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

September 24, 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**: Type.Article ⊓ R[λx.count(AuthorOf.x)] ≤ 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 \( \sqcap R[\lambda x.\text{count}(\text{AuthorOf}.x)] \leq 2 \)

"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 (NLU)

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

Lunar QA system of Woods (1973)
Why and How? Analogy with Compiler Design

- **NLU model is a kind of compiler, involves a transduction from NL to a formal (usually logical) language.**
Data-driven Semantic Parsing and NLU

1. Semantic Parsing
   - List samples that contain every major element

2. Knowledge Representation
   - \((\text{FOR EVERY } X / \text{ MAJORELT } : T; (\text{FOR EVERY } Y / \text{ SAMPLE } : (\text{CONTAINS } Y X); (\text{PRINTOUT } Y)))\)

3. Reasoning

\([sem] = \{S10019, S10059, \ldots \}\)

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

1. Semantic Parsing

List samples that contain every major element

\[ \text{database} \]

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

2. Knowledge Representation

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

3. Reasoning

▶ **Semantic Parser Induction:** Learn semantic parser (weighted transduction) from parallel text/meaning data, constrained SMT task.
Data-driven Semantic Parsing in a Nutshell

Desiderata: robust and domain agnostic models that require minimal amounts of hand engineering and data supervision.
Data-driven Semantic Parsing in a Nutshell

Training

Parallel Training Set
\[ D = \{(x_i, z_i)\}_{i}^{|D|} \]

→ Machine Learner

Testing

input

\[ x \]

→ Semantic Parsing

\[ \text{decoding} \]

→ model

\[ z \] reasoning

world

Desiderata: robust and domain agnostic models that require minimal amounts of hand engineering and data supervision.
Data-driven Semantic Parsing in a Nutshell

- **Challenge 1: Getting data?**
  - 2014[LREC], 2017c[INLG], 2017b[ACL], 2017a[EMNLP]

- **Challenge 2: Missing Data?**
  - 2018[NAACL]

- **Challenge 3: Deficient LFs?**
  - 2012[COLING], 2016[TACL]

**Parallel Training Set**

\[ D = \left\{ (x_i, z_i) \right\}_{i=1}^{D} \]

**Machine Learner**

**Model**

**Desiderata:** robust and domain agnostic models that require minimal amounts of hand engineering and data supervision.
Data-driven Semantic Parsing in a Nutshell

- **Challenge 1: Getting data?**
  - 2014[LREC], 2017c[INLG]
  - 2017b[ACL], 2017a[EMNLP]

- **Challenge 2: Missing Data?**
  - 2018[NAACL]

- **Challenge 3: Deficient LFs?**
  - 2012[COLING], 2016[TACL]

---

**Training**

Parallel Training Set

\[ D = \{(x_i, z_i)\}_{i=1}^{D} \]

\[ \text{Machine Learner} \]

**Testing**

**Desiderata:** robust and domain agnostic models that require minimal amounts of hand engineering and data supervision.
<Challenge 1>

Training

Parallel Training Set

$D = \left\{ (x_i, z_i) \right\}_{i}^{\left| D \right|}$

Machine Learner

\[ \text{model} \]

...
Semantic Parsing and Parallel Data

What state has the largest population?

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.
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 LF\s**: 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.

- **Underlying Challenge**: Finding parallel data tends to require considerable hand engineering effort (cf. Wang et al. (2015)).
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

- Tight coupling between high-level text and code, easy to extract text/code pairs automatically.

```java
@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)
```

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

<table>
<thead>
<tr>
<th>text</th>
<th>Returns the greater...</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>lang.Math long max( long... )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>text</th>
<th>Returns items from coll...</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>(core.random-sample prob...)</td>
</tr>
</tbody>
</table>
Source Code as a Parallel Corpus

- Tight coupling between high-level text and code, easy to extract text/code pairs automatically.

<table>
<thead>
<tr>
<th>Text</th>
<th>Returns the greater...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td><code>Math long max( long... )</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text</th>
<th>Returns items from coll...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td><code>(core.random-sample prob...)</code></td>
</tr>
</tbody>
</table>

**Function signatures:** Header-like representations, containing function name, arguments, return value, namespace.

```
Signature ::= lang Math long max ( long a, long b )
             namespace class return name named/typed arguments
```
## Resource 1: Standard Library Documentation (Stdlib)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Pairs</th>
<th>#Symbols</th>
<th>#Words</th>
<th>Vocab.</th>
<th>Example Pairs (x, z)</th>
</tr>
</thead>
</table>
| Java         | 7,183  | 4,072    | 82,696  | 3,721  | x: Compares this Calendar to the specified Object.  
          |        |          |         |        | z: boolean util.Calendar.equals(Object obj)                                         |
| Ruby         | 6,885  | 3,803    | 67,274  | 5,131  | x: Computes the arc tangent given y and x.  
          |        |          |         |        | z: Math.atan2(y,x) → Float                                                           |
| PHP<sub>en</sub> | 6,611  | 8,308    | 68,921  | 4,874  | x: Delete an entry in the archive using its name.  
          |        |          |         |        | z: bool ZipArchive::deleteName(string $name)                                          |
| Python       | 3,085  | 3,991    | 27,012  | 2,768  | x: Remove the specific filter from this handler.  
          |        |          |         |        | z: logging.Filterer.removeFilter(filter)                                             |
| Elisp        | 2,089  | 1,883    | 30,248  | 2,644  | x: Returns the total height of the window.  
          |        |          |         |        | z: (window-total-height window round)                                               |
| Geoquery     | 880    | 167      | 6,663   | 279    | x: What is the tallest mountain in America?  
          |        |          |         |        | z: (highest(mountain(loc_2(countryid usa))))                                         |

- Documentation for 16 APIs, 10 programming languages, 7 natural languages, from Richardson and Kuhn (2017b).

  - **Advantages:** zero annotation, highly multilingual, relatively large.
## Resource 1: Standard Library Documentation (Stdlib)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Pairs</th>
<th>#Symbols</th>
<th>#Words</th>
<th>Vocab.</th>
<th>Example Pairs (x, z)</th>
</tr>
</thead>
</table>
| Java    | 7,183  | 4,072    | 82,696 | 3,721  | x: Compares this Calendar to the specified Object.  
              z: boolean util.Calendar.equals(Object obj) |
| Ruby    | 6,885  | 3,803    | 67,274 | 5,131  | x: Computes the arc tangent given y and x.  
              z: Math.atan2(y,x) → Float |
| PHPen   | 6,611  | 8,308    | 68,921 | 4,874  | x: Delete an entry in the archive using its name.  
              z: bool ZipArchive::deleteName(string $name) |
| Python  | 3,085  | 3,991    | 27,012 | 2,768  | x: Remove the specific filter from this handler.  
              z: logging.Filterer.removeFilter(filter) |
| Elisp   | 2,089  | 1,883    | 30,248 | 2,644  | x: Returns the total height of the window.  
              z: (window-total-height window round) |
| Geoquery| 880    | 167      | 6,663  | 279    | x: What is the tallest mountain in America?  
              z: (highest(mountain(loc_2(countryid usa)))) |

- Documentation for 16 APIs, 10 programming languages, 7 natural languages, from Richardson and Kuhn (2017b).
  
  - **Advantages:** zero annotation, highly multilingual, relatively large.
### Resource 2: Python Projects (Py27)

<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>4,777</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>sqlalchemy</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>
<tr>
<td>geoquery</td>
<td>880</td>
<td>167</td>
<td>6,663</td>
<td>279</td>
</tr>
</tbody>
</table>

- 27 English Python projects from Github (Richardson and Kuhn, 2017a).
New Task: Text to Signature Translation

<table>
<thead>
<tr>
<th>text</th>
<th>Returns the greater of two long values</th>
</tr>
</thead>
<tbody>
<tr>
<td>signature</td>
<td>lang.Math long max( long a, long b )</td>
</tr>
</tbody>
</table>

- **Task**: Given text/signatures training pairs, learn a (quasi) *semantic parser*: text $\rightarrow$ signature (Richardson and Kuhn, 2017b)

  - **Assumption**: predicting within finite signature/translation space.
New Task: Text to Signature Translation

<table>
<thead>
<tr>
<th>text</th>
<th>Returns the greater of two long values</th>
</tr>
</thead>
<tbody>
<tr>
<td>signature</td>
<td>lang.Math long max( long a, long b )</td>
</tr>
</tbody>
</table>

- **Task**: Given text/signatures training pairs, learn a (quasi) semantic parser: text → signature (Richardson and Kuhn, 2017b)

- **Assumption**: predicting within finite signature/translation space.

- **Code Retrieval Analogy**: train/test split, at test time, retrieve function signature that matches input specification (Deng and Chrupała, 2014):

  - String APCIterator::key(void)
  - Int APCIterator::getTotalHits(void)
  - Int APCIterator::getSize(void)
  - Int APCIterator::getTotalSize(void)
  - Int Memcached::append(string $key)
  - "..."

  Accuracy @i? (exact match)
Initial approach: noisy-channel (nc) classical translation:

\[
\text{SEMPAR}^\text{nc}(x, z) = p_\theta(x \mid z) \times p_{1m}(z)
\]

- trans model
- valid expression (yes/no)?
Initial approach: noisy-channel (nc) classical translation:

\[
\text{SEMPAR}^{nc}(x, z) = p_{\theta}(x | z) \times p_{lm}(z)
\]

- \( p_{\theta}(x | z) \): trans model
- \( p_{lm}(z) \): valid expression (yes/no)?

Initial approach: noisy-channel (nc) classical translation:

\[
\text{SEMPAR}^{nc}(x, z) = p_{\theta}(x | z) \times p_{lm}(z)
\]

- \( p_{\theta}(x | z) \): trans model
- \( p_{lm}(z) \): valid expression (yes/no)?

\[\text{Initial approach: noisy-channel (nc) classical translation:}\]

\[
\text{SEMPAR}^{nc}(x, z) = p_{\theta}(x | z) \times p_{lm}(z)
\]

- \( p_{\theta}(x | z) \): trans model
- \( p_{lm}(z) \): valid expression (yes/no)?

1m: convenient for making strong assumptions about our output language, facilitates constrained decoding.
**Initial approach**: noisy-channel (nc) classical translation:

\[
\text{SEMPAR}^{nc}(x, z) = \underbrace{p_\theta(x \mid z)}_{\text{trans model}} \times \underbrace{p_{lm}(z)}_{\text{valid expression (yes/no)?}}
\]

- **lm**: convenient for making strong assumptions about our output language, facilitates **constrained decoding**.
- **code case**: make assumptions about what constitutes a **valid function** in a given API.
Text to Signature Translation: How Hard Is It?

- **Our Approach**: Lexical translation model (standard estimation), discriminative reranker, hard constraints on $p(z)$. 
Text to Signature Translation: How Hard Is It?

- **Our Approach**: Lexical translation model (standard estimation), discriminative reranker, hard constraints on $p(z)$. 

![Bar chart showing accuracy at 1 for different datasets]

- Our Noisy-Chanel SMT Model
- Discriminative Reranker
- Competitor Model/Baseline
- Blackbox PBSMT (Moses)
Our Approach: Lexical translation model (standard estimation), discriminative reranker, hard constraints on $p(z)$. 

![Graph showing accuracy comparison across different datasets. The graph compares Our Noisy-Chanel SMT Model, Discriminative Reranker, Competitor Model/Baseline, and Blackbox PBSMT (Moses) across Multilingual GeoQuery (4), Stdlib (16), and Py27 (27) datasets. The CCG: Kwiatkowski, T. et al. 2010 dataset is highlighted with a bar chart indicating higher accuracy.]
Our Approach: Lexical translation model (standard estimation), discriminative reranker, hard constraints on $p(z)$. 

![Bar chart showing accuracy on different datasets](chart.png)
Text to Signature Translation: How Hard Is It?

- **Our Approach**: Lexical translation model (standard estimation), discriminative reranker, hard constraints on $p(z)$.
Text to Signature Translation: How Hard Is It?

- **Our Approach**: Lexical translation model (standard estimation), discriminative reranker, hard constraints on $p(z)$.

- **Result**: achieving high accuracy is not easy, not a trivial problem.
Text to Signature Translation: How Hard Is It?

- **Our Approach**: Lexical translation model (standard estimation), discriminative reranker, hard constraints on $p(z)$. 

![Graph showing accuracy at 1 (avg.) for different datasets and models.](image)
Our Approach: Lexical translation model (standard estimation), discriminative reranker, hard constraints on $p(z)$. 
Text to Signature Translation: How Hard Is It?

- **Our Approach**: Lexical translation model (standard estimation), discriminative reranker, hard constraints on $p(z)$.

<table>
<thead>
<tr>
<th>text</th>
<th>Returns the index of the first occurrence of char in the string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moses</td>
<td>(start end occurrence lambda char string string string)</td>
</tr>
</tbody>
</table>

Accuracy @1 (avg.)

- **CCG**: Kwiatkowski, T. et al. 2010
- **Andreas, J. et al. 2013**

Datasets (# individual datasets)

- Multilingual GeoQuery (4)
- Stdlib (16)
- Py27 (27)
Text to Signature Translation: How Hard Is It?

- **Our Approach**: Lexical translation model (standard estimation), discriminative reranker, hard constraints on $p(z)$.

<table>
<thead>
<tr>
<th>Text</th>
<th>Returns the index of the first occurrence of char in the string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moses</td>
<td>(start end occurrence lambda char string string string)</td>
</tr>
</tbody>
</table>

- **Observation**: semantic parsing is not an unconstrained MT problem.
What do these results mean? Code Retrieval Again

Function{} Assistant

Please convert a character to a unicode string

PHP_en  Top 10  Generate  Search

private int HaruEncoder :: getUnicode ($character=int )

Converts the specified character to unicode

private int ord ( $string=string )

Returns the ascii value of the first character of string

private int HaruFont :: getUnicodeWidth ( $character=int )

Get the width of the character in the font

<table>
<thead>
<tr>
<th>Dataset(Avg.)</th>
<th>Accuracy @1 (average)</th>
<th>Accuracy @10 (average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stdlib</td>
<td>31.1</td>
<td>71.0</td>
</tr>
<tr>
<td>Py27</td>
<td>32.3</td>
<td>73.5</td>
</tr>
</tbody>
</table>
so far: Semantic parsing as constrained translation, API as parallel corpus.

</Challenge 1>
Challenge 2: Insufficient and Missing Data

Traditional approaches to semantic parsing train individual models for each available parallel dataset.

Underlying Challenge: 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?
Graph Based Approach

- Requirements: must generate well-formed output, be able to translate to target languages on demand.

- Idea: Exploit finite-ness of translation space, represent full search space as directed acyclic graph (DAG), add artificial language tokens.
Graph Based Approach

Requirements: must generate well-formed output, be able to translate to target languages on demand.

Idea: Exploit finite-ness of translation space, represent full search space as directed acyclic graph (DAG), add artificial language tokens.

Decoding (test time): Reduces to finding a path given an input $x$:

$x$ : The ceiling of a number

We formulate search in terms of single source shortest shortest-path (SSSP) search (Cormen et al., 2009) on DAGs.
Shortest Path Decoding in a Nutshell

- **Standard SSSP**: Traverse labeled edges $E$ (label $z$) in order (e.g., sorted or best-first order), and solve for each node $v$ the following recurrence:

$$d[v] = \min_{(u,v,z) \in E} \left\{ d[u] + \underbrace{w(u, v, z)}_{\text{edge score}} \right\}$$

- Use trained translation model to dynamically weight edges, general framework for directly comparing models (Richardson et al., 2018).

- Constrained decoding: ensure that output is well-formed, related efforts: (Krishnamurthy et al., 2017; Yin and Neubig, 2017).
Shortest Path Decoding in a Nutshell

- **Standard SSSP**: Traverse labeled edges $E$ (label $z$) in order (e.g., sorted or best-first order), and solve for each node $v$ the following recurrence:

  \[ d[v] = \min_{(u,v,z) \in E} \left( d[u] + \text{TRANS}(x, z) \right) \]

  node score \quad \text{incoming node score} \quad \text{translation}

- Use trained **translation model** to dynamically weight edges, general framework for directly comparing models (Richardson et al., 2018).
Shortest Path Decoding in a Nutshell

▶ **Standard SSSP**: Traverse labeled edges $E$ (label $z$) in order (e.g., sorted or best-first order), and solve for each node $v$ the following recurrence:

$$d[v] = \min_{(u,v,z) \in E} \left\{ d[u] + \text{TRANS}(x, z) \right\}$$

node score  \[\uparrow\]
incoming node score  \[\uparrow\]
translation

▶ Use trained **translation model** to dynamically weight edges, general framework for directly comparing models (Richardson et al., 2018).

▶ **constrained decoding**: ensure that output is well-formed, related efforts: (Krishnamurthy et al., 2017; Yin and Neubig, 2017).
DAG Decoding for Neural Semantic Parsing (Example)

- **Seq2Seq**: popular in semantic parsing (Dong and Lapata, 2016), variants of (Bahdanau et al., 2014), direct decoder model (unconstrained):

\[
p(z \mid x) = \text{CONDITIONALRNNLM}(z) = \prod_{i} p_{\Theta}(z_i \mid z_{<i}, x)
\]
DAG Decoding for Neural Semantic Parsing (Example)

- **Seq2Seq**: popular in semantic parsing (Dong and Lapata, 2016), variants of (Bahdanau et al., 2014), direct decoder model (unconstrained):

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

- **DAGs** \( \mathcal{G} = (V, E) \), **numerically sorted nodes** (acyclic), trained decoder.
DAG Decoding for Neural Semantic Parsing (Example)

- **Seq2Seq**: popular in semantic parsing (Dong and Lapata, 2016), variants of (Bahdanau et al., 2014), direct decoder model (unconstrained):

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

- **DAGs** \( \mathcal{G} = (V, E) \), **numerically sorted nodes** (acyclic), trained decoder.

0: \( d[b] \leftarrow 0.0 \)

1: **for** node \( v \in V \) in topologically sorted order
DAG Decoding for Neural Semantic Parsing (Example)

- **Seq2Seq**: popular in semantic parsing (Dong and Lapata, 2016), variants of (Bahdanau et al., 2014), direct decoder model (unconstrained):

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

- **DAGs** \( G = (V, E) \), **numerically sorted nodes** (acyclic), trained decoder.

0: \( d[b] \leftarrow 0.0 \)
1: for node \( v \in V \) in topologically sorted order
2: do \( d(v) = \min_{(u,v,z_j) \in E} \left\{ d(u) + -\log p_{\Theta}(z_j | z_{<j}, x) \right\} \)
DAG Decoding for Neural Semantic Parsing (Example)

- **Seq2Seq**: popular in semantic parsing (Dong and Lapata, 2016), variants of (Bahdanau et al., 2014), direct decoder model (unconstrained):

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

- **DAGs** \( \mathcal{G} = (V, E) \), **numerically sorted nodes** (acyclic), trained decoder.

  0: \( d[b] \leftarrow 0.0 \)
  1: for node \( v \in V \) in topologically sorted order
  2: \( d(v) = \min_{(u,v,z_j) \in E} \left\{ d(u) + - \log p_{\Theta}(z_j \mid z_{<j}, x) \right\} \)
  3: \( s[v] \leftarrow \text{RNN state for min edge and } z_j \)
DAG Decoding for Neural Semantic Parsing (Example)

- **Seq2Seq**: popular in semantic parsing (Dong and Lapata, 2016), variants of (Bahdanau et al., 2014), direct decoder model (unconstrained):

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

- **DAGs** \(G = (V, E)\), **numerically sorted nodes** (acyclic), trained decoder.

0: \(d[b] \leftarrow 0.0\)
1: for node \(v \in V\) in topologically sorted order
2: \(d(v) = \min_{(u, v, z_j) \in E} \left\{ d(u) + -\log p_{\Theta}(z_j \mid z_{<j}, x) \right\} \)
3: \(s[v] \leftarrow \text{RNN state for min edge and } z_j\)
4: return \(\min_{v \in V} \{d(v)\}\)
Training on Multiple Datasets: Does this help?

- **Strategy**: train models on multiple datasets (polyglot models), decoding to target languages and check for improvement.

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

Findings: Polyglot modeling can help improve accuracy depending on the model used, Seq2Seq models did not perform well on code datasets.
Training on Multiple Datasets: Does this help?

- **Strategy:** train models on multiple datasets (polyglot models), decoding to target languages and check for improvement.

<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></td>
<td>Best Seq2Seq SSSP</td>
<td>78.0</td>
</tr>
<tr>
<td>poly.</td>
<td>Lexical SMT SSSP</td>
<td>67.3</td>
</tr>
<tr>
<td></td>
<td>Best Seq2Seq SSSP</td>
<td>79.6</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>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>
<tr>
<td>Best Monolingual Model</td>
<td>32.4</td>
<td>73.5</td>
<td>46.5</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>
Training on Multiple Datasets: Does this help?

▶ **Strategy**: train models on multiple datasets (polyglot models), decoding to target languages and check for improvement.

<table>
<thead>
<tr>
<th>Multilingual Geoquery</th>
<th>Method</th>
<th>Acc@1 (averaged)</th>
</tr>
</thead>
<tbody>
<tr>
<td></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></td>
<td>Best Seq2Seq SSSP</td>
<td>78.0</td>
</tr>
<tr>
<td></td>
<td>Lexical SMT SSSP</td>
<td>67.3</td>
</tr>
<tr>
<td></td>
<td>Best Seq2Seq SSSP</td>
<td><strong>79.6</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stdlib</th>
<th>Method</th>
<th>Acc@1</th>
<th>Acc@10</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best Monolingual Model</td>
<td>29.9</td>
<td>69.2</td>
<td>43.1</td>
</tr>
<tr>
<td></td>
<td>Lexical SMT SSSP</td>
<td>33.2</td>
<td>70.7</td>
<td>45.9</td>
</tr>
<tr>
<td></td>
<td>Best Seq2Seq SSSP</td>
<td>13.9</td>
<td>36.5</td>
<td>21.5</td>
</tr>
<tr>
<td>py27</td>
<td>Best Monolingual Model</td>
<td>32.4</td>
<td>73.5</td>
<td>46.5</td>
</tr>
<tr>
<td></td>
<td>Lexical SMT SSSP</td>
<td><strong>41.3</strong></td>
<td>77.7</td>
<td>54.1</td>
</tr>
<tr>
<td></td>
<td>Best Seq2Seq SSSP</td>
<td>9.0</td>
<td>26.9</td>
<td>15.1</td>
</tr>
</tbody>
</table>

▶ **Findings**: Polyglot modeling can help improve accuracy depending on the model used, Seq2Seq models did not perform well on code datasets.
Any Language Decoding: translating between multiple APIs, letting the decoder decide output language.

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

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

<table>
<thead>
<tr>
<th>Source API (py):</th>
<th>Input: Compute the Moore-Penrose pseudo-inverse of a matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td>en, stats</td>
<td>matrices.matrix.base.pinv_solve( B, ... )</td>
</tr>
<tr>
<td>sympy</td>
<td>utils.pinvh( a, cond=None,rcond=None,... )</td>
</tr>
<tr>
<td>sklearn</td>
<td>tools.pinv2( a,cond=None,rcond=None )</td>
</tr>
</tbody>
</table>
New Tasks: Any/Mixed Language Decoding

- **Any Language Decoding**: translating between multiple APIs, letting the decoder decide output language.

<table>
<thead>
<tr>
<th></th>
<th>Source API (stdlib)</th>
<th>Input</th>
<th>Output Language</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(es, PHP)</td>
<td>Devuelve el mensaje asociado al objeto lanzado.</td>
<td>PHP</td>
<td>public string Throwable::getMessage ( void )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Java</td>
<td>public String lang.getMessage( void )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Clojure</td>
<td>(tools.logging.fatal throwable message &amp; more)</td>
</tr>
<tr>
<td>2</td>
<td>(ru, PHP)</td>
<td>конвертирует строку из формата UTF-32 в формат UTF-16.</td>
<td>PHP</td>
<td>string PDF_utf32_to_utf16 ( ... )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ruby</td>
<td>String#toutf16 =&gt; string</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Haskell</td>
<td>Encoding.encodeUtf16LE :: Text -&gt; ByteString</td>
</tr>
<tr>
<td>3</td>
<td>(en, stats)</td>
<td>Compute the Moore-Penrose pseudo-inverse of a matrix.</td>
<td>sympy</td>
<td>matrices.matrix.base.pinv_solve( B, ... )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sklearn</td>
<td>utils.pinvh( a, cond=None, rcond=None, ... )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>stats</td>
<td>tools.pinv2( a, cond=None, rcond=None )</td>
</tr>
</tbody>
</table>

**Challenge 2**: Can be used for finding missing data, data augmentation.
New Tasks: Any/Mixed Language Decoding

- **Any Language Decoding:** translating between multiple APIs, letting the decoder decide output language.

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

<table>
<thead>
<tr>
<th></th>
<th>Source API (stdlib): (ru, PHP)</th>
<th>Input: конвертирует строку из формата UTF-32 в формат UTF-16.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Language: <strong>PHP</strong></td>
<td>Translation: string PDF_utf32_to_utf16 ( ... )</td>
</tr>
<tr>
<td></td>
<td>Language: <strong>Ruby</strong></td>
<td>Translation: String#toutf16 =&gt; string</td>
</tr>
<tr>
<td></td>
<td>Language: <strong>Haskell</strong></td>
<td>Translation: Encoding.encodeUtf16LE :: Text -&gt; ByteString</td>
</tr>
</tbody>
</table>

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

**Challenge 2:** Can be used for finding missing data, data augmentation.
New Tasks: Any/Mixed Language Decoding

- **Mixed Language Decoding**: translating from input with NPs from multiple languages, introduced a new mixed GeoQuery test set.

<table>
<thead>
<tr>
<th>Mixed Lang.</th>
<th>Input: Wie hoch liegt der höchstgelegene punkt in Αλαμπάμα?</th>
</tr>
</thead>
<tbody>
<tr>
<td>LF:</td>
<td>answer(elevation_1(highest(place(loc_2(stateid('alabama'))))))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (averaged)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed Best Monolingual Seq2Seq</td>
<td>4.2</td>
</tr>
<tr>
<td>Polyglot Seq2Seq</td>
<td>75.2</td>
</tr>
</tbody>
</table>
</Challenge 2>
<Challenge 3>

Training

Parallel Training Set
\[ D = \left\{ (x_i, z_i) \right\}^{|D|}_i \]

Machine Learner

Testing

input

Semantic Parsing

\[ x \rightarrow \text{decoding} \rightarrow \text{sem} \]

\[ z \rightarrow \text{reasoning} \rightarrow \text{world} \]
Semantic Parsing and Entailment

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

All samples that contain a major element

→

Some sample that contains a major element

\[
\text{database} \quad \left[ \text{sem} \right] = \{S10019, S10059, \ldots \} \supseteq \{S10019\}
\]
Semantic Parsing and Entailment

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

**Entailment as a Unit Test:** For a set of target sentences, check that our semantic model (via some analysis for each sentence, e.g., an LF) accounts for particular entailment patterns observed between pairs of sentences; modify our model when such tests fail.

<table>
<thead>
<tr>
<th>sentence</th>
<th>analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>t <em>All samples that contain a major element</em></td>
<td>LFₜ</td>
</tr>
<tr>
<td>h <em>Some sample that contains a major element</em></td>
<td>LFₜ</td>
</tr>
</tbody>
</table>

- **Inference** $t \rightarrow h$: Entailment (RTE₁)
- **Inference** $h \rightarrow t$: Unknown (RTE)

---

¹Would a person reading $t$ ordinarily infer $h$? (Dagan et al., 2005)
Question: What happens if we *unit test* our semantic parsers?

Sportscaster: ≈1,800 Robocup soccer descriptions paired with logical forms (LFs) (Chen and Mooney, 2008).

<table>
<thead>
<tr>
<th>sentence</th>
<th>analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>t  Pink 3 passes to Pink 7</td>
<td>pass(pink3,pink7)</td>
</tr>
<tr>
<td>h  Pink 3 quickly kicks to Pink 7</td>
<td>pass(pink3,pink7)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>inference (human) t → h</th>
<th>Unknown (RTE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>inference (LF match) t → h</td>
<td>Entail (RTE)</td>
</tr>
</tbody>
</table>
Semantic Parsing and Entailment

▶ **Question**: What happens if we *unit test* our semantic parsers?

▶ **Sportscaster**: $\approx$1,800 Robocup soccer descriptions paired with logical forms (LFs) (Chen and Mooney, 2008).

<table>
<thead>
<tr>
<th>sentence</th>
<th>analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>The pink goalie passes to pink 7</td>
<td>pass(pink1,pink7)</td>
</tr>
<tr>
<td>Pink 1 kicks the ball</td>
<td>kick(pink1)</td>
</tr>
</tbody>
</table>

| inference (human) $t \rightarrow h$ | Entail (RTE) |
| inference (LF match) $t \rightarrow h$ | Contradict (RTE) |
Question: What happens if we unit test our semantic parsers?

Sportscaster: ≈1,800 Robocup soccer descriptions paired with logical forms (LFs) (Chen and Mooney, 2008).

<table>
<thead>
<tr>
<th>Inference Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Baseline</td>
<td>33.1%</td>
</tr>
<tr>
<td>RTE Classifier</td>
<td>52.4%</td>
</tr>
<tr>
<td>LF Matching</td>
<td>59.6%</td>
</tr>
</tbody>
</table>
Challenge 3: Deficient LFs, Missing Knowledge

- **Underlying Challenge:** 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.
Challenge 3: Deficient LFs, Missing Knowledge

- **Underlying Challenge:** 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, jointly optimize model to reason about entailment.

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

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

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.

Data: \( D = \{(t, h, z_i)\}_{i=1}^N \), 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: \( D = \{(t, h)_{i}, z_{i}\}_{i=1}^{N} \).  

Task: Learn (latent) proof \( y \).
Entailments are used to reason about target symbols and find holes in the analyses.

**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.

Data: \( D = \{(t, h)_i, z_i\}_{i=1}^N \), 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))

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

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

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

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

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

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

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

\[ \downarrow \text{(rule extraction)} \]

- kick(purple10)
- kick(purple10)
- block(purple7)
- block(purple9)
Semantic Parsing as Grammatical Inference

- Rules used to define a PCFG $G_{\theta}$, learn correct derivations.
- Learning: EM bootstrapping approach (Angeli et al., 2012), maximum (marginal) likelihood with beam search.

Purple 7 kicks to Purple 4

```
Z = \{\text{pass(purple7, purple4)}\}
```
Semantic Parsing as Grammatical Inference

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

Purple 7 kicks to Purple 4

$Z = \{\text{pass(purple7,purple4)}\}$
Semantic Parsing as Grammatical Inference

- Rules used to define a PCFG $\mathcal{G}_\theta$, learn correct derivations.

- **Learning:** EM bootstrapping approach (Angeli et al., 2012), maximum (marginal) likelihood with beam search.

Purple 7 kicks to Purple 4

$d_1$  play-transitive  player  $	ext{arg}_2$ purple4  $	ext{pass}$  $	ext{r}$ passes to  player  $	ext{arg}_1$ purple7

$d_2$  play-transitive  player  $	ext{arg}_2$ purple4  turnover  $	ext{r}$ passes to  player  $	ext{arg}_1$ purple7

$d_3$  play-transitive  player  $	ext{arg}_2$ purple8  kick  $	ext{r}$ passes to  player  $	ext{arg}_1$ purple7

$d_4$  play-transitive  player  $	ext{arg}_1$ purple7  $	ext{pass}$  $	ext{r}$ passes to  player  $	ext{arg}_2$ purple4

$\ldots\ldots\ldots\ldots$

$d_k$  play-transitive  player  $	ext{arg}_1$ purple7  $	ext{pass}$  $	ext{r}$ passes to  player  $	ext{arg}_2$ purple4

$k$-best list

$\theta^{t+1}$

$Z = \{\text{pass(purple7,purple4)}\}$
Joint Entailment Modeling and Reasoning

- Weakly-supervised semantic parsing (Liang et al., 2013; Berant et al., 2013), treat as partially-observed random process (Guu et al., 2017).

\[ x = (t, h), z \in \{\text{Entail}, \text{Contradict}, \text{Unknown}\} \]
Joint Entailment Modeling and Reasoning

- Weakly-supervised semantic parsing (Liang et al., 2013; Berant et al., 2013), treat as partially-observed random process (Guu et al., 2017).

\[ x = (t, h), \, z \in \{\text{Entail}, \text{Contradict}, \text{Unknown}\} \]

\[ p(z \mid x) = \sum_{y \in \mathcal{Y}_x} p(z \mid y) \times p_\theta(y \mid x) \]

\[ \text{valid inference?} \]

\[ \text{proof score} \]
Weakly-supervised semantic parsing (Liang et al., 2013; Berant et al., 2013), treat as partially-observed random process (Guu et al., 2017).

\[ x = (t, h), z \in \{\text{Entail, Contradict, Unknown}\} \]

\[
p(z \mid x) = \sum_{y \in \mathcal{Y}_x} p(z \mid y) \times p_\theta(y \mid x)
\]

- \( p(z \mid y) \): 1 if proof derives correct entailment, 0 otherwise
Joint Entailment Modeling and Reasoning

Weakly-supervised semantic parsing (Liang et al., 2013; Berant et al., 2013), treat as partially-observed random process (Guu et al., 2017).

\[ x = (t, h), \quad z \in \{\text{Entail}, \text{Contradict}, \text{Unknown}\} \]

\[
p(z \mid x) = \sum_{y \in \mathcal{Y}_x} p(z \mid y) \times p(\theta(y \mid x))
\]

\[ \uparrow \text{valid inference?} \]

\[ \uparrow \text{proof score} \]

\[ p(z \mid y) : 1 \text{ if proof derives correct entailment, 0 otherwise} \]

\[ p_\theta(y \mid x) : \text{Model proof structures and rules as PCFG, use variant of natural logic calculus (MacCartney and Manning, 2009).} \]
Joint Entailment Modeling and Reasoning

- Weakly-supervised semantic parsing (Liang et al., 2013; Berant et al., 2013), treat as partially-observed random process (Guu et al., 2017).

\[ x = (t, h), \quad z \in \{\text{Entail}, \text{Contradict}, \text{Unknown}\} \]

\[ p(z \mid x) = \sum_{y \in Y_x} \left[ p(z \mid y) \times p_\theta(y \mid x) \right] \]

- \( p(z \mid y) : 1 \) if proof derives correct entailment, 0 otherwise

- \( p_\theta(y \mid x) : \) Model proof structures and rules as PCFG, use variant of natural logic calculus (MacCartney and Manning, 2009).

- Results in an interesting probabilistic logic, efficient proof search via reduction to (P)CFG search.
Learning Entailment Rules

- Integrates a symbolic reasoner directly into the semantic parser, allows for joint training using a single generative model.
- **Learning**: Grammatical inference problem as before, maximum (marginal) likelihood with beam search ($\mathcal{Y}_x \approx \text{KBEST}(x)$).
Reasoning about Entailment

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

```
a. Sem_{sv}
  player_{arg1}  play-transitive
    purple9_c  pass_r  player_{arg2}
      pass_c  purple_c 6_c
        pass_p  purple 6  under pressure
          passes to purple
b. Sem_{sv}
  player_{arg1}  play-transitive
    purple9_c  pass_r  player_{arg2}
      pass_c  purple6_c
        pass_p  purple 6  under pressure
          passes to purple
```
Reasoning about Entailment

- Learned modifiers from example proofs trees.

(t, h):

(a beautiful pass to, passes to)

\[ \equiv c \upharpoonright \equiv \text{play-tran} = \equiv \text{play-tran} \]

\[ \equiv c \upharpoonright \equiv \text{pass/pass} \]

"a beautiful"/λ "pass to"/"passes to"

generalization:

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

(gets a free kick, freekick from the)

\[ \equiv c \upharpoonright \equiv \text{game-play} = \equiv \text{game-play} \]

\[ \equiv c \upharpoonright \equiv \text{freekick/freekick} \]

"gets a"/λ "free kick" / "freekick from the"

generalization:

[get(X) ≡ X]

(t, h):

(yet again passes to, kicks to)

\[ \equiv c \upharpoonright \equiv \text{play-tran} = \equiv \text{play-tran} \]

\[ \equiv c \upharpoonright \equiv \text{pass/pass} \]

"yet again"/λ "passes to"/"kicks to"

generalization:

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

(purple 10, purple 10 who is out front)

\[ =_{\text{player}} \uplus \exists c =_{\exists \text{player}} \]

\[ \equiv \text{player}_2 \]

\[ \equiv \text{purple10/purple10} \]

"purple 10"/"purple 10" λ/"who is out front"

generalization:

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

- **Learned lexical relations from example proof trees**

<table>
<thead>
<tr>
<th>$(t, h)$</th>
<th>$(\text{pink team is offsides}, \text{purple 9 passes})$</th>
<th>$(\text{bad pass...}, \text{loses the ball to})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis:</td>
<td>“pink team”/“purple 9”</td>
<td>“bad pass .. picked off by”/“loses the ball to”</td>
</tr>
<tr>
<td>relation:</td>
<td>pink team</td>
<td>purple9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$(t, h)$</th>
<th>$(\text{free kick for}, \text{steals the ball from})$</th>
<th>$(\text{purple 6 kicks to}, \text{purple 6 kicks})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis:</td>
<td>“free kick for”/“steals the ball from”</td>
<td>“kicks to”/“kicks’</td>
</tr>
<tr>
<td>relation:</td>
<td>free kick steal</td>
<td>pass (\sqsubseteq) kick</td>
</tr>
</tbody>
</table>
Learning from Entailment: Summary

- **New Evaluation**: Evaluating semantic parsers on recognizing textual entailment, check if we are learning the missing information.

<table>
<thead>
<tr>
<th>Inference Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Baseline</td>
<td>33.1%</td>
</tr>
<tr>
<td>RTE Classifier</td>
<td>52.4%</td>
</tr>
<tr>
<td>LF Matching</td>
<td>59.6%</td>
</tr>
<tr>
<td><strong>Logical Inference Model</strong></td>
<td><strong>73.4%</strong></td>
</tr>
</tbody>
</table>
Learning from Entailment: Summary

- **New Evaluation**: Evaluating semantic parsers on recognizing textual entailment, check if we are learning the missing information.

<table>
<thead>
<tr>
<th>Inference Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Baseline</td>
<td>33.1%</td>
</tr>
<tr>
<td>RTE Classifier</td>
<td>52.4%</td>
</tr>
<tr>
<td>LF Matching</td>
<td>59.6%</td>
</tr>
<tr>
<td><strong>Logical Inference Model</strong></td>
<td>73.4%</td>
</tr>
</tbody>
</table>

- Entailments prove to be a good learning signal for learning improved representations (joint models also achieve SOTA on original semantic parsing task).
</Challenge 3>
Conclusions and Looking Ahead

challenge 1: Getting data?
Solution: Look to the technical docs, source code as a parallel corpus.

challenge 2: Missing Data?
Solution: Polyglot modeling over multiple datasets.

challenge 3: Deficient LFs?
Solution: Learning from entailment, RTE training.

Parallel Training Set
\[ D = \{(x_i, z_i)\}_{i=1}^{D} \]

Machine Learner

Testing

input

Semantic Parsing

\[ x \rightarrow \text{decoding} \rightarrow \text{sem} \]
Conclusions and Looking Ahead

**Challenge 1: Getting data?**
Solution: Look to the technical docs, source code as a parallel corpus.

**Challenge 2: Missing Data?**
Solution: Polyglot modeling over multiple datasets.

**Challenge 3: Deficient LFs?**
Solution: Learning from entailment, RTE training.

**Parallel Training Set**

\[ D = \left\{ (x_i, z_i) \right\}_{i=1}^{\left| D \right|} \]

**Training**

**Testing**

**Technical topics:** graph-based constrained decoding, (probabilistic) logics for joint semantic parsing and reasoning.
Conclusions and Looking Ahead

- **Challenge 1**: Getting data? Solution: Look to the technical docs, source code as a parallel corpus.

- **Challenge 2**: Missing Data? Solution: Polyglot modeling over multiple datasets.

- **Challenge 3**: Deficient LFs? Solution: Learning from entailment, RTE training.

**Parallel Training Set**

\[ D = \{(x_i, z_i)\}_{i=1}^{|D|} \]

**Machine Learner**

**Input** → **Semantic Parsing** → **Output**

- **Technical topics**: graph-based constrained decoding, (probabilistic) logics for joint semantic parsing and reasoning.

- **Looking ahead**: more work on end-to-end NLU, neural learning from entailment?, structured decoding frameworks, code retrieval.
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:

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

What do signature actually mean? Signatures can be given a formal semantics (Richardson, 2018).

Might prove to be a good resource for investigating end-to-end NLU and symbolic reasoning, APIs contain loads of declarative knowledge.

see Neubig and Allamnis NAACL18 tutorial *Modeling NL, Programs and their Intersection.* and Allamanis et al. (2018).
Thank You
References


References III


