Probabilistic and Interpretable Models of Code

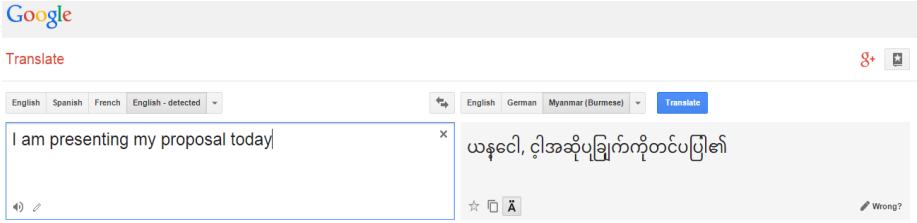
Martin Vechev
ETH Zurich and DeepCode.ai

ETH Zürich

DEEPCODE

Learning and probabilistic models based on Big Data have **revolutionized** entire fields

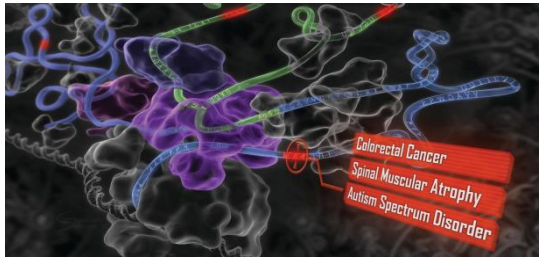
Natural Language Processing
(e.g., machine translation)



Computer Vision
(e.g., image captioning)

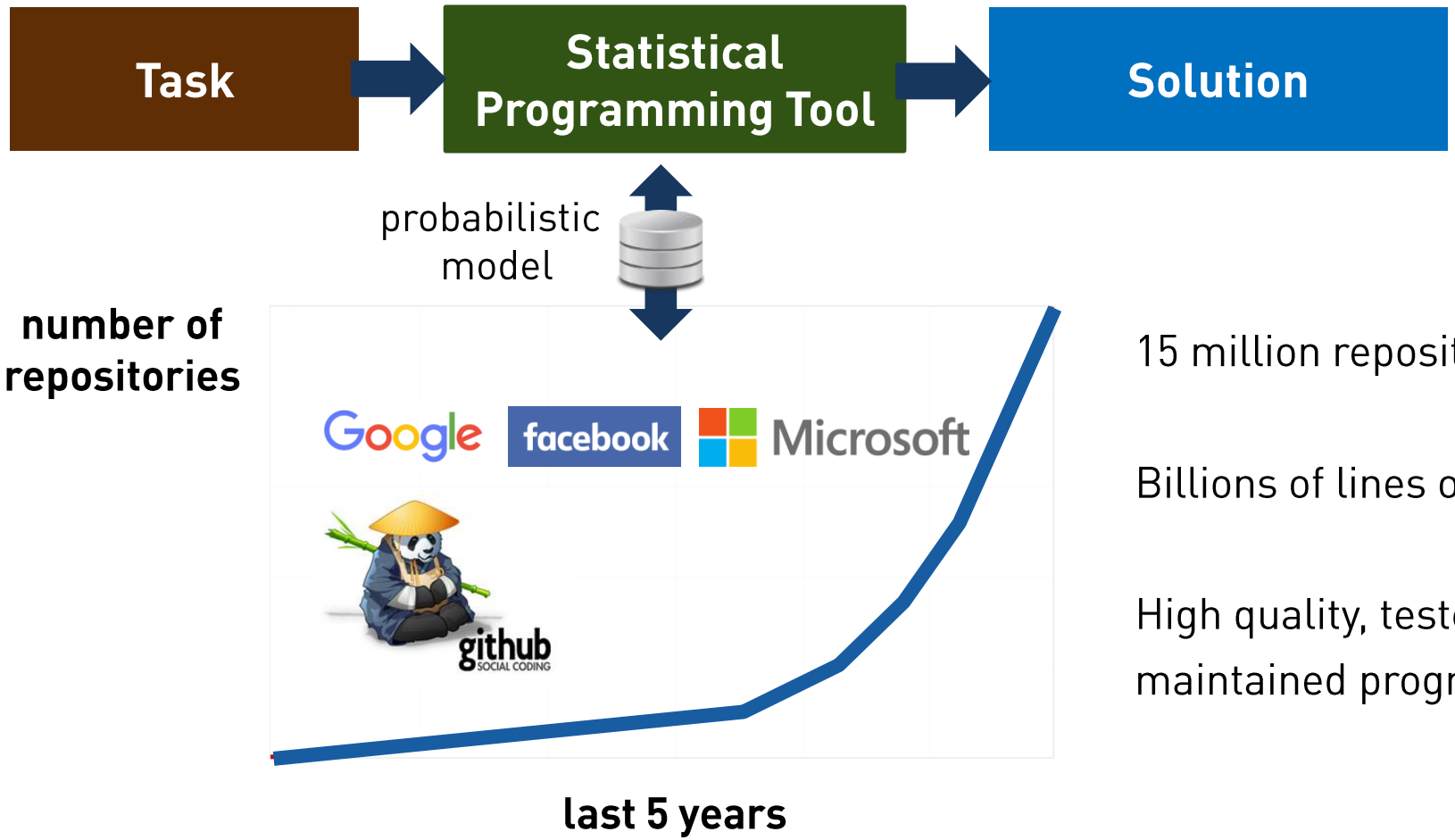
A group of people shopping at an outdoor market. There are many vegetables at the fruit stand.

Medical Computing
(e.g., disease prediction)



Can we bring this revolution to programmers?

Machine Learning for Programs



15 million repositories

Billions of lines of code

High quality, tested, maintained programs

Why now?

Advances in Programming Languages
[Automated Reasoning, Synthesis, Constraint Solving]

Confluence of streams

Advances in Machine Learning
[Deep Learning, Graphical Models, Language Models]

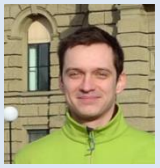
machine learning-based programming tools
new rules, new ideas, new opportunities

Data
[> 15 million public repositories]

Machine Learning for Programs

Probabilistically likely solutions to problems hard to solve otherwise

Joint work with :


Pavol Bielik


Veselin Raychev


Andreas Krause

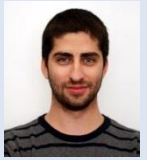

Christine Zeller


Svetoslav Karaivanov


Pascal Roos


Benjamin Bichsel


Timon Gehr


Petar Tsankov


Mateo Panzacchi

Publications

- Robust Relational Layout Synthesis from Examples, **ACM OOPSLA'18**
- Inferring Crypto API Rules from Code Changes, **ACM PLDI'18**
- Program Synthesis for Character Level Language Modeling, **ICLR'17**
- Learning a Static Analyzer from Data, **CAV'17**
- V. Raychev, PhD thesis, **ACM Doctoral Dissertation**, Honorable Mention
- Statistical Deobfuscation of Android Applications, **ACM CCS'16**
- Probabilistic Mode for Code with Decision Trees, **ACM OOPSLA'16**
- PHOG: Probabilistic Mode for Code, **ACM ICML'16**
- Learning Programs from Noisy Data, **ACM POPL'16**
- Predicting Program Properties from "Big Code", **ACM POPL'15, CACM'18**
- Code Completion with Statistical Language Models, **ACM PLDI'14**
- Machine Translation for Programming Languages, **ACM Onward'14**

Statistical Systems for Code

apk-deguard.com

DEGUARD

DEEP3

jsnice.org

JS NICE

SLANG

nice2predict.org

NICE 2 Predict

more: http://plml.ethz.ch

Dimensions:

Machine Learning for Programming

Applications

Intermediate Representation

Analyze Program (PL)

Train Model (ML)

Query Model (ML)

Dimensions:
Machine Learning for Programming

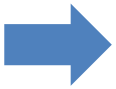
Applications	Code completion, Deobfuscation, Program synthesis, Feedback generation, Translation
Intermediate Representation	Sequences (sentences), Trees, Translation Table, Graphical Models (CRFs), Feature Vectors
Analyze Program (PL)	typestate analysis, scope analysis, control-flow analysis, alias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

Statistical Code Synthesis

[Code Completion with Statistical Language Models, ACM PLDI 2014
Veselin Raychev, Eran Yahav, M.V.]

```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
```

SLANG



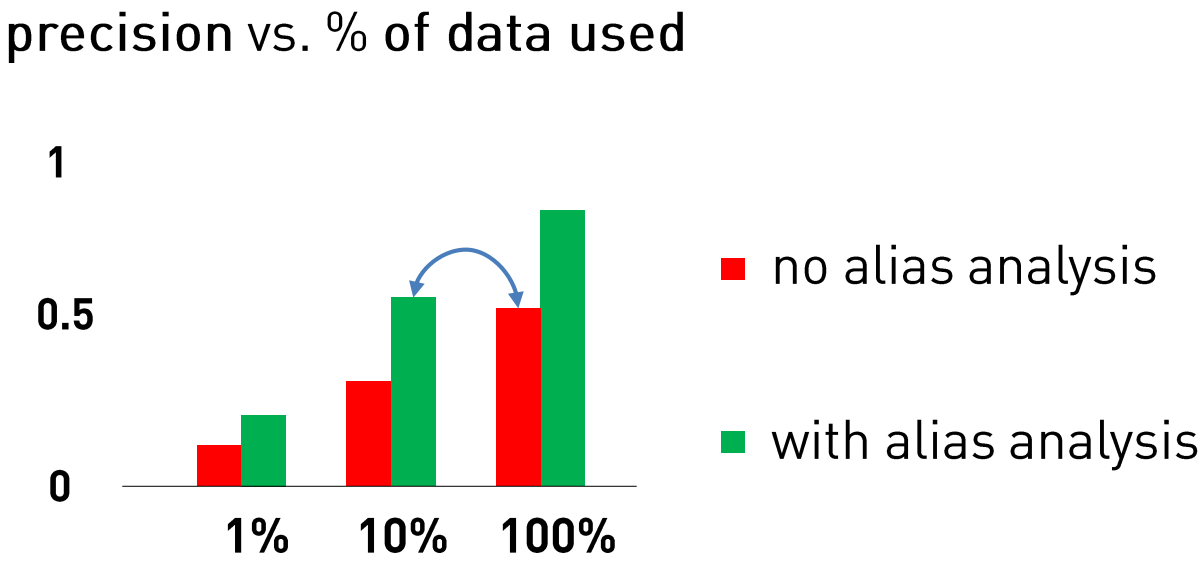
```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
camera.unlock();
SurfaceHolder holder = getHolder();
holder.addCallback(this);
holder.setType(SurfaceHolder.STP);
MediaRecorder r = new MediaRecorder();
r.setCamera(camera);
r.setAudioSource(MediaRecorder.AS);
r.setVideoSource(MediaRecorder.VS);
r.setOutFormat(MediaRecorder.MPEG4);
```

Statistical language models
Recurrent neural networks

+

Typestate analysis
Alias analysis

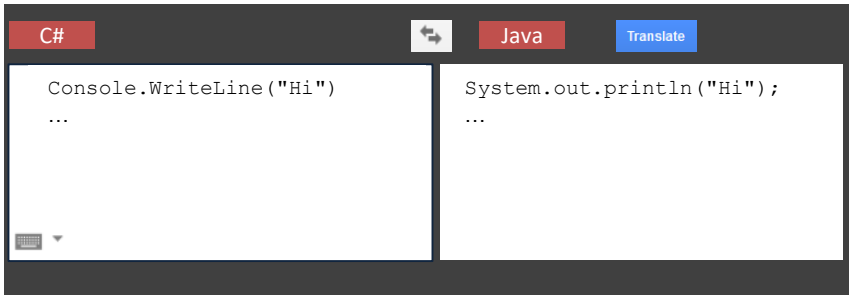
Importance of Static (Semantic) Analysis



static analysis benefit = 10x more data

Statistical Language Translation

[ACM Onward'14, Svetoslav Karaivanov, Veselin Raychev, M.V.]



Phrase-based Statistical Machine Translation

+

Prefix Parsing of Context-Free Grammars

Statistical Program de-obfuscation

Predicting program properties from "Big Code"

Veselin Raychev, Andreas Krause, M.V.

ACM POPL'15
SIGPLAN and ACM Research Highlight

```
function FZ(e, t) {   var n = [];
var r = e.length;   var i = 0;
for (; i < r; i += t)      if (i + t <
r)      n.push(e.substring(i, i +
t)); else
n.push(e.substring(i, r));
return n;
}
```

JSNice.org

```
function chunkData(str, step) {
  var colNames = [];
  var len = str.length;
  var i = 0;
  for (; i < len; i += step)
    if (i + step < len)
      colNames.push(str.substring(i, i + step));
    else
      colNames.push(str.substring(i, len));
  return colNames;
}
```

JSNice.org

JS NICE

✓ Every country ✓ 250,000+ users ✓ Top ranked tool

This Page Amsterdam @thispage_ams · Jul 16
Do you write ugly JavaScript code? Not to worry. JSNice will make it look like you are a superstar coder. Yai! - buff.ly/1HR4JL7

Ingvar Stepanyan @RReverser · Aug 6
JSNice.org became my must-have tool for code deobfuscation.
Expand Reply Retweet Favorite More

Brevity @seekbrevity · Jul 28
JSNice is an a amazing tool for de-minifying #javascript files. JSNice.org, Its great for #learning and reverse engineering.
Expand Reply Retweet Favorite More

Alvaro Sanchez @alvasavi · Jun 10
This is gold.
Statistical renaming, Type inference and Deobfuscation.
jsnice.org
Expand Reply Retweet Favorite More

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

Kamil Tomšík @cztomsik · Jun 6
tell me how this works!
de-minify #jquery #javascript incl. args, vars & #jsdoc
impressive! jsnice.org

CodeGeekz

COMMENTS MOST RECENT

Frici Komesz on: 20 Online Code Editors and Tools for Developers

HOME | WORDPRESS | WEB DESIGN | TYPOGRAPHY | RESOURCES | ARCHIVES | CONTACT | ADVERTISE

20 Essential Tools for Coders

BY GAVIN IN DEVELOPMENT · RESOURCES — 28 JUL, 2014

20 Best Freebies for Web Designers of Year 2014

JavaScript Diagrams

After spending a lot of time in a particular field, we normally get a lot of experience and we suddenly start doing things easily and in short span of time to maximize our efforts towards our goal. This same procedure also holds true in case of web designing also, Because it's all about getting an experience in a particular field or language. Learning web designing is not an easy task, but when you hold experience in the same field, it becomes much easier for you to get most of the work done in very short span of time.

JavaScript迷宮救星 — JS nICE

JS NICE

jsNice is tool that help you to make even make even obfuscated JavaScript code readable. JSnice is Statical Re-naming, Type Inference and De obfuscation.

1. JS Nice

JS NICE STATISTICAL RENAMING, TYPE INFERENCE AND DEOBFUSCATION ABOUT

ENTER JAVASCRIPT | NICIFY JAVASCRIPT | RESULT

Statistical Program de-obfuscation

[ACM CCS'16, Benjamin Bichsel, Petar Tsankov, Veselin Raychev, M.V.]

```
package a.b.c
class a extends SQLiteHelper
{
  SQLiteDatabase b;  public
a(Context ctx) {   b =
getWritableDatabase();  }
Cursor c(String str) {
return b.rawQuery(str); }}
```

apk-deguard.com

```
package com.example.dbhelper

class DBHelper extends
SQLiteHelper {
  SQLiteDatabase db;

  public DBHelper(Context ctx) {
    db = getWritableDatabase();
  }

  Cursor execSQL(String str) {
    return db.rawQuery(str);
  }
}
```

Funny Reddit post/comment

[–] **Tycon712** • **3 points** 2 days ago

Can someone tell me what the point of using Proguard is if there are tools out there like this?

permalink embed pocket

[–] **theheartbreakpug** • **6 points** 2 days ago

As far as I know, this is brand new. I asked the creator of ProGuard a week ago how hard it is to unobfuscate code after it's run through proguard. He said it strips all the names out of the code so it's essentially impossible. I'm super impressed by what they've done here.

Nice2Predict.org:

scalable structured prediction framework

[Pavol Bielik, Veselin Raychev, Matteo Panzacchi, Nick Baumann, M.V.]

fully, open sourced, Apache license

used by various groups worldwide

JS NICE

DEGUARD

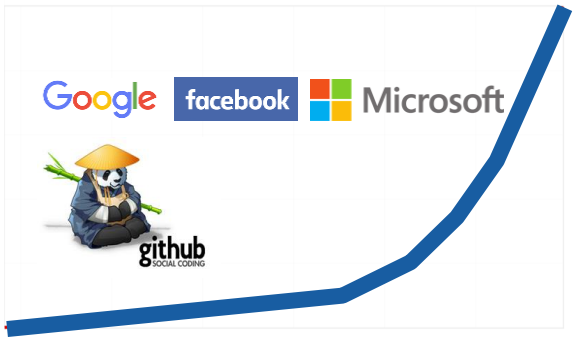
Your System Here

NICE 2 Predict

- Fast, Approximate MAP inference
- Fast, Parallel, Structured SVM and Pseudo-Likelihood Training
- Arbitrary factors and indicator functions

Fundamental Problem

Data



Learning

Model

Probabilistic Model

Widely Applicable

Efficient Learning

High Precision

Explainable Predictions

Training dataset ***D***

```
f.open("f2" | "r");
f.read();
```

```
f.open("f2" | "w");
f.write("c");
```

```
f.open("f1" | "r");
f.read();
```

query:

```
f.open("file" | "r");

f. ?
```

Training dataset ***D***

```
f.open("f2" | "r");
f.read();
```

```
f.open("f2" | "w");
f.write("c");
```

```
f.open("f1" | "r");
f.read();
```

query:

```
f.open("file" | "r");

f. ?
```

3-gram model on tokens

Hindle et. al., ACM ICSE'12

P(open | f.) ~ 3/6
P(read | f.) ~ 2/6
P(write | f.) ~ 1/6

context γ

Training dataset ***D***

```
f.open("f2" | "r");
f.read();
```

```
f.open("f2" | "w");
f.write("c");
```

```
f.open("f1" | "r");
f.read();
```

query:

```
f.open("file" | "r");

f. open
```

3-gram model on tokens

Hindle et. al., ACM ICSE'12

P(open | f.) ~ 3/6
P(read | f.) ~ 2/6
P(write | f.) ~ 1/6

context γ

Training dataset ***D***

```
f.open("f2" | "r");
f.read();
```

```
f.open("f2" | "w");
f.write("c");
```

```
f.open("f1" | "r");
f.read();
```

probabilistic model on APIs

Raychev et. al., ACM PLDI'14

```
P(read  | open ) ~ 2/3
P(write | open ) ~ 1/3
```

context γ

query:

```
f.open("file" | "r");

f. ?
```

3-gram model on tokens

Hindle et. al., ACM ICSE'12

```
P(open  | f. ) ~ 3/6
P(read  | f. ) ~ 2/6
P(write | f. ) ~ 1/6
```

context γ

Training dataset ***D***

```
f.open("f2" | "r");
f.read();
```

```
f.open("f2" | "w");
f.write("c");
```

```
f.open("f1" | "r");
f.read();
```

probabilistic model on APIs

Raychev et. al., ACM PLDI'14

P(read | open) ~ 2/3
P(write | open) ~ 1/3

context γ

query:

```
f.open("file" | "r");

f. read
```

3-gram model on tokens

Hindle et. al., ACM ICSE'12

P(open | f.) ~ 3/6
P(read | f.) ~ 2/6
P(write | f.) ~ 1/6

context γ

Training dataset ***D***

```
f.open("f2" | "r");
f.read();
```

```
f.open("f2" | "w");
f.write("c");
```

```
f.open("f1" | "r");
f.read();
```

probabilistic model on APIs

Raychev et. al., ACM PLDI'14

P(read | open) ~ 2/3
P(write | open) ~ 1/3

context γ

query:

```
f.open("file" | "r");

f. ?
```

3-gram model on tokens

Hindle et. al., ACM ICSE'12

P(open | f.) ~ 3/6
P(read | f.) ~ 2/6
P(write | f.) ~ 1/6

context γ

What should the context be?

"...All problems in computer science can be solved by **another level of indirection**..."

-- David Wheeler

"...All problems in computer science can be solved by **another level of indirection**..."

-- David Wheeler

key idea: synthesize a function f:

program

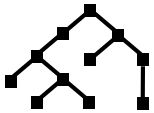
context

$\rightarrow \gamma$

Creating probabilistic models: our method

["Learning Programs from Noisy Data", ACM POPL'16, "PHOG: Probabilistic Model for Code", ACM ICML'16, "Probabilistic Model for Code with Decision Trees", ACM OOPSLA'16, "Synthesis for Char. Level Language Modeling, ICLR'17]

1. Pick a structure of interest, e.g., ASTs:



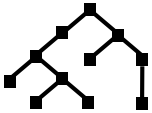
2. Define a DSL for expressing functions:
(can be Turing complete)

DSL

3. Synthesize $f_{best} \in$ DSL from Dataset $\boldsymbol{D}$:

$$f_{best} = \underset{f \in \text{DSL}}{\text{argmin}}\ \text{cost}(\boldsymbol{D}, f)$$

4. Use f_{best} to compute context and predict:

f_{best} () $\rightarrow$ γ

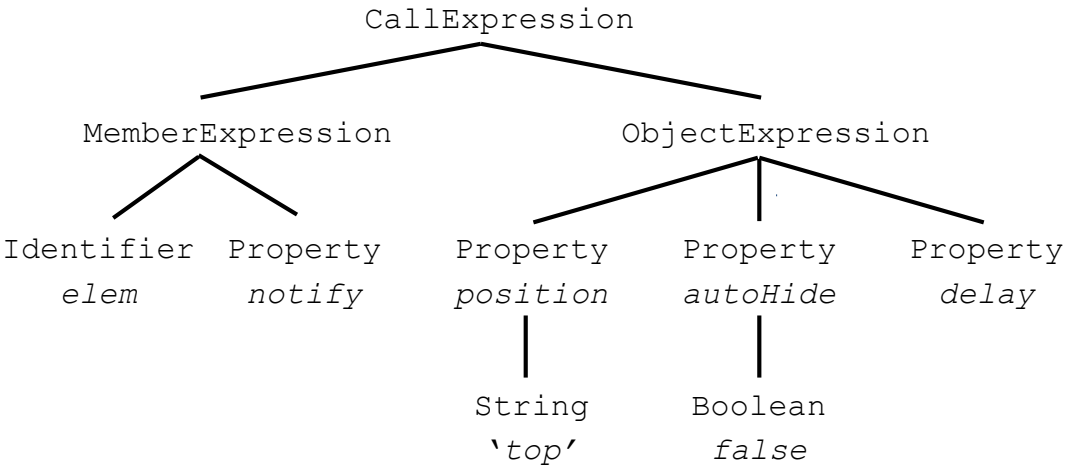
Step 1: Pick Structure of Interest

Let it be abstract syntax trees (ASTs) of programs

JavaScript program

```
elem.notify({
  position: 'top',
  autoHide: false,
  delay: 100
});
```

AST



Step 2: Define a DSL over structure

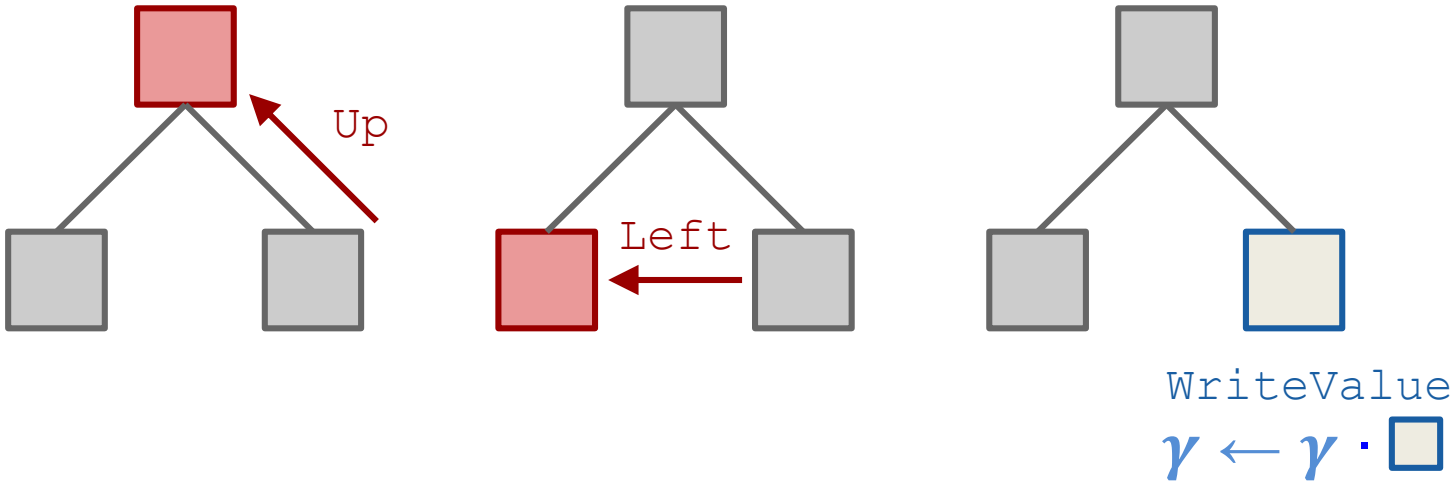
Syntax

```
TCond   ::=  ε | WriteOp TCond | MoveOp TCond

MoveOp  ::=  Up, Left, Right, DownFirst, DownLast,
             NextDFS, PrevDFS, NextLeaf,
             PrevLeaf,PrevNodeType, PrevNodeValue, PrevNodeContext

WriteOp ::=  WriteValue, WriteType, WritePos
```

Semantics



Step 3: synthesize $\boldsymbol{f_{best}}$

$$f_{best} = \underset{f \in \text{DSL}}{\text{argmin}}\ \text{cost}(\boldsymbol{D}, f)$$

Step 3: synthesize f_{best}

DSL

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

generate candidate f

dataset $\boldsymbol{D}$

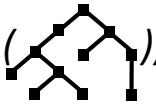
millions ($\approx 10^8$)

Synthesizer

f_{best} = argmin cost($\boldsymbol{D}$, f)
$f \in$ DSL

Build Probabilistic Model P

use $\boldsymbol{D}$ and f to compute

$P(element \mid \boldsymbol{f}$ ())

to scale: iterative synthesis on fraction of examples

$O(|D|)$

$cost(\boldsymbol{D}, f)$ = entropy(P)

Step 4: use f_{best} to predict

program

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

f_{best}	Context 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}

Deep3: Experimental Results

[Probabilistic Model of JavaScript]

Dataset **D:** 150,000 files Training Time: ~**100** hours f_{best} ~ **50,000** instr.

Probabilistic Model	**Accuracy** (APIs)
Last two tokens, Hindle et. al. [ICSE'12]	22.2%
Last two APIs, Raychev et. al. [PLDI'14]	30.4%
Deep3	**66.6%**

Details in: "Probabilistic Model for Code with Decision Trees", ACM OOPSLA'16

Applying the Concept to Natural Language

[Program Synthesis for Character Level Language Modeling, **ICLR'17**]

Dataset **D:**
Hutter Prize Wikipedia Dataset

Training Time: ~ **8** hours

f_{best} ~ **9,000** instr
uses a char-level DSL with state

Interpretable model, browse here:
http://www.srl.inf.ethz.ch/charmodel.html

Probabilistic Model	**Bits-per-Character**
7-gram (best)	1.94
Stacked LSTM (Graves 2013)	1.67
Char-based DSL synthesis	1.62
MRNN (Sutskever 2011)	1.60
MI-LSTM (Wu et al. 2016)	1.44
HM-LSTM* (Chung et al. 2016)	1.40

More Resources

Learning from Large Codebases,
PhD Thesis, ETH Zurich, 2016
ACM Doctoral Dissertation, Honorable Mention Award

Veselin Raychev

- http://plml.ethz.ch
- Dagstuhl Seminar on Big Code Analytics, Nov 2015
- Data sets, tools, challenges: http://learningfrombigcode.org

Summary

OPPORTUNITY

Advances in Programming Languages
[Automated Reasoning, Synthesis, Constraint Solving]

Advances in Machine Learning
[Deep Learning, Graphical Models, Language Models]

Data Google
[> 15 million public repositories]

Learning-based programming tools
new rules, new ideas, new opportunities

RICH PROBLEM SPACE

Applications	Code completion, Deobfuscation, Program synthesis, Feedback generation, Translation
Intermediate Representation	Sequences (sentences), Trees, Translation Table, Graphical Models (CRFs), Feature Vectors
Analyze Program (PL)	typestate analysis, scope analysis, control-flow analysis, alias analysis
Train Model (ML)	Neural Networks, N-gram language model, PCFGs, SVM, Structured SVM
Query Model (ML)	argmax P(y | x), y ∈ Ω, Greedy MAP inference

NEW PROBABILISTIC MODELS

1. **Pick** a structure of interest, e.g., trees:

2. **Define** a DSL for expressing functions: (can be Turing complete)

```
TCond   ::=  ε | WriteOp TCond | MoveOp TCond
MoveOp  ::=  Up, Left, Right, DownFirst, DownLast,
             NextDFS, PrevDFS, NextLeaf, PrevLeaf,
             PrevNodeType, PrevNodeValue,
WriteOp ::=  WriteValue, WriteType, WritePos
```

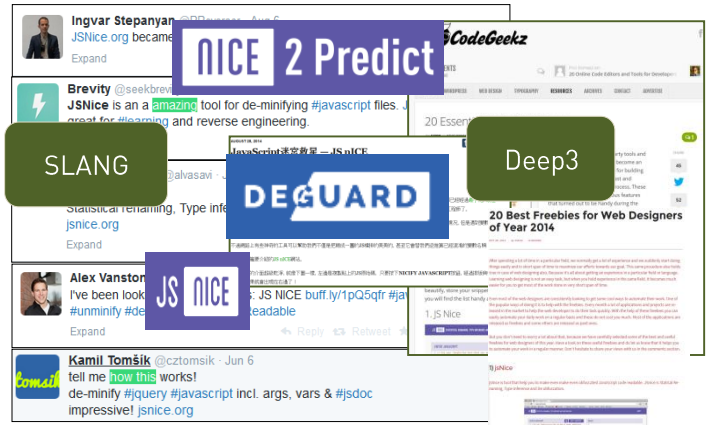
3. **Synthesize** $f_{best} \in$ DSL from Dataset D:

$$f_{best} = \underset{f \in \text{DSL}}{\operatorname{argmin}}\ \text{cost}(D, f)$$

4. **Use** f_{best} on new structures:

f_{best} () → y

PRACTICAL IMPACT



more: http://plml.ethz.ch

ETH Zürich