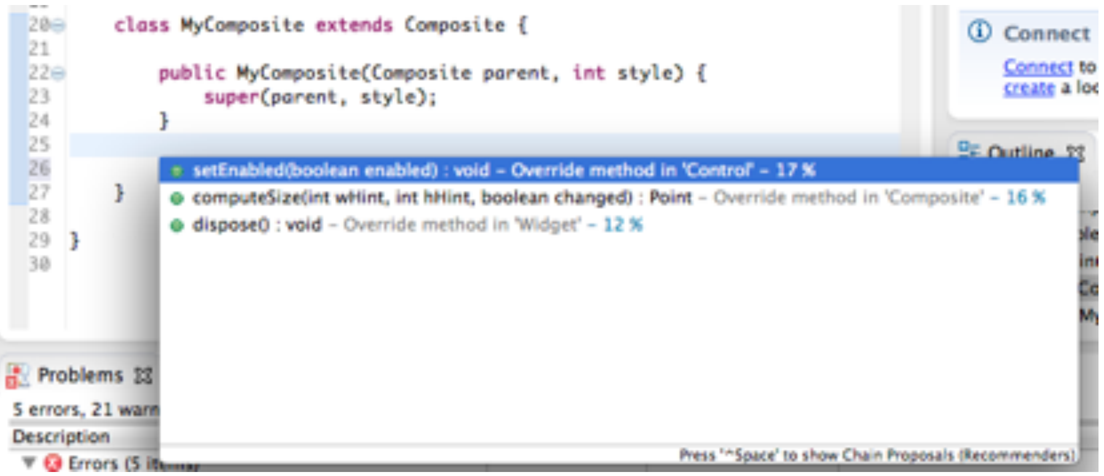
Statistical Models of Typed Syntax Trees

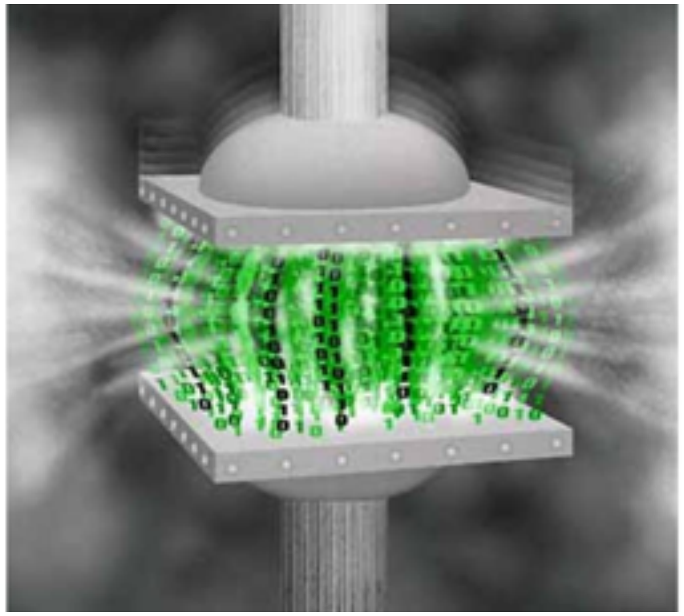
Cyrus Omar
Carnegie Mellon University

Many tools could benefit from an understanding of the statistics of *natural programs*.



code completion engines

predictive programming interfaces



code compression engines

Many tools could benefit from an understanding of the statistics of *natural programs*.

code smell detectors

Running Example

```
enum Planet {
  MERCURY,
  VENUS,
  …,
  NEPTUNE
}

class Astronomer {
  void observe(Planet p) {…}
}
```

```
class Sagan extends Astronomer {
  void beforeBed() {
    observe(favorite)
  }

  Planet favorite = Planet.NEPTUNE;
}
```

To develop a probability distribution for code, we need to first choose a **representation of code**

Running Example

```
enum Planet {
  MERCURY,
  VENUS,
  …,
  NEPTUNE
}

class Astronomer {
  void observe(Planet p) {…}
}
```

```
class Sagan extends Astronomer {
  void beforeBed() {
    observe(favorite)
  }

  Planet favorite = Planet.NEPTUNE;
}
```

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

Running Example

```
enum Planet {
  MERCURY,
  VENUS,
  …,
  NEPTUNE
}

class Astronomer {
  void observe(Planet p) {…}
}
```

```
class Sagan extends Astronomer {
  void beforeBed() {
    observe(favorite)
  }

  Planet favorite = Planet.NEPTUNE;
}
```

P(call var this .observe(field var this .favorite)

| ρ = **stmt**,

role

Our Approach: Programs are Syntax Trees

Running Example

```
enum Planet {
  MERCURY,
  VENUS,
  …,
  NEPTUNE
}

class Astronomer {
  void observe(Planet p) {…}
}
```

```
class Sagan extends Astronomer {
  void beforeBed() {
    observe(favorite)
  }

  Planet favorite = Planet.NEPTUNE;
}
```

P(call [var this] .observe(field [var this] .favorite)

$| \ \rho$ = **stmt**, τ = **void**, Γ)

role — type — typing context

Our Approach: Programs are *Typed* Syntax Trees

Running Example

```
enum Planet {
  MERCURY,
  VENUS,
  …,
  NEPTUNE
}

class Astronomer {
  void observe(Planet p) {…}
}
```

```
class Sagan extends Astronomer {
  void beforeBed() {
    observe(favorite)
  }

  Planet favorite = Planet.NEPTUNE;
}
```

$$= \mathbf{P}(\text{ call } [\text{var } \texttt{this}]\texttt{.observe(}[\text{field } [\text{var } \texttt{this}]\texttt{.favorite}]\texttt{)} \mid \phi = \texttt{call}, \rho = \texttt{stmt}, \tau = \texttt{void}, \Gamma)\,\mathbf{P}(\phi \mid \rho, \tau)$$

syntactic form — role — type — typing context

Bayes' rule!

Running Example

```
enum Planet {
  MERCURY,
  VENUS,
  …,
  NEPTUNE
}

class Astronomer {
  void observe(Planet p) {…}
}
```

```
class Sagan extends Astronomer {
  void beforeBed() {
    observe(favorite)
  }

  Planet favorite = Planet.NEPTUNE;
}
```

$= \mathbf{P}($ call [var this] .observe(field [var this] .favorite) $\mid \phi = \texttt{call}, \rho = \texttt{stmt}, \tau = \texttt{void}, \Gamma) \, \mathbf{P}(\phi \mid \rho, \tau)$

$= \mathbf{P}(\texttt{Sagan.observe} \mid \rho = \texttt{stmt}, \tau = \texttt{void}, \Gamma)$

$\mathbf{P}($ var this $\mid \rho = \texttt{targ}, \tau = \texttt{Sagan}, \Gamma)$

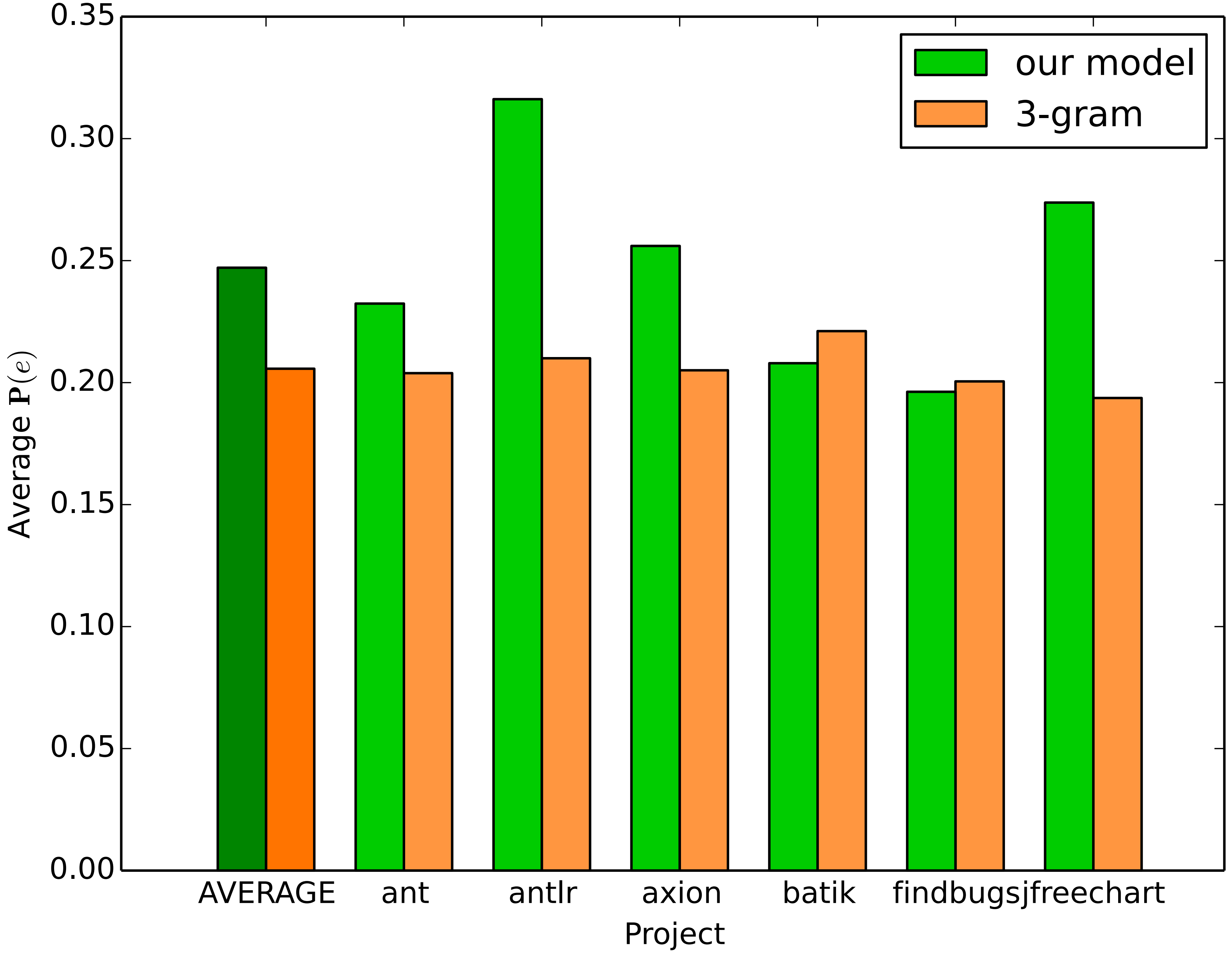
$\mathbf{P}($ field [var this] .favorite $\mid \rho = \texttt{arg}, \tau = \texttt{Planet}, \Gamma)$

$\mathbf{P}(\phi \mid \rho, \tau)$

We are starting to **implement** this model for Java using the **Eclipse JDT** for parsing and keeping track of Γ

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(call var `this` `.observe(` field var `this` `.favorite` `)`

| ρ = **`stmt`**, τ = **`void`**, Γ)

role type typing context