Constrained Decoding

Pradeep Dasigi
Carnegie Mellon University / Allen Institute for Artificial Intelligence
Introduction

• Traditional semantic parsers used grammar based parsing algorithms

• Neural semantic parsing has moved towards using encoder decoder models

• Decoders are usually recurrent neural networks that produce sequences
  • But they can produce outputs that are not valid (syntactically or semantically)!
Example from WikiTableQuestions

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

**Question:**

Which athlete was from South Korea after 2010?

\[
((\text{reverse athlete}) \quad \text{and} \quad \text{(nation south_korea)} \quad \text{(year ((reverse date) (>= 2010-mm-dd))))})
\]

[Pasupat and Liang, 2015]
Athlete | Nation | Olympics   | Medals |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Kim Yu-na</td>
<td>South Korea (KOR)</td>
<td>2010–2014</td>
<td>2</td>
</tr>
<tr>
<td>Tenley Albright</td>
<td>United States (USA)</td>
<td>1952-1956</td>
<td>2</td>
</tr>
</tbody>
</table>
Kim Yu-na
South Korea (KOR)
2010–2014
2

Tenley Albright
United States (USA)
1952-1956
2
Seq2Seq Output Space

<table>
<thead>
<tr>
<th>Athlete</th>
<th>Nation</th>
<th>Olympics</th>
<th>Medals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kim Yu-na</td>
<td>South Korea (KOR)</td>
<td>2010–2014</td>
<td>2</td>
</tr>
<tr>
<td>Tenley Albright</td>
<td>United States (USA)</td>
<td>1952-1956</td>
<td>2</td>
</tr>
</tbody>
</table>
Example from ATIS

flights from Dallas leaving after 4 in the afternoon

(lambda $0 e
  (and
    (> (departure_time $0) 1600:ti)
    (from $0 dallas:ci)))

[Hemphill et al., 1990; Dahl et al., 1994]
Seq2Seq Output Space

- flights
- from Dallas
- leaving after 4
- in the afternoon
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
- $0 \lambda ( )$
- e departure_and from dallas:ci
Constrained Decoding

- Constrain the output space to selections that matter
- **Inference:** Avoid invalid parses
- **Training:** Do not waste modeling power in distinguishing invalid parses from valid ones!

**Token-based Decoding:**
The output space is tokens, but they are constrained to be relevant at each time step.

**Grammar-based Decoding:**
The output space is production rules, and a grammar defines the constraints.
Constrained Decoding

- Constrain the output space to selections that matter
- **Inference**: Avoid invalid parses
- **Training**: Do not waste modeling power in distinguishing invalid parses from valid ones!

**Token-based Decoding**

Dong and Lapata. 2016. *Language to Logical Form with Neural Attention*. In ACL.

Dong and Lapata. 2018. *Coarse-to-Fine Decoding for Neural Semantic Parsing*. In ACL.


**Grammar-based Decoding**:


Yin and Neubig. 2017. *A Syntactic Neural Model for General Purpose Code Generation*. In ACL.

Krishnamurthy, Dasigi, and Gardner. 2017. *Neural Semantic Parsing with Type Constraints for Semi-Structured Tables*. In EMNLP.
Token-based Constrained Decoding
Constraining output structure: Seq2Tree

Flights from Dallas leaving after 4 in the afternoon

\[(\lambda \, 0 \, e \, (\text{and} \, (> \, (\text{departure\_time} \, 0) \, 1600:ti) \, (\text{from} \, 0 \, \text{dallas:ci})))\]
Constraining output structure: Seq2Tree

Flights from Dallas leaving after 4 in the afternoon

(lamba $0 e <n>)}

[Dong and Lapata, 2016]
Constraining output structure: Seq2Tree

Flights from Dallas leaving after 4 in the afternoon

(lambda $0 e (and <n> <n>))
Constraining output structure: Seq2Tree

Flights from Dallas leaving after 4 in the afternoon

(lambdas $0 \ e 
  \ (\text{and} 
    (\text{>} <n> 1600:ti) 
    <n>) ))
Constraining output structure: Seq2Tree

Flights from Dallas leaving after 4 in the afternoon

(lambda $0 e
  (and
    (> (departure_time $0) 1600:ti)
    <n>)))

[Dong and Lapata, 2016]
Constraining output structure: Seq2Tree

Flights from Dallas leaving after 4 in the afternoon

(lambda $0 e
  (and
    (> (departure_time $0) 1600:ti)
    (from $0 dallas:ci)))

[Dong and Lapata, 2016]
Constraining output structure: Seq2Tree

Flights from Dallas leaving after 4 in the afternoon

Need not explicitly model matching parentheses
Syntactically valid trees
Allows parent feeding
Semantically valid trees

[Dong and Lapata, 2016]
Empirical Comparison with Seq2Seq on GEO and ATIS

[GEO]

<table>
<thead>
<tr>
<th>Method</th>
<th>GEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq</td>
<td>84.6</td>
</tr>
<tr>
<td>Seq2Tree</td>
<td>87.1</td>
</tr>
</tbody>
</table>

[ATIS]

<table>
<thead>
<tr>
<th>Method</th>
<th>ATIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq</td>
<td>84.2</td>
</tr>
<tr>
<td>Seq2Tree</td>
<td>84.6</td>
</tr>
</tbody>
</table>

[Dong and Lapata, 2016]
Sketch-Constrained Seq2Tree

• Decoding in two steps:
  1. Decoder 1: Rough sketch conditioned on encoder output
  2. Decoder 2: Finer output constrained by the sketch, conditioned on the outputs of decoder 1 and encoder

[Dong and Lapata, 2018]
Grammar-based Constrained Decoding
Constraining output structure and types

<table>
<thead>
<tr>
<th>Athlete</th>
<th>Nation</th>
<th>Olympics</th>
<th>Medals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gillis Grafström</td>
<td>Sweden (SWE)</td>
<td>1920–1932</td>
<td>4</td>
</tr>
<tr>
<td>Kim Yu-na</td>
<td>South Korea (KOR)</td>
<td>2010–2014</td>
<td>2</td>
</tr>
<tr>
<td>Patrick Chan</td>
<td>Canada (CAN)</td>
<td>2014</td>
<td>2</td>
</tr>
</tbody>
</table>

Basic Types:
Row (r); Cell (c) (kim_yu_na; 2014; …); Number (n); Date (d)

Complex Types:
Column (<c,r>): athlete; nation; olympics; medals
Binary row operations (<r,<r,r>>): and; or
Reverse column operation (<<<c,r>,<r,c>>): reverse

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

[Krishnamurthy, Dasigi and Gardner, 2017]
Note on the notation of types

• Complex types
  • Example: column: \texttt{cell} \rightarrow \texttt{row} \ <c,r>  
  • Concrete example: (nation south_korea)

• Currying for functions with multiple arguments
  • Example: binary row operator: \texttt{row, row} \rightarrow \texttt{row}
    Rewritten as: \texttt{row} \rightarrow (\texttt{row} \rightarrow \texttt{row}) \ <r,<r,r>>
  • Concrete example:
    (and (nation south_korea) (medals 4))

• Higher order functions
  • Example: reverse: (\texttt{cell} \rightarrow \texttt{row}) \rightarrow (\texttt{row} \rightarrow \texttt{cell}) \ <<c,r>,<r,c>>
  • Concrete example:
    ((reverse athlete) (and (nation south_korea) (medals 4)))

<table>
<thead>
<tr>
<th>Athlete</th>
<th>Nation</th>
<th>Olympics</th>
<th>Medals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gillis</td>
<td>Sweden</td>
<td>1920–2014</td>
<td>4</td>
</tr>
<tr>
<td>Kim Yu-</td>
<td>South</td>
<td>2010–2014</td>
<td>2</td>
</tr>
<tr>
<td>Patrick</td>
<td>Canada</td>
<td>2014</td>
<td>2</td>
</tr>
</tbody>
</table>
Constraining output structure and types

<table>
<thead>
<tr>
<th>Athlete</th>
<th>Nation</th>
<th>Olympics</th>
<th>Medals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gillis Grafström</td>
<td>Sweden (SWE)</td>
<td>1920–1932</td>
<td>4</td>
</tr>
<tr>
<td>Kim Yu-na</td>
<td>South Korea (KOR)</td>
<td>2010–2014</td>
<td>2</td>
</tr>
<tr>
<td>Patrick Chan</td>
<td>Canada (CAN)</td>
<td>2014</td>
<td>2</td>
</tr>
</tbody>
</table>

Basic Types:
Row (r); Cell (c) (kim_yu_na; 2014; …); Number (n); Date (d)

Complex Types:
Column (<c,r>): athlete; nation; olympics; medals
Binary row operations (<r,<r,r>>): and; or
Reverse column operation (<<c,r>,<r,c>>): reverse

[ Krishnamurthy, Dasigi and Gardner, 2017]
Constraining output structure and types

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

[Krishnamurthy, Dasigi and Gardner, 2017]
Constraining output structure and types

\(((\text{reverse athlete}) \land (\text{nation south\_korea}) \land (\text{year } ((\text{reverse date}) \geq 2010\text{-mm-dd}))))

Generate the tree as a sequence of typed actions

[ Krishnamurthy, Dasigi and Gardner, 2017 ]
Which athlete was from South Korea after the year 2010?

Generated Actions
START $\rightarrow$ c

Logical Form
C

Non-terminal Stack

[Krishnamurthy, Dasigi and Gardner, 2017]
Which athlete was from South Korea after the year 2010?

Generated Actions
START → c

Logical Form
c

Non-terminal Stack

[ Krishnamurthy, Dasigi and Gardner, 2017 ]
Grammar-Constrained Decoding

Which athlete was from South Korea after the year 2010?

Generated Actions
START → c
c→(<r,c> r)

Logical Form
(<r,c> r)

Non-terminal Stack
Grammar-Constrained Decoding

Which athlete was from South Korea after the year 2010?

Generated Actions
START → c
c→(<r,c> r)

Logical Form
(<r,c> r)

Non-terminal Stack

[ Krishnamurthy, Dasigi and Gardner, 2017 ]
Which athlete was from South Korea after the year 2010?

Generated Actions
START → c
c→(r,c) r
<r,c>→(<c,r>,<r,c>) <c,r>

Logical Form
((<c,r>,<r,c>, <c,r>) r)

Non-terminal Stack
[ Krishnamurthy, Dasigi and Gardner, 2017 ]
Generated Actions
START → c
   c→(<r,c> r)
   <r,c>→(<c,r>,<r,c>> <c,r>)

Logical Form
((<<c,r>,<r,c>>, <c,r>) r)

Non-terminal Stack
Grammar-Constrained Decoding

Which athlete was from South Korea after the year 2010?

Generated Actions
START $\rightarrow$ c
  c $\rightarrow$ (<r,c> r)
  <r,c> $\rightarrow$ ([<c,r>,<r,c>> <c,r>]
  <<c,r>,<r,c>> $\rightarrow$ reverse

Logical Form
((reverse <c,r>) r)

Non-terminal Stack

[ Krishnamurthy, Dasigi and Gardner, 2017 ]
Grammar-Constrained Decoding

Which athlete was from South Korea after the year 2010?

Generated Actions
START → c
   c→(r,c) r)
   <r,c>→(<<c,r>,<r,c>> <c,r>)
   <<c,r>,<r,c>>→reverse
   <c,r>→athlete

Logical Form
((reverse athlete) r)

Non-terminal Stack

[ Krishnamurthy, Dasigi and Gardner, 2017 ]
Grammar-Constrained Decoding

Which athlete was from South Korea after the year 2010?

Generated Actions
START → 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

Logical Form
((reverse athlete)
(and r r))

Non-terminal Stack

[ Krishnamurthy, Dasigi and Gardner, 2017 ]
Which athlete was from South Korea after the year 2010?

Grammar-Constrained Decoding

Generated Actions
START → 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

Logical Form
((reverse athlete)
 (and (nation c) r))

Non-terminal Stack

[ Krishnamurthy, Dasigi and Gardner, 2017]
Grammar-Constrained Decoding

Which athlete was from South Korea after the year 2010?

Generated Actions
START $\rightarrow$ c
$c$ $\rightarrow$ $(<c,r> \text{ } r)$
$<r,c>$ $\rightarrow$ $(<<c,r>,<r,c>> \text{ } <c,r>)$
$<<c,r>,<r,c>>$ $\rightarrow$ reverse
$<c,r>$ $\rightarrow$ athlete
$r$ $\rightarrow$ $(<r,<r,r>> \text{ } r \text{ } r)$
$<r,<r,r>>$ $\rightarrow$ and
$r$ $\rightarrow$ $(<c,r> \text{ } c)$
$<c,r>$ $\rightarrow$ nation
$c$ $\rightarrow$ south_korea

Logical Form
$((\text{reverse} \text{ athlete})$ 
(and (nation south_korea)
$r))$
Which athlete was from South Korea after the year 2010?

Generated Actions
START → 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

Logical Form
((reverse athlete)
 (and (nation south_korea)
  (year c)))
Grammar-Constrained Decoding

Which athlete was from South Korea after the year 2010?

Generated Actions

START → 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

Logical Form

((reverse athlete)

(and (nation south_korea)

(year ((reverse date)

(≥ 2010-mm-dd))))

Non-terminal Stack

[ Krishnamurthy, Dasigi and Gardner, 2017 ]
Grammar-Constrained Decoding

Which athlete was from South Korea after the year 2010?

Generated Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>START → c</td>
<td>&lt;c,d&gt; → date</td>
</tr>
<tr>
<td>c → (&lt;r,c&gt; r)</td>
<td>d → (≥ d)</td>
</tr>
<tr>
<td>&lt;r,c&gt; → (&lt;&lt;c,r&gt;,&lt;r,c&gt;&gt; &lt;c,r&gt;)</td>
<td>d → 2010.mm.dd</td>
</tr>
<tr>
<td>&lt;&lt;c,r&gt;,&lt;r,c&gt;&gt;</td>
<td>reverse</td>
</tr>
</tbody>
</table>

Logical Form

((reverse athlete)
  (and (nation south_korea)
    (year ((reverse date)
      (≥ 2010-mm-dd))))

Need not explicitly model matching parentheses
Syntactically valid trees
Semantically valid trees

Non-terminal Stack

[c
[<r,c>
| south_korea |
| year |
| [<d,c> d] |
| [<c,d>,<d,c>> <c,d>] |
| [<c,d>,<d,c>> reverse] |
Empirical Comparison with Seq2Seq and Seq2Tree on WikiTableQuestions

[Krishnamurthy, Dasigi and Gardner, 2017]
Summary

- Constraining output forces decoder to generate only valid outputs
- Impose hard constraints instead of hoping the model would learn them
- Various hard constraints depending on output space
  - Token-level decoding (Seq2Tree, sketches, etc)
  - Grammar-based constraints