CODE ANALYSES USING NATURAL LANGUAGE QUESTIONS

Software Composition Seminar
Student: Michael Zbinden
Supervisor: Pooja Rani

29. May 2018
Problem

- Huge code base
- Finding classes

Example: *What are the deprecated classes?*

```java
/* Represents the Singleton object AddressBook */
public final class AddressBook {
    private static AddressBook book;
    private List<IAddress> entries;

    /* Load a CSV-String into the AddressBook */
    public void load(String csv) {
        entries.addAll(CsvParser.csv2iAddress(csv));
    }
}
```

```java
AccessibleResourceBundle ar = new AccessibleResourceBundle();
Date date = new Date(year: 2018, month: 3, date: 14);
```
Tools Support

- IntelliJ Search
Tools Support

- Eclipse Search
Moose
Visualize code in Moose

```plaintext
| view |
| view := RTMondrian new. |
| view shape circle |
| if: [:each | each isAnnotatedWith: 'Deprecated' ] |
| color: Color red. |
| view nodes: (self, (self flatCollect: #clientTypes)) asSet. |
| view edges connectFromAll: #clientTypes. |
| view layout force. |
| view view pushBackEdges. |
```

![Graph visualization](image)
Software Representation
Challenges

- Learn about tool
- Know the meta-model
- Learn query language
- Barrier
Solution

• Smalltalk Code

```smalltalk
self allModelClasses select:
    [ :each | each isAnnotatedWith:
        'Deprecated' ]
```

• Use Natural Language

What are the classes annotated with “deprecated”?
Generate Code

- Python experiment

A Syntactic Neural Model for General-Purpose Code Generation

Pengcheng Yin  
Language Technologies Institute  
Carnegie Mellon University  
pcyin@cs.cmu.edu

Graham Neubig  
Language Technologies Institute  
Carnegie Mellon University  
gneubig@cs.cmu.edu

Abstract

We consider the problem of parsing natural language descriptions into source code written in a general-purpose programming language like Python. Existing data-driven methods treat this problem as a language generation task without considering the underlying syntax of the target programming language. Informed by previous work in semantic parsing, in this paper we propose a novel neural architecture powered by a grammar model to explicitly capture the target syntax as prior knowledge. Experiments find this an effective way to scale up to generation of complex programs from natural language descriptions, achieving state-of-the-art results that well outperform previous code generation and semantic parsing approaches.

1 Introduction

Every programmer has experienced the situation where they know what they want to do, but do not have the ability to turn it into a concrete implementation. For example, a Python programmer

In parallel, the NLP community has developed methods for data-driven semantic parsing, which attempt to map NL to structured logical forms executable by computers. These logical forms can be general-purpose meaning representations (Clark and Curran, 2007; Banerescu et al., 2013), formalisms for querying knowledge bases (Tang and Mooney, 2001; Zettlemoyer and Collins, 2005; Berant et al., 2013) and instructions for robots or personal assistants (Artzi and Zettlemoyer, 2013; Quirk et al., 2015), among others. While these methods have the advantage of being learnable from data, compared to the programming languages (PLs) in use by programmers, the domain-specific languages targeted by these works have a schema and syntax that is relatively simple.

Recently, Ling et al. (2016) have proposed a data-driven code generation method for high-level, general-purpose PLs like Python and Java. This work treats code generation as a sequence-to-sequence modeling problem, and introduce methods to generate words from character-level models, and copy variable names from input descriptions. However, unlike most work in semantic parsing, it does not consider the fact that code has to be well-defined programs in the target syntax.
About Paper

- Translation problem
- Accuracy
- Neural network model
- Model adoption

<table>
<thead>
<tr>
<th></th>
<th>HS</th>
<th>DJANGO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieval System†</td>
<td>0.0</td>
<td>14.7</td>
</tr>
<tr>
<td>Phrasal Statistical MT†</td>
<td>0.0</td>
<td>31.5</td>
</tr>
<tr>
<td>Hierarchical Statistical MT†</td>
<td>0.0</td>
<td>9.5</td>
</tr>
<tr>
<td>NMT</td>
<td>1.5</td>
<td>60.4</td>
</tr>
<tr>
<td>SEQ2TREE</td>
<td>1.5</td>
<td>53.4</td>
</tr>
<tr>
<td>SEQ2TREE–UNK</td>
<td>13.6</td>
<td>62.8</td>
</tr>
<tr>
<td>LPN†</td>
<td>4.5</td>
<td>65.6</td>
</tr>
<tr>
<td>Our system</td>
<td>16.2</td>
<td>75.8</td>
</tr>
</tbody>
</table>

Ablation Study
- frontier embed. 16.7
- parent feed. 10.6
- copy terminals 3.0
+ unary closure
- unary closure 10.1
Tools used

• Framework: Theano

• Library: NLTK

• AST: astor

• Language: Python
**Paper**

**Input:** sort my_list in descending order

**Code:** `sorted(my_list, reverse=True)`
Challenges in Running Model

- Complex neural network
- Lots of components
- Running Code
- Dataset available for python
Findings

- Difficult to prepare dataset
- Creating neural network
- Neural Network libraries
- NLP library
Learning challenges

• Learning about moose

• Creating a simple query

• Analyzing complex neural network

• Python Dependency
Future work

• Prepare dataset

• Decoupling Code

• Adapt classes for other datasets