Semantic Parsing and its Applications in Code Generation

Mentors: Navin Goyal and Monojit Choudhury
Problem Overview
Problem Overview

NL → Parser → Logical Form

- Lambda - DCS
- DSL
- Programming Language
What are the ids of stations that are located in San Francisco and have average bike availability above 10?

\[
\text{Interface} = (\text{SELECT id FROM station WHERE city = "San Francisco") INTERSECT (SELECT station_id FROM status GROUP BY station_id HAVING avg(bikes_available) > 10})
\]

<table>
<thead>
<tr>
<th>S.No.</th>
<th>ID No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13225</td>
</tr>
<tr>
<td></td>
<td>....</td>
</tr>
<tr>
<td></td>
<td>....</td>
</tr>
<tr>
<td>n</td>
<td>37724</td>
</tr>
</tbody>
</table>
Problem Overview: Semantic Code Search/Generation

- Programmer-oriented use-case
- Search for code by functionality
- Generate code via NL

Source: https://githubengineering.com/towards-natural-language-semantic-code-search/
Problem Overview: Robot Navigation

- Communicating with robots using NL
- Conversion of instruction to DSL
- Context-dependent instructions

**Previous instruction:**
Go to the tree on the right

**Previous interpretation:**
$$\lambda a. \text{move}(a) \land \lambda x. \text{tree}(x) \land \text{right-of}(x, \text{rock}) \land \text{to}(a, x)$$

**Current instruction:**
Go to the other tree

**Current interpretation:**
$$\lambda a. \text{move}(a) \land \lambda x. \text{tree}(x) \land \neg \text{right-of}(x, \text{rock}) \land \text{to}(a, x)$$

Traditional vs Neural Parsing

**Traditional**
- Manual grammar+lexicon creation
- Deterministic or probabilistic parsing
- Highly accurate parsing
- Restricted domain

**Neural**
- Parsing as sequence-to-sequence generation problem
- Data-driven
- Robust, scalable
- Margin of error
Problem Overview

Why is it difficult?

Precision vs. robustness

SHRDLU  CHAT-80

Brittle, narrow coverage

Fuzzy, partial understanding

Precise, complete understanding

Robust, broad coverage

Objectives and Progress
Model Objectives

- NL2Regex
- Study, implement different parsing techniques
- Beat the state-of-the-art model

Data Objectives

- Study existing datasets
- Identify failure points in datasets
- Analyze efficiency of data collection techniques
Model Objectives
NL-RX dataset (Locascio et.al 2016)

- 10,000 pairs of NL descriptions and regex
- Grammar-based generation + paraphrasing

Lines start with number and contains the string “dog”  **Paraphrase**  Lines which start with a number and contain the string “dog” in it.
Semantic Parsing Models: Current SOTA

- Seq2Seq w/ attention

**Advantages:**
- Quick to train
- Robust to variation

**Disadvantages:**
- No structural integrity in logical form
- Unable to handle large nesting in logical forms

Source: [Language to Logical Form with Neural Attention (Dong and Lapata 2016)](http://example.com)
Semantic Parsing Models: Coarse2Fine

Two stages of encoding-decoding:
1. NL is encoded, sketch is decoded
2. Sketch is encoded, logical form is decoded

Advantages:
- Structure is encoded, guides decoding throughout
- Work of encoding-decoding is divided

Disadvantages:
- May still result in syntax errors

Source: Coarse-to-Fine Decoding for Neural Semantic Parsing (Dong and Lapata 2018)
Semantic Parsing Models: Seq2Tree

- Tree decoder instead of sequence decoder

**Advantages:**
- Leverages tree/nested nature of code during decoding

**Disadvantages:**
- Structure is not encoded explicitly, does not guide the decoding

Source: [Language to Logical Form with Neural Attention (Dong and Lapata 2016)](http://example.com)
Semantic Parsing Models: Abstract Syntax Networks

- Recursive calls of decoding modules

**Advantages:**
- Leverages recursive nature of general programs
- Output is always syntactically correct

**Disadvantages:**
- Lack of effective encoding of NL
- Not generalizable to all semantic parsing problems

Source: Abstract Syntax Networks for Code Generation and Semantic Parsing (Rabinovich et al. 2017)
Semantic Parsing Models: Multi-Task Learning Models

- Joint training of multiple tasks
- Common loss function

**Advantages:**
- Learns more informed representation of NL
- Encoding of NL is more advanced

**Disadvantages:**
- No structural integrity of decoding
<table>
<thead>
<tr>
<th>Model</th>
<th>Exact Matching Accuracy</th>
<th>DFA-Equals Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (Seq2Seq + Copy)</td>
<td>38.96%</td>
<td>55.24%</td>
</tr>
<tr>
<td>Current SOTA</td>
<td>38.6%</td>
<td>58.2%</td>
</tr>
<tr>
<td>Coarse2Fine</td>
<td>42.52%</td>
<td>59.68%</td>
</tr>
<tr>
<td>Multi-task Network (MQAN)</td>
<td><strong>44.96%</strong></td>
<td><strong>61.92%</strong></td>
</tr>
</tbody>
</table>
1. Incorrect Paraphrasing

Synth: lines having either a lower-case letter, the string “dog”, or a number before a capital letter
Paraphrase: lines containing a lower-case letter and the word dog, followed by a number, then a capital letter

Correct regex: ( ( [a-z] ) | (dog) | ([0-9]) ) .* ( [A-Z] ) *
Predicted regex: ( ( [a-z] ) & (dog) ) .* ( [0-9] .* [A-Z] .* ) *

2. Transferred ambiguity

Synth: lines with the string “dog” before the string “truck” or the string “ring”, 6 or more times
Paraphrase: lines with string “dog” before string “truck” or string “ring”, 6 or more times

Correct regex: ( ( (dog) .* (truck) .* ) | (ring) ) {6,}
Predicted regex: ( (dog) .* (truck) .* ) | ( (ring) {6,} )

3. Large syntactic variation

Synth: lines containing a character and a lower-case letter
Paraphrase: a character and a lower cased letter is required of lines

Correct regex: .* ( . ) & ( [a-z] ) *
Predicted regex: ( ( . ) & ( [a-z] ) ) .* ( [0-9] ) *
Data Objectives
Existing Datasets: NL2Program Datasets

1. Hearthstone

   class DireWolfAlpha(MinionCard):
   def __init__(self):
       super().__init__(
           "Dire Wolf Alpha", 2, CHARACTER_CLASS.ALL,
           CARD_RARITY.COMMON, minion_type=MINION_TYPE.BEAST)

def create_minion(self, player):
    return Minion(2, 2, auras=[
        Aura(ChangeAttack(1), MinionSelector(Adjacent()))])

2. NL2Bash

   find . -type f | sort -nk 5,5 | tail -5
du -a . | sort -rh | head -n5

3. Django

   localedir = os.path.join(app_config.path, 'locale')

4. CoNaLa

   How can I send a signal from a python program?

   os.kill(os.getpid(), signal.SIGUSR1)
<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>NL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATIS</td>
<td>Airline Booking</td>
<td>What flights from any city land at airport_code0?</td>
</tr>
<tr>
<td>GeoQuery</td>
<td>US Geography</td>
<td>could you tell me what is the highest point in the state of Utah?</td>
</tr>
<tr>
<td>WikiSQL</td>
<td>Various (e.g. Movies, Sports, History)</td>
<td>Srdjan Dragojevic worked on a film which earned what nomination?</td>
</tr>
<tr>
<td>Spider</td>
<td>Various (e.g. Games, Class schedules, U.S. government)</td>
<td>For every student who is registered for some course, how many courses are they registered for?</td>
</tr>
</tbody>
</table>
Existing Datasets: Sequential, Context-Dependent Datasets

**Navi**

Instructions:
- Place your back against the wall of the T intersection
- Turn left
- Go forward along the pink flowered carpet hall two segments to the intersection with the brick hall

**SCONE**

- Empty out the leftmost beaker of purple chemical
- Then, add the contents of the first beaker to the second
- Mix it
- Then, drain 1 unit from it
- Same for 1 more unit
Factors to determine data quality

Natural Language

- **NL Variation**
  - Lexical
  - Phrasal
  - Syntactic

- **NL Quality**: Grammatical errors, mispellings, etc.

- **Level of Anaphora**

- **Domain span**

Logical Forms

- **LF Variation**: Coverage
- **LF Complexity**: Nesting (depth)
- **LF Consistency**: Dense distribution of LFs
- **LF Quality**: Syntactic and semantic accuracy
## Some Qualitative Observations

<table>
<thead>
<tr>
<th>Dataset</th>
<th>NL variation</th>
<th>NL Quality</th>
<th>Level of Anaphora</th>
<th>LF variation</th>
<th>LF complexity</th>
<th>LF consistency</th>
<th>LF Quality</th>
<th>Domain Span</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL2Regex</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Django</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>WikiSQL</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Spider</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Scone</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Quantitative Analysis Metrics

**Natural Language**

1. Size of vocabulary
2. Av. length of datapoint
3. Level of anaphora
4. N-gram variation
5. Zipf distribution of words

**Logical Forms**

1. Av. number of nodes in AST (Gen. purpose programs only)
2. Av. number of operators/operands
3. “N-gram variation”
Some Quantitative Results: Level of Anaphora

**Level of anaphora**: % of datapoints where anaphora was detected.

**Inferences**:

1. Anaphora resolution should be *explicitly* modeled into parsers for sequential parsing problems.

2. It is *not the prime focus* for other types of problems.
Some Quantitative Results: Zipf distribution of words

Zipf Absolute Slope:

Slope of plot of log(freq. of word) vs. log(rank of word)

The closer to 1, the better the frequency distribution of words

Inferences:

1. Good datasets have **high** Zipf slope (Spider, Conala, Hearthstone)
2. NL2Regex has **poor** distribution
3. Seq+Context-dep datasets **don’t focus** on accurate distribution
Data Cleaning

- How to abstract away the task and logical form complexity from NL variation?

- Three step cleaning:
  - Replace named entities with `<NE>`
  - Replace words which are not common words and have frequency > 3 with `<KW>`
  - Replace those with frequency < 3 with `<NE>`

Django NL query:

```python
call the function _create_cache with argument alias
```

NL2Regex NL query:

```regex
lines with the string 'dog' at least 2 times
```
Some Quantitative Results: N-gram NL variation

3-gram NL variation:

1. Sort unique 3-grams in descending order of frequency
2. Take top 20% of this list, and find % of datapoints which contain these 3-grams
3. The higher the %, the less variation there is.

Inferences:
1. Django dataset and NL2Regex datasets comparable.
2. Spider maintains variation level, whereas variation of WikiSQL is lesser than expected
Some Quantitative Results: 3-gram NL variation vs. 3-gram Code Variation

Without cleaning

With cleaning
Devised a generalized set of methods used for data collection.

**LF-phase collection**

Inputs:
1. Web/Internet
2. Grammar/Lexicon
3. World State

Process:
1. Scrape
2. Generative Model

**NL-phase Collection**

Inputs:
1. NL description
2. LF description
3. World description

Process:
1. Generate
2. Extract
3. Paraphrase
### LF-phase Classification

<table>
<thead>
<tr>
<th>Input</th>
<th>Scrape</th>
<th>Generative Model</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web/Internet</td>
<td>CodeNN, Conala, Geoquery, Hearthstone, IFTTT, NL2Bash</td>
<td>WebQuestions</td>
<td></td>
</tr>
<tr>
<td>Grammar+Lexicon</td>
<td>Invalid</td>
<td>NL2Regex, Overnight, WikiSQL</td>
<td></td>
</tr>
<tr>
<td>World State</td>
<td>Invalid</td>
<td>SCONE</td>
<td>ATIS, Spider</td>
</tr>
</tbody>
</table>

### NL-phase Classification

<table>
<thead>
<tr>
<th>Input</th>
<th>Generate</th>
<th>Extract</th>
<th>Paraphrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL description</td>
<td>WebQuestions</td>
<td>CodeNN, Conala, Geoquery, Hearthstone, NL2Bash, IFTTT</td>
<td>NL2Regex, WikiSQL, Overnight</td>
</tr>
<tr>
<td>LF description</td>
<td>Conala, NL2Bash</td>
<td>Invalid</td>
<td></td>
</tr>
<tr>
<td>World description</td>
<td>ATIS, SCONE, Spider</td>
<td>Invalid</td>
<td>Invalid</td>
</tr>
</tbody>
</table>
Future Work

1. Collect small regex datasets with different methods
2. Analyze the data and determine efficient data collection methods and strategies.
3. Measure code complexity with advanced measures such as:
   a. Halstead complexity
   b. Cyclometric complexity
Thank you