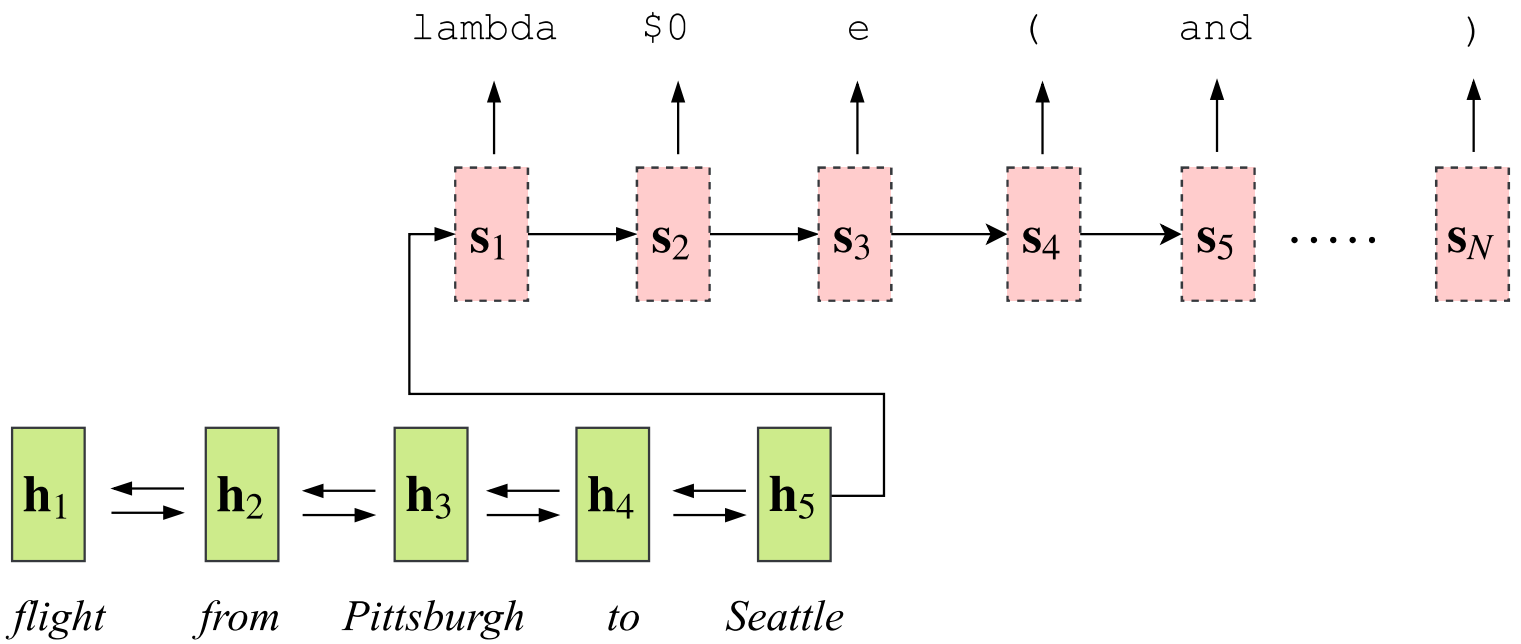
Semantic Parsing as Sequence-to-Sequence Transduction

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

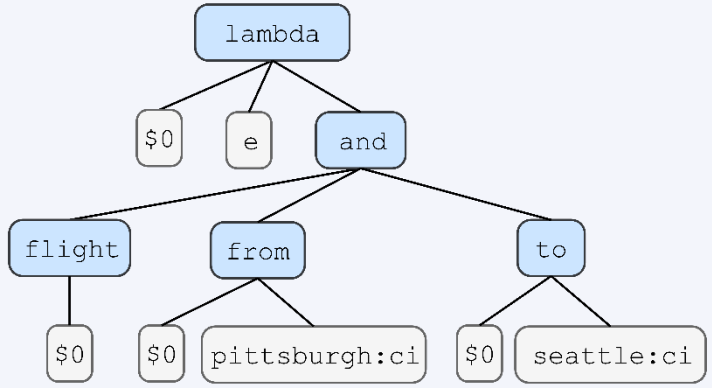
- **Meaning Representations** (e.g., a database query) have strong underlying structures!
- **Issue** Using vanilla seq2seq models ignore the rich structures of meaning representations, and could generate invalid outputs that are not trees

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

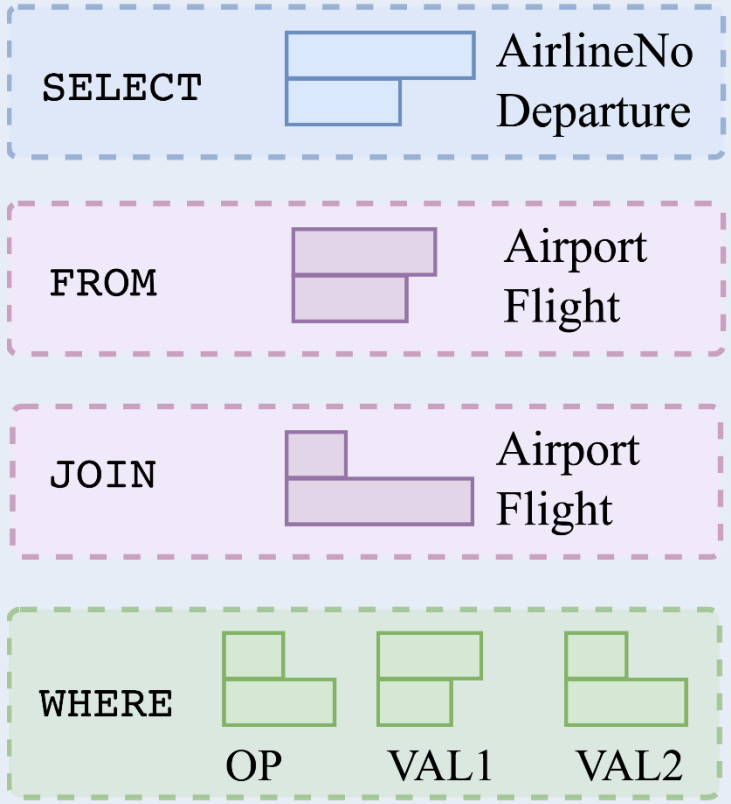
Input Utterance

Show me flights from Pittsburgh to Berkeley

Flight	
FlightNo	UniqueId
Departure	foreign key
Arrival	foreign key

Airport	
Name	UniqueId
CityName	string
PublicTransport	boolean

Predict Programs Following Task-Specific Program Structures



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

Grammar/Syntax-driven Semantic Parsing

- Previously introduced methods could generate tree-structured representations but cannot guarantee they are grammatically 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

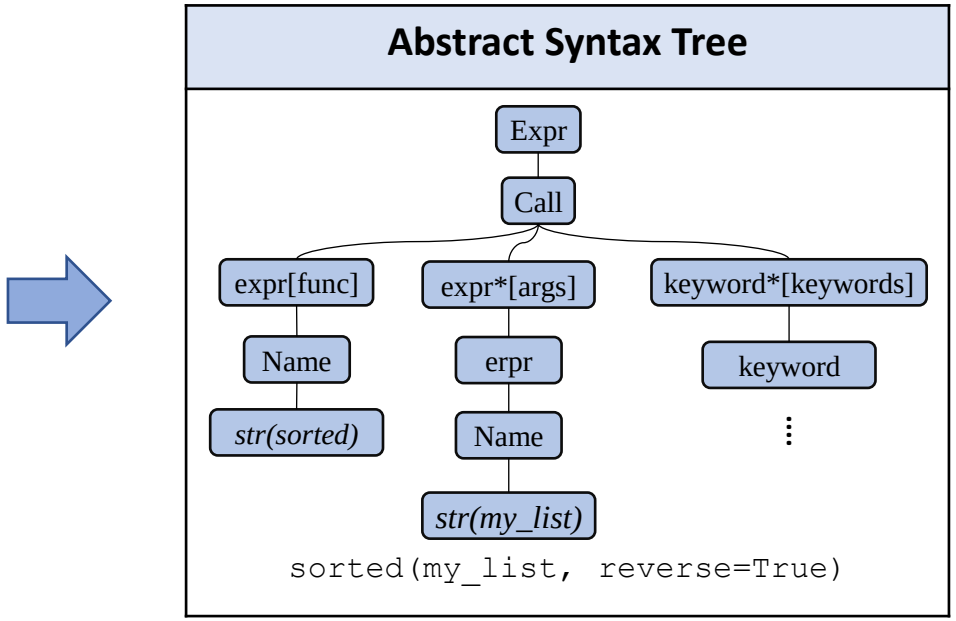
expr ⟼ Name | Call

Call ⟼ expr[*func*] expr*[*args*] keyword*[*keywords*]

If ⟼ expr[*test*] stmt*[*body*] stmt*[*orelse*]

For ⟼ expr[*target*] expr*[*iter*] stmt*[*body*] stmt*[*orelse*]

FunctionDef ⟼ identifier[*name*] expr*[*iter*] stmt*[*body*] stmt*[*orelse*]



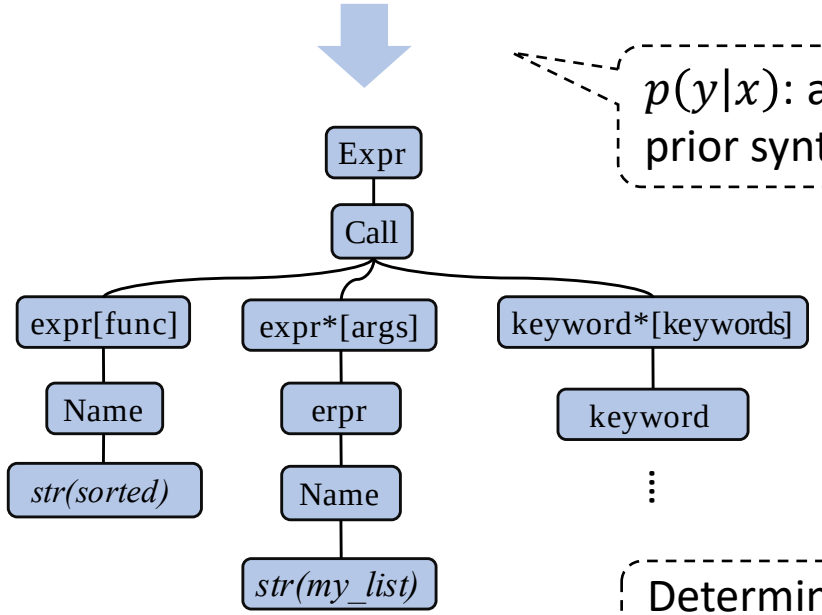
[Yin and Neubig, 2017; Rabinovich *et al.*, 2017]

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*

Generated AST (y)



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

Deterministic transformation (using Python `astor` library)

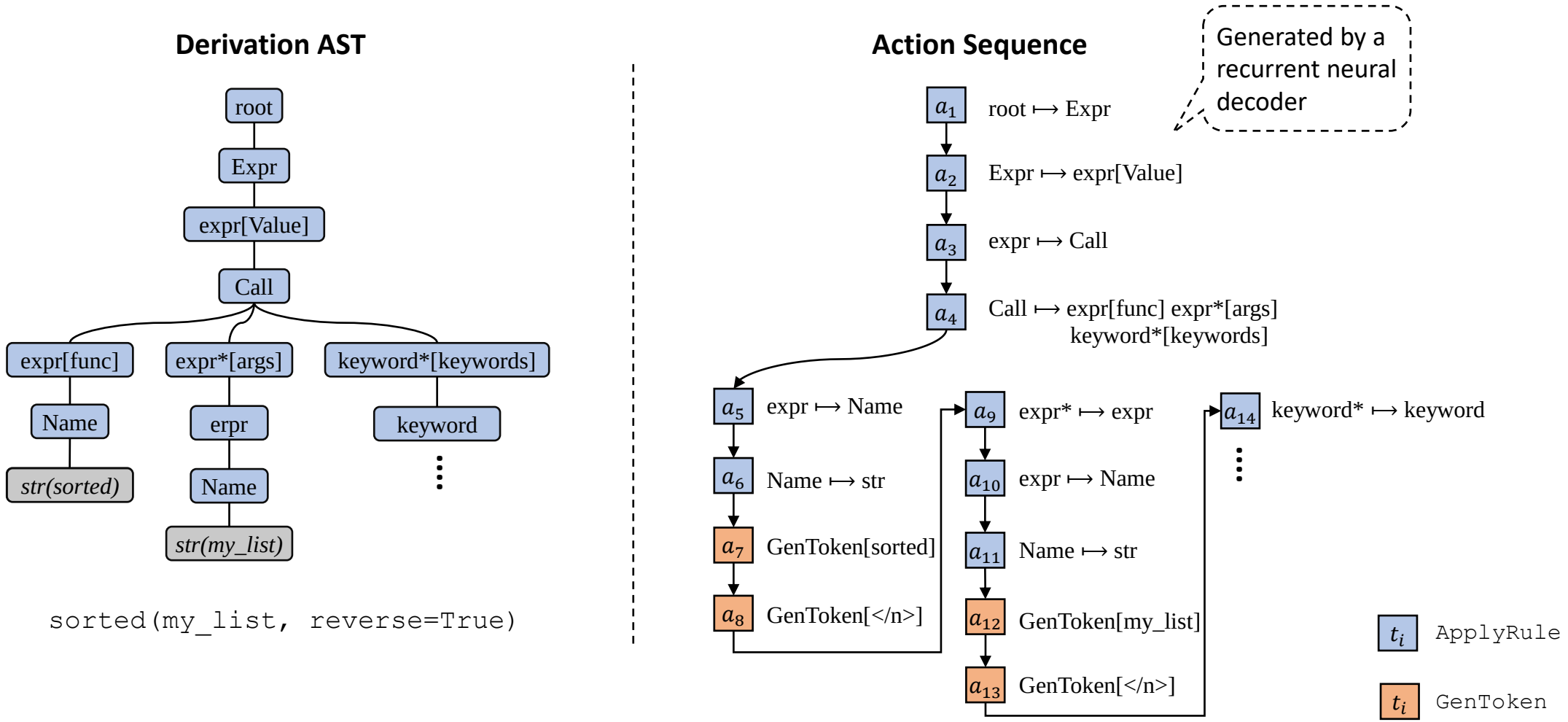
Surface Code (c)

```
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

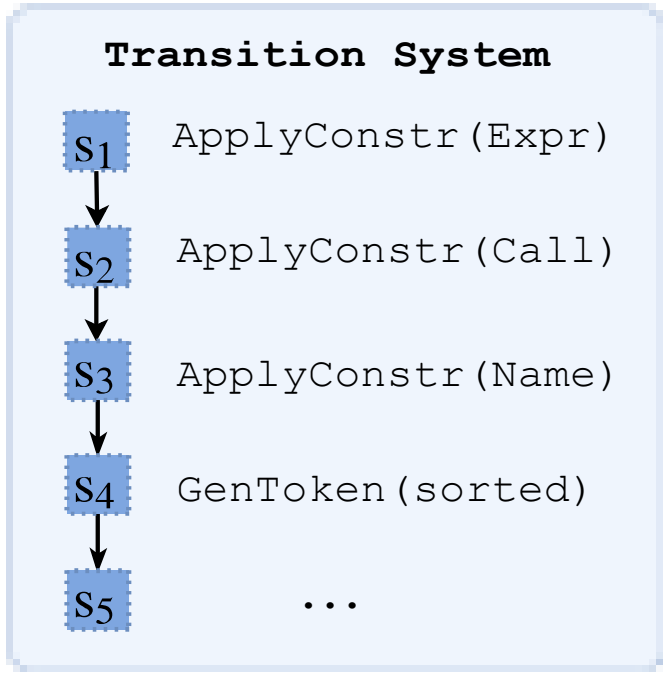
- Convenient interface to specify task-dependent grammar in plain text
- Customizable conversion from abstract syntax trees to domain-specific programs
- Built-in support for many languages: Python, SQL, Lambda Calculus, Prolog…

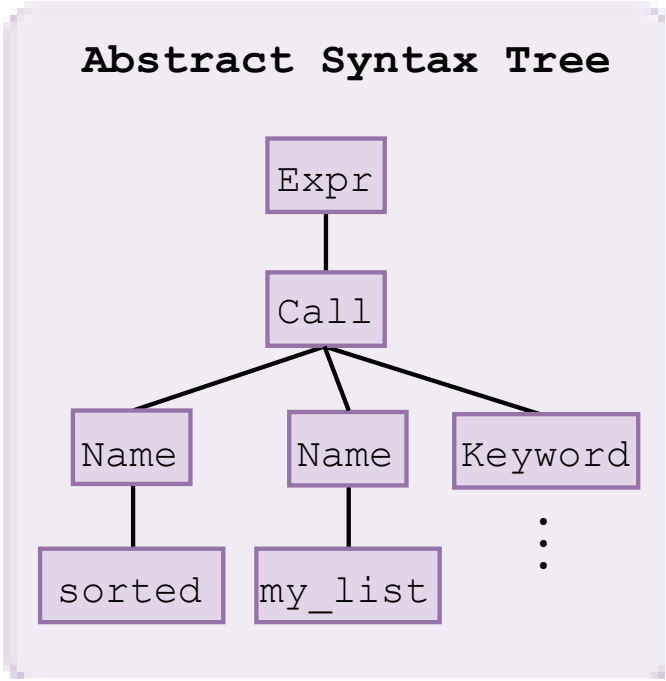
Grammar Specification

```
stmt ↦ FunctionDef(identifiler name,
                   arguments args, stmt* body)
     | Expr(expr value)
expr ↦ Call(expr func, expr* args,
                       keyword* keywords)
     |  Name(identifier id)
     |  Str(string id)
```

Input Utterance

Sort my_list in descending order



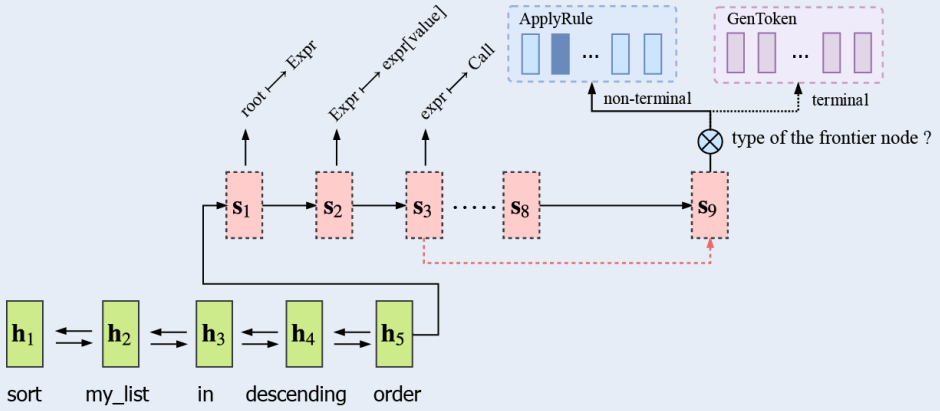


github.com/pcyin/tranX

[Yin and Neubig 2018, Yin and Neubig 2019]

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

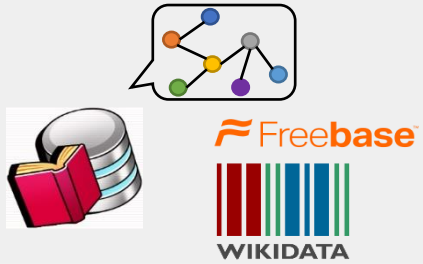
Show me flights from Pittsburgh to Seattle

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 Learning of Semantic Parsers

Incorporate external resources, unlabeled data, noisy data, etc. (*a.k.a Data Augmentation*)

Software library documentations

class `collections.`**`deque`**([*iterable*[, *maxlen*]])
Returns a new deque object initialized ...
`append`(*x*)
Add *x* to the right side of the deque. class methods
`rotate`(*n=1*)
Rotate the deque *n* steps to the right. ...

`heapq.`**`nlargest`**(*n*, *iterable*, *key=None*) top-level functions
Return a list with the *n* largest elements from ...

pre-process

d=collections.deque(*iterable*) d=collections.deque(*iterable*,*maxlen*)
d.append(*x*)
d.rotate() d.rotate(*n=1*)
heapq.nlargest(*n*,*iterable*) heapq.nlargest(*n*,*iterable*,*key=None*)

Stack Overflow Q&A

Removing duplicates in lists ← Intent

406 138

Pretty much I need to write a program to check if a list has any duplicates and if it does it removes them and returns a new list with the items that werent duplicated/removed. This is what I have but to be honest I do not know what to do.

```
def remove_duplicates():
    t = ['a', 'b', 'c', 'd']
    t2 = ['a', 'c', 'd']
    for t in t2:
        t.append(t.remove())
    return t
```

Question

780

The common approach to get a unique collection of items is to use a `set`. Sets are *unordered* collections of *distinct* objects. To create a set from any iterable, you can simply pass it to the built-in `set()` function. If you later need a real list again, you can similarly pass the set to the `list()` function.

The following example should cover whatever you are trying to do:

```
>>> t = [1, 2, 3, 1, 2, 5, 6, 7, 8]
>>> t
[1, 2, 3, 1, 2, 5, 6, 7, 8]
>>> list(set(t))
[1, 2, 3, 5, 6, 7, 8]
>>> s = [1, 2, 3]
>>> list(set(t) - set(s))
[8, 5, 6, 7]
```

Context 1
Snippet 1

Answers

As you can see from the example result, the original order is not maintained. As mentioned above, sets themselves are unordered collections, so the order is lost. When converting a set back to a list, an arbitrary order is created.

222

FWIW, the new (v2.7) Python way for removing duplicates from an iterable while keeping it in the original order is:

```
>>> from collections import OrderedDict
>>> list(OrderedDict.fromkeys('abracadabra'))
['a', 'b', 'r', 'c', 'd']
```

Context 2
Snippet 2

Weakly-supervised Learning of Semantic Parsers

Data augmentation may suffer from distribution mismatch!

class `collections.`**`deque`**([*iterable*[, *maxlen*]])
Returns a new deque object initialized ...
`append`(*x*)
Add *x* to the right side of the deque.
`rotate`(*n=1*)
Rotate the deque *n* steps to the right. ...

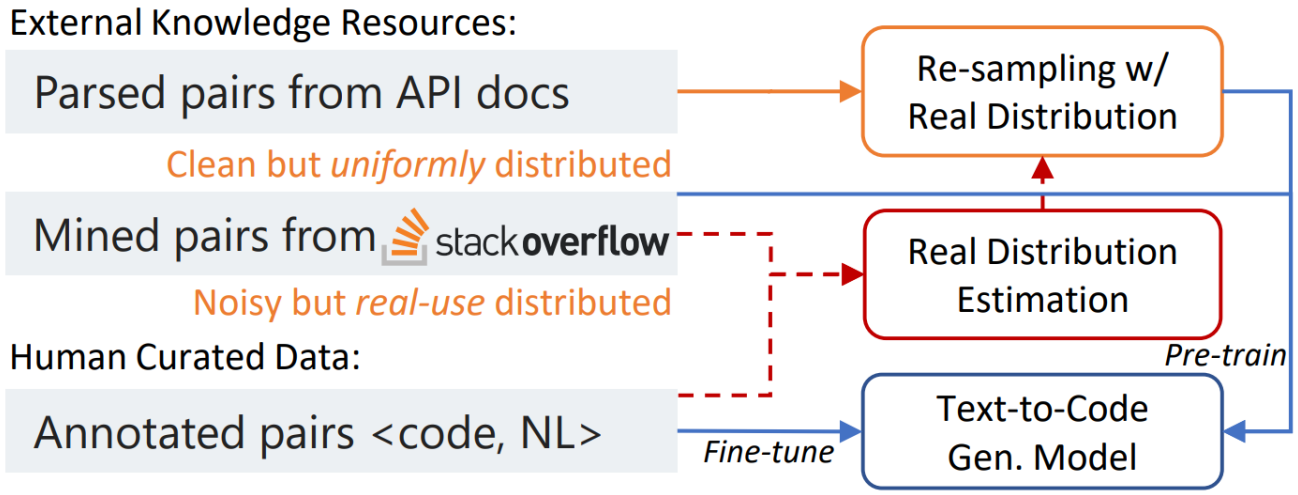
class methods

`heapq.`**`nlargest`**(*n*, *iterable*, *key=None*)
Return a list with the *n* largest elements from ...

top-level functions

pre-process

d=collections.deque(*iterable*) d=collections.deque(*iterable*,*maxlen*)
d.append(*x*)
d.rotate() d.rotate(*n=1*)
heapq.nlargest(*n*,*iterable*) heapq.nlargest(*n*,*iterable*,*key=None*)



[Xu *et al.*, 2020]

GitHub Copilot

OpenAI Codex

Data is important…

OpenAI Codex uses no labeled data, and deals with code linearly without explicitly modeling structures (Transformer-based language models)

```
#!/usr/bin/env ts-node

import { fetch } from "fetch-h2";

// Determine whether the sentiment of text is positive
// Use a web service
async function isPositive(text: string): Promise<boolean> {
  const response = await fetch(`http://text-processing.com/api/sentiment/`, {
    method: "POST",
    body: `text=${text}`,
    headers: {
      "Content-Type": "application/x-www-form-urlencoded",
    },
  });
  const json = await response.json();
  return json.label === "pos";
}
```

Copilot

GitHub Copilot

OpenAI Codex

Data is important...

```python
import datetime

def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of triples (date, value, currency).
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
        2016-01-02 -34.01 USD
        2016-01-03 2.59 DKK
        2016-01-03 -2.72 EUR
    """
    expenses = []
    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
                         float(value),
                         currency))
    return expenses
```

Copilot

GitHub Copilot

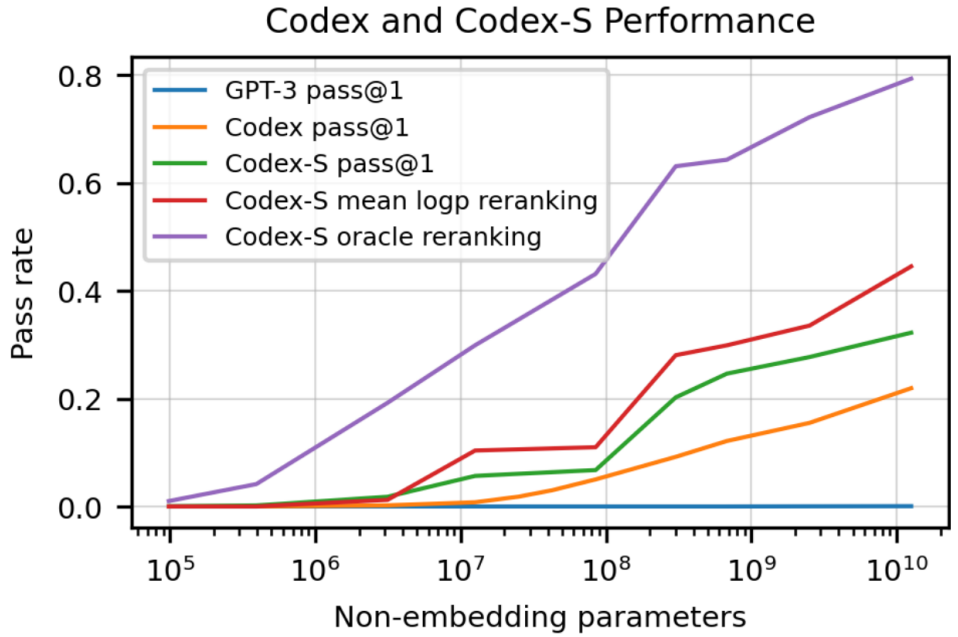
OpenAI Codex

Data is important...

```go
package main

type CategorySummary struct {
    Title        string
    Tasks        int
    AvgValue     float64
}

func createTables(db *sql.DB) {
    db.Exec("CREATE TABLE tasks (id INTEGER PRIMARY KEY, title TEXT, value INTEGER, category TEXT)")
}

func createCategorySummaries(db *sql.DB) ([]CategorySummary, error) {
    var summaries []CategorySummary
    rows, err := db.Query("SELECT category, COUNT(category), AVG(value) FROM tasks GROUP BY category"
    if err != nil {
        return nil, err
    }
    defer rows.Close()

    for rows.Next() {
        var summary CategorySummary
        err := rows.Scan(&summary.Title, &summary.Tasks, &summary.AvgValue)
        if err != nil {
            return nil, err
        }
        summaries = append(summaries, summary)
    }
    return summaries, nil
}
```

Pretraining Comes at a Price

Codex for Python training data: 54 million public software repositories hosted on GitHub, containing 179 GB of unique Python files. The model contains 12 billion parameters.



No access to this amount of data? Recall previously introduced models that exploits induction bias (structures, etc.).

Recap: Workflow of a Semantic Parser

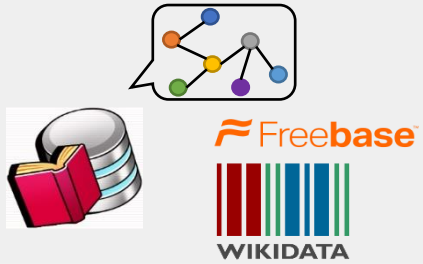
User's Natural Language Query

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



Execution Results (Answer)

1. AS 119
2. AA 3544 -> AS 1101
3. …