Determining API Usage Policies from Method Documentation

Tej Chajed, Jimmy Koppel
6.864 Fall 2015

Introduction: API constraints

- API methods have constraints (e.g., number of arguments, positive values)
- Documentation conveys constraints in natural language

```
/**
 * Adds the specified component to the layout, using the
 * specified constraint object.  For border layouts, the
 * constraint must be one of the following constants:
 * NORTH, SOUTH, EAST, WEST, or CENTER.
 * ...
 */
public void addLayoutComponent(Component comp,
        Object constraints) {
}
```

- **Learn these constraints**

Approach: static analysis as ground truth

- Run a **static analysis** to get properties about method usages
- Call those results typical (assume usages correct)
- Learn to predict the results from documentation
- **Predict analysis result** for unseen usages from documentation

Application: constantness of arguments

Should a parameter be a statically known value (**constant**), or can it be a dynamic, runtime value (**non-constant**)? (note: has nothing to do with C++ `const` arguments)

- Not a typical constraint
- Interesting restriction: makes sense in some cases (*e.g.*, permissions for `open()`)
- Ground truth static analysis is extremely reliable

Learning samples

For each usage, get:

- documentation for called method
- whether each argument is called with a constant

Aggregate for each (method, arg) pair the **fraction of uses** that had a constant argument

Convert fraction of constant usages to a **classification problem**:

fraction constant	class
0	never constant
0–0.1	probably non-constant
0.1–0.9	maybe constant
0.9–1.0	constant

Data: Java from the MUSE corpus

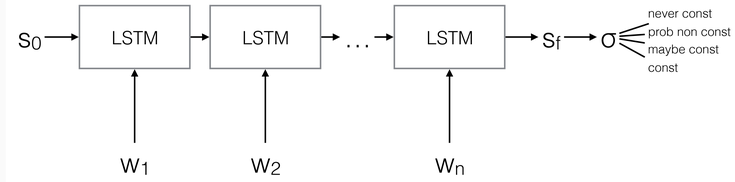
MUSE

- Extracted 36,435 documented API methods with arguments
- Total of 11M usages
- Produced 8,700 training examples (after aggregating within each documented method), which we split:

train	dev	eval
70%	15%	15%

Neural Network model

Learn word embeddings along with LSTM model to capture context in documentation text.



Results

Trained on $\sim$ 6000 examples.

classifier	dev	eval
Baseline (prior distribution) always predicts "never constant"	57%	61%
Maximum entropy with bag-of-words model	62%	67%
Neural Net	61%	61%