# Building Semantic Parsers

Matt Gardner, Allen Institute for Artificial Intelligence

So you want to build a parser…

where do you start?

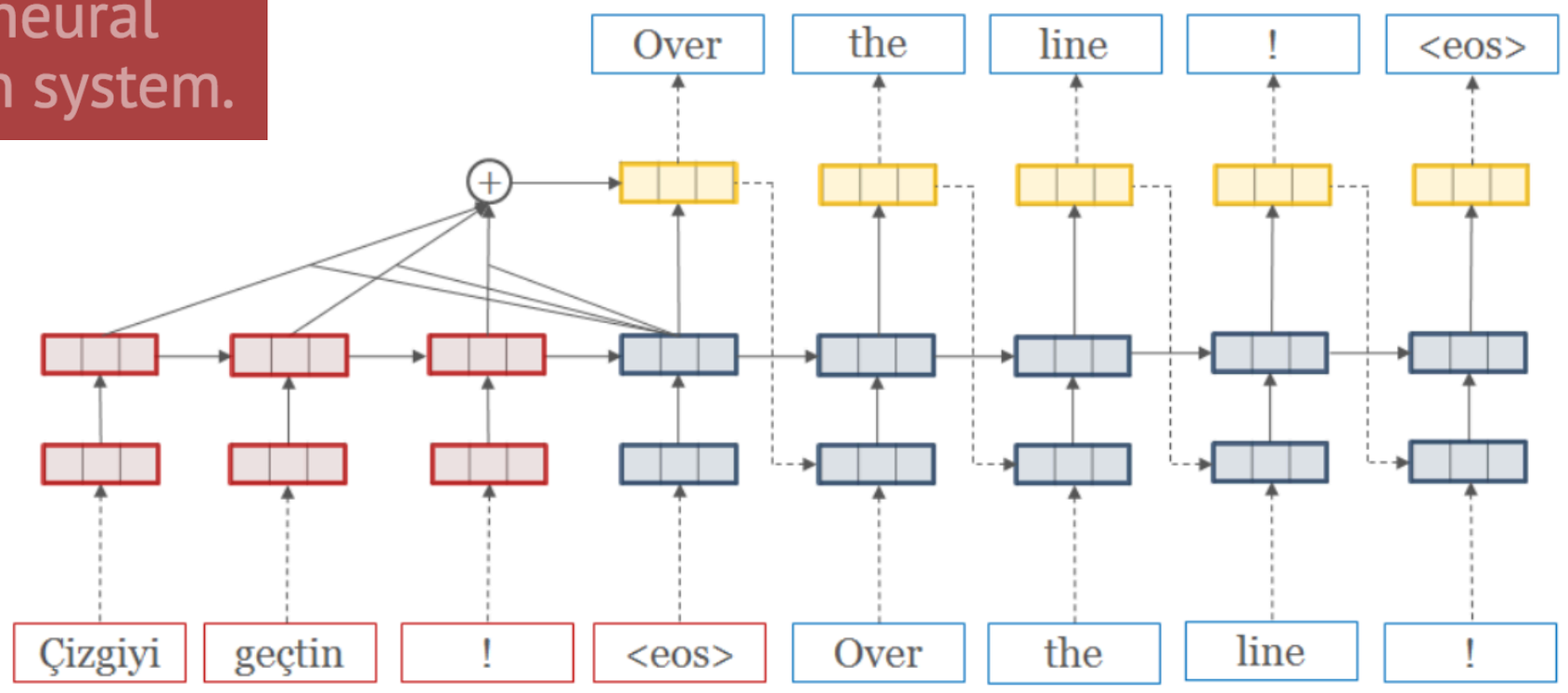
what should you be thinking about?

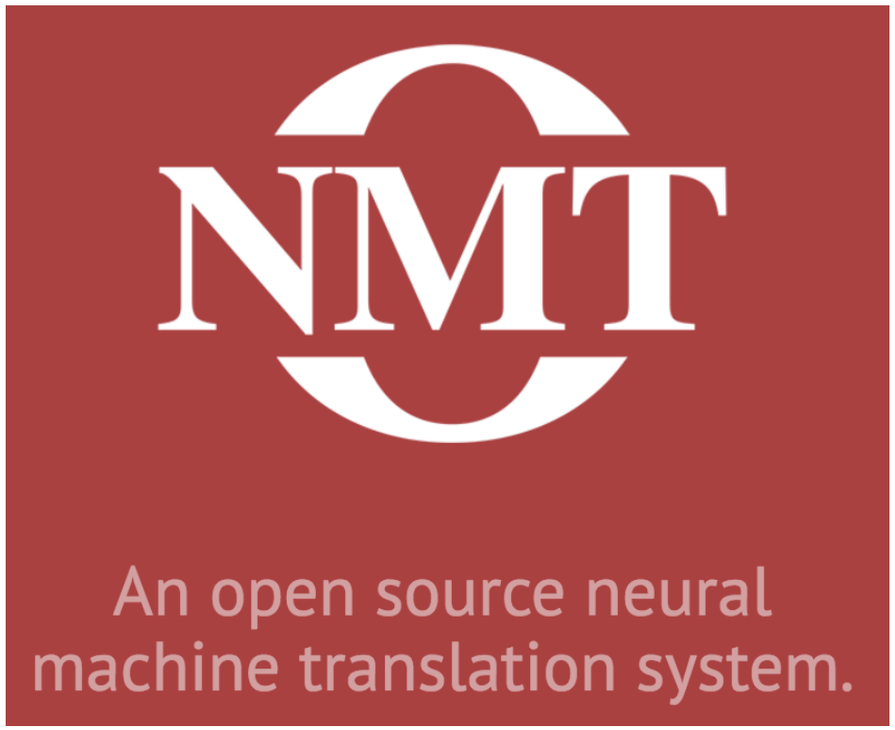
# Lots of code available

- Pre-neural frameworks
    - SEMPRE (Stanford) https://github.com/percyliang/sempre
    - SPF (Cornell / UW) https://github.com/clic-lab/spf
    - WASP (UT Austin) http://www.cs.utexas.edu/~ml/wasp/
- Neural frameworks
    - OpenNMT (Harvard) (needs hacking to do constrained decoding) http://opennmt.net/
    - AllenNLP (AI2) (coming soon) https://github.com/allenai/allennlp
- Code for single papers
    - https://github.com/donglixp/coarse2fine
    - https://github.com/clic-lab/atis
    - https://github.com/udiNaveh/nlvr_tau_nlp_final_proj
    - https://github.com/sriniiyer/nl2sql
    - https://github.com/allenai/pnp/tree/wikitables2
    - … (just look for papers, most have code these days)

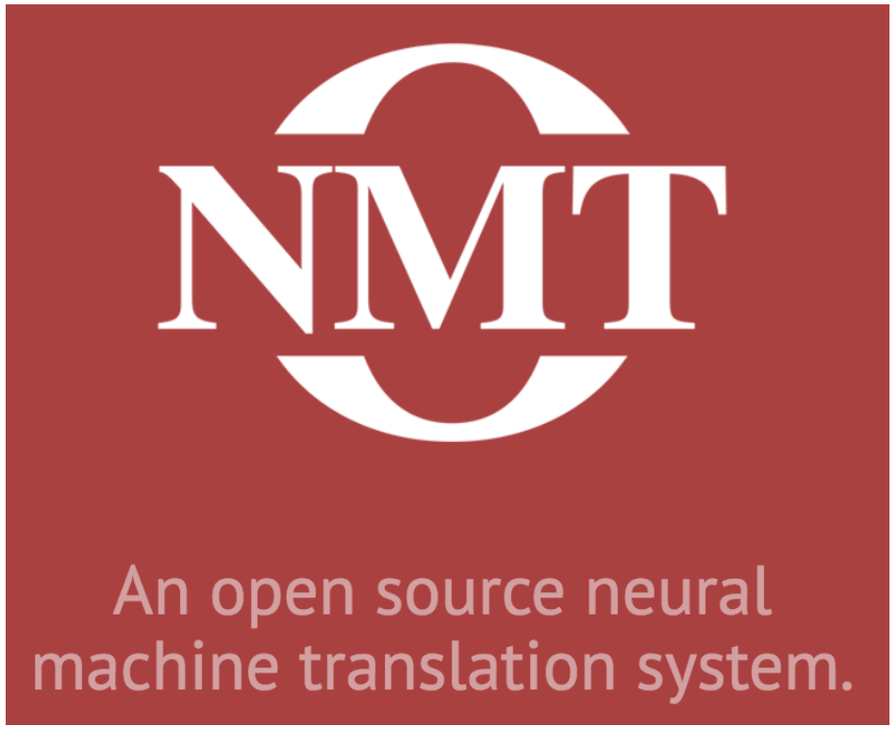
NMT

An open source neural machine translation system.






- Problem: you really want to do some kind of constrained decoding



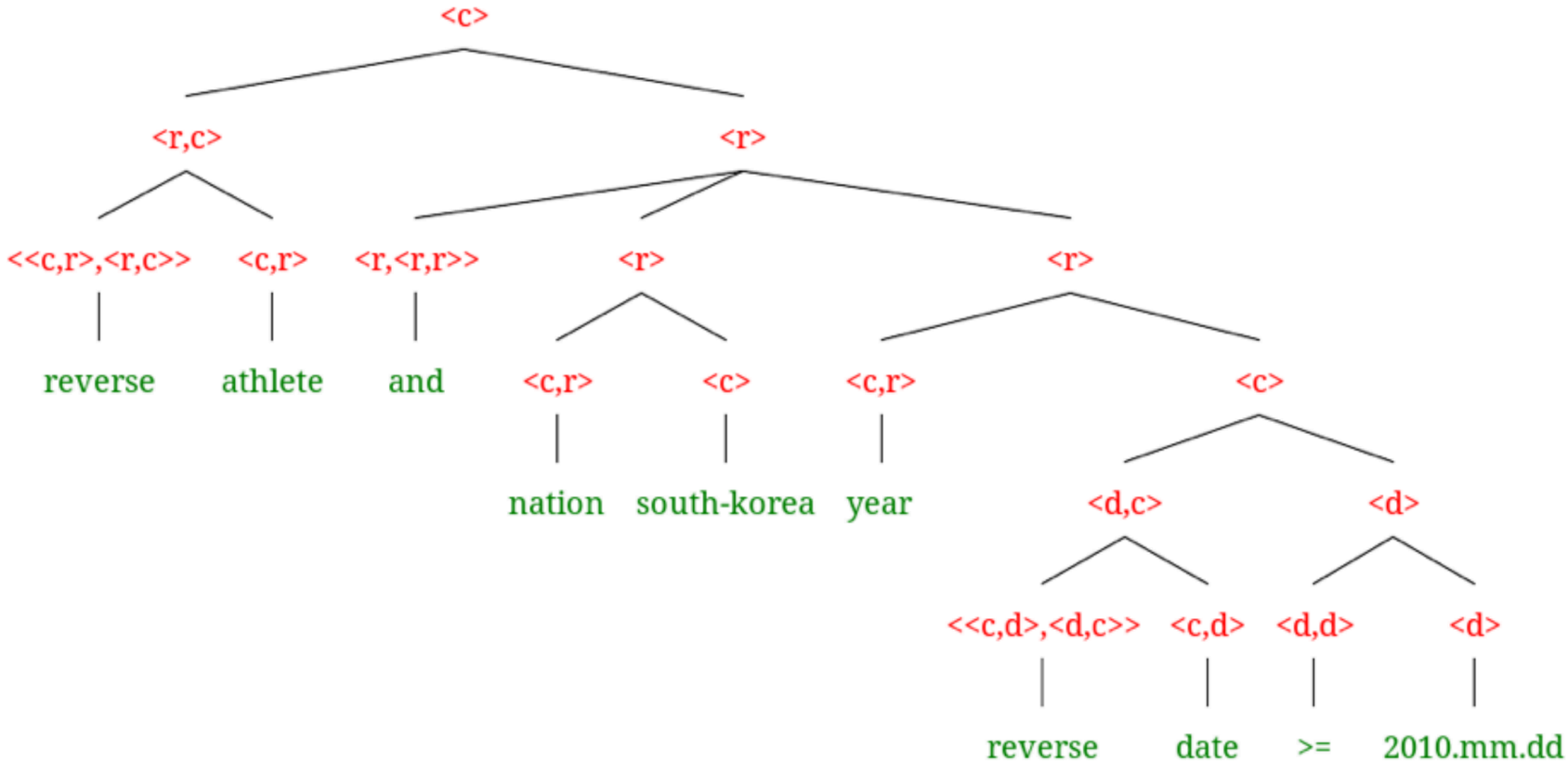

- Problem: you really want to do some kind of constrained decoding
- Dong and Lapata have done a lot of work using OpenNMT, with seq2seq and seq2tree models, so check out their code if you want to go this route

# What you need on top of seq2seq

# What you need on top of seq2seq

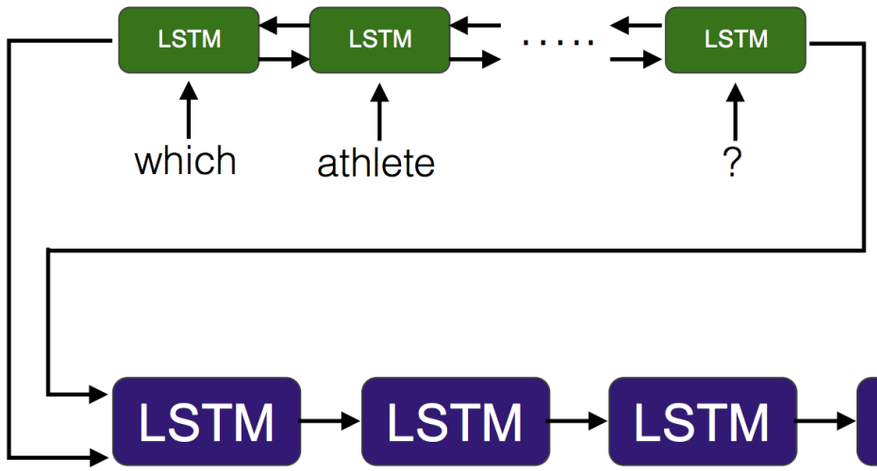
1. Convert programs to action sequences

```
((reverse athlete) (and (nation south_korea) (year ((reverse date) (>=
2010-mm-dd)))
```

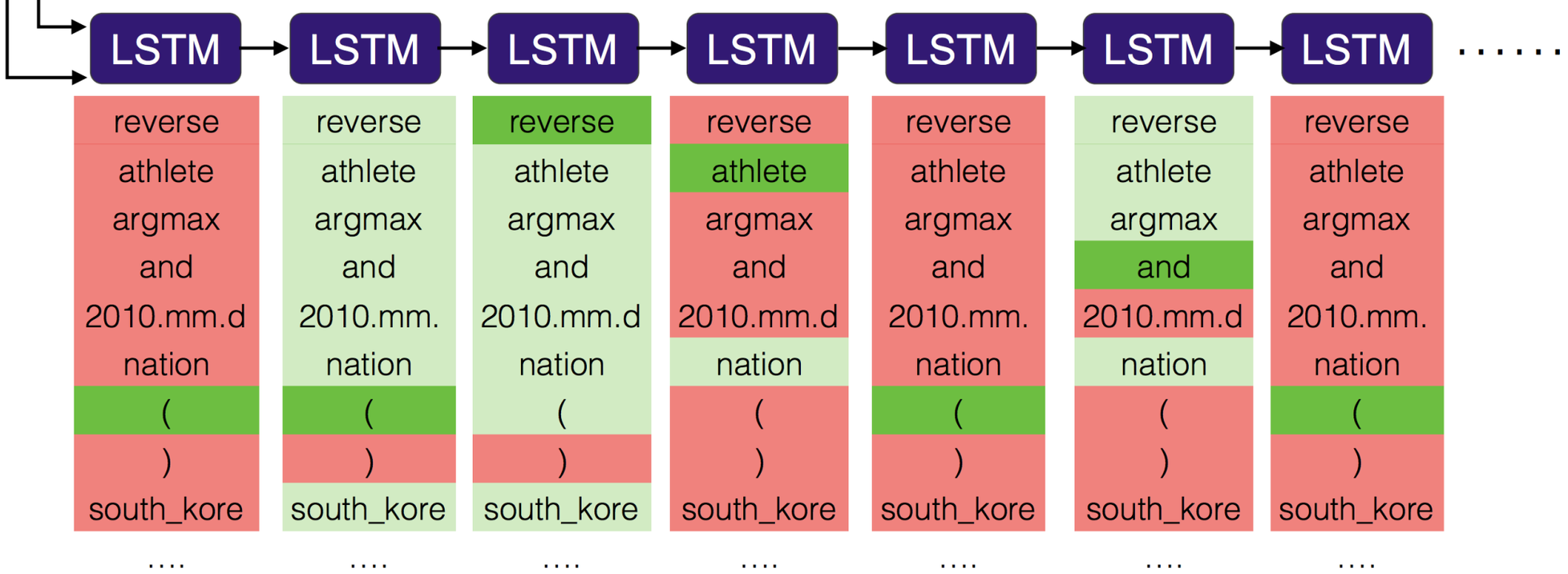


# What you need on top of seq2seq

## 2. What actions are valid at every timestep?




| Athlete | Nation | Olympics | Medals |
|---|---|---|---|
| Kim Yu-na | South Korea (KOR) | 2010–2014 | 2 |
| Tenley Albright | United States (USA) | 1952-1956 | 2 |

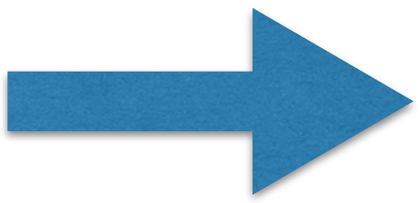


# What you need on top of seq2seq

## 3. Convert action sequences back to programs

**Generated Actions**

c
c→(<r,c> r)
<r,c>→(<<c,r>,<r,c>> <c,r>)
<<c,r>,<r,c>>→**reverse**
<c,r>→**athlete**
r→(<r,<r,r>> r r)
<r,<r,r>>→**and**
r→(<c,r> c)
<c,r>→**nation**
c→**south_korea**
r→(<c,r> c)
<c,r>→**year**
c→(<d,c> d)
<d,c>→(<<c,d>,<d,c>> <c,d>)
<<c,d>,<d,c>>→**reverse**
<c,d>→**date**
d→(**>=** d)
d→**2010.mm.dd**



**Logical Form**

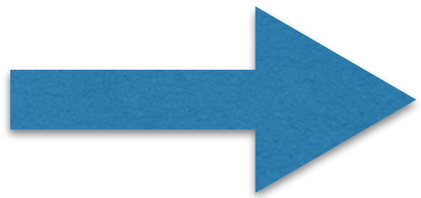
```
((reverse athlete)
 (and (nation south_korea)
      (year ((reverse date)
             (>= 2010-mm-dd)))
```

# What you need on top of seq2seq

4. (sometimes) A way to execute programs

**Logical Form**

```
((reverse athlete)
 (and (nation south_korea)
      (year ((reverse date)
             (>= 2010-mm-dd)))
```



| Athlete | Nation | Olympics | Medals |
|---|---|---|---|
| Gillis Grafström | Sweden (SWE) | 1920–1932 | 4 |
| Evgeni Plushenko | Russia (RUS) | 2002–2014 | 4 |
| Karl Schäfer | Austria (AUT) | 1928–1936 | 2 |
| Katarina Witt | East Germany (GDR) | 1984–1988 | 2 |
| Tenley Albright | United States (USA) | 1952-1956 | 2 |
| Kim Yu-na | South Korea (KOR) | 2010–2014 | 2 |
| Patrick Chan | Canada (CAN) | 2014 | 2 |

# What you need on top of seq2seq

5. If you don’t have labeled logical forms: a different way to train

# What you need on top of seq2seq

1. Convert programs to action sequences

2. What actions are valid at every timestep?

3. Convert action sequences back to programs

4. (sometimes) A way to execute programs

5. If you don’t have labeled logical forms: a different way to train

# A few additional considerations

# Token-based or grammar-based?

| | Token-based | Grammar-based |
|---|---|---|
| **Programs to actions** | **Trivial** | **Harder** |
| **What actions are valid?** | **Harder** | **Trivial** |
| **Actions to programs** | **Trivial** | **Harder** |

This decision also has modeling implications - one might be easier on the model than the other

# Programs to actions

- This can be surprisingly difficult to get right - don't underestimate how much work it is to get a good grammar!
- The way that you define the action space can have a large impact on your model performance
    - If you write the language, the closer it is to your utterances, the better
    - If you're using a programming language, you still might want to simplify / collapse parts of the grammar
- Particularly important: which parts of your grammar are specific to individual instances?
    - In WikiTableQuestions: table cells, numbers
    - In source code: the other classes and methods in the current scope, the allowed methods on an object

# AllenNLP

An open-source NLP research library, built on PyTorch

- The only neural framework for semantic parsing
- Still in progress, but early version is available now, official release in the next month or two
- You get all of the benefits of AllenNLP (configurability, easy ELMo, easy demos, …), plus…

AllenNLP

An open-source NLP research library, built on PyTorch

# Grammars

- An easy way to define lisp-like languages, if you are writing your own (can be bypassed if you're not)

- Handles going from programs to actions and back again, and getting the valid actions at each grammar state

AllenNLP

An open-source NLP research library, built on PyTorch

# Model

- Semantic parsing model is a state machine
- Start in the initial grammar state; model ranks valid transitions between states
- Transition function and state representation are re-usable across models
- State machines are more general than semantic parsers - you can use this for other tasks, too

AllenNLP

An open-source NLP research library, built on PyTorch

# Training

- Allows for many ways of training the state machine
  - Fully-supervised (e.g., maximum likelihood)
  - Weakly supervised (e.g., maximum marginal likelihood, or answer-only)
  - Reinforcement learning (with a reward function)

AllenNLP

An open-source NLP research library, built on PyTorch

# Datasets

- WikiTableQuestions
- Cornell NLVR
- ATIS
- Kushman open algebra questions
- CONCODE
- Other text-to-SQL datasets
- …

# Tutorial Summary

- Seq2Seq-based models have taken over semantic parsing research
- Datasets
- Constrained Decoding
- Language to Code
- Language in Context
- Building semantic parsers

Slides will be at https://github.com/allenai/acl2018-semantic-parsing-tutorial

Happy semantic parsing!