Statistical Deobfuscation with “Big Code”

14.05.2018

“Big Code”

A new still untapped corpus of data.

Atlassian Bitbucket

GitLab

Plenty of well-maintained code repositories

But using the data requires understanding program analysis **and** machine learning

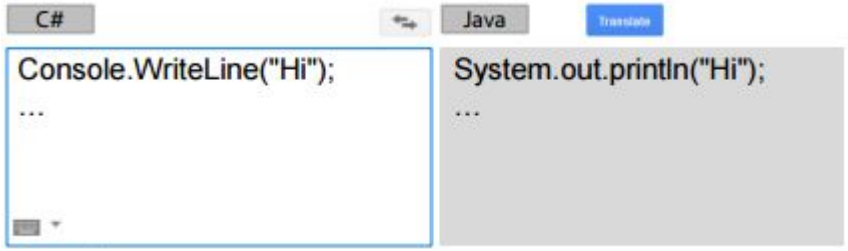
80+ million projects hosted only in GitHub

“Big Code” applications

Write new code:
Code Completion

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

Port code:
Programming Language Translation



Understand code/security:
Code Deobfuscation
Type Prediction

JSNICE DEGUARD

Find issues:
Statistical Defect Prediction

DEEPCODE Bad for security

```
...
MessageDigest.getInstance(“MD5”);
```

Code completion, Slang PLDI'14

Statistical API code completion system

Motivated by large number of shared APIs used by many projects. Developed for Java/Android

Apache Commons
http://commons.apache.org/

tapestry

Slang: Capabilities

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

?

```
MediaRecorder rec = new MediaRecorder();
```

?

```
rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
```

?

```
rec.setOutputFile("file.mp4");
...
```

...

Slang: Capabilities

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

camera.unlock();

MediaRecorder rec = new MediaRecorder();

rec.setCamera(camera);

rec.setAudioSource(MediaRecorder.AudioSource.MIC);
rec.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
rec.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);

rec.setAudioEncoder(1);
rec.setVideoEncoder(3);

rec.setOutputFile("file.mp4");
...
```

Key insight

Think of regularities in code like regularities in natural languages

i.e. can be captured by the right machine learning model

We want to learn that

```
MediaRecorder rec = new MediaRecorder();
```

is before

```
rec.setCamera(camera);
```

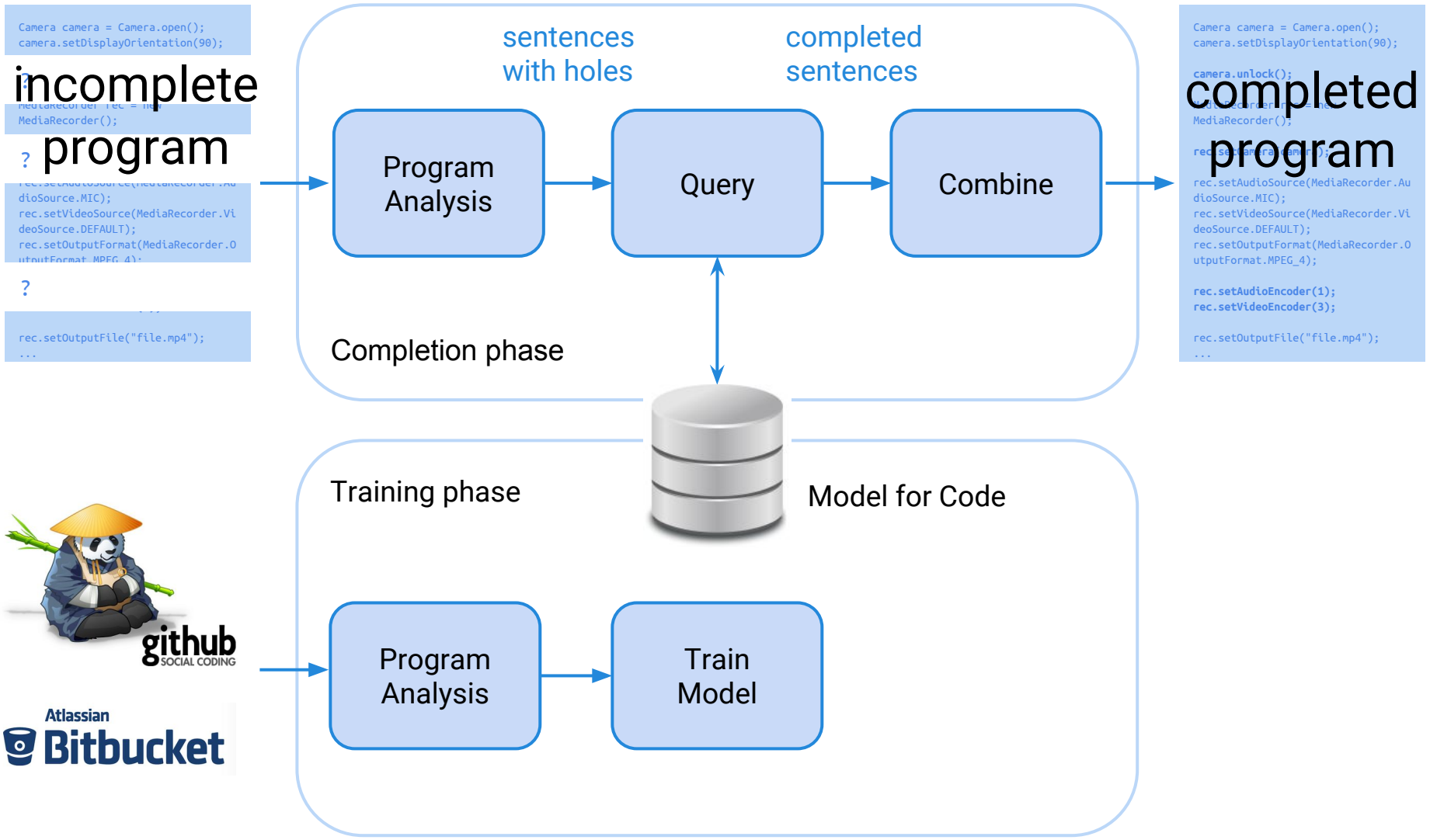
like in natural languages

```
Hello
```

is before

```
World !
```

The Slang system

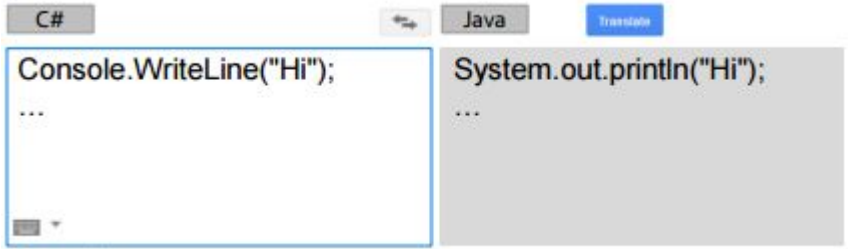


“Big Code” applications

Write new code:
Code Completion

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

Port code:
Programming Language Translation



Understand code/security:
Code Deobfuscation
Type Prediction

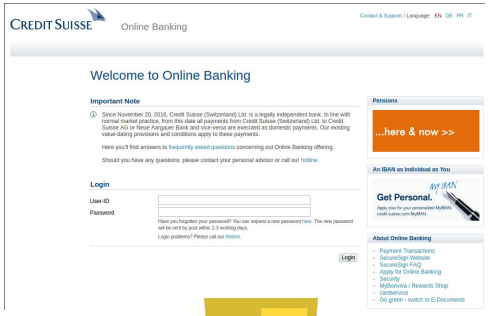
JSNICE DEGUARD

Find issues:
Statistical Defect Prediction

DEEPCODE Bad for security

```
...
MessageDigest.getInstance(“MD5”);
```

Deobfuscation



JavaScript

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

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

Security Analyst

Online systems

Web
JavaScript

http://jsnice.org/

Android
Applications

http://apk-deguard.com/

Example deobfuscation

What is this doing?

```
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;
}
```

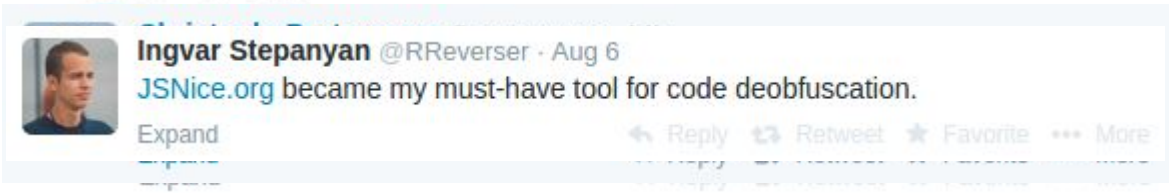
JS NICE

Types hard to predict:
Google Closure Compiler, TypeScript,
Facebook Flow all infer {?}

```
/**
 * @param {string} str
 * @param {number} step
 * @return {?}
 */
function chunkData(str, step) {
  /** @type {Array} */
  var colNames = [];
  var len = str.length;
  /** @type {number} */
  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;
}
```

predict variable names

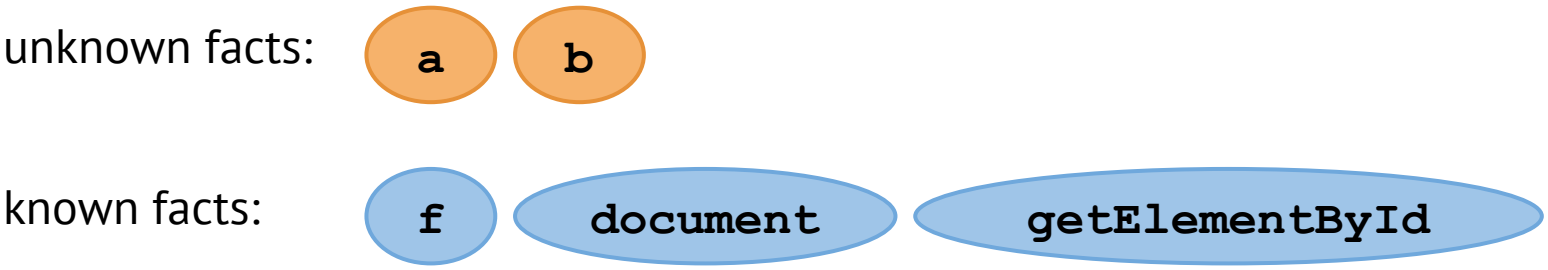
easy to infer



Predicting names

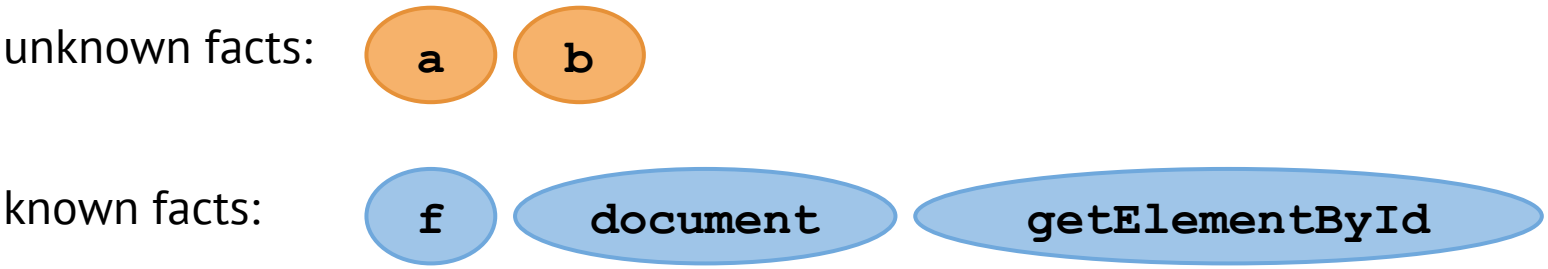
What can be “soundly” renamed in JavaScript?

```
function f(a) {
  var b = document.getElementById(a);
  return b;
}
```



Predicting names

Goal: predict **unknown** facts given these **known** facts



Predicting type annotations

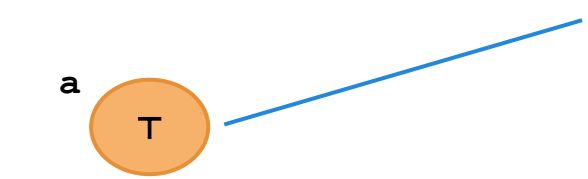
Type annotations for Google Closure Compiler

(available training data)

```
function inc(a) {
  return a+1;
}
```

Standard type inference returns:

unknown facts:



to be predicted

known facts:

Key point: All facts are related

(for both, names and types)

Challenges

- Predicted facts are dependent
- Predictions must satisfy constraints
- Must learn from huge codebases
- Prediction should be fast (real time)

General prediction approach

We phrased the general problem of predicting program facts into a framework based on:

- **Model:** Conditional Random Fields (CRFs)
- **Query:** MAP inference
- **Learning:** Structured SVM

CRF - introduction

Example: choose the right words

I go to a

venue)		city)		transport)
lecture	in	Zug	by	bus
vacation		Zurich		tram

constraints:

1) It is May 14 ⇒ lecture
2) Zug has no trams

CRF - introduction

Example: choose the right words

I go to a

v)
lecture

in

c)
Zug

Zurich

by

t)
bus

tram

constraints:

1) It is May 14 ⇒ lecture
2) Zug has no trams

CRF - introduction

A possible choice (the most likely):

I go to a v) lecture in c) Zurich by t) tram

constraints:

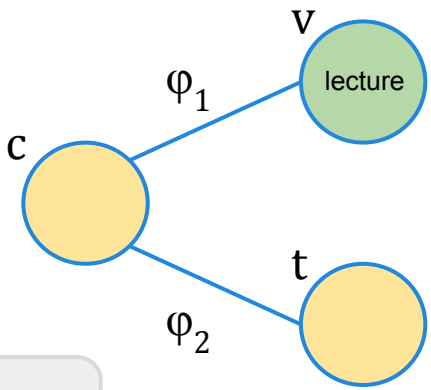
1) It is May 14 ⇒ lecture
2) Zug has no trams

Conditional Random Fields

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

- Undirected graphical model
- Captures **dependencies** between facts to be predicted
- Captures **conditional distribution** on known facts

I go to a [v) lecture] in [c) Zug / Zurich] by [t) bus / tram]



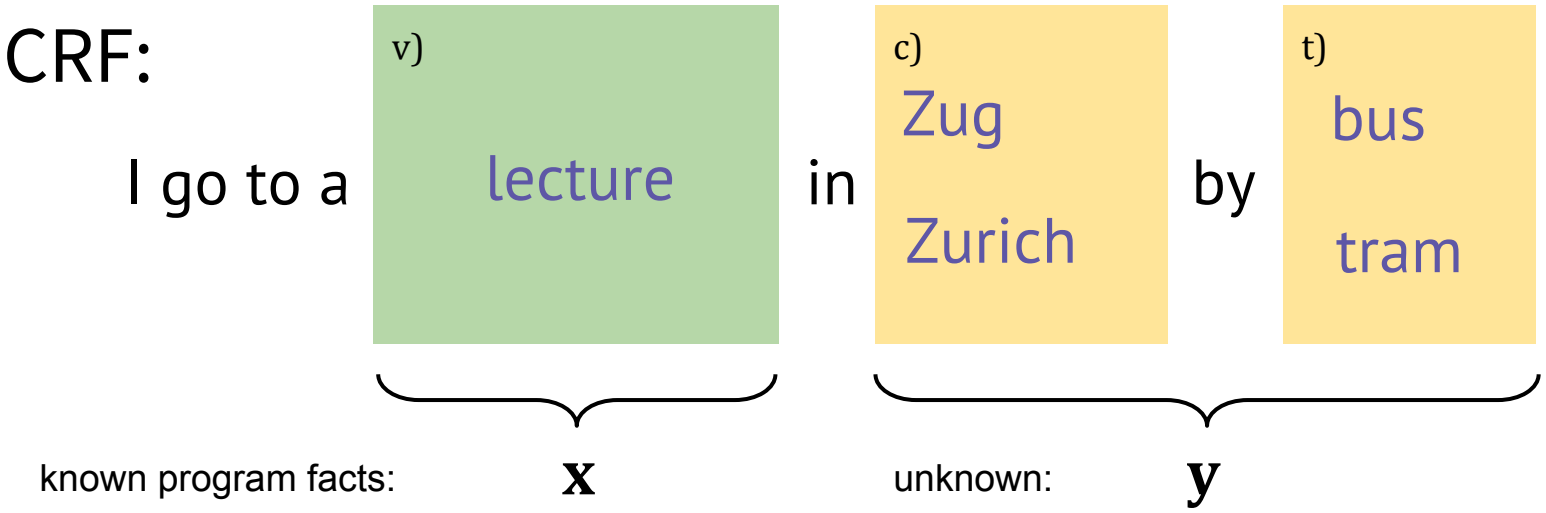
v - known fact, c and t - random variables (unknown facts)

φ_1, φ_2 - scoring functions

$$\mathbf{P}(c,t|v) = \varphi_1(v,c) * \varphi_2(c,t) / Z(v)$$

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

CRF notation



Key Point: CRFs model conditional probabilities
a.k.a discriminative model

CRF notation

CRF:

$$\mathbf{P(y|x)} = 1/\mathbf{Z} \prod \varphi_i(\mathbf{x},\mathbf{y})$$ (factor graph)

known program facts: $\mathbf{x}$ unknown: $\mathbf{y}$

Key Point: CRFs model conditional probabilities
a.k.a discriminative model

General prediction approach

- **Model:** Conditional Random Fields (CRFs)
- **Query:** MAP inference
- **Learning:** Structured SVM

Query

Our goal is to find the most likely assignment of $\mathbf{y}$ 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: for this query 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**:

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

Good news: for this query the expensive partition function $\mathbf{Z}(\mathbf{x})$ is unnecessary

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

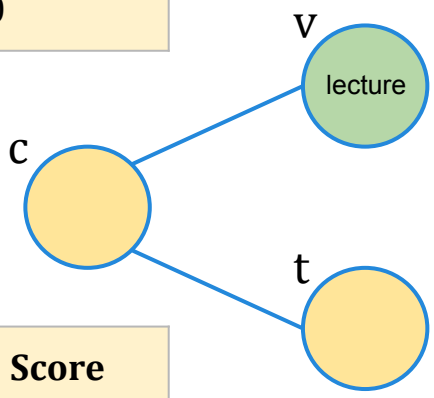
Our solution: approximate algorithms

MAP inference example

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

φ_1

v	c	Score
lecture	Zug	2
lecture	Zurich	10



φ_2

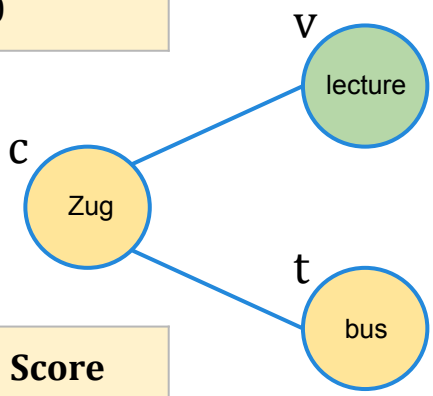
c	t	Score
_ *(any city)*	bus	5
Zurich	tram	10
Zug	tram	0

MAP inference example

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

φ_1

v	c	Score
lecture	Zug	2
lecture	Zurich	10



φ_2

c	t	Score
_ *(any city)*	bus	5
Zurich	tram	10
Zug	tram	0

Maximize product of scores:

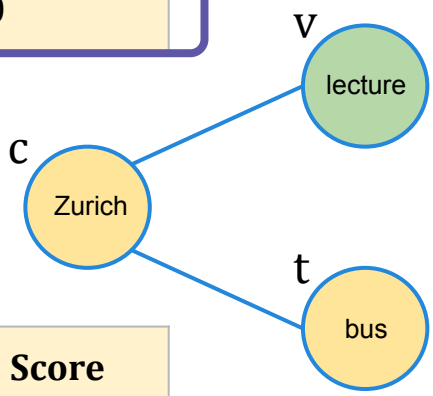
$$2*5 = 10$$

MAP inference example

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

φ_1

v	c	Score
lecture	Zug	2
lecture	Zurich	10



φ_2

c	t	Score
_ *(any city)*	bus	5
Zurich	tram	10
Zug	tram	0

Maximize product of scores:

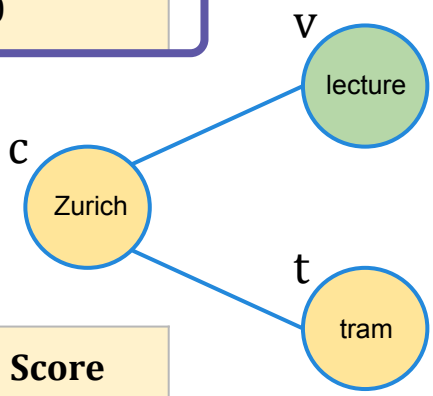
$$10*5 = 50$$

MAP inference example

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

φ_1

v	c	Score
lecture	Zug	2
lecture	Zurich	10



φ_2

c	t	Score
_ *(any city)*	bus	5
Zurich	tram	10
Zug	tram	0

Maximize product of scores:

$$10*10 = 100$$

Guaranteed to satisfy constraints

General prediction approach

- **Model:** Conditional Random Fields (CRFs)
- **Query:** MAP inference
- **Learning:** Structured SVM

Learning

CRF - two equivalent notations

$$P(y|x) = 1/Z \prod_i \varphi_i(x,y)$$

$$P(y|x) = 1/Z \exp \sum_i \lambda_i f_i(x,y)$$

φ_1

v	c	Score
lecture	Zug	2
lecture	Zurich	10

$f_1 = 1$ **if** v=lecture $\wedge$ c=Zug, 0 **otherwise** $\lambda_1 = \log 2$

$f_2 = 1$ **if** v=lecture $\wedge$ c=Zurich, 0 **otherwise** $\lambda_2 = \log 10$

Learning

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

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:

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

for all training data samples

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

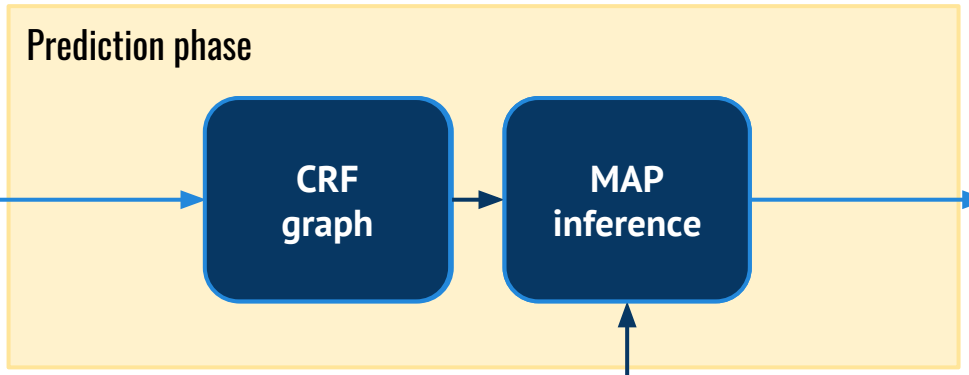
Training procedure:

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

- Stochastic (sub-) gradient descent
- Uses MAP inference as a subroutine
- no partition function Z

General approach: summary

```
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;
}
```




```
/**
 * @param {string} str
 * @param {number} step
 * @return {?}
 */
function chunkData(str, step) {
  /** @type {Array} */
  var colNames = [];
  var len = str.length;
  /** @type {number} */
  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;
}
```


feature weights

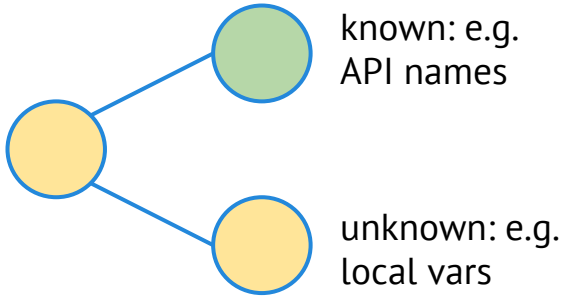
Remaining question

Predicting names and types are two **instantiations** of the **general approach** discussed before.

Remaining:

We need to build CRF graphs from programs.

```
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;
}
```



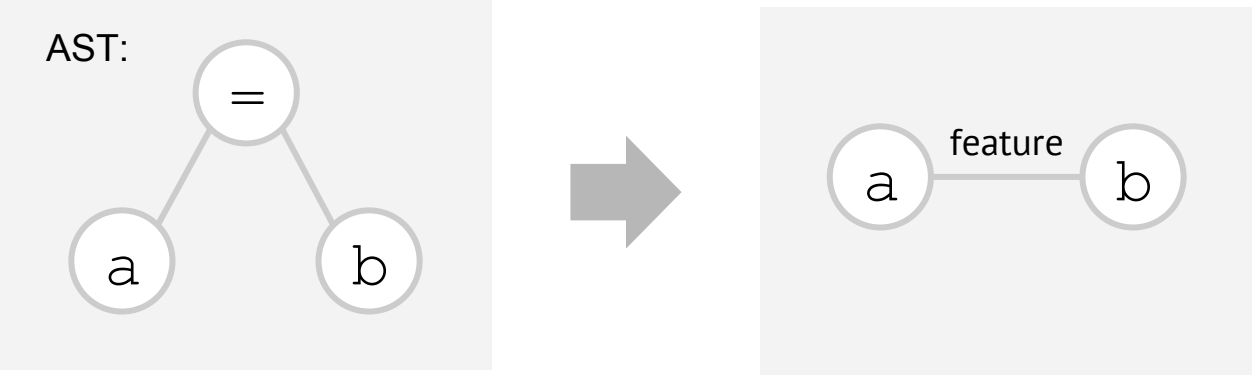
Nodes and constraints: already discussed

Features

Edges in CRF graph define features

We put edges depending on AST relationships of elements

- some typing rules are subset of these features



Training

Training data for names and types

10’517 JavaScript projects from GitHub

Simple detector for minified files removes code with no good names and types

325’501 JavaScript files

Learned 7’627’484 features for names and 70’052 features for types

Finally, we took 50 different JavaScript projects to evaluate on (from bitbucket)

Evaluation results

	Names Accuracy	Types Precision	Types Recall
Baseline	25.3%	37.8%	-
Independent predictions	54.1%	84.0%	56.0%
JSNice	**63.4%**	81.6%	**66.9%**

Structured Prediction is Critical

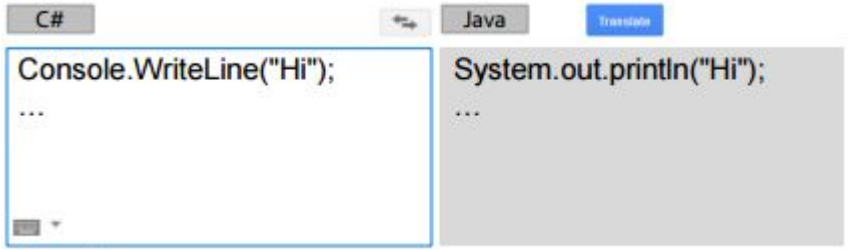
Types: more programs typecheck with predicted types than with manually provided types

“Big Code” applications

Write new code:
Code Completion

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

Port code:
Programming Language Translation



Understand code/security:
Code Deobfuscation
Type Prediction

JSNICE DEGUARD

Find issues:
Statistical Defect Prediction

DEEPCODE Bad for security

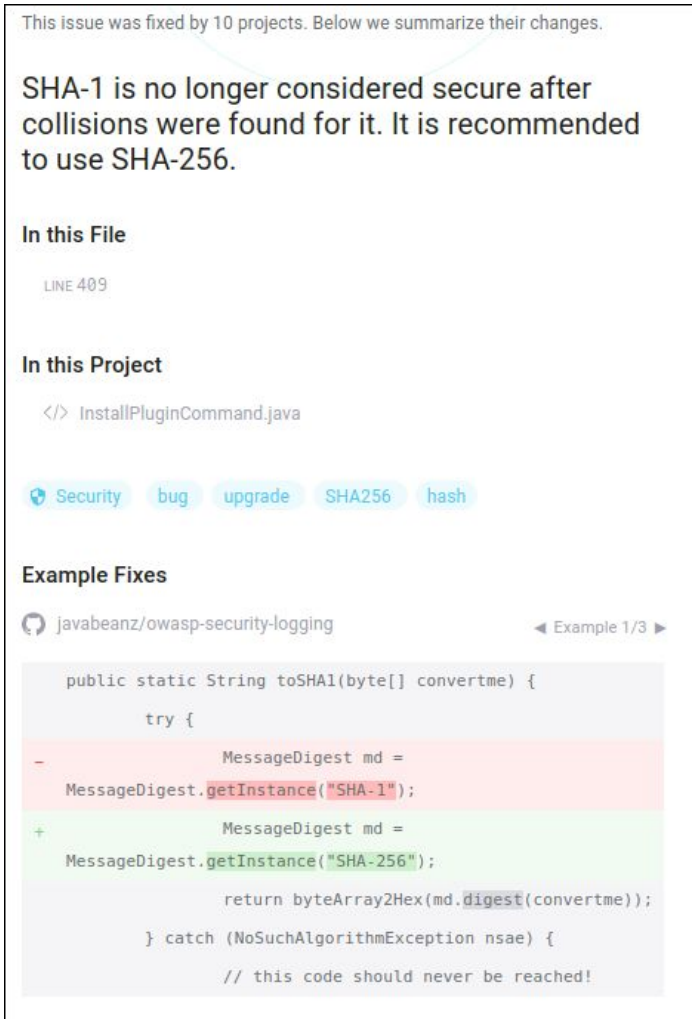
```
...
MessageDigest.getInstance(“MD5”);
```

DeepCode

ETH spin-off

just got its first funding

Learned static analysis makes probabilistically likely suggestions for your code



DEEPCODE

Spinoff ETH zürich

Shameless plug

We are hiring

If you want to work on
program analysis and/or
machine learning

or want to create a great deepcode.ai app