NL2Bash: A Corpus and Semantic Parser for Natural Language Interface to the Linux Operating System

find system log files older than a month

find / -name "*.log" -mtime +30

Victoria Lin†  Chenglong Wang†  Luke Zettlemoyer†
Michael D. Ernst†
{xilin,clwang,lsz,mernst}@cs.washington.edu
OUTLINE

Problem Definition

natural language → machine commands
Problem Definition

- natural language
- machine commands

Convenient vs. powerful
OUTLINE

Problem Definition

natural language

machine commands

convenient

powerful

Domain
OUTLINE

Problem Definition

natural language \rightarrow machine commands

convenient \rightarrow powerful

Domain

Adaptions from state-of-the-art neural machine translation models

Data-Driven Approaches
Problem Definition

natural language \rightarrow machine commands

convenient \rightarrow powerful

Corpus Construction

Domain

Data-Driven Approaches

Adaptions from state-of-the-art neural machine translation models
OUTLINE

Problem Definition

natural language \text{ --- } \text{machine commands}

Convenient \text{ --- } Powerful

Corpus Construction

Domain

Data-Driven Approaches

Adaptions from state-of-the-art neural machine translation models

System Performance

Qualitative Analysis

Live Demo
PROBLEM DEFINITION

• Natural Language → Command Translation
  - Generating **one-liners**
    - In most command languages complex semantics can be represented in short syntactic forms
    - Other work: code block generation (Polosukhin and Skidanov ’18)
  - **Single-turn interaction** between the user & the system (building block for multi-turn system)
    - Other work: conversational natural language programming assistant (Pandita et. al. ’18)
    - Semantic parsing can be a building block conversational programming assistant
DOMAIN - BASH

• Potentially Wide User Base
  - Most Linux users know bash, but not mastering it

• Command Interface Language

• Generalizable to other command languages
- find all '*.c' files under $HOME directory which contain the string "Salesforce"

```bash
find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "Salesforce" {} 
```
BASH EXAMPLE

- find all '*.c' files under $HOME directory which contain the string “Salesforce"

```bash
find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "Salesforce" {}
```

Head command
BASH EXAMPLE

- find all '*.c' files under $HOME directory which contain the string “Salesforce"

    find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "Salesforce" {}
BASH EXAMPLE

- find all '*.c' files under $HOME directory which contain the string “Salesforce"

```bash
find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "Salesforce" {}
```
- find all '*.c' files under $HOME directory which contain the string "Salesforce"

```
find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "Salesforce" {}
```
RELATED WORK

• Neural Networks: Natural Language → Formal Languages
  ✓ NL → Syntactic parse trees (Vinyals et. al. ’14)
  ✓ NL → Regular expression (Locascio et. al. ’16)
  ✓ NL → Logical forms (Li & Lapata ’16)
  ✓ NL → Python (Wang et. al. ’16)
  ✓ NL → Python (Yin & Neubig ’17, Rabinovich et. al. ’17)

Rule-Based Systems

Statistical Models over Discrete Structures
RELATED WORK

• Neural Networks: Natural Language → Formal Languages
  ✓ NL → Syntactic parse trees (Vinyals et. al. ’14)
  ✓ NL → Regular expression (Locascio et. al. ’16)
  ✓ NL → Logical forms (Li & Lapata ’16)
  ✓ NL → Python (Wang et. al. ’16)
  ✓ NL → Python (Yin & Neubig ’17, Rabinovich et. al. ’17)

Adapted from NMT methods for natural language translation
RELATED WORK

• Neural Networks: Natural Language $\rightarrow$ Formal Languages
  - ✓ NL $\rightarrow$ Syntactic parse trees (Vinyals et. al. ’14)
  - ✓ NL $\rightarrow$ Regular expression (Locascio et. al. ’16)
  - ✓ NL $\rightarrow$ Logical forms (Li & Lapata ’16)
  - ✓ NL $\rightarrow$ Python (Wang et. al. ’16)
  - ✓ NL $\rightarrow$ Python (Yin & Neubig ’17, Rabinovich et. al. ’17)

Expressive $\longrightarrow$ Simplest Data Representation
RELATED WORK

• Neural Networks: Natural Language → Formal Languages
  ✓ NL → Syntactic parse trees (Vinyals et. al. ’14)
  ✓ NL → Regular expression (Locascio et. al. ’16)
  ✓ NL → Logical forms (Li & Lapata ’16)
  ✓ NL → Python (Wang et. al. ’16)

Target Domain: Shallow Syntax Structure (No Formal Grammar)
SEQUENCE-TO-SEQUENCE NEURAL NETWORK

Encoder RNN

Decoder RNN

find [fname] files in current folder

<start>
SEQ2SEQ + COPYING

- find all '*.c' files under $HOME directory whose content has the string "salesforce"

find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "salesforce" {}

✗ Large number of out-of-vocabulary words (arguments)
SEQ2SEQ + COPYING

- find all '*.c' files under $HOME directory whose content has the string "salesforce"

```
find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "salesforce" {}
```

✗ Large number of out-of-vocabulary words

Incorporating Copying Mechanism in Sequence-to-Sequence Learning, Gu et. al. EMNLP 2016
SEQ2SEQ + COPYING

- find all '*.c' files under $HOME directory whose content has the string “salesforce"

```
find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "salesforce" {}
```

✗ Many command arguments are source tokens transformed through atomic string edits

Incorporating Copying Mechanism in Sequence-to-Sequence Learning, Gu et. al. EMNLP 2016
SEQ2SEQ + COPYING

- find all '*.c' files under $HOME directory whose content has the string "salesforce"

find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "salesforce" {}

✗ Many command arguments are source tokens transformed through atomic string edits

Character models? Very long sequences…
find all ` *. c ' files under ` $ HOME ` directory whose content has the string " salesforce "

```
find " $ HOME " -name " * . c " -print0 | xargs -0 -I {} grep " salesforce " {}
```

Split the constant tokens in both the source and target sequences into a sequence of sub-tokens consists of the following:

1. Consecutive sub-sequences of alphabetical letters
2. Consecutive sub-sequences of digits
3. All other special tokens

Run CopyNet on the sub-tokens
SUB-TOKEN COPYING

- find all ‘* . c’ files under $HOME directory whose content has the string “salesforce”

```
find " $HOME " -name " * . c " -print0 | xargs -0 -I {} grep " salesforce " {}
```

Enables learning of

1. Substring addition
2. Substring deletion
3. Substring replacement
4. Semantics of the special characters such as “$”, quotation marks, “*”, etc.
DATA COLLECTION

- Bash programmers hired upwork™

- Collect bash commands and their natural language descriptions from the web

✓ web interface to control the collection process
## BASH COMMAND FILTERING

- **Bash Command**

<table>
<thead>
<tr>
<th>In-scope</th>
<th>Single command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical connectives</td>
<td>&amp;&amp;,</td>
</tr>
<tr>
<td>Nested command</td>
<td>pipeline</td>
</tr>
<tr>
<td></td>
<td>command substitution $()</td>
</tr>
<tr>
<td></td>
<td>process substitution &lt;()</td>
</tr>
<tr>
<td>I/O redirection</td>
<td>&lt;, &lt;&lt;</td>
</tr>
<tr>
<td>Variable assignment</td>
<td>=</td>
</tr>
<tr>
<td>Parameters</td>
<td>e.g. $1, $HOME</td>
</tr>
<tr>
<td>Multi-statement</td>
<td>if, for, while, until, etc.</td>
</tr>
<tr>
<td>Regex structure</td>
<td>e.g. x<em>y</em></td>
</tr>
<tr>
<td>Non-bash programs</td>
<td>triggered by awk, java, etc.</td>
</tr>
</tbody>
</table>

- **Out-of-scope**

- **Non-bash programs**
  - triggered by awk, java, etc.
DATA STATISTICS

• 12,609 pairs —> 9,301 pairs after filtering
• 8,090 train, 609 dev, 606 test
• 100+ unique bash commands, 537 unique flags
TOP-50 COMMAND HISTOGRAM

The rest combined: 984
EVALUATION METHODOLOGY

• Manual Evaluation (Multiple Correct Solutions)

  - 3 bash programmers (hired via upwork™) judged the top-3 suggestions of each test example
    ‣ Full command correctness
    ‣ Command template correctness

find [path] -name [regex] -print0 | xargs -0 -I {} grep [regex] {}

  - Final judgement: majority vote

  - Inter-annotator agreement: 0.89, 0.83, 0.80
BASILINES

- Vanilla Seq2Seq (Sutskever et. al. ‘14)
- CopyNet (Gu et. al. ‘17)
- Three-stage translation model (Lin et. al. ‘17)
  1. Convert both NL and bash command to templates
  2. Apply Seq2Seq translation on the templates
  3. Fill arguments using heuristics
SYSTEM PERFORMANCE (Dev Set)

- **Seq2Seq**
- **CopyNet**
- **Sub-token CopyNet**
- **Stage-wise**

Acc-F-1
Acc-F-3
Acc-T-1
Acc-T-3
SYSTEM PERFORMANCE (Dev Set)

- Acc-F-1
- Acc-F-3
- Acc-T-1
- Acc-T-3

- Seq2Seq
- CopyNet
- Sub-token CopyNet
- Stage-wise
SYSTEM PERFORMANCE (Dev Set)

Acc-F-1
Acc-F-3
Acc-T-1
Acc-T-3

Seq2Seq
CopyNet
Sub-token CopyNet
Stage-wise
Sub-token CopyNet has the best full command translation accuracy.
Sub-token CopyNet has the best full command translation accuracy.
QUALITATIVE ANALYSIS

• Live Demo: http://tellina.rocks

• Split ‘/usr/bin/gcc’ into 10 files of about equal size

• Which files in the computer were modified more than 30 days ago and larger than 500M

• Find all *company* (case-insensitive) files/directories under /basedir with null character as the delimiter
Corpus: 10k real-world bash commands, paired with human-written English descriptions

Data-driven baselines: motivated by SOTA neural machine translation approaches copying, sub-token modeling

Huge space for improvements

To appear in LREC 2018 conference proceedings

Contact: xilin@salesforce.com
BKI - SEQ2SEQ OUTPUT PROBABILITY

Generation Probability

|target vocabulary|
BKII - COPYNET OUTPUT PROBABILITY

Generation Probability

Copy Probability

|target vocabulary| |source sequence|
BKII - COPYNET (Gu et. al. 2016)

Generation Probability

Copy Probability

|target vocabulary| |source sequence|

\[ p(y_t|s_t, y_{t-1}, c_t, M) = p(y_t, g|s_t, y_{t-1}, c_t, M) + p(y_t, c|s_t, y_{t-1}, c_t, M) \]

“hidden state”  “copying context”
**BKIV- COPYNET (Gu et. al. 2016)**

**Generation Logit**

$$\text{softmax}()$$

|target vocabulary|

$$p(y_t, g|\cdot) = \begin{cases} 
\frac{1}{Z} e^{\psi_g(y_t)}, & y_t \in \mathcal{V} \\
0, & y_t \in \mathcal{X} \cap \overline{\mathcal{V}} \\
\frac{1}{Z} e^{\psi_g(\text{UNK})}, & y_t \not\in \mathcal{V} \cup \mathcal{X} 
\end{cases}$$

**Copy Logit**

|source sequence|

$$p(y_t, c|\cdot) = \begin{cases} 
\frac{1}{Z} \sum_{j:x_j=y_t} e^{\psi_c(x_j)}, & y_t \in \mathcal{X} \\
0, & \text{otherwise}
\end{cases}$$
BKV - SPEED-UP EXPERT SOURCING

Figure 2. Data Collection Interface Screenshot
natural language input:
find all log files older than 15 days

Stage 1: rule-based open-vocabulary entity recognition

entity mentions: {filename: “log”, timespan: “15 days”}
natural language template: find all [filename] files older than [timespan]

Stage 2: NL template to program template translation

synthesized program templates:
find . -name “*.log” -mtime +15d

Stage 3: Argument filling and post-processing

find . -type f -name “*.log” -mtime +15d

...