New Resources and Ideas for Semantic Parsing

Kyle Richardson

Institute for Natural Language Processing (IMS)

School of Computer Science, Electrical Engineering and Information Technology
University of Stuttgart

May 28, 2018

Collaborators: Jonas Kuhn (advisor, Stuttgart) and Jonathan Berant (work on "polyglot semantic parsing", Tel Aviv)
Main Topic: Semantic Parsing

- **Task**: mapping text to formal meaning representations (ex., from Herzig and Berant (2017)).

  **Text**: *Find an article with no more than two authors.*

  →

  **LF**: \( \text{Type}\cdot\text{Article} \sqcap R[\lambda x.\text{count}(\text{AuthorOf}\cdot x)] \leq 2 \)
Main Topic: Semantic Parsing

- **Task**: mapping text to formal meaning representations (ex., from Herzig and Berant (2017)).

  **Text**:  
  Find an article with no more than two authors.  
  →  
  **LF**: Type.Article \( \sqcap \mathbf{R}[\lambda x.\text{count}(\text{AuthorOf}.x)] \leq 2 \)

"Machines and programs which attempt to answer English question have existed for only about five years… Attempts to build machine to test logical consistency date back to at least Roman Lull in the thirteenth century… Only in recent years have attempts been made to translate mechanically from English into logical formalisms…"

Classical Natural Language Understanding

- Conventional **pipeline model**: focus on capturing deep inference and entailment.

List samples that contain every major element

\[ \text{sem} = \{S10019, S10059, \ldots\} \]

Lunar QA system of Woods (1973)
Classical Natural Language Understanding

List samples that contain every major element

1. Semantic Parsing
   \[(\text{FOR EVERY } X / \text{MAJORELT}: T; \text{FOR EVERY } Y / \text{SAMPLE}: (\text{CONTAINS } Y \ X); \text{PRINTOUT } Y))\]

2. Knowledge Representation

3. Reasoning

\[[\text{sem}] = \{S10019, S10059, \ldots\}\]

<table>
<thead>
<tr>
<th>Sub-problem</th>
<th>Problem Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Semantic Parsing</td>
<td>Translating input to sem, input $\rightarrow$ sem</td>
</tr>
<tr>
<td>2. Knowledge Representation</td>
<td>Defining a sufficiently expressive sem language.</td>
</tr>
<tr>
<td>3. Reasoning/Execution</td>
<td>Going from sem to denotations in the real-world.</td>
</tr>
</tbody>
</table>
NLU model is a kind of compiler, involves a **transduction** from NL to a formal (usually logical) language.
Data-driven Semantic Parsing and NLU

Diagram:

- **input**: List samples that contain every major element
- **sem**: (FOR EVERY X /
  MAJORELT : T;
  (FOR EVERY Y /
  SAMPLE : (CONTAINS Y X);
  (PRINTOUT Y)))
- **database**: 
  \[
  \text{[}sem\text{]} = \{S10019, S10059, \ldots\}
  \]

**Data-driven NLU**: Asks an empirical question: Can we learn NLU models from examples? Building a NL compiler by hand is hard....
Data-driven Semantic Parsing and NLU

**Semantic Parser Induction:** Learn semantic parser (weighted transduction) from parallel text/meaning data, constrained SMT task.

**Resource Problem:** Where does the parallel data come from, what do we learn from? Does not occur 'in the wild'
Talk Overview

<table>
<thead>
<tr>
<th>Sub-problem</th>
<th>Problem Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Semantic Parsing</td>
<td>Translating input to sem, input → sem</td>
</tr>
<tr>
<td>2. Knowledge Representation</td>
<td>Defining a sufficiently expressive sem language.</td>
</tr>
<tr>
<td>3. Reasoning/Execution</td>
<td>Going from sem to denotations in the real-world.</td>
</tr>
</tbody>
</table>

- **Resource Problem:** Using technical documentation as a parallel corpus (problems 1,4), from Richardson and Kuhn (2017b,a)
- **Polyglot Modeling:** Building semantic parsers from multiple datasets, ”polyglot decoding” (problems 1,4), from Richardson et al. (2018)
- **Learning from Entailment:** Integrating entailment into a semantic parsing pipeline (problems 1,2,3), from Richardson and Kuhn (2016).
<Resource Problem>
Semantic Parsing and Parallel Data

What state has the largest population?

\[ z = \text{argmax} (\lambda x. \text{state } x) \lambda x. \text{population } x) \]

- **Learning from LFs**: Pairs of text \( x \) and logical forms \( z \), \( D = \{(x, z)_i\}_i^n \), learn \( \text{sem} : x \rightarrow z \)
- **Modularity**: Study the translation independent of other semantic issues.
What state has the largest population?

\[
\text{z} = \text{argmax} (\lambda x. (\text{state } x) \lambda x. (\text{population } x))
\]

- **Learning from LF**: Pairs of text \(x\) and logical forms \(z\), \(D = \{(x, z)_i\}_i^n\), learn \(\text{sem}: x \rightarrow z\)

- **Modularity**: Study the translation independent of other semantic issues.

- **Resource issue**: Finding *parallel data*, current lack of resources.
Source Code and API Documentation

* Returns the greater of two long values
* @param a an argument
* @param b another argument
* @return the larger of a and b
* @see java.lang.Long#MAX_VALUE
*/

```
public static Long max(long a, long b)
```

➤ **Source Code Documentation:** High-level descriptions of internal software functionality paired with code.
Source Code and API Documentation

* Returns the greater of two long values
* @param a an argument
* @param b another argument
* @return the larger of a and b
* @see java.lang.Long#MAX_VALUE
*
public static Long max(long a, long b)

- **Source Code Documentation:** High-level descriptions of internal software functionality paired with code.
- **Idea:** Treat as a parallel corpus (Allamanis et al., 2015; Gu et al., 2016; Iyer et al., 2016), or synthetic semantic parsing dataset.
Source Code as a Parallel Corpus

► Observation 1: Tight coupling between high-level text and code.

* Returns the greater of two long values
  *
  * @param a an argument
  * @param b another argument
  * @return the larger of a and b
  * @see java.lang.Long#MAX_VALUE
  */
  public static Long max(long a, long b)

(ns ... clojure.core)
(defn random-sample
  "Returns items from coll with random probability of prob (0.0 - 1.0)"
  ([prob] ...)
  ([prob coll] ...))
Source Code as a Parallel Corpus

- **Observation 1:** Tight coupling between high-level text and code.

```
* Returns the greater of two long values
*
* @param a an argument
* @param b another argument
* @return the larger of a and b
* @see java.lang.Long#MAX_VALUE
*/

class MathUtil {
  public static Long max(long a, long b)
}
```

```
(ns ... clojure.core)

defn random-sample
  "Returns items from coll with random probability of prob (0.0 - 1.0)"
  ([prob] ...)
  ([prob coll] ...))
```

- **Function signatures:** Header-like representations, containing function name, (optionally typed) arguments, (optional) return value, namespace.
Observation 1: Tight coupling between high-level text and code.

* Returns the greater of two long values
* @param a an argument
* @param b another argument
* @return the larger of a and b
* @see java.lang.Long#MAX

```
public static Long max(long a, long b)
```

```
(ns ... clojure.core)

(defn random-sample
  "Returns items from coll with random probability of prob (0.0 – 1.0)"
  ([prob] ...) ([prob coll] ...))
```

Function signatures: Header-like representations, containing function name, (optionally typed) arguments, (optional) return value, namespace.

* Returns the greater of two long values
* math.util Long max(long a, long b)
* Returns items from coll with random...
* (core.random-sample prob coll)
Observation 2: There are many languages, hence many datasets.

```java
public static Long max(long a, long b)
```

```clojure
(ns ... clojure.core)

(defn random-sample
  "Returns items from coll with random probability of prob (0.0 - 1.0)"
  ([prob] ...) ([prob coll] ...))
```

```python
# zipfile.py
"""Read and write ZIP files"""

class ZipFile(object):

  """Class to open ... zip files."""
  def write(filename,arcname,....):
    """Put the bytes from filename into the archive under the name."""
```

```plaintext
--| Mostly functions for reading and showing RealFloat like values
module Numeric

-- | Show non-negative Integral numbers in base 10.
showInt :: Integral a => a -> ShowS
```
Observation 3: Many NLs, hence many multilingual datasets.

namespace ArrayIterator;
/*
 * Appends values as the last element
 * @param value The value to append
 * @see ArrayIterator::next()
 */
public void append(mixed $value)

namespace ArrayIterator;
/*
 * Ajoute une valeur comme dernier élément
 * @param value La valeur à ajouter
 * @see ArrayIterator::next()
 */
public void append(mixed $value)

namespace ArrayIterator;
/*
 * Dobavlyaet znachenie value, kak posledni element massiva.
 * @param value znachenie, kotoroe nuzhno dobavit'.
 * @see ArrayIterator::next()
 */
public void append(mixed $value)

namespace ArrayIterator;
/*
 * Anade el valor cómo el último elemento.
 * @param value El valor a anadir.
 * @see ArrayIterator::next()
 */
public void append(mixed $value)
Beyond raw pairs: Background Information

**Observation 4:** Code collections contain rich amount of background info.

```plaintext
NAME : dappprof  
profile user and lib function usage.

SYNOPSIS  
dappprof [-ac..] .. -p PID | command

DESCRIPTION  
--a  print all data  
--p PID  examine the PID

EXAMPLES  
Run and examine the ‘‘df -h’’ command  
dappprof command=‘‘df -h’’

Print elapsed time for PID 1871  
dappprof -p PID=1871

SEE ALSO  
dapptrace(1M), dtrace(1M), ...
```

**Descriptions:** textual descriptions of parameters, return values, ...

**Cluster information:** pointers to related functions/utilities, ...

**Syntactic information:** function/code syntax
<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Pairs</th>
<th>#DescrSymbols</th>
<th>#Words</th>
<th>Vocab.</th>
<th>Example Pairs ((x, z)), <strong>Goal:</strong> learn a function (x \rightarrow z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>7,183</td>
<td>4,804</td>
<td>4,072</td>
<td>82,696</td>
<td>3,721</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x:) Compares this Calendar to the specified Object. (z:) boolean util.Calendar.equals(Object obj)</td>
</tr>
<tr>
<td>Ruby</td>
<td>6,885</td>
<td>1,849</td>
<td>3,803</td>
<td>67,274</td>
<td>5,131</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x:) Computes the arc tangent given (y) and (x). (z:) Math.atan2(y,x) (\rightarrow) Float</td>
</tr>
<tr>
<td>PHP(_{en})</td>
<td>6,611</td>
<td>13,943</td>
<td>8,308</td>
<td>68,921</td>
<td>4,874</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x:) Delete an entry in the archive using its name. (z:) bool ZipArchive::deleteName(string $name)</td>
</tr>
<tr>
<td>Python</td>
<td>3,085</td>
<td>429</td>
<td>3,991</td>
<td>27,012</td>
<td>2,768</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x:) Remove the specific filter from this handler. (z:) logging.Filterer.removeFilter(filter)</td>
</tr>
<tr>
<td>Elisp</td>
<td>2,089</td>
<td>1,365</td>
<td>1,883</td>
<td>30,248</td>
<td>2,644</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x:) Returns the total height of the window. (z:) (window-total-height window round)</td>
</tr>
<tr>
<td>Haskell</td>
<td>1,633</td>
<td>255</td>
<td>1,604</td>
<td>19,242</td>
<td>2,192</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x:) Extract the second component of a pair. (z:) Data.Tuple.snd :: (a, b) (\rightarrow) b</td>
</tr>
<tr>
<td>Clojure</td>
<td>1,739</td>
<td>–</td>
<td>2,569</td>
<td>17,568</td>
<td>2,233</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x:) Returns a lazy seq of every nth item in coll. (z:) (core.take-nth n coll)</td>
</tr>
<tr>
<td>C</td>
<td>1,436</td>
<td>1,478</td>
<td>1,452</td>
<td>12,811</td>
<td>1,835</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x:) Returns current file position of the stream. (z:) long int ftell(FILE *stream)</td>
</tr>
<tr>
<td>Scheme</td>
<td>1,301</td>
<td>376</td>
<td>1,343</td>
<td>15,574</td>
<td>1,756</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x:) Returns a new port and the given state. (z:) (make-port port-type state)</td>
</tr>
<tr>
<td>Geoquery</td>
<td>880</td>
<td>–</td>
<td>167</td>
<td>6,663</td>
<td>279</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x:) What is the tallest mountain in America? (z:) (highest(mountain(loc_2(countryid usa))))</td>
</tr>
</tbody>
</table>

- Standard library documentation for 9+ programming languages, 7 natural languages, from Richardson and Kuhn (2017b).
Resource 2: Open source Python projects

<table>
<thead>
<tr>
<th>Project</th>
<th># Pairs</th>
<th># Symbols</th>
<th># Words</th>
<th>Vocab.</th>
</tr>
</thead>
<tbody>
<tr>
<td>scapy</td>
<td>757</td>
<td>1,029</td>
<td>7,839</td>
<td>1,576</td>
</tr>
<tr>
<td>zipline</td>
<td>753</td>
<td>1,122</td>
<td>8,184</td>
<td>1,517</td>
</tr>
<tr>
<td>biopython</td>
<td>2,496</td>
<td>2,224</td>
<td>20,532</td>
<td>2,586</td>
</tr>
<tr>
<td>renpy</td>
<td>912</td>
<td>889</td>
<td>10,183</td>
<td>1,540</td>
</tr>
<tr>
<td>pyglet</td>
<td>1,400</td>
<td>1,354</td>
<td>12,218</td>
<td>2,181</td>
</tr>
<tr>
<td>kivy</td>
<td>820</td>
<td>861</td>
<td>7,621</td>
<td>1,456</td>
</tr>
<tr>
<td>pip</td>
<td>1,292</td>
<td>1,359</td>
<td>13,011</td>
<td>2,201</td>
</tr>
<tr>
<td>twisted</td>
<td>5,137</td>
<td>3,129</td>
<td>49,457</td>
<td>4,830</td>
</tr>
<tr>
<td>vispy</td>
<td>1,094</td>
<td>1,026</td>
<td>9,744</td>
<td>1,740</td>
</tr>
<tr>
<td>orange</td>
<td>1,392</td>
<td>1,125</td>
<td>11,596</td>
<td>1,761</td>
</tr>
<tr>
<td>tensorflow</td>
<td>5,724</td>
<td>4,321</td>
<td>45,006</td>
<td>4,672</td>
</tr>
<tr>
<td>pandas</td>
<td>1,969</td>
<td>1,517</td>
<td>17,816</td>
<td>2,371</td>
</tr>
<tr>
<td>sqlalchmy</td>
<td>1,737</td>
<td>1,374</td>
<td>15,606</td>
<td>2,039</td>
</tr>
<tr>
<td>pyspark</td>
<td>1,851</td>
<td>1,276</td>
<td>18,775</td>
<td>2,200</td>
</tr>
<tr>
<td>nupic</td>
<td>1,663</td>
<td>1,533</td>
<td>16,750</td>
<td>2,135</td>
</tr>
<tr>
<td>astropy</td>
<td>2,325</td>
<td>2,054</td>
<td>24,567</td>
<td>3,007</td>
</tr>
<tr>
<td>sympy</td>
<td>5,523</td>
<td>3,201</td>
<td>52,236</td>
<td>4,777</td>
</tr>
<tr>
<td>ipython</td>
<td>1,034</td>
<td>1,115</td>
<td>9,114</td>
<td>1,771</td>
</tr>
<tr>
<td>orator</td>
<td>817</td>
<td>499</td>
<td>6,511</td>
<td>670</td>
</tr>
<tr>
<td>obspy</td>
<td>1,577</td>
<td>1,861</td>
<td>14,847</td>
<td>2,169</td>
</tr>
<tr>
<td>rdkit</td>
<td>1,006</td>
<td>1,380</td>
<td>9,758</td>
<td>1,739</td>
</tr>
<tr>
<td>django</td>
<td>2,790</td>
<td>2,026</td>
<td>31,531</td>
<td>3,484</td>
</tr>
<tr>
<td>ansible</td>
<td>2,124</td>
<td>1,884</td>
<td>20,677</td>
<td>2,593</td>
</tr>
<tr>
<td>statsmodels</td>
<td>2,357</td>
<td>2,352</td>
<td>21,716</td>
<td>2,733</td>
</tr>
<tr>
<td>theano</td>
<td>1,223</td>
<td>1,364</td>
<td>12,018</td>
<td>2,152</td>
</tr>
<tr>
<td>nltk</td>
<td>2,383</td>
<td>2,324</td>
<td>25,823</td>
<td>3,151</td>
</tr>
<tr>
<td>sklearn</td>
<td>1,532</td>
<td>1,519</td>
<td>13,897</td>
<td>2,115</td>
</tr>
</tbody>
</table>

- 27 Python projects from Github, from Richardson and Kuhn (2017a), similar to Barone and Sennrich (2017)
Summary of Current Resources

- **API Datasets**: Stdlib collection and Py27, consists of 45 APIs across 11 programming languages, 8 natural languages.

- **Other Resources**: Function Assistant, tool for extracting parallel datasets from Python projects

- **forthcoming**: around 460 Python/Java API datasets for data-to-text generation (Richardson et al., 2017).

https://github.com/yakazimir/Code-Datasets
Text to Signature Translation: How hard is it?

- **Task**: For each API dataset of text/signature pairs (each within a finite signature space), learn a sp: text → signature.

- **Question** Can background info. from API help?
Text to Signature Translation: How hard is it?

▶ **Task**: For each API dataset of text/signature pairs (each within a finite signature space), learn a \text{sp: text} \rightarrow \text{signature}.

▶ **Question** Can background info. from API help?

▶ **SMT Baseline**: (Deng and Chrupała, 2014), sequence prediction model.

\[ k\text{-best signature translation list} \]

\begin{itemize}
\item \text{SMT Model}
\item Gets the total cache size
\item Task specific decoder
\end{itemize}

\begin{itemize}
\item Discriminative Model
\item \text{reranked k-best}
\end{itemize}

\begin{itemize}
\item \text{Evaluation}
\item Accuracy @i?
\end{itemize}

\begin{itemize}
\item \text{Evaluation}
\item \text{gets the total cache size}
\item \text{task specific decoder}
\end{itemize}

\begin{itemize}
\item \text{SMT Model}
\item \text{Discriminative Model}
\item \text{reranked k-best}
\end{itemize}
Background API Information

**Reranker Model:** See-also annotations, abstract syntax info., parameter descriptions,...

\[
\phi(x,z) = \begin{aligned}
\text{Model score: } & \text{is it in top 5..10?} \\
\text{Alignments: } & (\text{hyperbolic, cosh}), (\text{cosine, cosh}), \ldots \\
\text{Phrases: } & (\text{hyperbolic cosine, cosh}), (\text{of arg, float $\arg$}), \ldots \\
\text{See also classes: } & (\text{hyperbolic, } \{\text{cos, acosh, sinh} \ldots\}), \ldots \\
\text{In descriptions: } & (\text{arg, float $\arg$}) \\
\text{Matches/Tree position: } & \ldots \\
\end{aligned}
\]
Text to Signature Translation: How hard is it?

\( k\)-best signature translation list

```
- string APCIterator::key(void)
- int APCIterator::getTotalHits(void)
- int APCIterator::getSize(void)
int APCIterator::getTotalSize(void)
- int Memcached::append(string $key)
...
```

SMT Model

```
 Gets the total cache size
```

Task specific decoder

```
Accuracy @i?
```

Evaluation

```
Dataset (Avg.)
Standard Library Docs. 12.7 32.0 19.2 28.9 67.7 41.9 31.1 71.1 44.5
Py27 22.9 50.6 32.4 29.3 67.4 42.5 32.4 73.5 46.5
```

SMT

```
- int APCIterator::getTotalSize(void)
- int APCIterator::getSize(void)
- string APCIterator::key(void)
- int Memcached::append(string $key)
- int APCIterator::getTotalHits(void)
```

Discriminative Model

```
reranked k-best
```

Evaluation

```
Accuracy @1
```

```
accuracy @1  accuracy @10  MRR
```
What do these results mean?

Dataset (Avg.)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Term Matching</th>
<th>SMT</th>
<th>SMT + Reanker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Library Docs.</td>
<td>12.7 32.0 19.2</td>
<td>28.9 67.7 41.9</td>
<td><strong>31.1 71.1 44.5</strong></td>
</tr>
<tr>
<td>Py27</td>
<td>22.9 50.6 32.4</td>
<td>29.3 67.4 42.5</td>
<td><strong>32.4 73.5 46.5</strong></td>
</tr>
</tbody>
</table>

accuracy @1   accuracy @10   MRR
Semantic Parsing in Tech Docs: General Findings

- **How hard is it?**: Certainly not trivial, simple SMT models do alright, but lots of room for improvement

- **New Challenges**: Highly sparse vocabulary, very hard to apply existing semantic parsing and MT methods (more about this next).
The Semantics of Function Signatures

Returns the greater of two long values

<table>
<thead>
<tr>
<th>Signature (informal)</th>
<th>lang Math long max(long a, long b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized</td>
<td>java lang Math::max(long:a, long:b) -&gt; long</td>
</tr>
</tbody>
</table>

Expansion to Logic

\[
\lambda x_1 \lambda x_2 \exists v \exists f \exists n \exists c \; eq(v, max(x_1, x_2)) \land \text{fun}(f, \text{max}) \land \text{type}(v, \text{long}) \\
\land \text{lang}(f, \text{java}) \\
\land \text{var}(x_1, a) \land \text{param}(x_1, f, 1) \land \text{type}(x_1, \text{long}) \\
\land \text{var}(x_2, b) \land \text{param}(x_2, f, 2) \land \text{type}(x_2, \text{long}) \\
\land \text{namespace}(n, \text{lang}) \land \text{in_namespace}(f, n) \\
\land \text{class}(c, \text{Math}) \land \text{in_class}(f, c)
\]

▶ **Disclaimer**: not real logical forms, but we can formalize the function signature languages and define a translation to logic (Richardson, 2018).

▶ **Reasoning**: A lot of declarative knowledge can be extracted from libraries directly, and via natural language.
</Resource Issue>
Traditional approaches to semantic parsing train individual models for each available parallel dataset.

Resource Problem: Datasets tend to be small, hard and unlikely to get certain types of parallel data, e.g., (de,Haskell).
Code Domain: Projects often Lack Documentation

- Ideally, we want each dataset to have tens of thousands of documented functions.
- Most projects have 500 or less documented functions.
Polyglot Models: Training on Multiple Datasets

- **Idea**: concatenate all datasets into one, build a single-model with shared parameters, capture redundancy (Herzig and Berant, 2017).

- **Polyglot Translator**: translates from any input language to any output (programming) language.
Polyglot Models: Training on Multiple Datasets

- Idea: concatenate all datasets into one, build a single-model with shared parameters, capture redundancy (Herzig and Berant, 2017).

- Polyglot Translator: translates from any input language to any output (programming) language.

1. **Multiple Datasets**: Does this help learn better translators?

2. **Zero-Short Translation** (Johnson et al., 2016): Can we translate between different APIs and unobserved language pairs?
Polyglot Models: Training on Multiple Datasets

- **Challenge:** Building a polyglot decoder, or translation mechanism that facilitates crossing between (potentially unobserved) language pairs.
Polyglot Models: Training on Multiple Datasets

- **Challenge**: Building a polyglot decoder, or translation mechanism that facilitates crossing between (potentially unobserved) language pairs.

  - **Constraint 1**: Ensure well-formed code output (not guaranteed in ordinary MT, cf. Cheng et al. (2017); Krishnamurthy et al. (2017))

  - **Constraint 2**: Must be able to translate to target APIs/programming languages on demand.
**Graph Based Approach**

▶ **Idea**: Exploit finite-ness of target translation space, represent full search space as directed acyclic graph (DAG).
**Graph Based Approach**

▶ **Idea**: Exploit finite-ness of target translation space, represent full search space as directed acyclic graph (DAG).

▶ **Trick**: Prepend to each signature an artificial token that identifiers the API project or programming language (Johnson et al., 2016).
Graph Based Approach

- **Idea**: Exploit finite-ness of target translation space, represent full search space as directed acyclic graph (DAG).

- **Trick**: Prepend to each signature an artificial token that identifies the API project or programming language (Johnson et al., 2016).

- **Decoding**: Reduces to finding a path given an input $x$:

  $$ x : \text{The ceiling of a number} $$

  Can be solved using variant of single-source shortest path (SSSP) problem (Cormen et al., 2009), extendible to $k$-SSSP paths.
Graph Decoder: Shortest Path Decoding

Standard SSSP: assumes a DAG $G = (V, E)$, a weight function: $w : E \rightarrow \mathbb{R}$, (initialized) vector $d \in \infty^{|V|}$, unique source node $b$

0: $d[b] \leftarrow 0.0$
1: for vertex $u \in V$ in top sorted order
2: do $d(v) = \min_{(u,v,z) \in E} \{d(u) + w(u,v,z)\}$
3: return $\min_{v \in V} \{d(v)\}$
Graph Decoder: Shortest Path Decoding

- **Standard SSSP:** assumes a DAG $G = (V, E)$, a weight function: $w : E \rightarrow \mathbb{R}$, (initialized) vector $d \in \infty^{|V|}$, unique source node $b$

  0: $d[b] \leftarrow 0.0$
  1: for vertex $u \in V$ in top sorted order
  2: do $d(v) = \min_{(u,v,z) \in E} \{d(u) + w(u,v,z)\}$
  3: return $\min_{v \in V} \{d(v)\}$

- **Variant:** replace $w(..)$ with translation model, dynamically generates weights correspond. to translation scores for $x$ and labels in SSSP search.
Neural Sequence to Sequence Models

▶ **Encoder Model**: neural sequence model, builds a *distributed* representation of the source sentence and its words $x = (h_1, h_2, ..., h_{|x|})$.

▶ **Decoder Model**: RNN language model additionally conditioned on input $x$/Encoder states.

$$p(z \mid x) = \prod_{i} p_{\Theta}(z_i \mid z_{<i}, x)$$
Neural Sequence to Sequence Models

- **Encoder Model**: neural sequence model, builds a *distributed* representation of the source sentence and its words \( x = (h_1, h_2, .., h_{|x|}) \):

- **Decoder Model**: RNN language model additionally conditioned on input \( x \)/Encoder states.

\[
p(z \mid x) = \prod_{i} p_{\Theta}(z_i \mid z_{<i}, x)
\]

- **Modification** (at decode/test time): Constrain search (each new \( z_i \)) to allowable transitions and paths in the graph.
Graph Decoder: Shortest Path Decoding

- **Standard SSSP:** assumes a DAG $G = (V, E)$, a weight function: $w : E \rightarrow \mathbb{R}$, (initialized) vector $d \in \infty^{|V|}$, unique source node $b$

  0: $d[b] \leftarrow 0.0$
  1: for each vertex $u \in V$ in top sorted order
  2: do $d(v) = \min_{(u, v, z) \in E} \{d(u) + w(u, v, z)\}$
  3: return $\min_{v \in V} \{d(v)\}$

- **Translation models:** Any model can be used, we experiment with lexical translation models (see paper) and attentive encoder-decoder models.
Graph Decoder: Shortest Path Decoding

- **Standard SSSP:** assumes a DAG $G = (V, E)$, a weight function: $w : E \rightarrow \mathbb{R}$, (initialized) vector $d \in \infty^{|V|}$, unique source node $b$

0: $d[b] \leftarrow 0.0$
1: for each vertex $u \in V$ in top sorted order
2: do $d(v) = \min_{(u,v,z) \in E} \left\{ d(u) + w(u,v,z) \right\}$
3: return $\min_{v \in V} \left\{ d(v) \right\}$

- **Translation models:** Any model can be used, we experiment with lexical translation models (see paper) and attentive encoder-decoder models.

- **Neural Variant:** assumes input $x$, $G$, neural decoder parameters $\Theta$ (trained normally), $d$, and $s$ (state map):

0: $d[b] \leftarrow 0.0$
1: for each vertex $u \in V$ in top sorted order
2: do $d(v) = \min_{(u,v,z) \in E} \left\{ -\log p_{\Theta}(z | z_{<i}, x) + d(u) \right\}$
3: $s[v] \leftarrow$ RNN state for min edge
4: return $\min_{v \in V} \left\{ d(v) \right\}$
Polyglot vs. Monolingual Decoding

- The difference is the type of input data, and starting point (i.e., source node) in the graph search.
- **Any Language Decoding**: Letting the decoder decide.
Polyglot vs. Monolingual Decoding

- The difference is the type of input data, and starting point (i.e., source node) in the graph search.
- **Any Language Decoding**: Letting the decoder decide.

<table>
<thead>
<tr>
<th>1. Source API (stdlib): (es, PHP)</th>
<th><strong>Input</strong>: Devuelve el mensaje asociado al objeto lanzado.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td><strong>Language</strong>: PHP, Java, Clojure</td>
</tr>
<tr>
<td><strong>Translation</strong>:</td>
<td>public string Throwable::getMessage ( void )</td>
</tr>
<tr>
<td></td>
<td>public String lang.getMessage( void )</td>
</tr>
<tr>
<td></td>
<td>(tools.logging.fatal throwable message &amp; more)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Source API (stdlib): (ru, PHP)</th>
<th><strong>Input</strong>: конвертирует строку из формата UTF-32 в формат UTF-16.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td><strong>Language</strong>: PHP, Ruby, Haskell</td>
</tr>
<tr>
<td><strong>Translation</strong>:</td>
<td>string PDF.utf32.to_utf16 ( ... )</td>
</tr>
<tr>
<td></td>
<td>String#toutf16 =&gt; string</td>
</tr>
<tr>
<td></td>
<td>Encoding.encodeUtf16LE :: Text -&gt; ByteString</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Source API (py): (en, stats)</th>
<th><strong>Input</strong>: Compute the Moore-Penrose pseudo-inverse of a matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td><strong>Project</strong>: sympy, sklearn, stats</td>
</tr>
<tr>
<td><strong>Translation</strong>:</td>
<td>matrices.matrix.base.pinv_solve( B, ... )</td>
</tr>
<tr>
<td></td>
<td>utils.pinvh( a, cond=None,rcond=None,... )</td>
</tr>
<tr>
<td></td>
<td>tools.pinv2( a,cond=None,rcond=None )</td>
</tr>
</tbody>
</table>
Polyglot vs. Monolingual Decoding

- The difference is the type of input data, and starting point (i.e., source node) in the graph search.

- **Any Language Decoding**: Letting the decoder decide.

<table>
<thead>
<tr>
<th>Source API (stdlib): (es, PHP)</th>
<th>Input: Devuelve el mensaje asociado al objeto lanzado.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td>Language: <strong>PHP</strong></td>
</tr>
<tr>
<td></td>
<td>Language: <strong>Java</strong></td>
</tr>
<tr>
<td></td>
<td>Language: <strong>Clojure</strong></td>
</tr>
<tr>
<td>2. Source API (stdlib): (<strong>ru</strong>, PHP)</td>
<td><strong>Input</strong>: конвертирует строку из формата UTF-32 в формат UTF-16.</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>Language: <strong>PHP</strong></td>
</tr>
<tr>
<td></td>
<td>Language: <strong>Ruby</strong></td>
</tr>
<tr>
<td></td>
<td>Language: <strong>Haskell</strong></td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>Project: <strong>sympy</strong></td>
</tr>
<tr>
<td></td>
<td>Project: <strong>sklearn</strong></td>
</tr>
<tr>
<td></td>
<td>Project: <strong>stats</strong></td>
</tr>
</tbody>
</table>

- Can be used for extracting declarative knowledge about function equivalences (e.g., for the logical approach introduced).
Our Focus: Does training on multiple datasets (i.e., polyglot models) improve monolingual decoding?

- Monolingual models: current best models, primarily SMT based.
Polyglot vs. Monolingual Decoding: Tech Doc Task

- **Our Focus**: Does training on multiple datasets (i.e., *polyglot models*) improve monolingual decoding?
  - **Monolingual models**: current best models, primarily SMT based.

<table>
<thead>
<tr>
<th></th>
<th>Method</th>
<th>Acc@1</th>
<th>Acc@10</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>stdlib</em></td>
<td>Best Monolingual Model</td>
<td>29.9</td>
<td>69.2</td>
<td>43.1</td>
</tr>
<tr>
<td>mono.</td>
<td>Lexical SMT SSSP</td>
<td>33.2</td>
<td>70.7</td>
<td>45.9</td>
</tr>
<tr>
<td>poly.</td>
<td>Best Seq2Seq SSSP</td>
<td>13.9</td>
<td>36.5</td>
<td>21.5</td>
</tr>
<tr>
<td><em>py27</em></td>
<td>Best Monolingual Model</td>
<td>32.4</td>
<td>73.5</td>
<td>46.5</td>
</tr>
<tr>
<td>mono.</td>
<td>Lexical SMT SSSP</td>
<td>41.3</td>
<td>77.7</td>
<td>54.1</td>
</tr>
<tr>
<td>poly.</td>
<td>Best Seq2Seq SSSP</td>
<td>9.0</td>
<td>26.9</td>
<td>15.1</td>
</tr>
</tbody>
</table>
Polyglot vs. Monolingual Decoding: Tech Doc Task

- **Our Focus**: Does training on multiple datasets (i.e., *polyglot models*) improve monolingual decoding?

- **Monolingual models**: current best models, primarily SMT based.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc@1</th>
<th>Acc@10</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdlib</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mono.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>poly.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexical SMT SSSP</td>
<td>33.2</td>
<td>70.7</td>
<td>45.9</td>
</tr>
<tr>
<td>Best Seq2Seq SSSP</td>
<td>13.9</td>
<td>36.5</td>
<td>21.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc@1</th>
<th>Acc@10</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>py27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mono.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>poly.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexical SMT SSSP</td>
<td>41.3</td>
<td>77.7</td>
<td>54.1</td>
</tr>
<tr>
<td>Best Seq2Seq SSSP</td>
<td>9.0</td>
<td>26.9</td>
<td>15.1</td>
</tr>
</tbody>
</table>

- **Findings**: Polyglot models can improve performance using SMT models, do not work for Seq2Seq models.
Our Focus: Does training on multiple datasets (i.e., polyglot models) improve monolingual decoding?

Monolingual models: current best models, primarily SMT based.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc@1</th>
<th>Acc@10</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Monolingual Model</td>
<td>29.9</td>
<td>69.2</td>
<td>43.1</td>
</tr>
<tr>
<td>Lexical SMT SSSP</td>
<td>33.2</td>
<td>70.7</td>
<td>45.9</td>
</tr>
<tr>
<td>Best Seq2Seq SSSP</td>
<td>13.9</td>
<td>36.5</td>
<td>21.5</td>
</tr>
</tbody>
</table>

Findings: Polyglot models can improve performance using SMT models, do not work for Seq2Seq models.

Polyglot Modeling on Benchmark SP Tasks

▶ **Our Focus:** Does this help on benchmark semantic parsing tasks?
Polyglot Modeling on Benchmark SP Tasks

 conforme a Our Focus: Does this help on benchmark semantic parsing tasks?

<table>
<thead>
<tr>
<th>Multilingual Geoquery</th>
<th>Method</th>
<th>Acc@1 (averaged)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mono.</td>
<td>UBL Kwiatkowski et al. (2010)</td>
<td>74.2</td>
</tr>
<tr>
<td></td>
<td>TreeTrans Jones et al. (2012)</td>
<td>76.8</td>
</tr>
<tr>
<td></td>
<td>Lexical SMT SSSP</td>
<td>68.6</td>
</tr>
<tr>
<td>poly.</td>
<td>Best Seq2Seq SSSP</td>
<td>78.0</td>
</tr>
</tbody>
</table>

▶ Multilingual Geoquery: monolingual/polyglot models on Geoquery in en, de, gr, th, polyglot setting improves accuracy, neural Seq2Seq models perform best (consistent with recent findings, (Dong and Lapata, 2016)).
Our Focus: Does this help on benchmark semantic parsing tasks?

<table>
<thead>
<tr>
<th>Multilingual Geoquery</th>
<th>Method</th>
<th>Acc@1</th>
<th>(averaged)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mono.</td>
<td>UBL Kwiatkowski et al. (2010)</td>
<td>74.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TreeTrans Jones et al. (2012)</td>
<td>76.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lexical SMT SSSP</td>
<td>68.6</td>
<td></td>
</tr>
<tr>
<td>poly.</td>
<td>Best Seq2Seq SSSP</td>
<td>78.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lexical SMT SSSP</td>
<td>67.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Best Seq2Seq SSSP</td>
<td>79.6</td>
<td></td>
</tr>
</tbody>
</table>

Multilingual Geoquery: monolingual/polyglot models on Geoquery in en, de, gr, th, polyglot setting improves accuracy, neural Seq2Seq models perform best (consistent with recent findings, (Dong and Lapata, 2016)).

Recall that these same Seq2Seq models do not work in the technical documentation tasks.
Benchmark SP Tasks: Mixed Language Decoding

Introduced a new *mixed language* GeoQuery test set, each sentence contains NPs from two or more languages.

```
Mixed Lang. Input: Wie hoch liegt der höchstgelegene punkt in Αλάμπαμα?
LF: answer(elevation_1(highest(place(loc_2(stateid('alabama'))))))
```

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc@1 (averaged)</th>
<th>Acc@10 (averaged)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed Best Monolingual Seq2Seq</td>
<td>4.2</td>
<td>18.2</td>
</tr>
<tr>
<td>Polyglot Seq2Seq</td>
<td><strong>75.2</strong></td>
<td>90.0</td>
</tr>
</tbody>
</table>
Learning from multiple datasets: Summary

► **Take Away:** polyglot modeling can be a useful technique for improving semantic parsing and transfer learning.

► **New Ideas:** Translating between datasets and languages, mixed language parsing, *hardness of classical SMT decoding*..?

► **Technical Docs:** has features of a low-resource translation task, difficult especially for neural modeling.
<Learning from Entailment>

(high level overview)
Semantic Parsing and Entailment

- **Entailment**: One of the *basic aims* of semantics. (Montague (1970))
- Representations should be *grounded* in judgements about entailment.

\[
\text{database} \ni \{\text{S10019, S10059, } \ldots \} \supseteq \{\text{S10019}\}
\]
Unit Testing our Semantic Parsers

- **Minimal requirement:** A semantic parser should be able to recognize certain types of entailments.

- **RTE:** Would a human reading $t$ infer $h$? Dagan et al. (2005)

<table>
<thead>
<tr>
<th>sentence</th>
<th>Logical Form (from corpus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$  Pink3 passes to Pink7</td>
<td>pass(pink3, pink7)</td>
</tr>
<tr>
<td>$h$  Pink3 quickly passes to Pink7</td>
<td>pass(pink3, pink7)</td>
</tr>
</tbody>
</table>

| Inference                        | $t \rightarrow h$ Uncertain (human)    |
Unit Testing our Semantic Parsers

- **Minimal requirement**: A semantic parser should be able to recognize certain types of entailments.

- **RTE**: Would a human reading \( t \) infer \( h \)? Dagan et al. (2005)

<table>
<thead>
<tr>
<th>sentence</th>
<th>Logical Form (from corpus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t ) Pink3 passes to Pink7</td>
<td>pass(pink3,pink7)</td>
</tr>
<tr>
<td>( h ) Pink3 quickly passes to Pink7</td>
<td>pass(pink3,pink7)</td>
</tr>
</tbody>
</table>

Inference

- \( t \to h \) Uncertain (human)
- \( t \to h \) **Entail** (LF only)

Sportscaster corpus (Chen and Mooney (2008))
Unit Testing our Semantic Parsers

- **Minimal requirement:** A semantic parser should be able to recognize certain types of entailments.

- **RTE:** Would a human reading $t$ infer $h$? Dagan et al. (2005)

<table>
<thead>
<tr>
<th>sentence</th>
<th>Logical Form (from corpus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$ <em>Pink3 passes to Pink7</em></td>
<td>pass(pink3, pink7)</td>
</tr>
<tr>
<td>$h$ <em>Pink3 kicks the ball</em></td>
<td>kick(Pink3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inference</th>
<th>$t \rightarrow h$ Entail (human)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t \rightarrow h$ Contradict (LF only)</td>
<td></td>
</tr>
</tbody>
</table>
Unit Testing our Semantic Parsers

- **Minimal requirement**: A semantic parser should be able to recognize certain types of entailments.
- **RTE**: Would a human reading $t$ infer $h$? Dagan et al. (2005)

<table>
<thead>
<tr>
<th>sentence</th>
<th>Logical Form (from corpus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$ Pink3 passes to Pink7</td>
<td>pass(pink3,pink7)</td>
</tr>
<tr>
<td>$h$ Pink3 kicks the ball</td>
<td>kick(Pink3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inference</th>
<th>$t \rightarrow h$ Entail (human)</th>
<th>$t \rightarrow h$ Contradict (LF only)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Inference Prediction</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Baseline</td>
<td>0.33</td>
</tr>
<tr>
<td>Logical Form Matching</td>
<td>0.60</td>
</tr>
</tbody>
</table>
General Problem: Semantic representations are underspecified, fail to capture entailments, background knowledge missing.

Goal: Capture the missing knowledge and inferential properties of text, incorporate entailment information into learning.
How to improve this? Test Driven Learning…

- **General Problem:** Semantic representations are underspecified, fail to capture entailments, background knowledge missing.

- **Goal:** Capture the missing knowledge and inferential properties of text, incorporate entailment information into learning.

- **Solution:** Use entailment information (EI) and logical inference as weak signal to train parser, in Richardson and Kuhn (2016).

<table>
<thead>
<tr>
<th>Paradigm and Supervision</th>
<th>Dataset ( D = )</th>
<th>Learning Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning from LFs ( {(input_i, LF_i)}_i^N )</td>
<td>input ( \xrightarrow{\text{Trans.}} ) LF</td>
<td></td>
</tr>
<tr>
<td>Learning from Entailment ( {(input_t, input'_h, EI_i)}_i^N )</td>
<td>(input(_t), input(_h')) ( \xrightarrow{\text{Proof}} ) EI</td>
<td></td>
</tr>
</tbody>
</table>
Learning from Entailment: Illustration

- Entailments are used to reason about target symbols and find holes in the analyses.

\[
\begin{align*}
\text{input: } (t,h) & \quad t \quad \text{pink3} \quad \lambda \quad \text{passes to} \quad \text{pink1} \\
\text{a} & \quad \text{quickly} \\
\text{h} & \quad \text{pink3} \quad \text{passes to} \quad \text{pink1} \\
\end{align*}
\]

\[
\begin{align*}
\text{EI} \quad \text{y} & \quad \text{Uncertain} \\
\text{world} & \\
\end{align*}
\]

Data:  \( D = \{ ((t, h)_i, z_i) \}_{i=1}^N \), generic logical calculus. Task:  learn (latent) proof \( y \)
Learning from Entailment: Illustration

- Entailments are used to reason about target symbols and find holes in the analyses.

\[
\text{input: (t,h)} \quad \begin{array}{c}
\text{a} \\
\text{h}
\end{array} \quad \begin{array}{c}
pink3 \\
\text{quickly}
\end{array} \quad \lambda \quad \text{passes to} \quad \begin{array}{c}
pink1 \\
\text{quickly}
\end{array}
\]

\[
\begin{array}{c}
y \quad \lambda \sqsubseteq c \\
\text{passes to} \\
pink1 \sqsubseteq \lambda
\end{array}
\]

\[
\begin{array}{c}
z
\end{array}
\]

\[
\text{EI}
\]

\[
\text{world}
\]

\[
\text{Data: } D = \{((t, h)_i, z_i)\}_{i=1}^N, \text{ generic logical calculus. Task: learn (latent) proof y}
\]
Learning from Entailment: Illustration

- Entailments are used to reason about target symbols and find holes in the analyses.

\[ \text{input: } (t, h) \]

\[ \begin{array}{c}
\text{a} \\
\text{h} \\
\text{t}
\end{array} \]

\[ \begin{array}{c}
pink3 \\
pink3 \\
pink3
\end{array} \]

\[ \begin{array}{c}
\lambda/\exists c \\
\lambda \\
\lambda
\end{array} \]

\[ \begin{array}{c}
passes to \\
pass/kick \\
passes to
\end{array} \]

\[ \begin{array}{c}
pink1/\lambda \\
pink1/\lambda \\
pink1/\lambda
\end{array} \]

\[ \begin{array}{c}
pink3/pink3 \\
pink3/pink3 \\
pink3/pink3
\end{array} \]

\[ \begin{array}{c}
pink3 \\
pink3 \\
pink3
\end{array} \]

\[ \begin{array}{c}
\equiv \\
\equiv \\
\equiv
\end{array} \]

\[ \begin{array}{c}
\lambda \sqsubseteq c \\
\lambda \sqsubseteq c \\
\lambda \sqsubseteq c
\end{array} \]

\[ \begin{array}{c}
\text{pass } \sqsubseteq \text{kick, pink1 } \sqsubseteq \lambda \\
\text{quickly passes to pink1 } \sqsubseteq \text{kicks} \\
\text{passes to pink1 } \# \text{ quickly kicks}
\end{array} \]

\[ \begin{array}{c}
pink3 \sqsubseteq pink3 \\
pink3 \sqsubseteq pink3 \\
pink3 \sqsubseteq pink3
\end{array} \]

\[ \begin{array}{c}
pink3 \equiv pink3 \\
pink3 \equiv pink3 \\
pink3 \equiv pink3
\end{array} \]

\[ \begin{array}{c}
\text{passes to pink1 } \# \text{ quickly kicks}
\end{array} \]

\[ \begin{array}{c}
pink3 \text{ passes to pink1 } \# \text{ pink3 quickly kicks}
\end{array} \]

\[ z \text{ Uncertain} \]

\[ \text{Data: } D = \{(t, h)_i, z_i\}_{i=1}^N, \text{ generic logical calculus. Task: learn (latent) proof } y \]
Learning from Entailment: Illustration

Entailments are used to reason about target symbols and find holes in the analyses.

Data:  \( D = \{(t_i, h_i, z_i)\}_{i=1}^N \), generic logical calculus. Task: learn (latent) proof \( y \)
Learning from Entailment: Illustration

Entailments are used to reason about target symbols and find holes in the analyses.

Data: \( D = \{(t, h), z_i\}_{i=1}^N \), generic logical calculus. Task: learn (latent) proof \( y \)
Grammar Approach: Sentences to Logical Form

- Use a semantic CFG, rules constructed from target representations using small set of templates (Börschinger et al. (2011))

\[(x: \text{purple 10 quickly kicks}, z: \{\text{kick(purple 10)}, \text{block(purple 7)}, \ldots\})\]

\[\downarrow \text{(rule extraction)}\]
Grammar Approach: Sentences to Logical Form

- Use a semantic CFG, rules constructed from target representations using small set of templates (Börschinger et al. (2011))

\[(x: \text{purple 10 quickly kicks}, z: \{\text{kick(purple10), block(purple7)}, \ldots\})\]

\[\downarrow \text{(rule extraction)}\]
Use a semantic CFG, rules constructed from target representations using small set of templates (Börschinger et al. (2011))

\((x : \text{purple} 10 \text{ quickly kicks, } z : \{\text{kick(purple10)}, \text{block(purple7)}, \ldots\})\)

↓ (rule extraction)

\[
\begin{array}{c}
\checkmark \quad \checkmark \quad \times \quad \times \\
\text{kick(purple10)} \quad \text{kick(purple10)} \quad \text{block(purple7)} \quad \text{block(purple9)}
\end{array}
\]
Semantic Parsing as Grammatical Inference

- Rules used to define a PCFG $G_\theta$, learn correct derivations.
- **Learning:** EM bootstrapping approach (Angeli et al. (2012))

Purple 7 kicks to Purple 4

$Z = \{\text{pass(purple7, purple4)}\}$
Rules used to define a PCFG $G_\theta$, learn correct derivations.

**Learning:** EM bootstrapping approach (Angeli et al. (2012))

$Z = \{\text{pass(purple7,purple4)}\}$
Rules used to define a PCFG $\mathcal{G}_\theta$, learn correct derivations.

**Learning:** EM bootstrapping approach (Angeli et al. (2012))

Purple 7 kicks to Purple 4

```
Z = \{\text{pass(purple7,purple4)}\}
```
Learning Entailment Rules

- Rules define an inference PCFG $G'_{\theta}$, learn correct proofs, uses natural logic calculus from MacCartney and Manning (2009).
- **Learning**: Grammatical inference problem as before, EM bootstrapping.

**Input** $d$

$t$: pink 1 kicks

$h$: pink 1 quickly passes to pink 2

**Beam Parser $\theta^t$**

**Interpretation**

**World**

$z = \text{Uncertain}$

$k$-best list

$\theta^{t+1}$
Reasoning about Entailment

- Improving the internal representations (before, a, after, b).

![Diagram](https://via.placeholder.com/150)

a. Improving the internal representations (before):

```
player_{arg1} \ \\
purple_{9_c} \ \\
\vdots \ \\
purple_{9} \ \\
| \ \\
pass_r \ \\
pass_c \ \\
\vdots \ \\
pass_p \ \\
passes to \ \\
purple \ \\
6\under{pressure}
```

b. Improving the internal representations (after):

```
player_{arg1} \ \\
purple_{9_c} \ \\
\vdots \ \\
purple_{9} \ \\
| \ \\
pass_r \ \\
pass_c \ \\
\vdots \ \\
pass_p \ \\
passes to \ \\
\under{pressure}
```
Reasoning about Entailment

- Learned modifiers from example proofs trees.

\[(t, h)\]:

(a beautiful pass to, passes to)

\[\sqsubseteq_c \sqsupseteq \text{play-tran} = \sqsubseteq_c \text{play-tran}\]

\[\sqsubseteq_c / \lambda \text{ pass/pass}\]

analysis:

"a beautiful"/\lambda "pass to"/"passes to"

generalization:

\[\text{beautiful}(X) \sqsubseteq X\]

( gets a free kick, freekick from the)

\[\sqsubseteq_c \sqsupseteq \text{game-play} = \sqsubseteq_c \text{game-play}\]

\[\sqsubseteq_c / \lambda \text{ freekick/freekick}\]

analysis:

"gets a"/\lambda "free kick" / "freekick from the"

generalization:

\[\text{get}(X) \equiv X\]

\[(t, h)\]:

(yet again passes to, kicks to)

\[\sqsubseteq_c \sqsupseteq \text{play-tran} = \sqsubseteq_c \text{play-tran}\]

\[\sqsubseteq_c / \lambda \text{ pass/pass}\]

analysis:

"yet again"/\lambda "passes to"/"kicks to"

generalization:

\[\text{yet-again}(X) \sqsubseteq X\]

(purple 10, purple 10 who is out front)

\[\equiv_{\text{player}_{\arg 2}} \sqsupseteq \sqsubseteq_c = \sqsubseteq_{\text{player}_{\arg 2}}\]

\[\equiv_{\text{player}_{\arg 2}} \equiv_{\text{player}_{\arg 2}} \sqsubseteq_c \text{ purple10/purple10} \lambda/ \sqsubseteq_c\]

analysis:

"purple 10"/"purple 10" \lambda/"who is out front"

generalization:

\[X \sqsubseteq \text{out_front}(X)\]
Reasoning about Entailment

Learned lexical relations from example proof trees

\((t, h)\): \((\text{pink team is offsides}, \text{purple 9 passes})\)

\(\text{team}_{\text{arg1}}\)

\(\text{substitute}\)

\(\text{pink team/purple9}\)

\(\text{"pink team"/"purple 9"}\)

\(\text{relation: pink team | purple9}\)

\(\square \text{play-tran}\)

\(\text{substitute}\)

\(\text{bad pass/turnover}\)

\(\text{relation: bad pass \sqsubseteq turnover}\)

\(\boxed{45}\)

\((t, h)\): \((\text{free kick for, steals the ball from})\)

\(\text{game-play}\)

\(\text{substitute}\)

\(\text{free kick/steal}\)

\(\text{"free kick for"/"steals the ball from"}\)

\(\text{relation: free kick | steal}\)

\(\square \text{play-tran.}\)

\(\text{substitute}\)

\(\text{pass/kick}\)

\(\text{"kicks to"/"kicks’}\)

\(\text{relation: pass \sqsubseteq kick}\)
Reasoning about Entailment: Summary

▶ **New Ideas**: Evaluating semantic parsers on RTE.
Reasoning about Entailment: Summary

- **New Ideas:** Evaluating semantic parsers on RTE.

<table>
<thead>
<tr>
<th>Inference Prediction</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Baseline</td>
<td>0.33</td>
</tr>
<tr>
<td>Logical Form Matching</td>
<td>0.60</td>
</tr>
<tr>
<td>Learning from Entailment</td>
<td><strong>0.73</strong></td>
</tr>
</tbody>
</table>
Reasoning about Entailment: Summary

▶ **New Ideas**: Evaluating semantic parsers on RTE.

<table>
<thead>
<tr>
<th>Inference Prediction</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Baseline</td>
<td>0.33</td>
</tr>
<tr>
<td>Logical Form Matching</td>
<td>0.60</td>
</tr>
<tr>
<td><strong>Learning from Entailment</strong></td>
<td><strong>0.73</strong></td>
</tr>
</tbody>
</table>

▶ **Take Away**: Using entailment as a weak signal can help improve representations being learned, help theory construction.
Reasoning about Entailment: Summary

- **New Ideas**: Evaluating semantic parsers on RTE.

<table>
<thead>
<tr>
<th>Inference Prediction</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Baseline</td>
<td>0.33</td>
</tr>
<tr>
<td>Logical Form Matching</td>
<td>0.60</td>
</tr>
<tr>
<td><strong>Learning from Entailment</strong></td>
<td><strong>0.73</strong></td>
</tr>
</tbody>
</table>

- **Take Away**: Using entailment as a weak signal can help improve representations being learned, help theory construction.

- **Conceptually**: Training our semantic parsers to be more like a semanticist, work backwards from entailments to representations.
</Learning from Entailment>
Conclusions

- **Natural language understanding:** pursue a data-driven approach, centering around semantic parser induction.

- Explore many areas of the problem, developed new resources and techniques, looking at the problem holistically.
  
  - Easy to get stuck on the translation problem, must be integrated within a broader theory of KR and reasoning.
Conclusions

- **Natural language understanding:** pursue a data-driven approach, centering around semantic parser induction.
- Explore many areas of the problem, developed new resources and techniques, looking at the problem holistically.
  - Easy to get stuck on the translation problem, must be integrated within a broader theory of KR and reasoning.
- **Question:** To what extend can we use source code *in the wild* as a KR for deep NLU?
Thank You
References


