Machine Learning for Programming

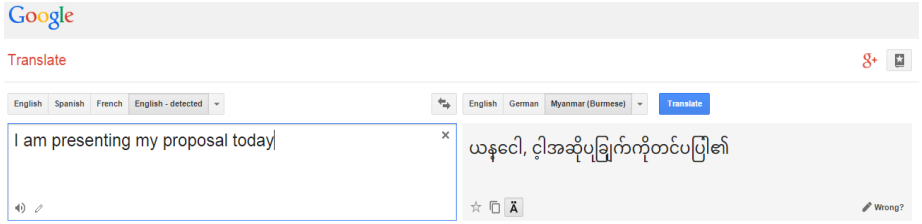
Martin Vechev
ETH Zurich and DeepCode.ai

ETH Zürich

DEEPCODE

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

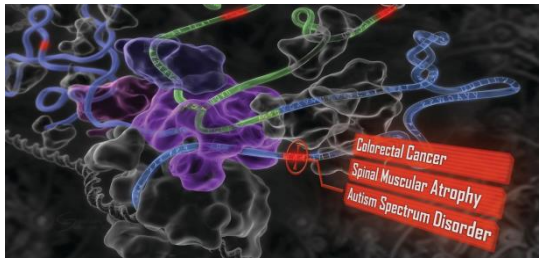
Natural Language Processing
(e.g., machine translation)



Computer Vision
(e.g., image captioning)

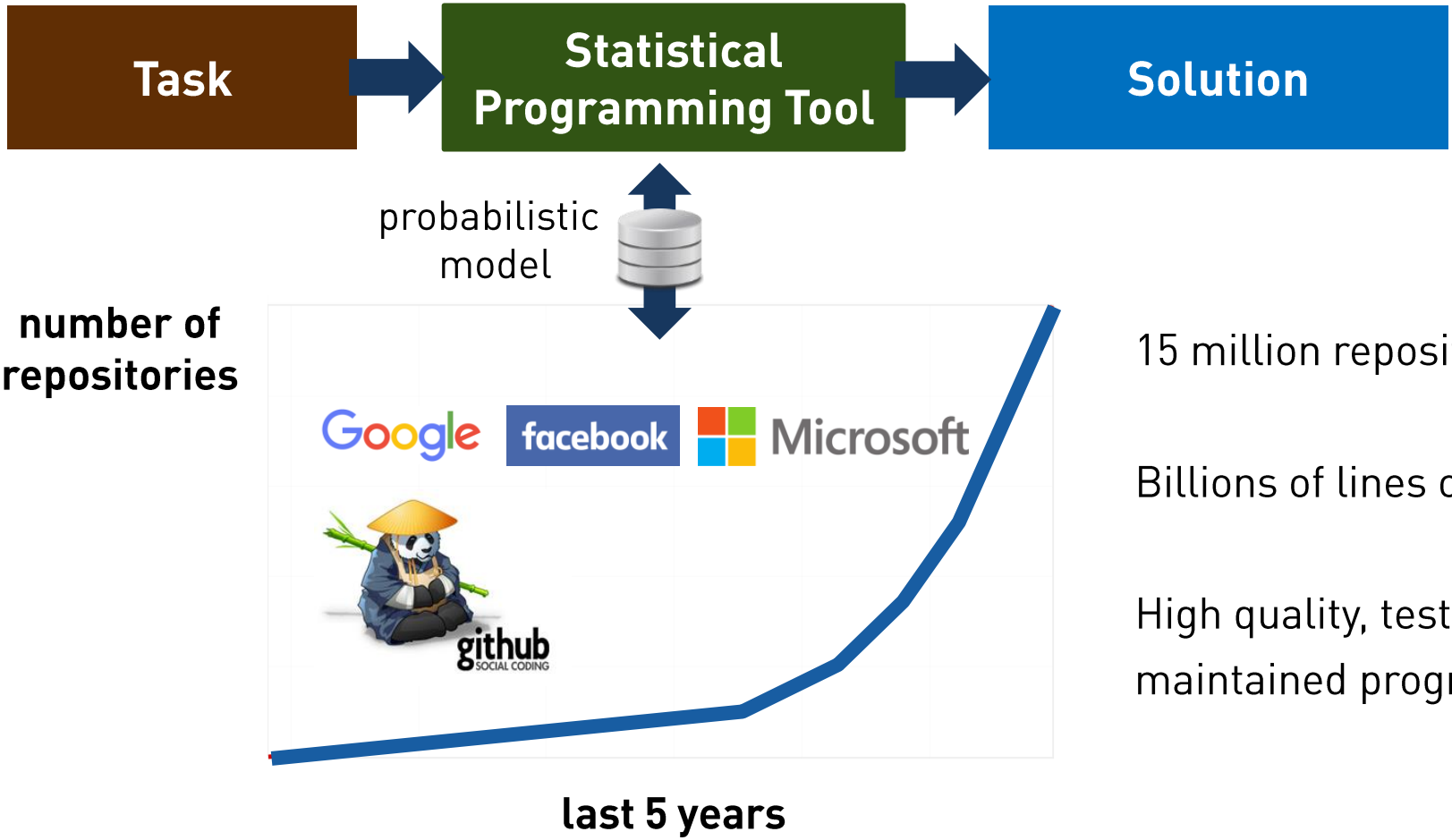
A group of people shopping at an outdoor market. There are many vegetables at the fruit stand.

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]

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

Data
[> 15 million public repositories]

Confluence of streams

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

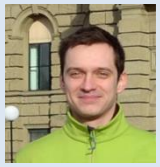
Machine Learning for Programs

Probabilistically likely solutions to problems hard to solve otherwise

Joint work with :


Veselin Raychev


Andreas Krause


Pavol Bielik

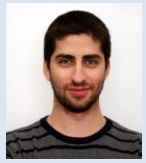

Christine Zeller


Svetoslav Karaivanov


Pascal Roos


Benjamin Bichsel


Timon Gehr


Petar Tsankov


Mateo Panzacchi

Publications

- Program Synthesis for Character Level Language Modeling, **ICLR'17**
- Learning a Static Analyzer from Data, **CAV'17**
- 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**
- Code Completion with Statistical Language Models, **ACM PLDI'14**
- Machine Translation for Programming Languages, **ACM Onward'14**

Statistical Engines

apk-deguard.com

DEGUARD

DEEP3

jsnice.org

JS NICE

nice2predict.org

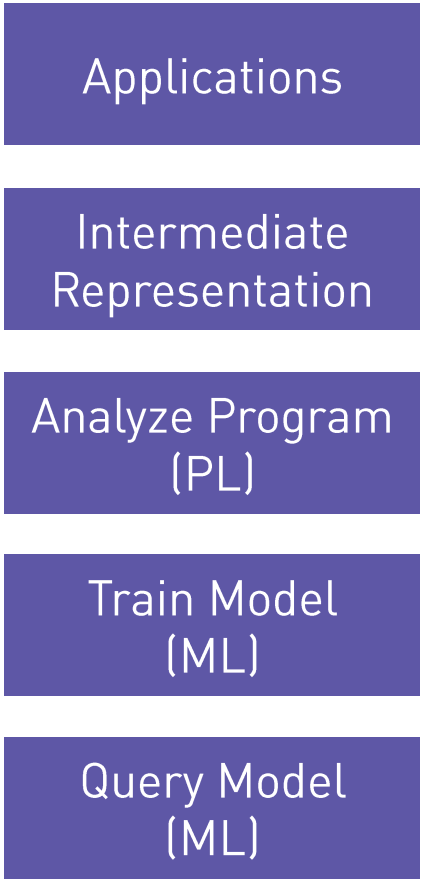
SLANG

NICE 2 Predict

more: http://plml.ethz.ch

Dimensions:

Machine Learning for Programming



Dimensions:
Machine Learning for Programming

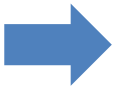
Applications	Code completion	Deobfuscation	Program synthesis	Feedback generation	Translation
Intermediate Representation	Sequences (sentences)	Trees	Translation Table	Feature Vectors	Graphical Models (CRFs)
Analyze Program (PL)	typestate analysis	scope analysis	control-flow analysis	alias analysis	
Train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



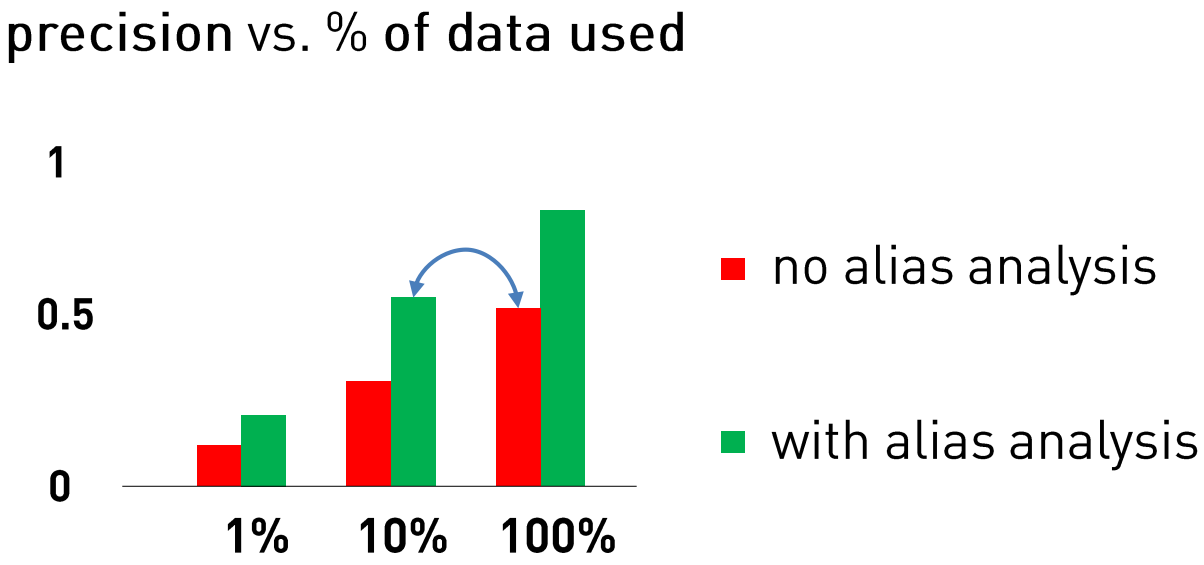
```
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
camera.unlock();
SurfaceHolder holder = getHolder();
holder.addCallback(this);
holder.setType(SurfaceHolder.STP);
MediaRecorder r = new MediaRecorder();
r.setCamera(camera);
r.setAudioSource(MediaRecorder.AS);
r.setVideoSource(MediaRecorder.VS);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

Statistical language models
Recurrent neural networks

+

Typestate analysis
Alias analysis

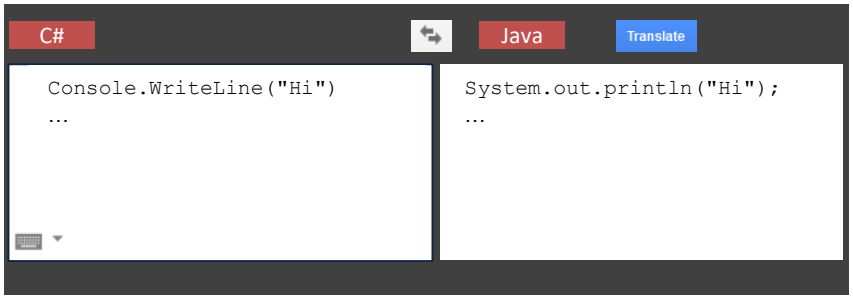
Importance of Static (Semantic) Analysis



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 ✓ 200,000 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

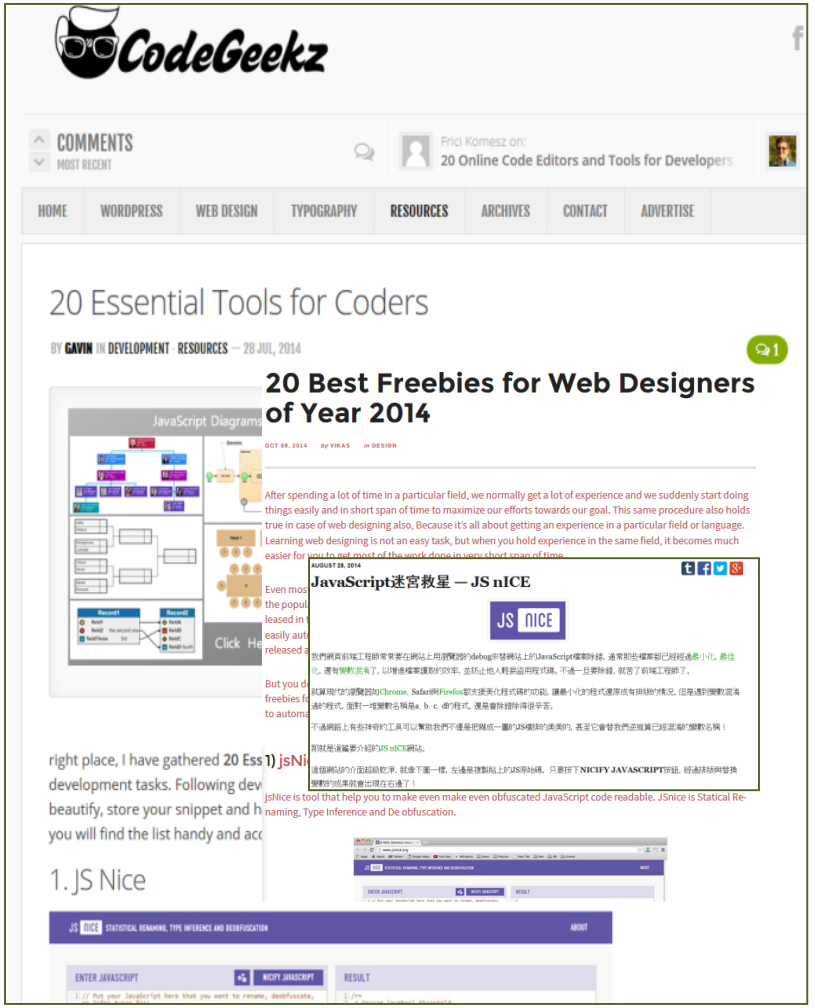
Ingvar Stepanyan @RReverser · Aug 6
JSNice.org became my must-have tool for code deobfuscation.
Expand Reply Retweet Favorite More

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

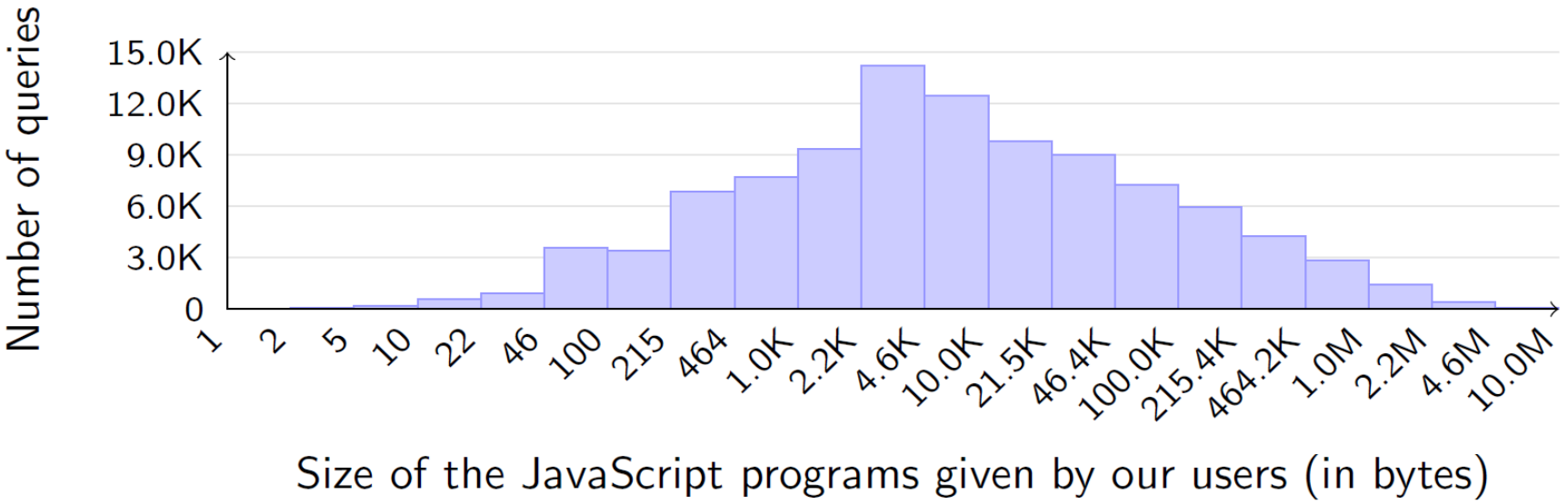
Alex Vanston @mrvdot · Jun 7
I've been looking for this for years: JS NICE buff.ly/1pQ5qfr #javascript #unminify #deobfuscate #makeItReadable
Expand Reply Retweet Favorite More

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



JSNice: over a year of usage

May 2015 – May 2016



Total: 12GB of code

JSNice

```
function chunkData(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;
```

```
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;
```

Key Challenges

Facts to be predicted are dependent

Prediction should be fast (real-time)
(hard, millions of possible choices)

Key Idea

Phrase the problem of predicting program facts as

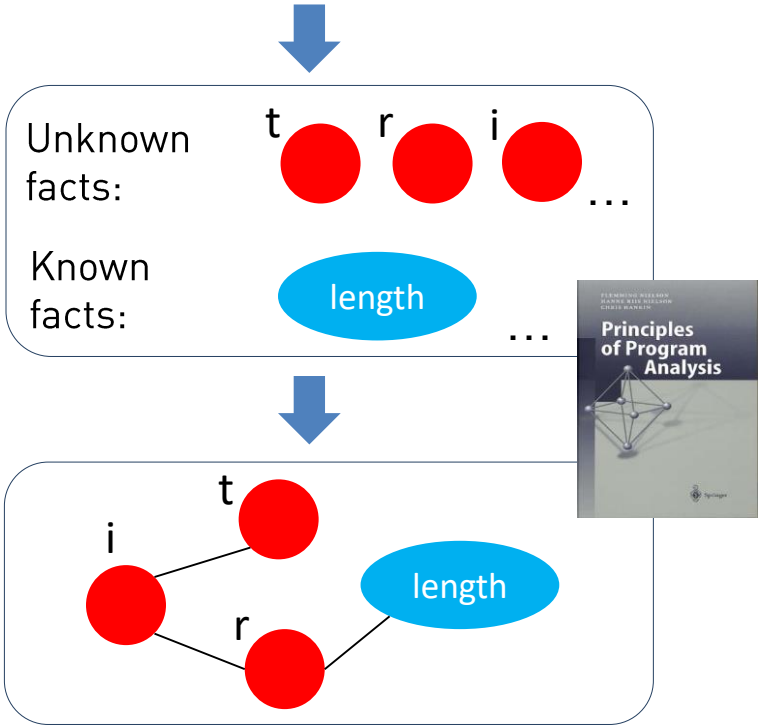
Structured Prediction for Programs

MAP inference

```
function chunkData(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;
```

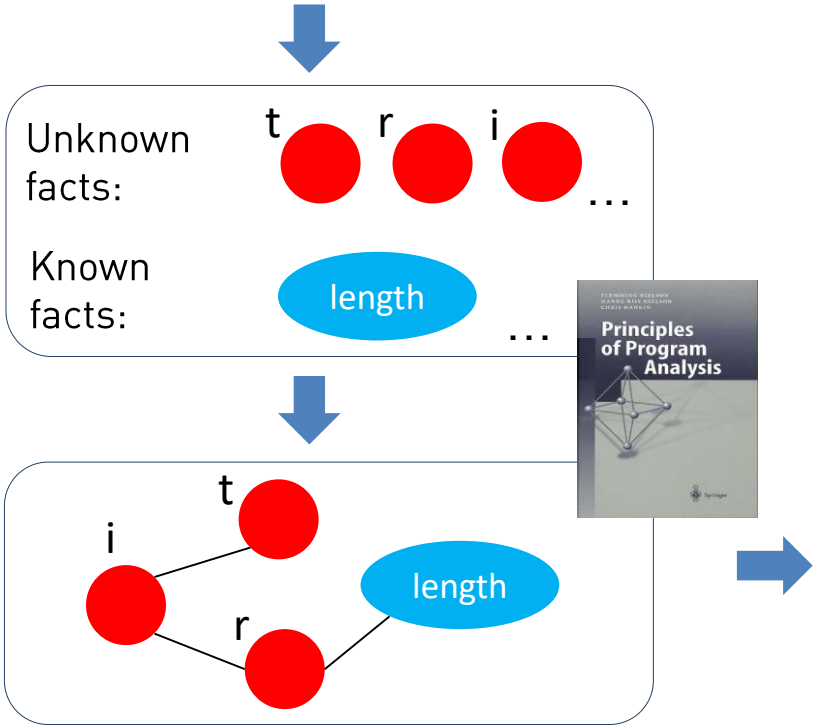
MAP inference

```
function chunkData(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;
```

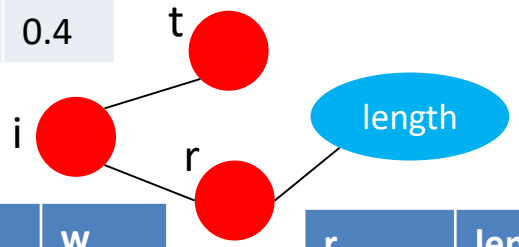


MAP inference

```
function chunkData(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;
```



i	t	w
i	step	0.5
j	step	0.4

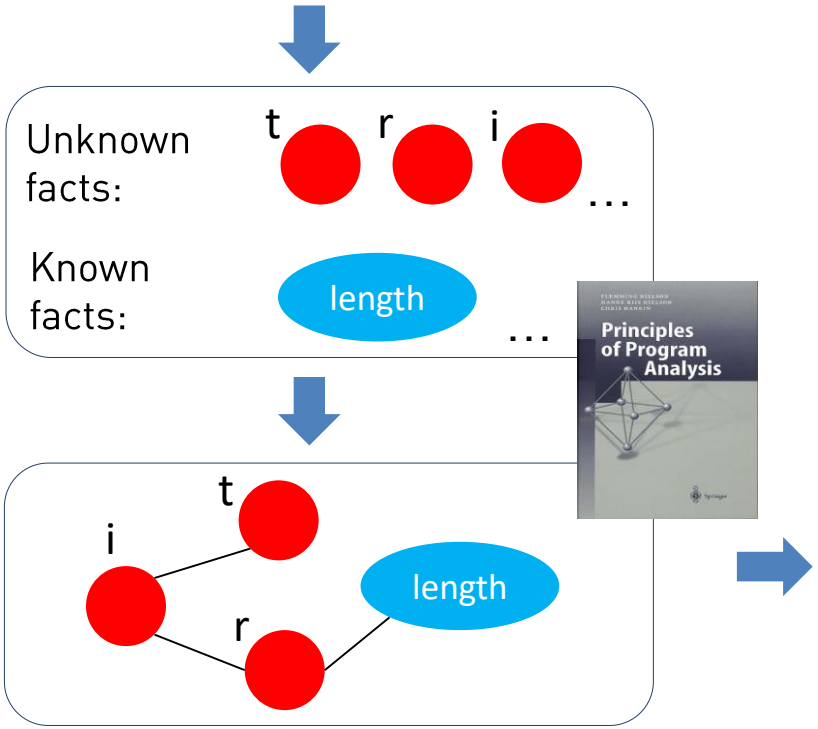


i	r	w
i	len	0.6
i	length	0.3

r	length	w
length	length	0.5
len	length	0.3

MAP inference

```
function chunkData(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;
```



argmax $\boldsymbol{w}^T \boldsymbol{f}(i, t, r, length)$

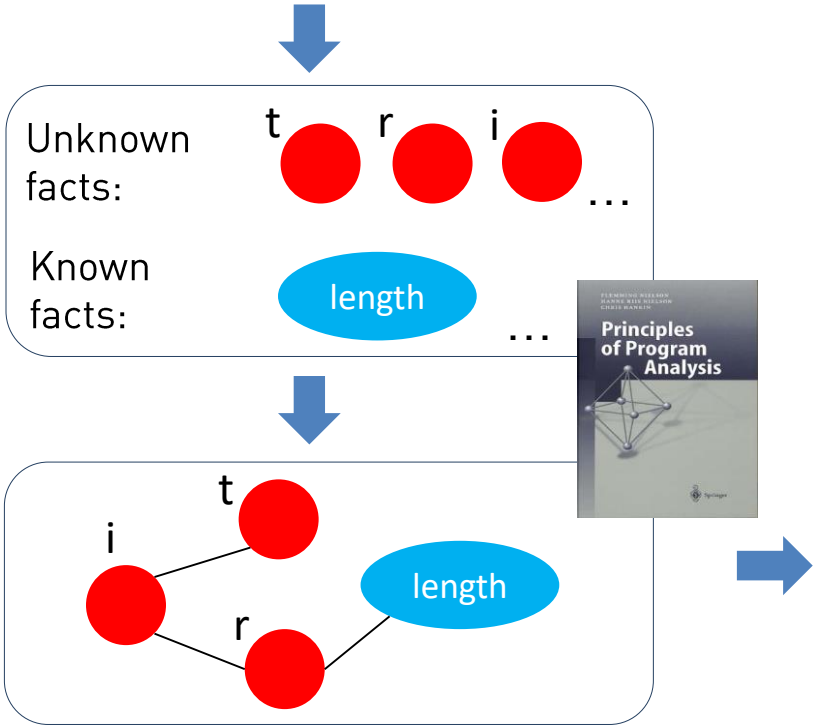
i	t	w
i	step	0.5
j	step	0.4

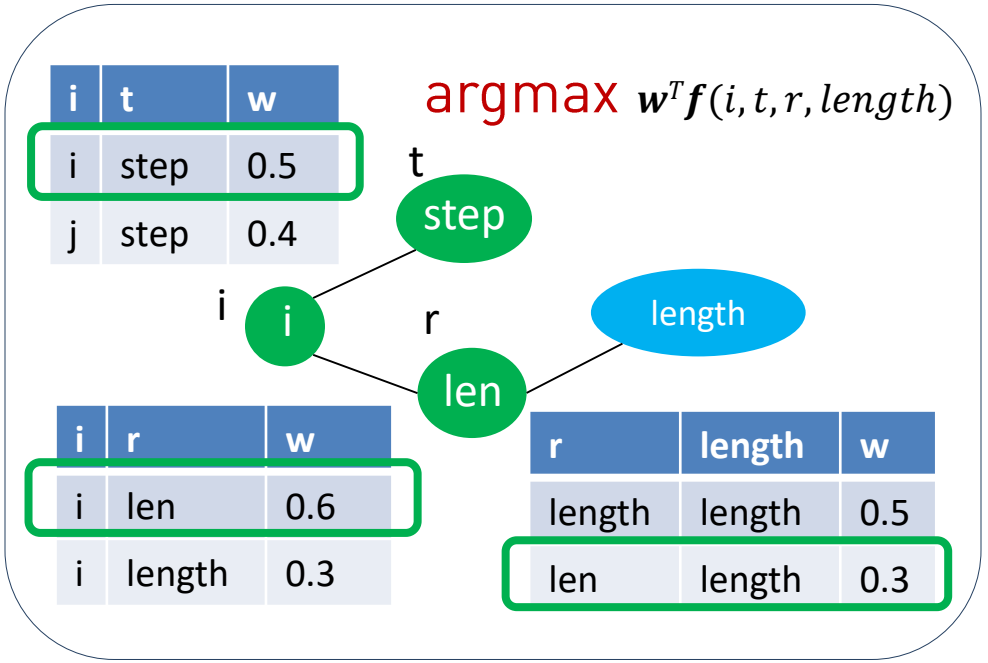
i	r	w
i	len	0.6
i	length	0.3

r	length	w
length	length	0.5
len	length	0.3

MAP inference

```
function chunkData(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;
```

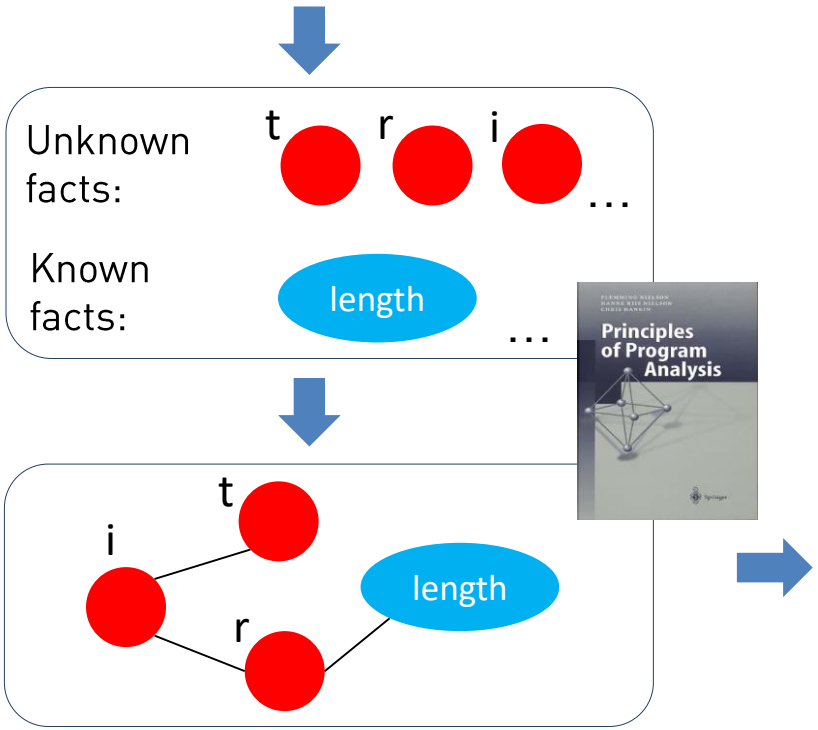


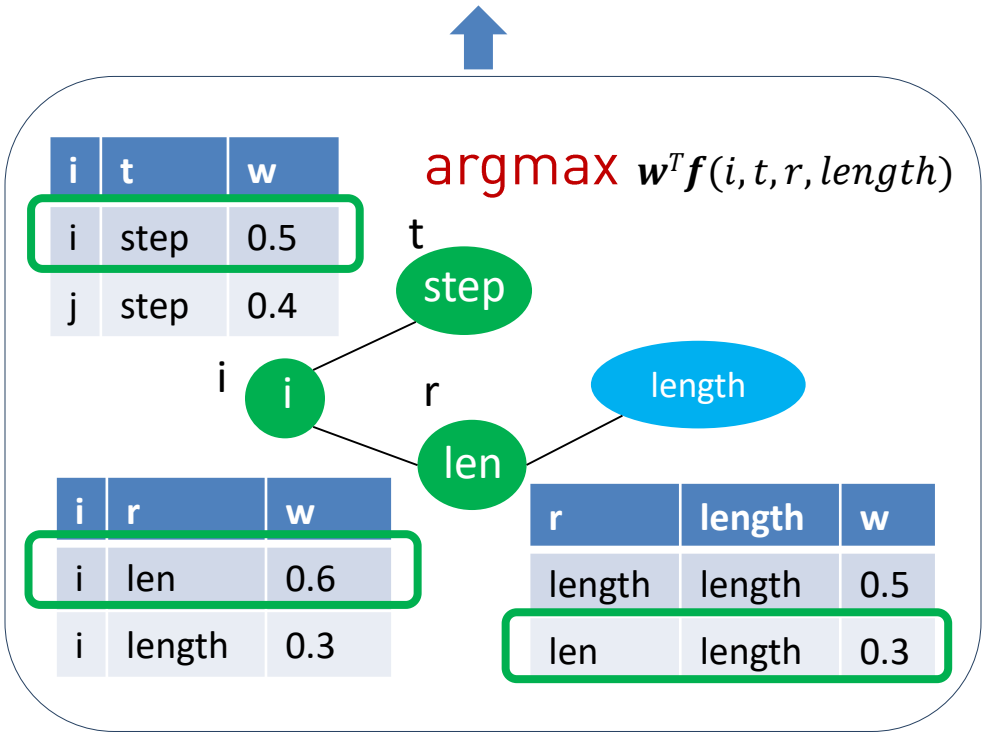


MAP inference

```
function chunkData(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;
```

```
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;
```





A glimpse of learning…

[N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007]

$$P(y \mid x) = \frac{1}{Z(x)} \exp(\boldsymbol{w}^T \boldsymbol{f}(y, x))$$

Given a data set: $D = \{ \mathbf{x}^j , \mathbf{y}^j \}_{j=1..n}$ learn weights $\boldsymbol{w}^T$

Optimization objective (max-margin training):

$$\forall j \; \forall \mathbf{y} \; \underbrace{\Sigma \, w_i f_i(\mathbf{x}^{(j)},\mathbf{y}^{(j)})}_{\text{Given prediction}} \geq \underbrace{\Sigma \, w_i f_i(\mathbf{x}^{(j)},\mathbf{y})}_{\text{any other prediction}} + \underbrace{\boldsymbol{\Delta}(\mathbf{y},\mathbf{y}^{(j)})}_{\text{margin}}$$

for all samples

Given prediction is better than any other prediction by a margin

Avoids expensive computation of the partition function $Z(x)$

JSNice

[Predicting program properties from Big Code, ACM POPL 2015]

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

Atlassian Bitbucket

program analysis

MAP inference

transform

Prediction Phase

150MB

Learning Phase

Conditional Random Field

$P(y \mid x)$

program analysis

SSVM learning

max-margin training

alias,call analysis

7M feature functions for names
70K feature functions for types

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

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.

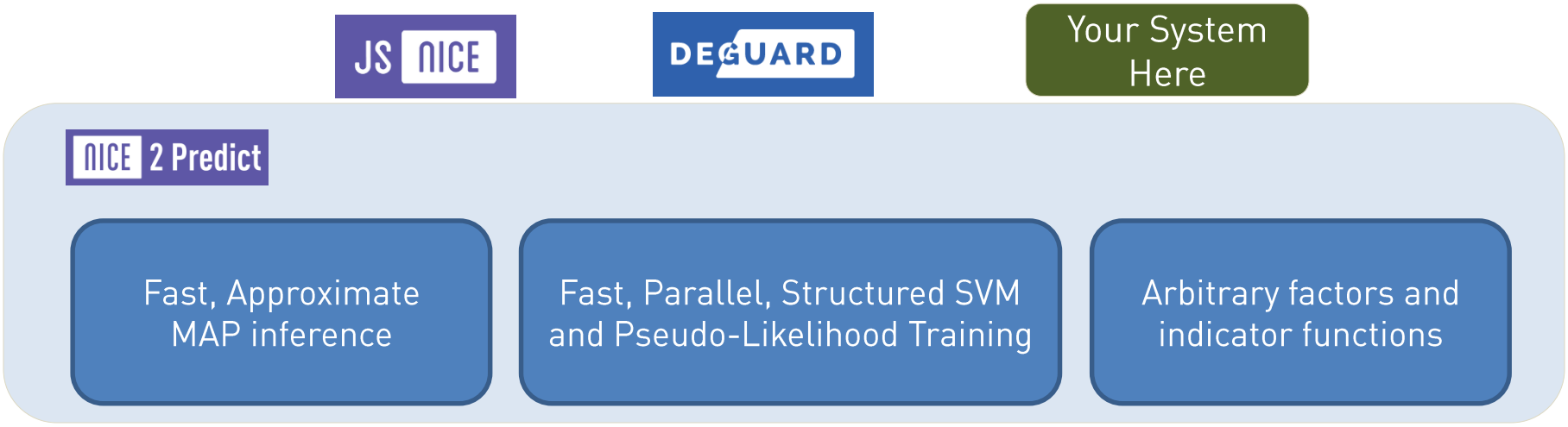
Nice2Predict.org:

scalable structured prediction framework

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

fully, open sourced, Apache license

used by various groups worldwide

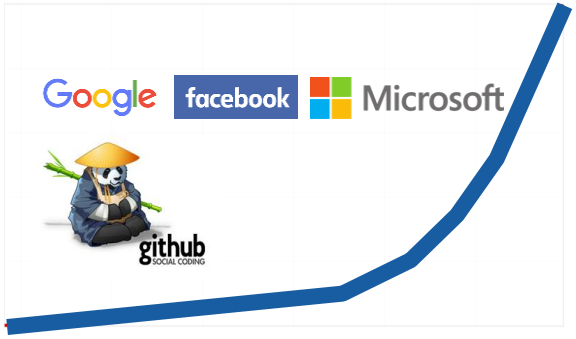


Fundamental Problem

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:

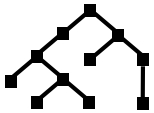
program context

$\rightarrow \gamma$

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:



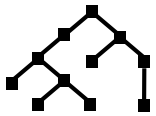
2. Define a DSL for expressing functions:
(can be Turing complete)

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} () → γ

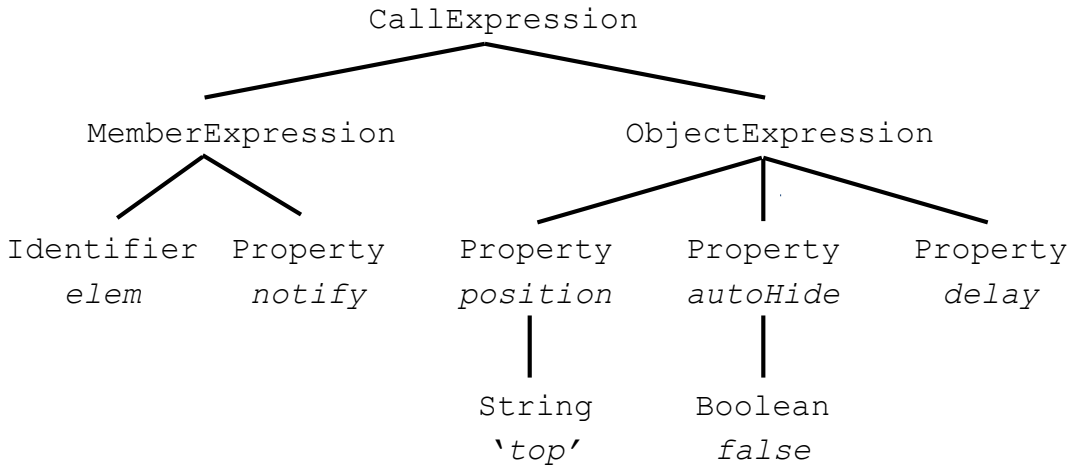
Step 1: Pick Structure of Interest

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

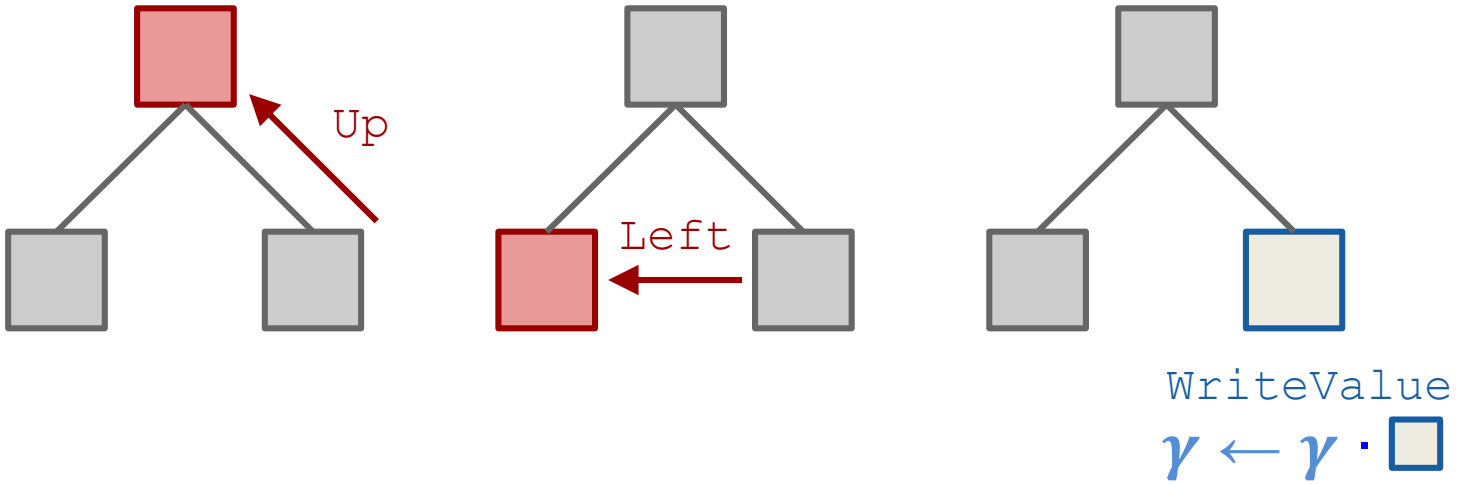
Syntax

```
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 f_{best}

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

Step 3: synthesize f_{best}

DSL

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

generate candidate f

dataset D

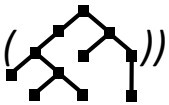
millions ($\approx 10^8$)

Synthesizer

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

Build Probabilistic Model P

use D and f to compute

$P(element \mid f($  $))$

to scale: iterative synthesis on fraction of examples

$O(|D|)$

cost(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

Learning a Static Analyzer from Data

[Pavol Bielik, Veselin Raychev, M.V., CAV'17]

```
function isBig(v) {
 return v < this.length
}
[12, 5].filter(isBig);
```

VarPtsTo("global", *h*)
checkIfInsideMethodCall
checkMethodCallName
checkReceiverType
checkNumberOfArguments ...

VarPtsTo(*this*, *h*)

Can be understood by experts

Found issues in Facebook's Flow

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 Github Google
[> 15 million public repositories]

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

RICH PROBLEM SPACE

Applications	Code completion, Deobfuscation, Program synthesis, Translation, Feedback generation
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, PCFGs, N-gram language model, 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
```

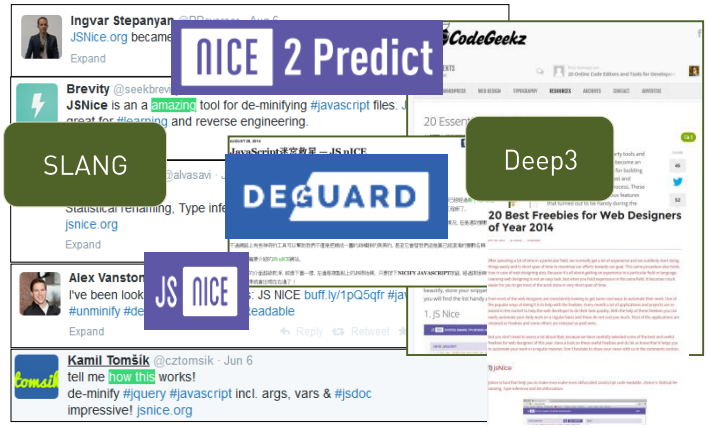
3. **Synthesize** $f_{best} \in$ DSL from Dataset D:

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

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

$$f_{best}\ (\ \ ) \rightarrow y$$

PRACTICAL IMPACT



more: http://plml.ethz.ch

ETH Zürich