Learning from “Big Code”

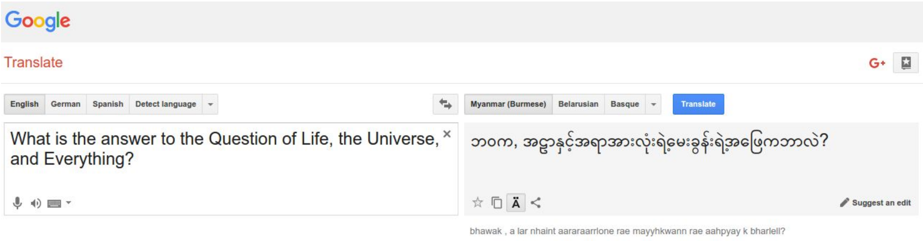
21th PeWe, April 7, 2017

Pavol Bielik
Software Reliability Lab
Department of Computer Science
ETH Zurich

European Research Council
Established by the European Commission
Supporting top researchers from anywhere in the world

Big Data Impact

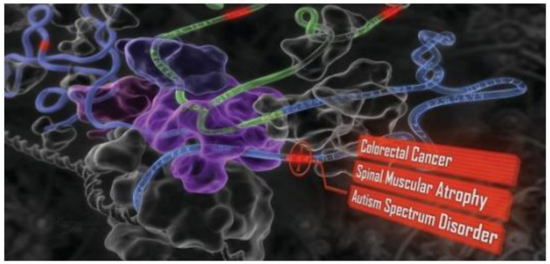
Natural Language Processing
(e.g., machine translation)



Computer Vision
(e.g., image captioning)

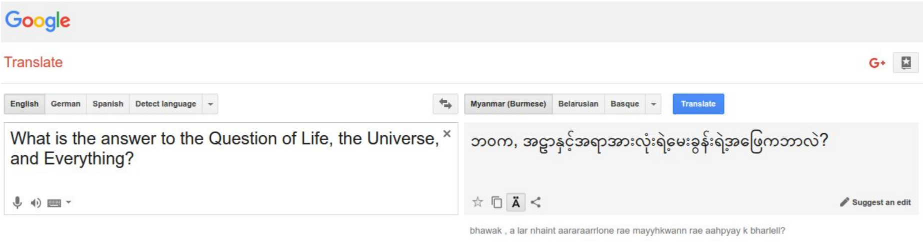
→ A group of people shopping at an outdoor market. There are many vegetables at the fruit stand.

Medical Computing
(e.g., disease prediction)



Big Data Impact

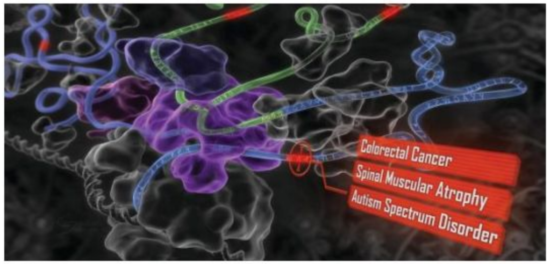
Natural Language Processing
(e.g., machine translation)



Computer Vision
(e.g., image captioning)

→ A group of people shopping at an outdoor market. There are many vegetables at the fruit stand.

Medical Computing
(e.g., disease prediction)



Can we bring this revolution to programmers?

Vision

Create new kinds of software tools that leverage **massive codebases** to solve problems **beyond** what is possible with traditional techniques.

number of repositories

last 8 years

15 million repositories

Billions of lines of code

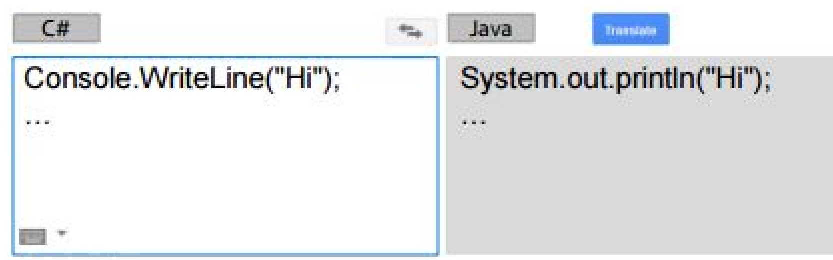
High quality, tested, maintained programs

Statistical Software Tools

Write new code:
Code Completion

```
Camera camera = Camera.open();
camera.SetDisplayOrientation(90);
                ?
```

Port code:
Programming Language Translation



Understand code/security:
JavaScript Deobfuscation
Type Prediction

JS NICE

Debug code:
Statistical Bug Detection

likely error

```
...
for x in range(a):
    print a[x]
```

All of these benefit from the "Big Code" and lead to applications not possible with previous techniques

Statistical Software Tools

Write new code:
Code Completion

```
Camera camera = Camera.open();
camera.SetDisplayOrientation(90);
            ?
```

Port code:
Programming Language Translation

C# Java Translate

```
Console.WriteLine("Hi");
...
```

```
System.out.println("Hi");
...
```

Understand code/security:
JavaScript Deobfuscation
Type Prediction

JS NICE

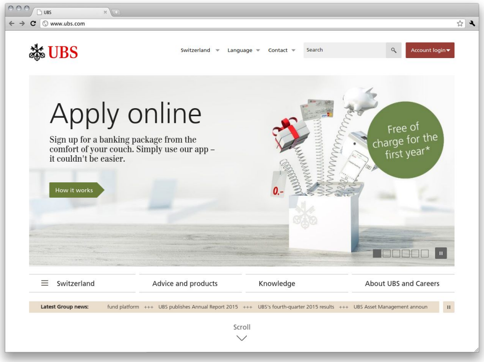
Debug code:
Statistical Bug Detection

likely error

```
...
for x in range(a):
    print a[x]
```

All of these benefit from the "Big Code" and lead to applications not possible with previous techniques

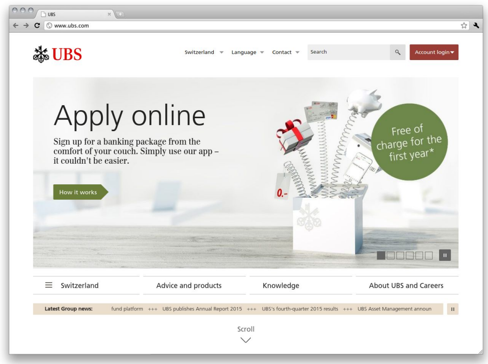
Code understanding/security



+

JavaScript

Code understanding/security

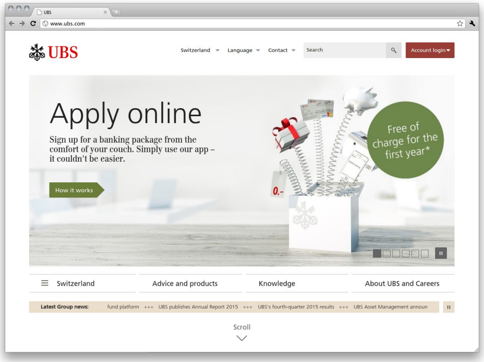


+

JavaScript

`savePassword(user, password)` → **Obfuscation/ Minification** → `B(d, c)`

Code understanding/security

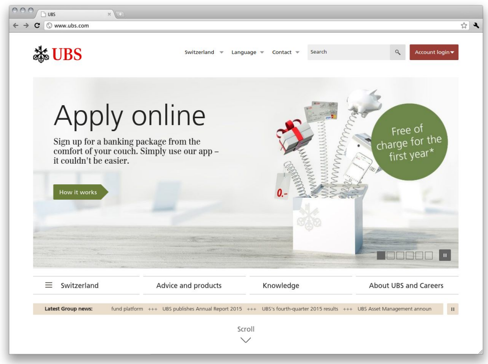


+

iPhone JavaScript android

`savePassword(user, password)` → **Obfuscation/ Minification** → `B(d, c)`

Code understanding/security



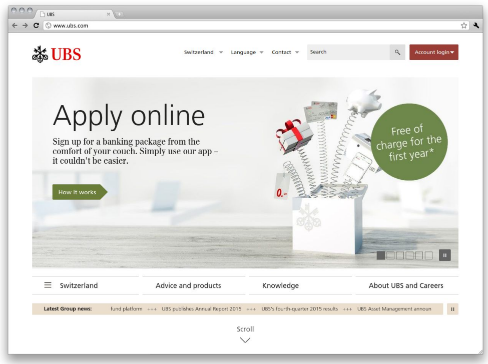
+

iPhone JavaScript android

Security Analyst

`savePassword(user, password)` → **Obfuscation/ Minification** → `B(d, c)`

Code understanding/security



+

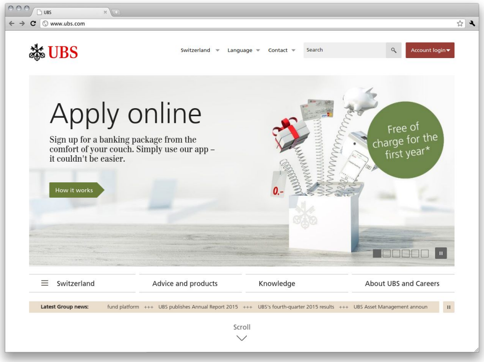
iPhone JavaScript ANDROID

Security Analyst

`savePassword(user, password)` → **Obfuscation/ Minification** → `B(d, c)`

`savePassword(user, password)` ← **?** ← `B(d, c)`

Code understanding/security



+

iPhone JavaScript android

Security Analyst

`savePassword(user, password)` → **Obfuscation/ Minification** → `B(d, c)`

`savePassword(user, password)` ← **JS NICE** ← `B(d, c)`

Demo

available online:

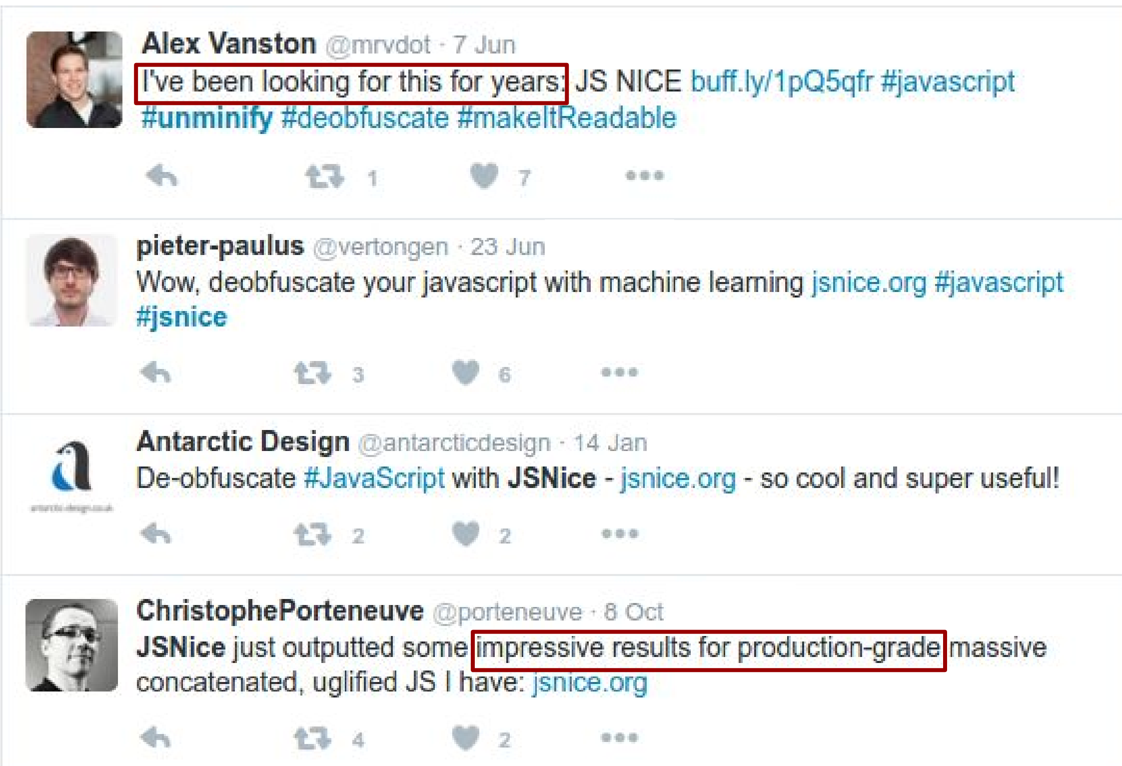
www.jsnice.org

Impact

30, 000 Users in 1st week

Impact

30, 000 Users in 1st week



Alex Vanston @mrvdot · 7 Jun
I've been looking for this for years: JS NICE buff.ly/1pQ5qfr #javascript #unminify #deobfuscate #makeItReadable
1 7

pieter-paulus @vertongen · 23 Jun
Wow, deobfuscate your javascript with machine learning jsnice.org #javascript #jsnice
3 6

Antarctic Design @antarcticdesign · 14 Jan
De-obfuscate #JavaScript with **JSNice** - jsnice.org - so cool and super useful!
2 2

ChristophePorteneuve @porteneuve · 8 Oct
JSNice just outputted some impressive results for production-grade massive concatenated, uglified JS I have: jsnice.org
4 2

Impact

30, 000 Users in 1st week

How to build such tools?

Write new code:
Code Completion

```
Camera camera = Camera.open();
camera.SetDisplayOrientation(90);
                    ?
```

Port code:
Programming Language Translation

C# → Java

```
Console.WriteLine("Hi");
...
```

```
System.out.println("Hi");
...
```

Understand code/security:
JavaScript Deobfuscation
Type Prediction

JS NICE

Debug code:
Statistical Bug Detection

```
...
for x in range(a):
    print a[x]
```

likely error

JS NICE Statistical Type Inference and Renaming

Applications

```
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```

Predicting types | Predicting names

```
// @param {string} url
// @param {Object} client
// @param {string} data
function get(url, client, data) {
    client.open("GET", url, false);
    client.send(data);
}
```

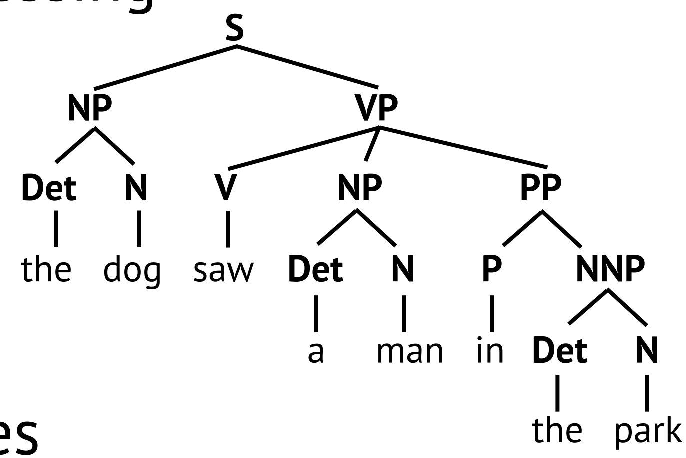
What is a suitable program representation?

Applications

Intermediate Representation

Natural Language Processing

The dog saw a man in the park →



Programming Languages

JavaScript Java →

?

What is a suitable program representation?

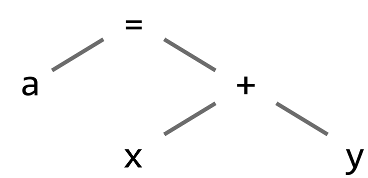
Applications

Intermediate Representation

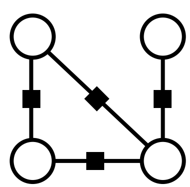
Sequences

```
req    → {<open, 0>, <send, 0>}
source → {..., <open, 2>}
```

Trees



Graphical Models



Feature Vectors

```
req    → (0,0,1,1,0)
source → (1,0,0,0,0)
...
```

What is a suitable program representation?

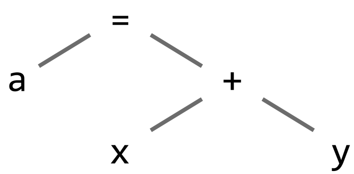
Applications

Intermediate Representation

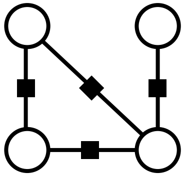
Sequences

```
req    → {<open, 0>, <send, 0>}
source → {..., <open, 2>}
```

Trees



Graphical Models



Feature Vectors

```
req    → (0,0,1,1,0)
source → (1,0,0,0,0)
...
```

JS NICE

Representation

Applications

Intermediate Representation

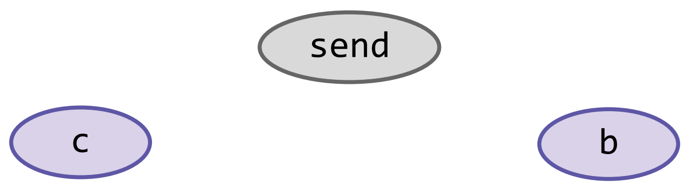
```
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```

JS NICE Representation

Applications

Intermediate Representation

```
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```

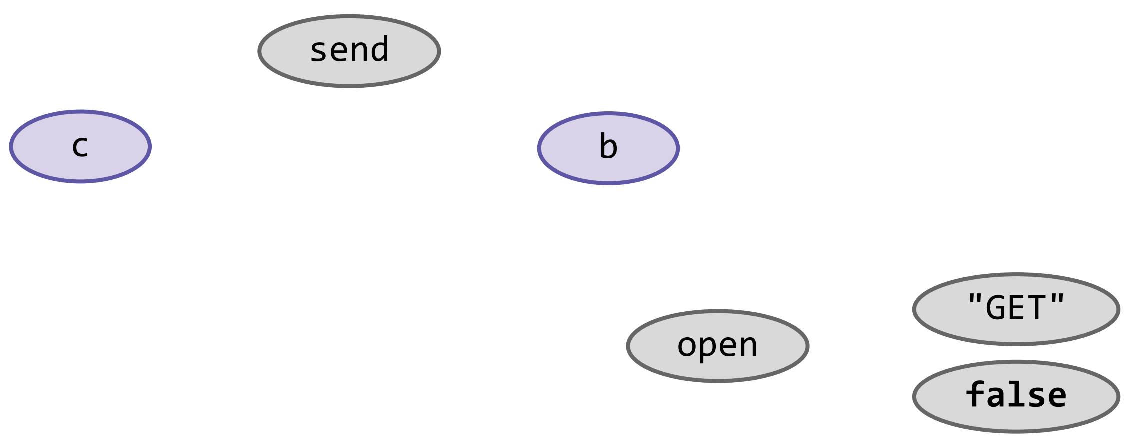


JS NICE Representation

Applications

Intermediate Representation

```
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```

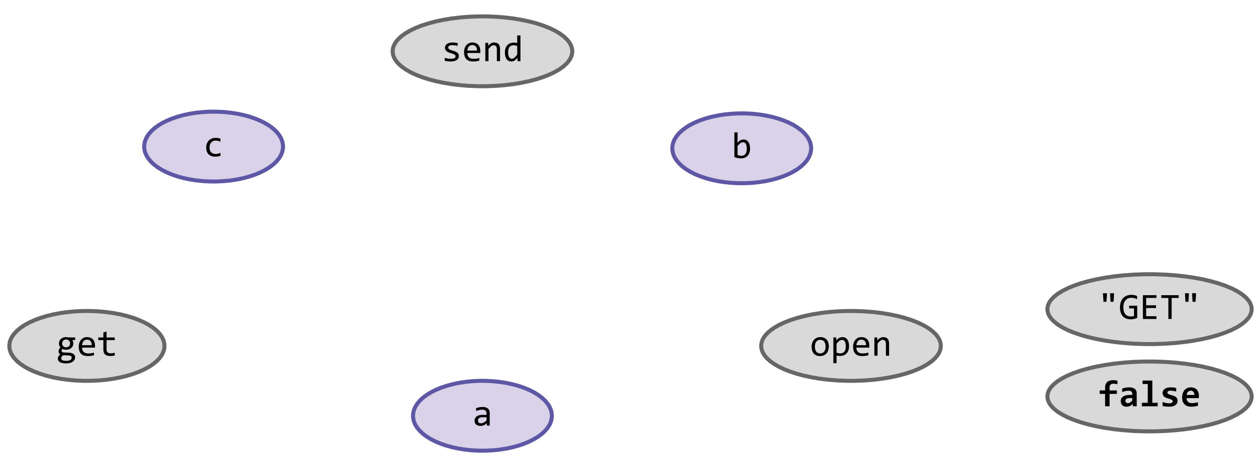


JS NICE Representation

Applications

Intermediate Representation

```
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```



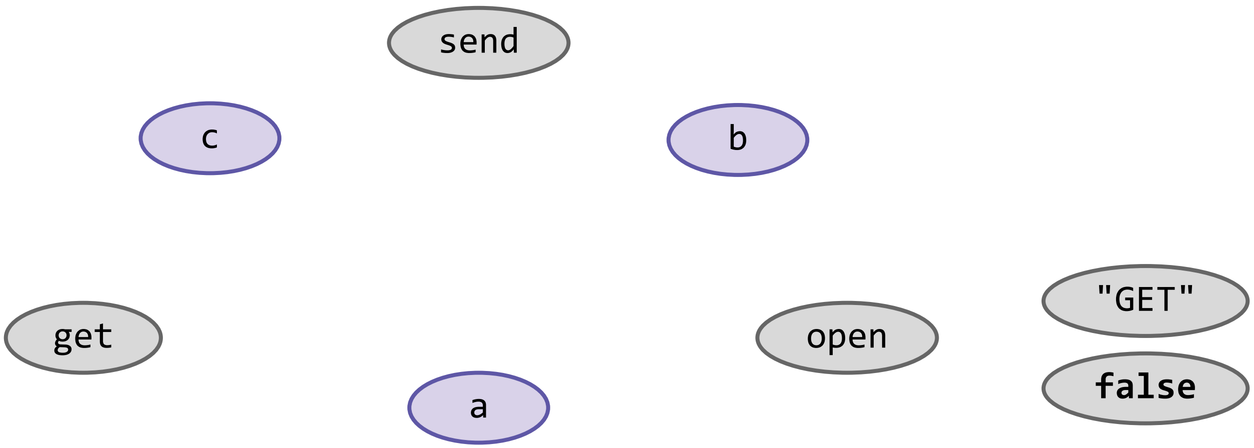
JS NICE Representation

Applications

Intermediate Representation

Analyze Program (PL)

```
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```



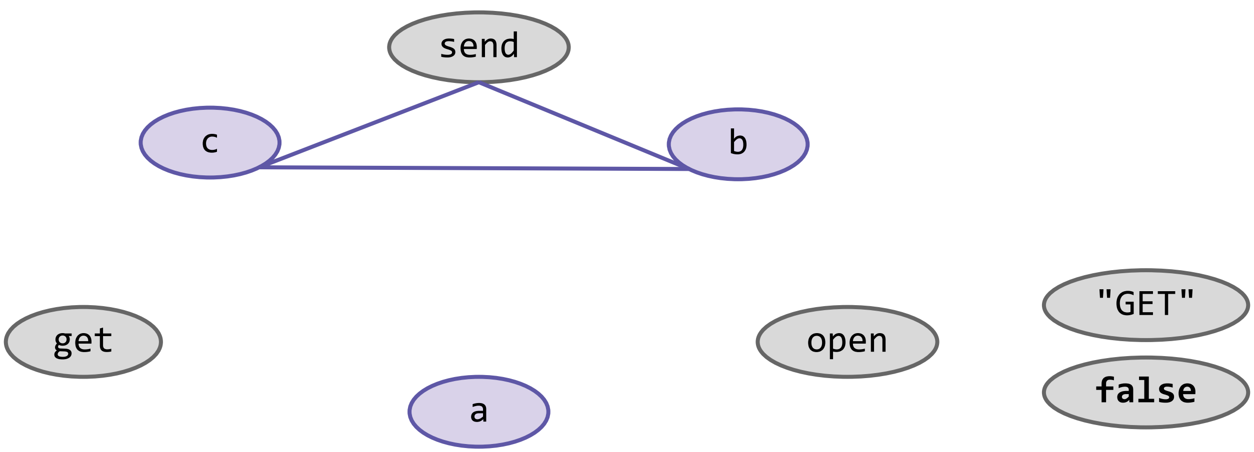
JS NICE Representation

Applications

Intermediate Representation

Analyze Program (PL)

```
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```



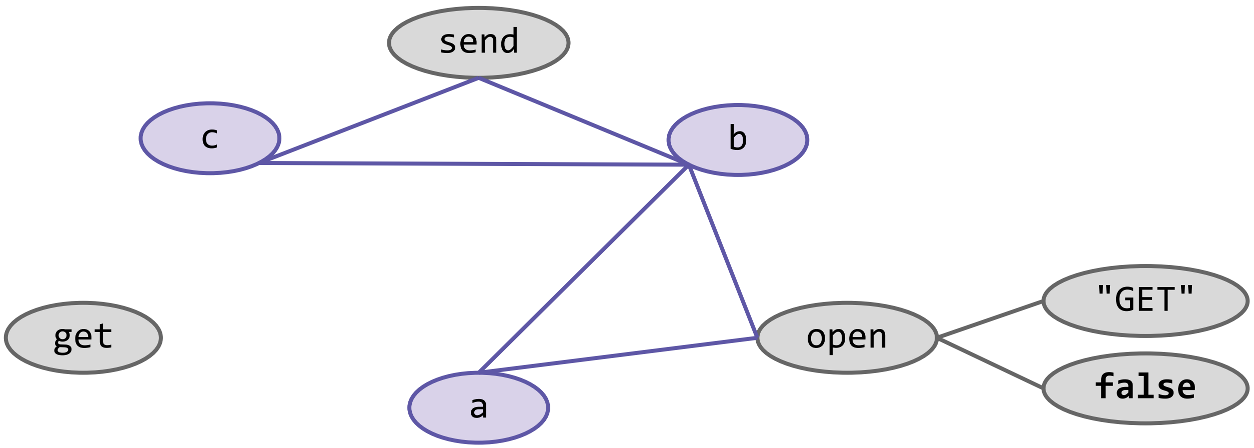
JS NICE Representation

Applications

Intermediate Representation

Analyze Program (PL)

```
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```



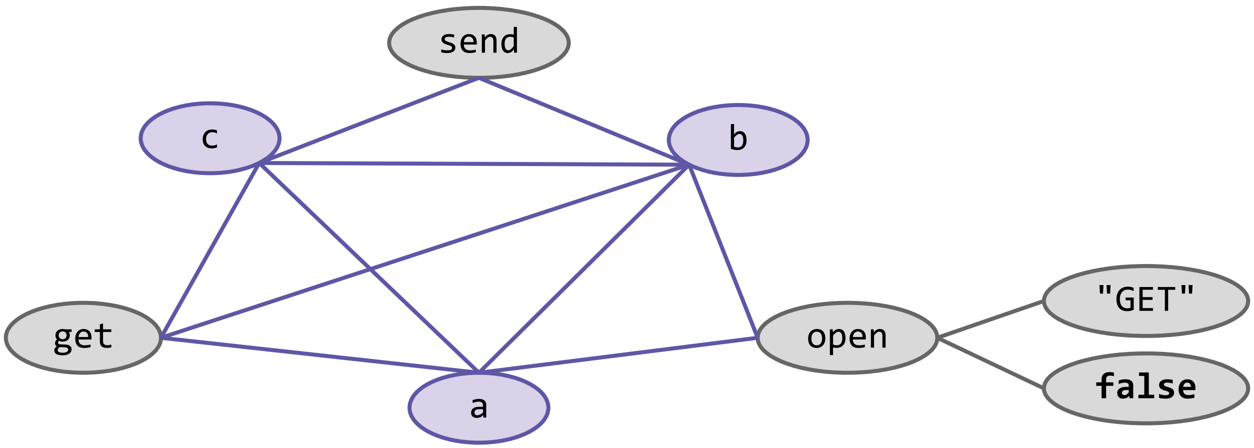
JS NICE Representation

Applications

Intermediate Representation

Analyze Program (PL)

```
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```



JS NICE Representation

Applications

Intermediate Representation

Analyze Program (PL)

Call graph analysis

`Who are the callers of get function?`

Alias analysis

b and c point to the same object?

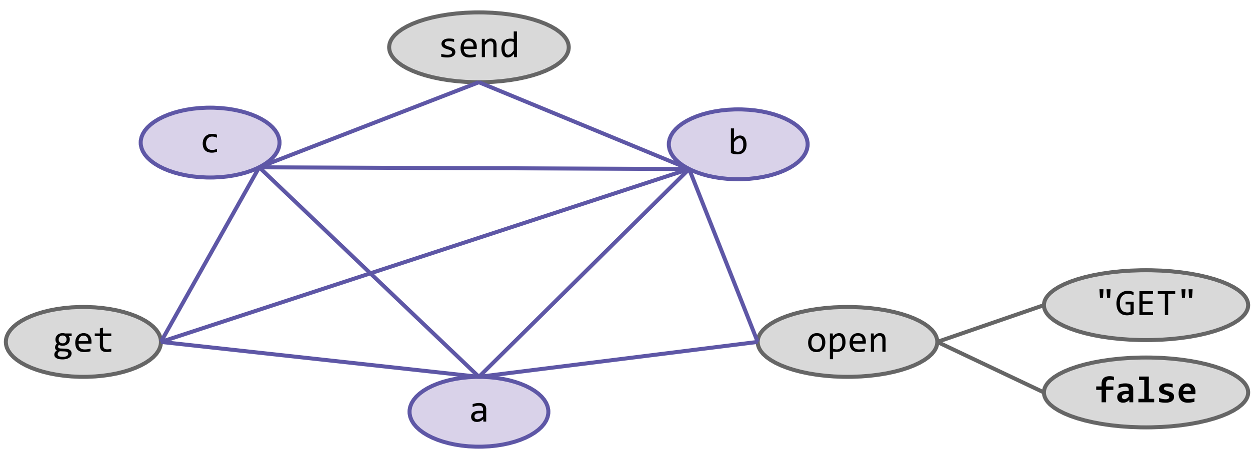
```
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```

Scope analysis

`Program location where b was defined?`

Typestate analysis

`State of the object b?`



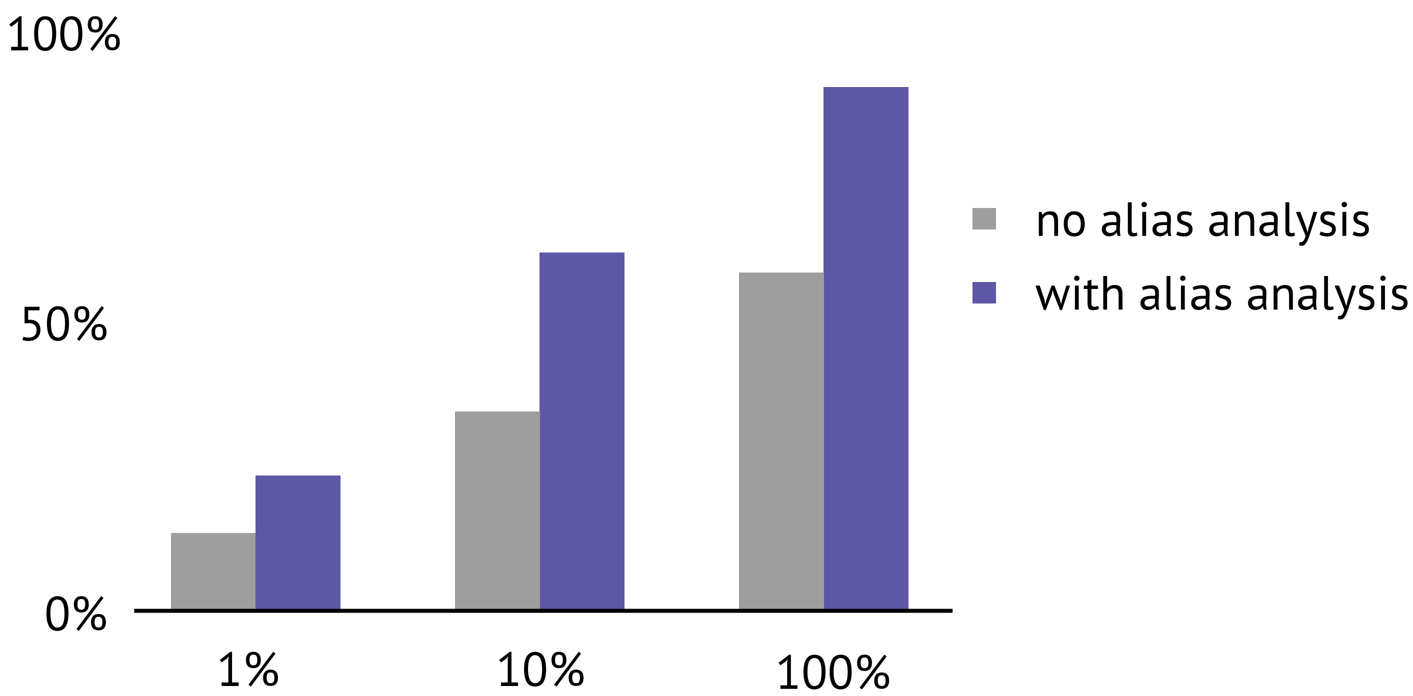
Importance of Program Analysis

Applications

Intermediate Representation

Analyze Program (PL)

[Precision vs % of data used]



Markov Network

Applications

Intermediate Representation

Analyze Program (PL)

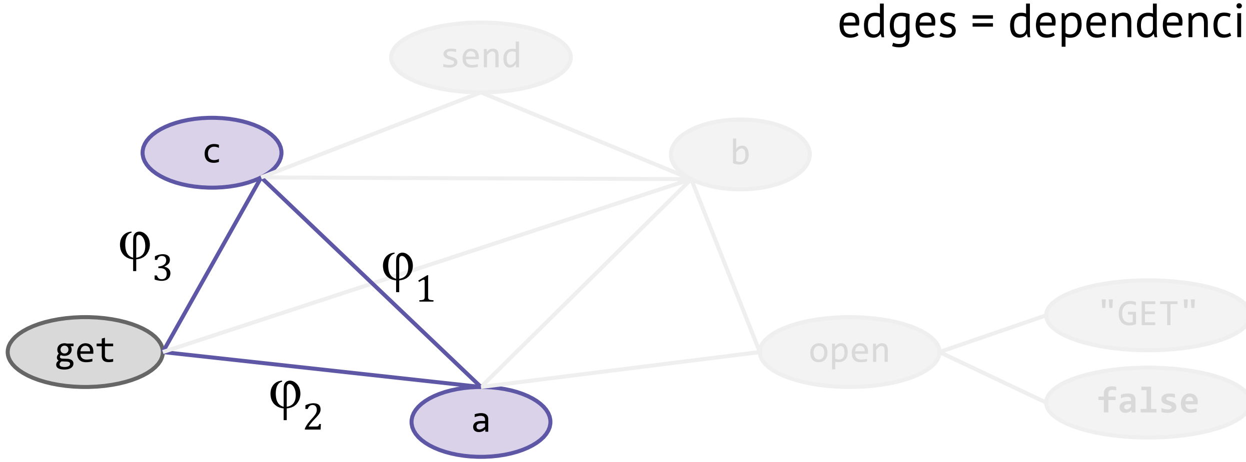
Train Model (ML)

Undirected graphical model

Captures **dependencies** between facts to be predicted

nodes = random variables

edges = dependencies



$$P(get, c, a) = \varphi_1(a, c) * \varphi_2(a, get) * \varphi_2(c, get) * \ldots / Z(get, c, a)$$

makes **P** a valid probability distribution
very expensive to compute

Conditional Random Fields

(J. Lafferty, A. McCallum, F. Pereira, ICML 2001)

Applications

Intermediate Representation

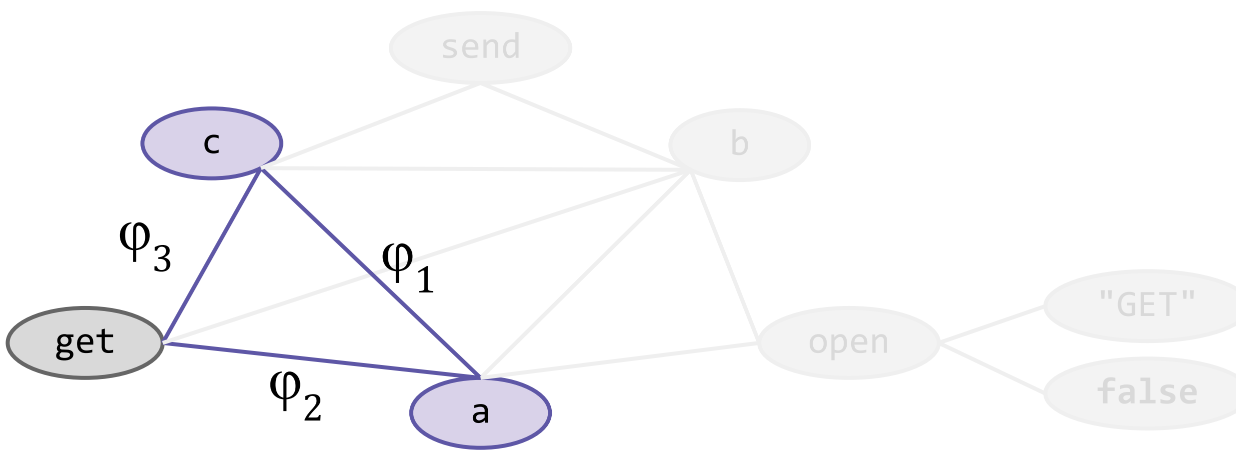
Analyze Program (PL)

Train Model (ML)

Undirected graphical model

Captures **dependencies** between facts to be predicted

Captures **conditional distribution** on known facts

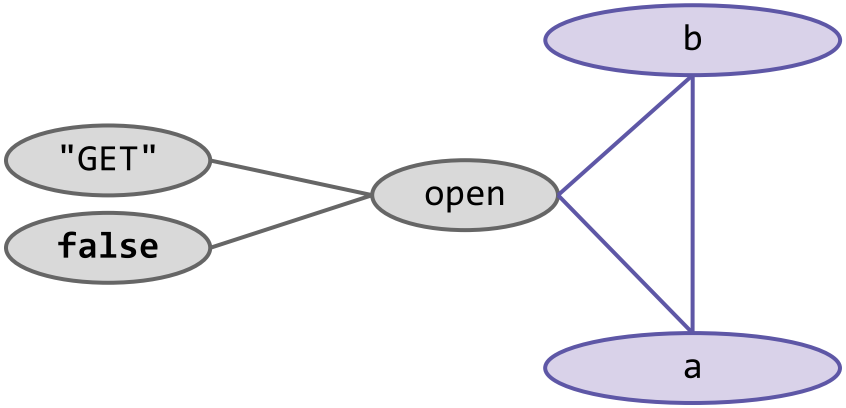


$$P(\mathbf{Y} \mid \mathbf{X}) = \varphi_1(a, c) * \varphi_2(a, get) * \varphi_2(c, get) * \ldots / Z(\mathbf{Y})$$

makes **P** a valid probability distribution
very expensive to compute

MAP inference example

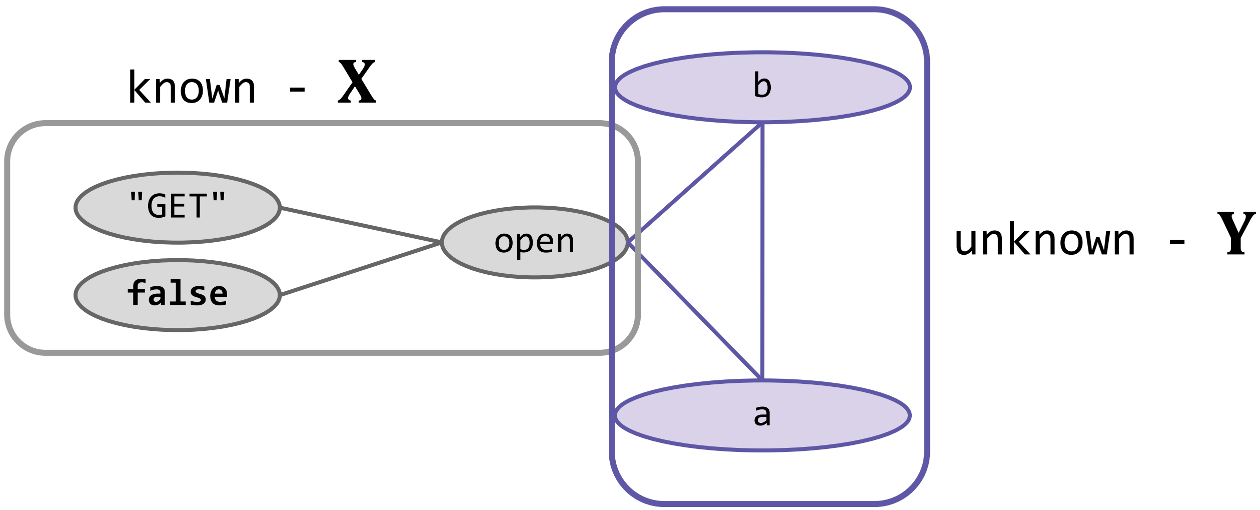
$$\text{argmax}_{\mathbf{Y}} \ \mathbf{P}(\mathbf{Y}|\mathbf{X})$$




```
b.open("GET", a, false);
```

MAP inference example

$$\text{argmax}_{\mathbf{Y}}\ \mathbf{P}(\mathbf{Y}|\mathbf{X})$$



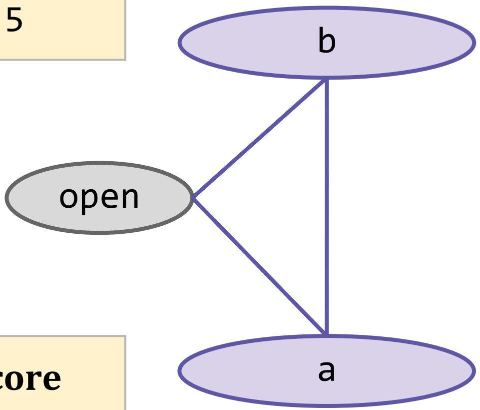

```
b.open("GET", a, false);
```

MAP inference example

$$\operatorname{argmax}_{c,t}\ \mathbf{P}(c,t|v{=}open)$$

φ_1

v	c	Score
open	req	6
open	client	5



t	c	Score
client	url	8
client	link	5
req	link	2

φ_2

v	t	Score
open	link	7
open	url	5
open	address	2

```
b.open("GET", a, false);
```

MAP inference example

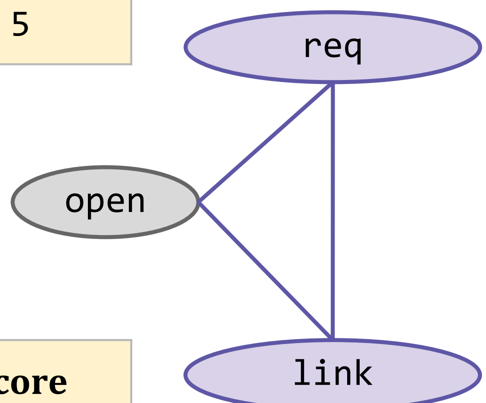
$$\text{argmax}_{c,t}\ \mathbf{P}(c,t|v=\text{open})$$

φ_1

v	c	Score
open	req	6
open	client	5

φ_2

v	t	Score
open	link	7
open	url	5
open	address	2



Maximize product of scores:

$$6*7*2 = 84$$

t	c	Score
client	url	8
client	link	5
req	link	2

```
b.open("GET", a, false);
```

MAP inference example

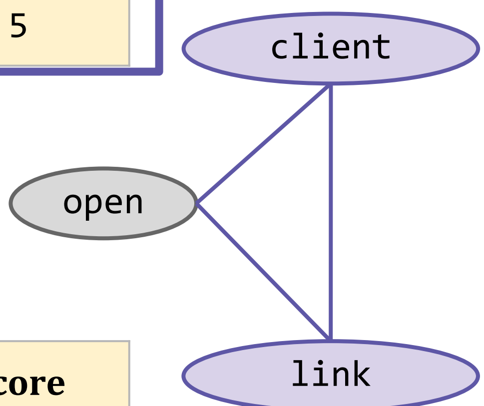
$$\operatorname{argmax}_{c,t} \mathbf{P}(c,t|v=open)$$

φ_1

v	c	Score
open	req	6
open	client	5

Maximize product of scores:

5*7*5 = 175



t	c	Score
client	url	8
client	link	5
req	link	2

φ_2

v	t	Score
open	link	7
open	url	5
open	address	2

```
b.open("GET", a, false);
```

MAP inference example

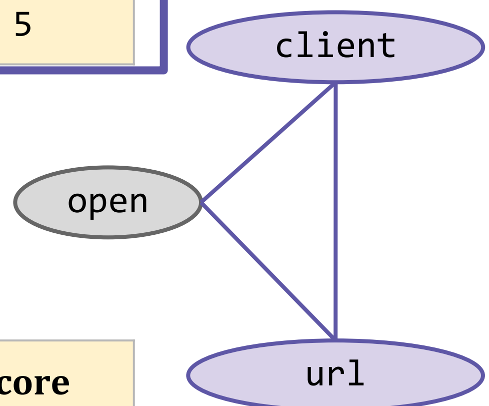
$$\text{argmax}_{c,t}\ \mathbf{P}(c,t|v=\text{open})$$

φ_1

v	c	Score
open	req	6
open	client	5

Maximize product of scores:

$$5*5*8 = 200$$



t	c	Score
client	url	8
client	link	5
req	link	2

φ_2

v	t	Score
open	link	7
open	url	5
open	address	2

```
b.open("GET", a, false);
```

Query

Our goal is to find the most likely assignment of $\mathbf{y}$ that satisfies the constraints, also known as **MAP inference**:

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y}'}\, \mathbf{P}(\mathbf{y}'|\mathbf{x}) = \mathrm{argmax}_{\mathbf{y}'}\, 1/\mathbf{Z} \prod \varphi_i(\mathbf{x},\mathbf{y})$$

Good news:

the expensive partition function $\mathbf{Z}(\mathbf{x})$ is unnecessary

Query

Our goal is to find the most likely assignment of $\mathbf{y}$ that satisfies the constraints, also known as **MAP inference**:

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y}'} \mathbf{P}(\mathbf{y}'|\mathbf{x}) = \mathrm{argmax}_{\mathbf{y}'} 1/\mathbf{Z} \prod \varphi_i(\mathbf{x},\mathbf{y})$$

Good news:

the expensive partition function $\mathbf{Z}(\mathrm{x})$ is unnecessary

Bad news:

computing the argmax is still NP-hard (Max-SAT)

Query

Our goal is to find the most likely assignment of $\mathbf{y}$ that satisfies the constraints, also known as **MAP inference**:

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y}'}\, \mathbf{P}(\mathbf{y}'|\mathbf{x}) = \mathrm{argmax}_{\mathbf{y}'}\, 1/\mathbf{Z} \prod \varphi_i(\mathbf{x},\mathbf{y})$$

Good news:

many approximate algorithm exists (Variational Methods, EM, Gibbs sampling, Elimination Algorithm, Junction-Tree algorithm)

Query

Our goal is to find the most likely assignment of **y** that satisfies the constraints, also known as **MAP inference**:

$$\mathbf{y} = \text{argmax}_{\mathbf{y}'} \mathbf{P}(\mathbf{y}'|\mathbf{x}) = \text{argmax}_{\mathbf{y}'} 1/\mathbf{Z} \prod \varphi_i(\mathbf{x},\mathbf{y})$$

Good news:

many approximate algorithm exists (Variational Methods, EM, Gibbs sampling, Elimination Algorithm, Junction-Tree algorithm)

Bad news:

still too slow for learning

Query

Our goal is to find the most likely assignment of **y** that satisfies the constraints, also known as **MAP inference**:

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y}'} \mathbf{P}(\mathbf{y}'|\mathbf{x}) = \mathrm{argmax}_{\mathbf{y}'} 1/Z \prod \varphi_i(\mathbf{x},\mathbf{y})$$

Good news:

approximate algorithms designed to fit our setting

Learning

$$\mathbf{P}(\mathbf{y}|\mathbf{x}) = 1/Z \;\; \exp \Sigma\, \lambda_i f_i(\mathbf{x},\mathbf{y})$$

Learning finds weights λ_i from training data

$$D = \{\, \mathbf{x}^{(j)} , \mathbf{y}^{(j)} \,\}_{j=1..n}$$

programs with facts of interest already manually annotated

Big codebase to learn from
Programmers have spent countless hours to develop, maintain and annotate

Structured SVM

Generalizes SVM, learns weights such that:

$$\forall j \;\forall \mathbf{y} \sum \lambda_i f_i(\mathbf{x}^{(j)},\mathbf{y}^{(j)}) \geq \sum \lambda_i f_i(\mathbf{x}^{(j)},\mathbf{y}) + \Delta(\mathbf{y},\mathbf{y}^{(j)})$$

for all training data samples

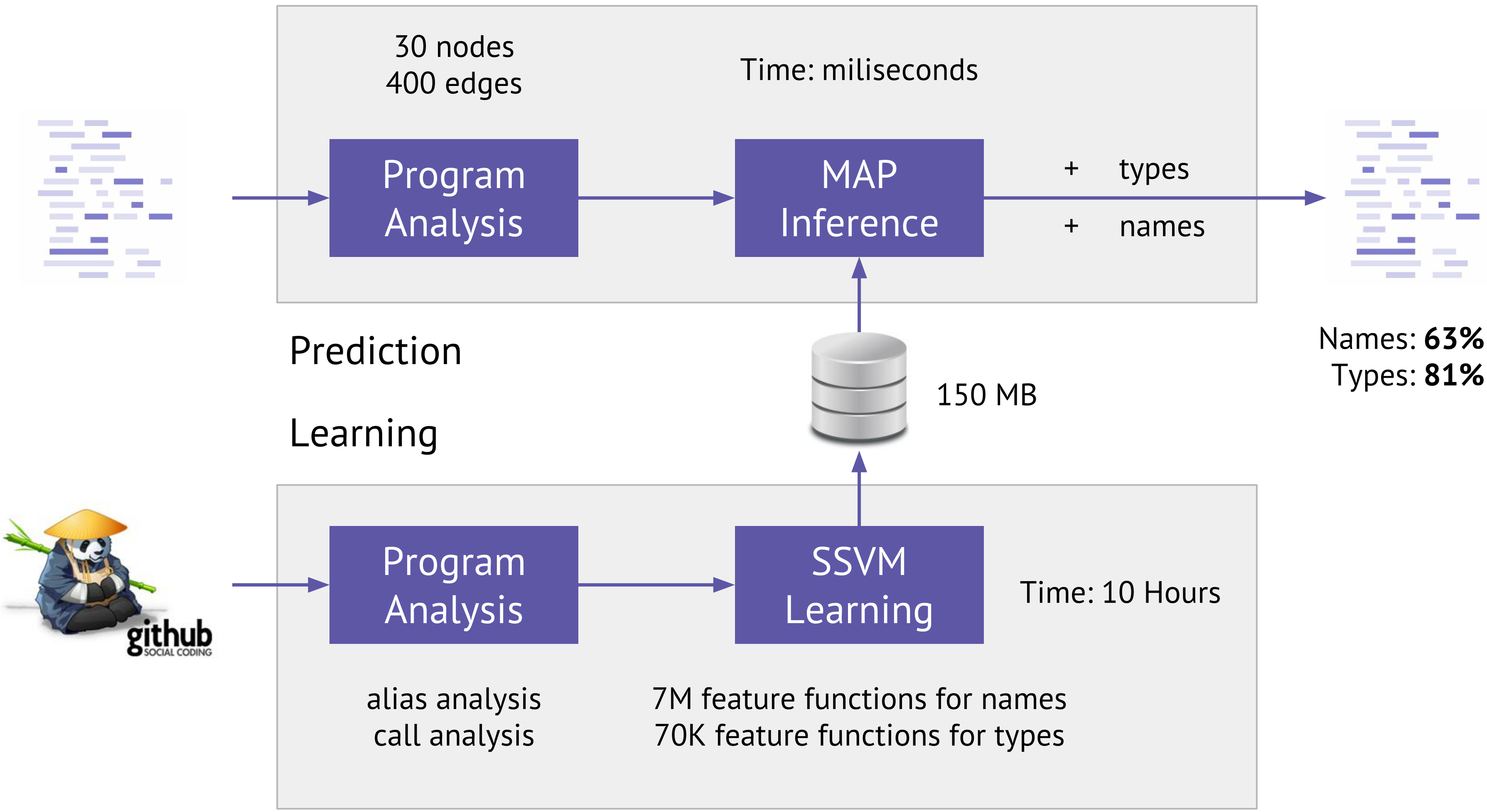
the **given prediction** is better than **any other prediction** by at least a **margin**

Training procedure:

N.Ratliff, J. Bagnell, M. Zinkevich: (Online) Subgradient Methods for Structured Prediction, AIStats'07

Memory efficient
Fast and scalable

Structured Prediction for Programs



Programming with "Big Code"

Applications	Code completion, Program synthesis, Translation, Deobfuscation, Feedback generation	
Intermediate Representation	Sequences (sentences), Translation Table, Trees	Graphical Models, Feature Vectors
Analyze Program (PL)	alias analysis, control-flow analysis, scope analysis, typestate analysis	
Train Model (ML)	Neural Networks, SVM, N-gram language model	Structured SVM
Query Model	$\text{argmax}_{y \in \Omega} P(y \mid x)$	Greedy MAP Inference

More information and tutorials at: http://www.nice2predict.org/ http://plml.ethz.ch/

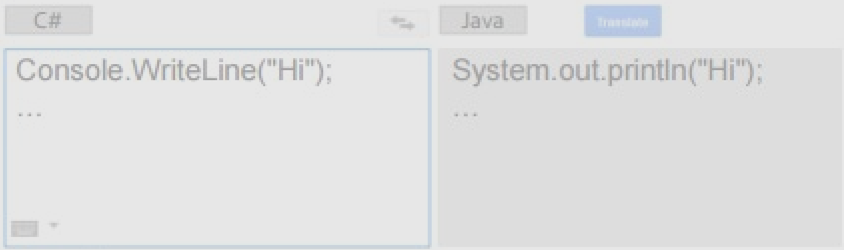
NICE 2 Predict

Statistical Software Tools

Write new code:
Code Completion

```
Camera camera = Camera.open();
camera.SetDisplayOrientation(90);
              ?
```

Port code:
Programming Language Translation



Understand code/security:
JavaScript Deobfuscation
Type Prediction

JS NICE

Debug code:
Statistical Bug Detection

likely error

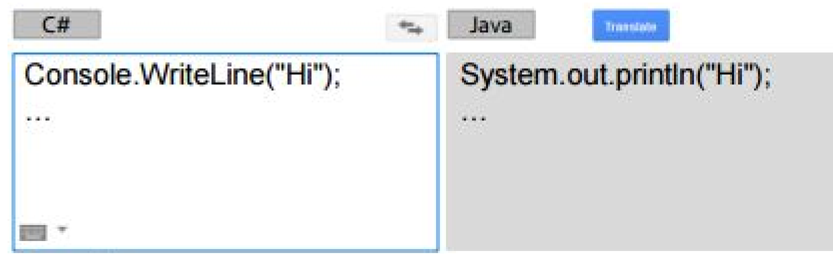
```
...
for x in range(a):
    print a[x]
```

Statistical Software Tools

Write new code:
Code Completion

```
Camera camera = Camera.open();
camera.SetDisplayOrientation(90);
              ?
```

Port code:
Programming Language Translation



Understand code/security:
JavaScript Deobfuscation
Type Prediction

JS NICE

Debug code:
Statistical Bug Detection

```
...
for x in range(a):
     print a[x]
```

likely error

Probabilistic Model for Code

Model is a key part of the Statistical Programming Tools

Goal: score programs

Select best among several candidates

Probabilistic Model for Code

Model is a key part of the Statistical Programming Tools

Goal: score programs

Select best among several candidates

Example: Which function is more likely?

(a)

```
function area(a) {
  return a.width * a.height
}
```

(b)

```
function area(a) {
  return a.width * a.close()
}
```

Statistical Code Completion

Model is a key part of Statistical Programming Tools

Goal: score programs

Select best among several candidates

Example:

```
function area(a) {
  return a.width * a.
}
```

Candidate	Likelihood
height	Very likely
width	Less likely
open	impossible
close	impossible

Probabilistic Model for Code

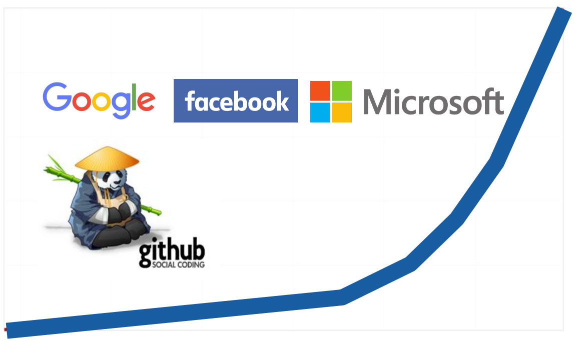
Directly applicable to code completion, but is a ***key statistical component*** for many others tasks:
e.g. natural language to code,
statistical bug localization

Model Requirements

Existing Programs

Learning

Model



Probabilistic Model

Widely Applicable

Efficient Learning

High Precision

Explainable Predictions

Observation

Regularities in code are similar
to regularities in natural language

The quick brown fox jumps over the lazy ?

Observation

Regularities in code are similar
to regularities in natural language

The quick brown fox jumps over the lazy dog

Observation

Regularities in code are similar
to regularities in natural language

The quick brown fox jumps over the lazy dog

```
file = open(filename, "r")
file.??
```

Observation

Regularities in code are similar
to regularities in natural language

The quick brown fox jumps over the lazy dog

```
file = open(filename, "r")
file.read()
```

N-gram Language Model

Conditional probability only on previous n-1 words

$$P(w_i \mid w_1 \dots w_{i-1}) \approx P(w_i \mid \underbrace{w_{i-n+1} \dots w_{i-1}}_{\text{n-1 words}}) \approx \frac{\#(w_{i-n+1} \dots w_{i-1} w_i)}{\#(w_{i-n+1} \dots w_{i-1})}$$

#(n-gram) - number of occurrences of n-gram in training data

N-gram Language Model

Conditional probability only on previous n-1 words

$$P(w_i \mid w_1 \ldots w_{i-1}) \approx P(w_i \mid \underbrace{w_{i-n+1} \ldots w_{i-1}}_{\text{n-1 words}}) \approx \frac{\#(w_{i-n+1} \ldots w_{i-1} w_i)}{\#(w_{i-n+1} \ldots w_{i-1})}$$

Training is achieved by counting n-grams (3-gram)

$$P(\text{jumped} \mid \text{The quick brown fox}) \approx P(\text{jumped} \mid \text{brown fox}) \approx \frac{\#(\text{brown fox jumped})}{\#(\text{brown fox})}$$

#(n-gram) - number of occurrences of n-gram in training data

N-gram Language Model

Training

```
return f . height * scale;

f . open (mode) ;

f . close () ;

2 * f . width;

f . close ()
```

Prediction

```
f.width + f. ?
```

N-gram Language Model

Training (3-gram model)

```
return f . height * scale;

f . open (mode) ;

f . close () ;

2 * f . width;

f . close ()
```

3-gram

Prediction

```
f.width + f. ?
```

N-gram Language Model

Training (3-gram model)

```
return f . height * scale;

f . open (mode) ;

f . close () ;

2 * f . width;

f . close ()
```

3-gram

Prediction

f.width + f. **?**

	P
close	0.4
open	0.2
width	0.2
height	0.2

N-gram Language Model

Conditioning	Accuracy
Last two tokens, Hindle et. al. [ICSE'12]	22.2%

Main Problem:

```
f.width + f. ?
```

?	P
close	0.4
open	0.2
width	0.2
height	0.2

Bad context leads to bad probability estimates

Better Context

Training

```
return x.width * x.height

return y.width * y.height

area = s.width * s.height

s.width = s.width + 10

q.depth * q.width * q.height
```

Prediction

```
f.width + f. ?
```

Better Context

Training

```
return x.width * x.height

return y.width * y.height

area = s.width * s.height

s.width = s.width + 10

q.depth * q.width * q.height
```

Prediction

```
f.width + f. ?
```

height	0.8
width	0.2
open	0.0
close	0.0

Context: relevant for this prediction

Better Context

Conditioning	Accuracy
Last two tokens, Hindle et. al. [ICSE'12]	22.2%
Last two APIs, Raychev et. al. [PLDI'14]	30.4%

JavaScript APIs

Conditioning	**Accuracy**
Last two tokens, Hindle et. al. [ICSE'12]	22.2%
Last two APIs, Raychev et. al. [PLDI'14]	30.4%

is this the best we can do?

JavaScript APIs

Conditioning	Accuracy
Last two tokens, Hindle et. al. [ICSE'12]	22.2%
Last two APIs, Raychev et. al. [PLDI'14]	30.4%

is this the best we can do?

Program synthesis	**66.4%**

JavaScript APIs

Conditioning	Accuracy
Last two tokens, Hindle et. al. [ICSE'12]	22.2%
Last two APIs, Raychev et. al. [PLDI'14]	30.4%
Last three APIs	
Declaration Site + Last two APIs	
Variable Name + Method Name + Last API	
...	

How do we know that which is the best context?

JavaScript APIs

Conditioning	Accuracy
Last two tokens, Hindle et. al. [ICSE'12]	22.2%
Last two APIs, Raychev et. al. [PLDI'14]	30.4%
Last three APIs	
Declaration Site + Last two APIs	
Variable Name + Method Name + Last API	
...	

How do we know that which is the best context?

JavaScript APIs

Identifiers

Strings

Numbers

Arguments

Properties

Statements

RegExp

Structure

Solution: Synthesise the Best Model

`{width} = f( x.width + x. ? )`

Synthesise a function *f* from a domain specific language that explains the data

Function Examples

$$f(p_1) = \{ \quad \}$$

```
for (j = 0; j < groups.length; j++) {
  idsInGroup = groups[j].filter(
    function(id) { return id >= 42; }
  );
  if (idsInGroup.length == 0) {
      ?
  }
}
```

Function Examples

$$f(p_1) = \{\ \texttt{for},\ \texttt{if},\ \texttt{length==0}\ \}$$

```
for (j = 0; j < groups.length; j++) {
   idsInGroup = groups[j].filter(
     function(id) { return id >= 42; }
   );
   if (idsInGroup.length == 0) {
        ?
   }
}
```

Function Examples

$$f(p_1) = \{ \text{for}, \text{if}, \text{length==0} \}$$

```
for (j = 0; j < groups.length; j++) {
  idsInGroup = groups[j].filter(
    function(id) { return id >= 42; }
  );
  if (idsInGroup.length == 0) {
        ?
  }
}
```

$$f(p_2) = \{ \quad \}$$

```
elem.notify( ..., {
    position: 'top',
    hide: false,
      ?
} );
```

Function Examples

$$f(p_1) = \{\ \texttt{for},\ \texttt{if},\ \texttt{length==0}\ \}$$

```
for (j = 0; j < groups.length; j++) {
   idsInGroup = groups[j].filter(
     function(id) { return id >= 42; }
   );
  if (idsInGroup.length == 0) {
           ?
  }
}
```

$$f(p_2) = \{\ \texttt{notify},\ \texttt{position},\ \texttt{hide}\ \}$$

```
elem.notify( ..., {
    position: ‘top’,
    hide: false,
        ?
} );
```

Overview

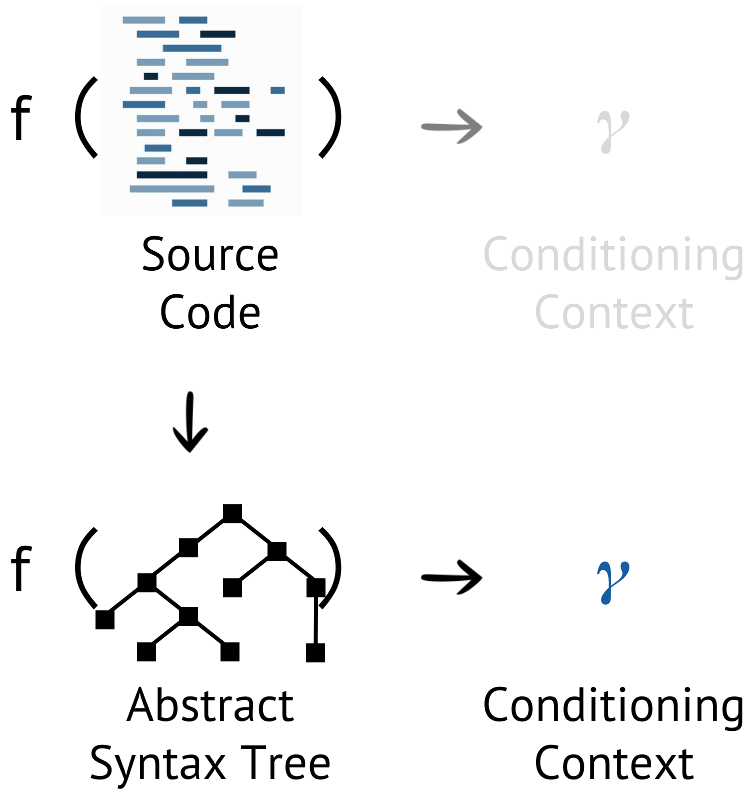
f () → γ

Source Code

Conditioning Context

Synthesise a function ***f*** from a domain specific language that explains the data

Overview



Synthesise a function ***f*** from a domain specific language that explains the data

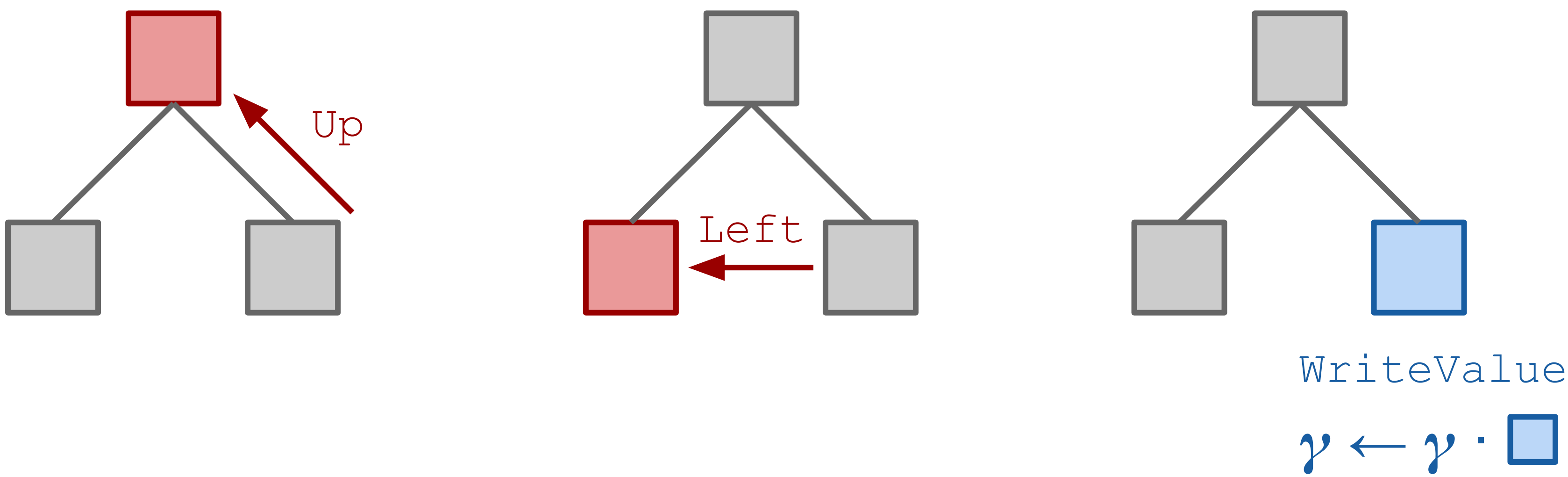
Function Representation

In general:
Unrestricted programs (Turing complete)

Our Work:
TCond Language for navigating over trees
and accumulating context

```
TCond      ::=  ε | WriteOp TCond | MoveOp TCond | BranchProg
BranchProg ::=  if pred(x) then TCond else TCond
MoveOp     ::=  Up, Left, Right, DownFirst, DownLast,
                NextDFS, PrevDFS, NextLeaf, PrevLeaf,
                PrevNodeType, PrevNodeValue, PrevNodeContext
WriteOp    ::=  WriteValue, WriteType, WritePos
```

Expressing functions: TCond Language




```
TCond      ::=  ε | WriteOp TCond | MoveOp TCond | BranchProg
BranchProg ::=  if pred(x) then TCond else TCond
MoveOp     ::=  Up, Left, Right, DownFirst, DownLast,
                NextDFS, PrevDFS, NextLeaf, PrevLeaf,
                PrevNodeType, PrevNodeValue, PrevNodeContext
WriteOp    ::=  WriteValue, WriteType, WritePos
```

Example

Query TCond γ

```
elem.notify(
   ... ,
   ... ,
   {
     position: ‘top’,
     hide: false,
     ?
   }
);
```

Example

Query	TCond	γ
	Left	{}
	WriteValue	{hide}

```
elem.notify(
   ... ,
   ... ,
   {
     position: 'top',
     hide: false,
     ?
   }
);
```

Example

Query	TCond	γ
	Left	{}
	WriteValue	{hide}
	Up	{hide}
	WritePos	{hide, 3}

```
elem.notify(
   ... ,
   ... ,
  {
     position: 'top',
     hide: false,
     ?
  }
);
```

Example

Query

```
elem.notify(
   ... ,
   ... ,
  {
    position: 'top',
    hide: false,
    ?
  }
);
```

TCond	γ
Left	{}
WriteValue	{hide}
Up	{hide}
WritePos	{hide, 3}
Up	{hide, 3}
DownFirst	{hide, 3}
DownLast	{hide, 3}
WriteValue	{hide, 3, notify}

Example

```
elem.notify(
   ... ,
   ... ,
  {
    position: 'top',
    hide: false,
    ?
  }
);
```

Query	TCond	γ
	Left	{}
	WriteValue	{hide}
	Up	{hide}
	WritePos	{hide, 3}
	Up	{hide, 3}
	DownFirst	{hide, 3}
	DownLast	{hide, 3}
	WriteValue	{hide, 3, notify}

↓

{ Previous Property, Parameter Position, API name }

Results

Probabilistic Model of JavaScript Language

20k Learning *100k Training* *50k Blind Set*

GitHub

JavaScript APIs

Conditioning	Accuracy
Last two tokens, Hindle et. al. [ICSE'12]	22.2%
Last two APIs, Raychev et. al. [PLDI'14]	30.4%
Program synthesis	**66.4%**

JavaScript Structure

Model	Accuracy
PCFG	51.1%
N-gram	71.3%
Naïve Bayes	44.2%
SVM	70.5%
Program synthesis	**81.5%**

JavaScript Values

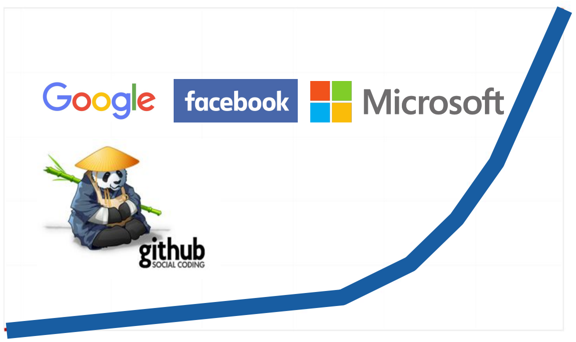
<table>
<tr><th></th><th>Accuracy</th><th>Example</th></tr>
<tr><td>Identifier</td><td>62%</td><td><code>contains = <u>jQuery</u> …</code></td></tr>
<tr><td>Property</td><td>65%</td><td><code>start = list.<u>length</u>;</code></td></tr>
<tr><td>String</td><td>52%</td><td><code>'[' + attrs + <u>']'</u></code></td></tr>
<tr><td>Number</td><td>64%</td><td><code>canvas(xy[0], xy[<u>1</u>], …)</code></td></tr>
<tr><td>RegExp</td><td>66%</td><td><code>line.replace(<u>/(|)+/</u>, …)</code></td></tr>
<tr><td>UnaryExpr</td><td>97%</td><td><code>if (!events || <u>!</u>…)</code></td></tr>
<tr><td>BinaryExpr</td><td>74%</td><td><code>while (++index <u><</u> …)</code></td></tr>
<tr><td>LogicalExpr</td><td>92%</td><td><code>frame = frame <u>||</u> …</code></td></tr>
</table>

Model Requirements

Existing Programs

Learning

Model



→

Probabilistic Model

Widely Applicable

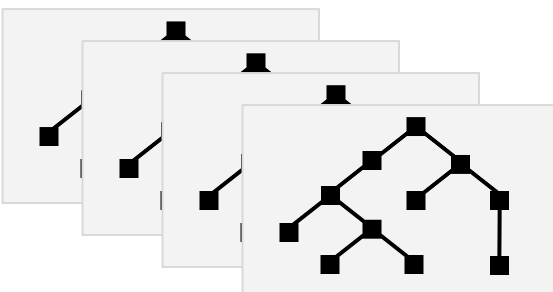
Efficient Learning

High Precision

Explainable Predictions

Learning

Dataset



```
TCond   ::=  ε | WriteOp TCond | MoveOp TCond
MoveOp  ::=  Up, Left, Right, ...
WriteOp ::=  WriteValue, WriteType, ...
```

TCond Language

Program Synthesis
Enumerative search
Genetic programming
Decision tree learning
MCMC

↓

$$f_{best} = \arg\min_{f \in TCond} cost(D, f)$$

↑

$|d| << |D|$
$|cost(d, f) - cost(D,f)| < \varepsilon$
Representative sampling

Applications

Writing new Code

Finding Bugs

Porting Code

Allocation site Analysis

Points-to Analysis

Handwriting Recognition

Speech Recognition

Probabilistic Model

Static Analysis

Neural Networks

Character-level Language Models

Google facebook Microsoft

UBS

github SOCIAL CODING

The quick brown fox jumps over the lazy dog

Work @ ETH Zurich

Prof.
Martin
Vechev

Veselin
Raychev

Pavol
Bielik

Christine
Zeller

Pascal
Roos

Benjamin
Bischel

Svetoslav
Karaivanov

Benjamin
Mularczyk

Learning from "Big Code"

Applications	Name and Types Prediction ...
Intermediate Representation	Graphical Models Feature Vectors
Analyze Program (PL)	
Train Model (ML)	Structured SVM
Query Model	Greedy MAP Inference

NICE 2 Predict

Key Idea:

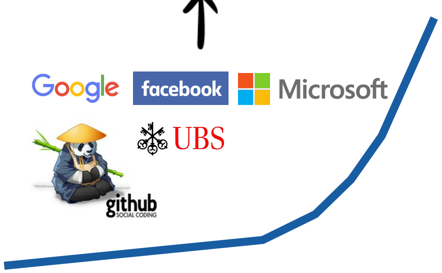
Learn a function *f* that explains the data. The function dynamically obtains the best conditioning context for a given query.

Probabilistic Model

↑

$$f_{best} = \underset{f \in DSL}{\arg\min}\ cost(D, f)$$

↑



http://plml.ethz.ch/