Statistical Models of Typed Syntax Trees

Cyrus Omar
Carnegie Mellon University
Many tools could benefit from an understanding of the statistics of natural programs.
Many tools could benefit from an understanding of the statistics of natural programs.
To develop a probability distribution for code, we need to first choose a representation of code.
class Sagan extends Astronomer {
    void beforeBed() {
        observe(favorite)
    }
    Planet favorite = Planet.NEPTUNE;
}

enum Planet {
    MERCURY,
    VENUS,
    ...
    NEPTUNE
}

class Astronomer {
    void observe(Planet p) {...}
}

P(observe, (favorite), ) | (())

- n-grams (Hindle et al., ICSE 2012; Allamanis & Sutton, MSR 2013)
- topic modeling + part of speech analysis (Nguyen et al., FSE 2013)

Previous Work: Programs are Token Sequences
Our Approach: Programs are Syntax Trees
Our Approach: Programs are \textit{Typed Syntax Trees}
Many kinds of tools can benefit from better models of the statistical structure:

- Code compression engines
- Many kinds of tools can benefit from better models of the statistical structure
- Bandwidth (e.g. mobile phones, wearable computers like Google Glass, or predictive interfaces)

To avoid these limitations, we investigate models where probabilistic assignments often assign non-zero probabilities to malformed and ill-typed programs, to program tokens conditioned on their surrounding tokens:

- Previous Work: Token Sequences
- 3-15) preceding tokens
- 3-15) preceding tokens
- Form-Specific Distributions
- Example

\( \mathbb{P}(\phi | \rho, \tau) \)

\( \phi = \text{call}, \rho = \text{stmt}, \tau = \text{void}, \Gamma \)

- Syntactic form
- Role
- Type
- Typing context

Bayes' rule!
Many kinds of tools can benefit from better models of the statistical structure.

SLAMC bandwidth (e.g., mobile phones, wearable computers like Google Glass, or for conveying commands via devices with limited

-gram

SLAMC

• SLAMC

Code completion engines

Example

Automated bug fixing tools

Motivation

SLAMC typed syntax trees

Statistical Modeling of Typed Syntax Trees

Our objective is to learn $n$ types, strings and enums, method calls and variables, respectively.

as an argument to a method or in any other context

e is the set of variables of in scope, and the structural context,

Role ($\phi$, $\rho$, $\tau$, $\Gamma$)

$\phi \rightarrow \text{call, } \rho \rightarrow \text{stmt, } \tau \rightarrow \text{void, } \Gamma \mapsto \mathbf{P}(\phi | \rho, \tau)$

$\mathbf{P}(\text{Sagan.observe} | \rho \rightarrow \text{stmt, } \tau \rightarrow \text{void, } \Gamma)$

$\mathbf{P}(\text{var this} | \rho \rightarrow \text{targ, } \tau \rightarrow \text{Sagan, } \Gamma)$

$\mathbf{P}(\text{field var this.favorite} | \rho \rightarrow \text{arg, } \tau \rightarrow \text{Planet, } \Gamma)$

$\mathbf{P}(\phi | \rho, \tau)$
We are starting to implement this model for Java using the Eclipse JDT for parsing and keeping track of $\Gamma$

http://www.github.com/cyrus-/syzygy
To test our implementation, we perform **10-fold cross-validation** of our model on a corpus of several large open source projects and compare it to the **3-gram model** used in Hindle et. al, 2012.

(collaboration with **Salil Joshi** and **Flavio Cruz**)
We are taking a first-principles approach to source code prediction that combines the foundational techniques of both statistics and semantics.

\[
P(\text{call \ var \ this \ .observe(field \ var \ this \ .favorite)} | \rho = \text{stmt}, \mathcal{T} = \text{void}, \Gamma)
\]

role type typing context