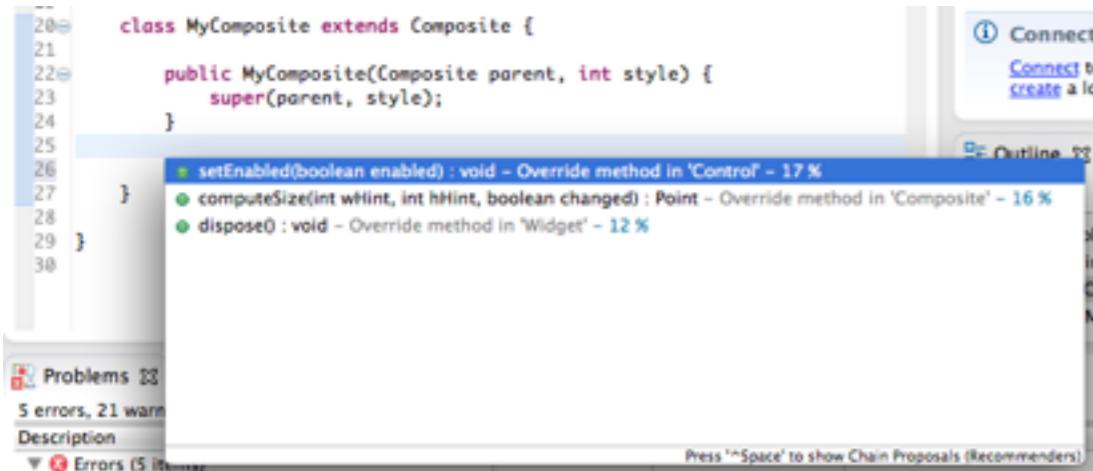


Statistical Models of Typed Syntax Trees

Cyrus Omar

Carnegie Mellon University

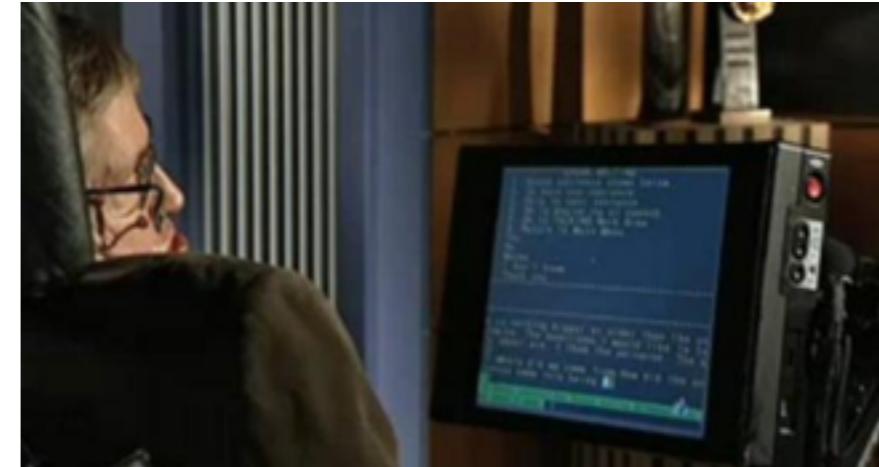
Many tools could benefit from an understanding of the statistics of *natural programs*.



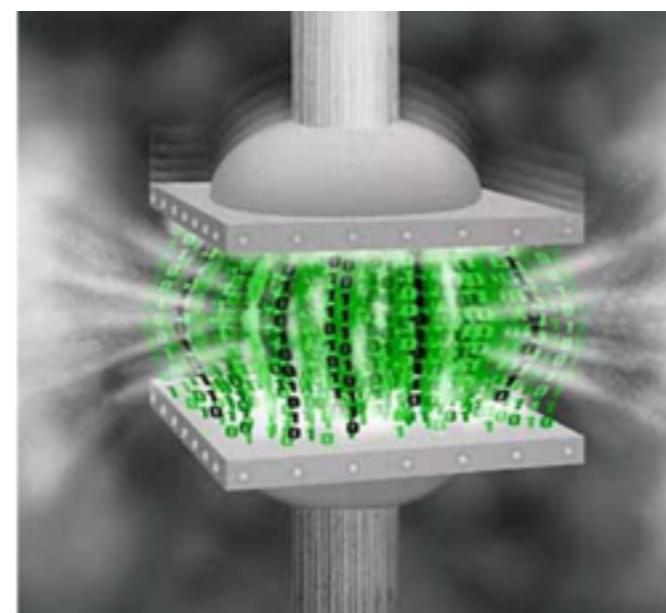
```
20: class MyComposite extends Composite {  
21: 22:     public MyComposite(Composite parent, int style) {  
23:         super(parent, style);  
24:     }  
25: 26:     @Override  
27:     public void setEnabled(boolean enabled) {  
28:         super.setEnabled(enabled);  
29:     }  
30: }  
  
setEnabled(boolean enabled) : void - Override method in 'Control' - 17 %  
computeSize(int wHint, int hHint, boolean changed) : Point - Override method in 'Composite' - 16 %  
dispose0 : void - Override method in 'Widget' - 12 %
```

Problems 23
5 errors, 21 warnings
Description
Errors (5 items)

code completion engines

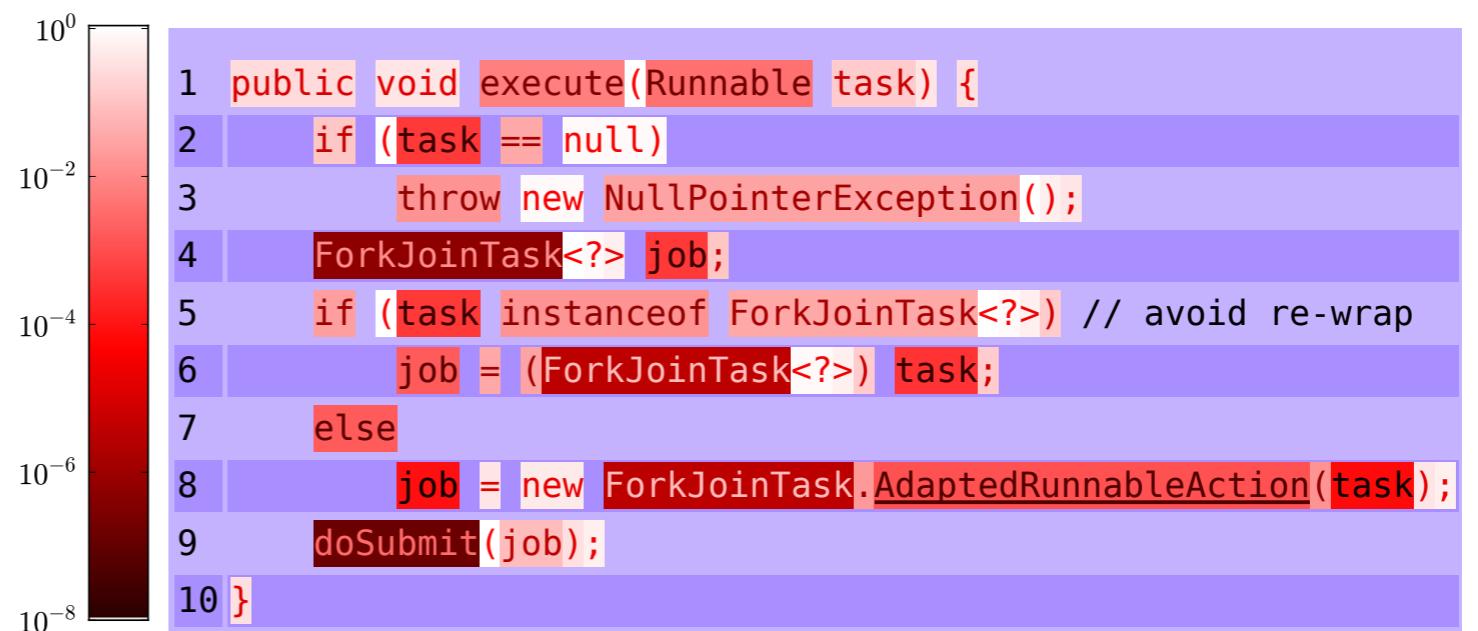


predictive programming interfaces



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;  
}
```

$P(\text{observe} \mid (\text{favorite} \mid (\{ \mid \text{) } \mid \text{) }))$

- ***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)
| $\rho = \text{stmt,}$
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)

| $\rho = \text{stmt}, \tau = \text{void}, \Gamma$)

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;  
}
```

$= P(\boxed{\text{call}} \boxed{\text{var this}} \boxed{\text{.observe}}(\boxed{\text{field}} \boxed{\text{var this}} \boxed{\text{.favorite}}))$

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

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;
}

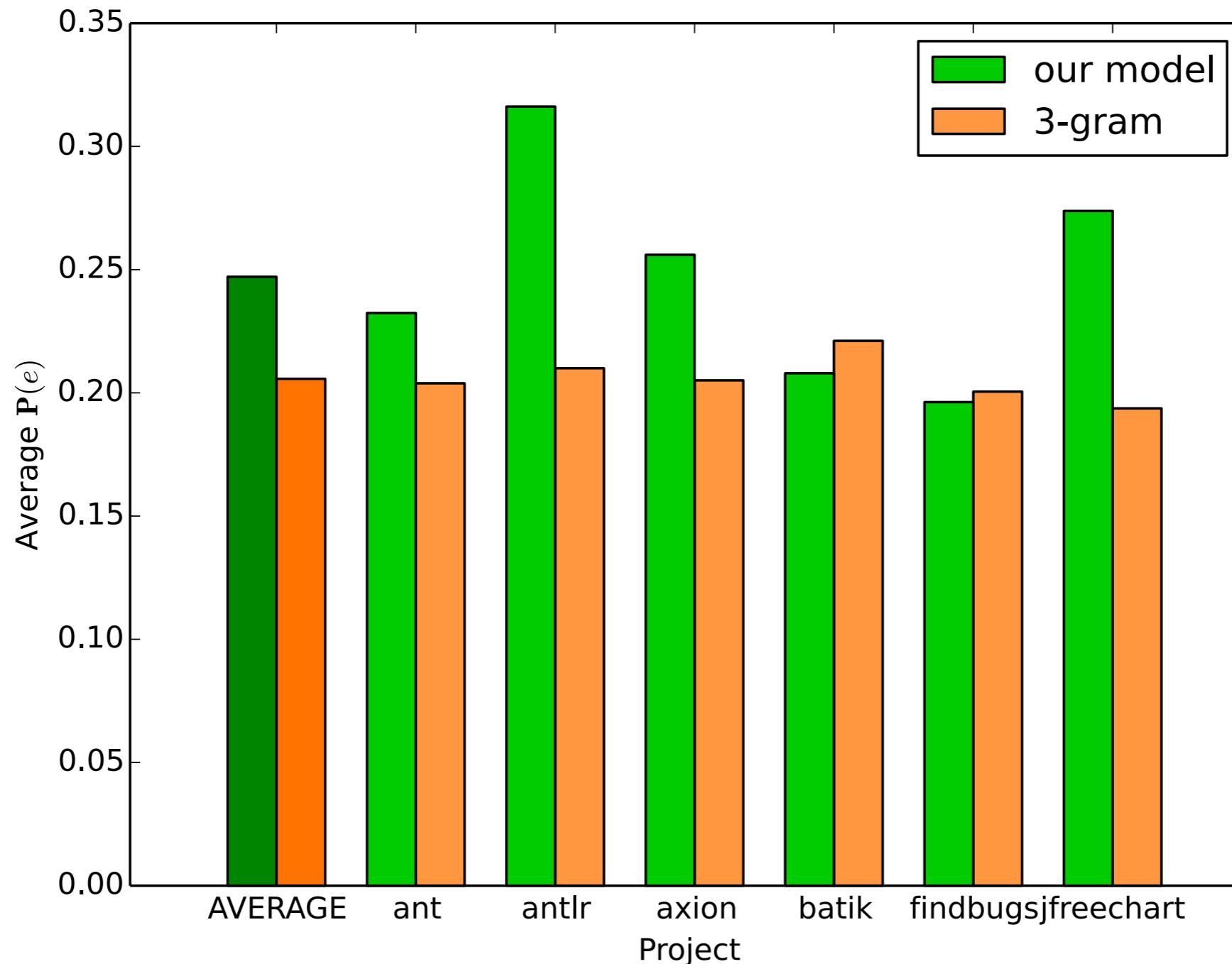
```

$$\begin{aligned}
 &= \mathbf{P}(\boxed{\mathbf{call}} \boxed{\mathbf{var} \; \mathbf{this}} \mathbf{.observe}(\boxed{\mathbf{field}} \boxed{\mathbf{var} \; \mathbf{this}} \mathbf{.favorite})) \\
 &\quad | \phi = \mathbf{call}, \rho = \mathbf{stmt}, \tau = \mathbf{void}, \Gamma) \mathbf{P}(\phi | \rho, \tau) \\
 &= \mathbf{P}(\mathbf{Sagan}.\mathbf{observe} | \rho = \mathbf{stmt}, \tau = \mathbf{void}, \Gamma) \\
 &\quad \mathbf{P}(\boxed{\mathbf{var} \; \mathbf{this}} | \rho = \mathbf{targ}, \tau = \mathbf{Sagan}, \Gamma) \\
 &\quad \mathbf{P}(\boxed{\mathbf{field}} \boxed{\mathbf{var} \; \mathbf{this}} \mathbf{.favorite} | \rho = \mathbf{arg}, \tau = \mathbf{Planet}, \Gamma) \\
 &\quad \mathbf{P}(\phi | \rho, \tau)
 \end{aligned}$$

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**.

$$\mathbf{P}(\text{call } \boxed{\text{var this}} \cdot \text{observe}(\text{field } \boxed{\text{var this}} \cdot \text{favorite}) \mid \rho = \mathbf{stmt}, \tau = \mathbf{void}, \Gamma)$$

role type typing
 context