Machine Learning for Programming

Martin Vechev
Department of Computer Science
ETH Zurich
How can we leverage “Big Code”?
Machine Learning for Programming

Probabilistically likely solutions to problems impossible to solve otherwise

Publications:
Predicting Program Properties from “Big Code”, *ACM POPL’15*
Programming with Big Code, *SNAPL 2015*
Code Completion with Statistical Language Models, *ACM PLDI’14*
Machine Translation for Programming Languages, *ACM Onward’14*
Statistical Feedback Generation for Programs, *ETH TR*
Fast and Precise Statistical Code Completion, *ETH TR*

Tools:

**JSNICE (used worldwide)**
statistical de-obfuscation

**SLANG**
statistical code synthesis

**SAGE**
statistical feedback generation

Scene Completion

Input
Scene Completion

Program Completion

```java
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
```
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);
Martin is talking at the TCE now.
Programming Language Translation

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

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

Image de-noisification
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;
}

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: Impact

✓ Used in 191 countries
✓ Top ranked tool for JavaScript
✓ 1,000+ Tweets
✓ 30,000 users in 1st week

“Predicting program properties from Big Code”, V. Raychev, M. V., A. Krause, ACM POPL’15
Machine Learning for Programming

Applications

Intermediate Representation

Analyze Program (PL)

Train Model (ML)

Query Model (ML)
## Machine Learning for Programming

<table>
<thead>
<tr>
<th>Applications</th>
<th>Code completion</th>
<th>Deobfuscation</th>
<th>Program synthesis</th>
<th>Feedback generation</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate Representation</td>
<td>Sequences (sentences)</td>
<td>Translation Table</td>
<td>Trees</td>
<td>Graphical Models (CRFs)</td>
<td>Feature Vectors</td>
</tr>
<tr>
<td>Analyze Program (PL)</td>
<td>typestate analysis</td>
<td>scope analysis</td>
<td>control-flow analysis</td>
<td>alias analysis</td>
<td></td>
</tr>
<tr>
<td>Train Model (ML)</td>
<td>Neural Networks</td>
<td>SVM</td>
<td>N-gram language model</td>
<td>Structured SVM</td>
<td></td>
</tr>
<tr>
<td>Query Model (ML)</td>
<td>$\text{argmax } P(y \mid x)$</td>
<td>$y \in \Omega$</td>
<td>Greedy MAP inference</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Machine Learning for Programming

<table>
<thead>
<tr>
<th>Applications</th>
<th>Code completion</th>
<th>Program synthesis</th>
<th>Feedback generation</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate Representation</td>
<td>Sequences (sentences)</td>
<td>Translation Table</td>
<td>Graphical Models (CRFs)</td>
<td>Feature Vectors</td>
</tr>
<tr>
<td>Analyze Program (PL)</td>
<td>typestate analysis</td>
<td>control-flow analysis</td>
<td>alias analysis</td>
<td></td>
</tr>
<tr>
<td>Train Model (ML)</td>
<td>scope analysis</td>
<td>SVM</td>
<td>Structured SVM</td>
<td></td>
</tr>
<tr>
<td>Query Model (ML)</td>
<td>Neural Networks</td>
<td>N-gram language model</td>
<td>Greedy MAP inference</td>
<td></td>
</tr>
</tbody>
</table>

More information and tutorials at:  
Goal

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

unknown facts: \( a \) \( b \)

known facts: \( f \) \( \text{document} \) \( \text{getElementById} \)
function f(a) {
    var b = document.getElementById(a);
    return b;
}

unknown facts: a b

known facts: f document.getElementById

Predict unknown facts given some known facts
Challenges

Facts to be predicted are dependent

Millions of candidate choices

Must quickly learn from huge codebases

Prediction should be fast (real time)
Key Idea

Phrase the problem of predicting program facts as

Structured Prediction for Programs
Conditional Random Field
(J. Lafferty, A. McCallum, F. Pereira, ICML 2001)

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

Undirected Probabilistic Graphical Model
Captures dependence between facts to be predicted
Represents a conditional distribution on known facts
Conditional Random Field

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

Example: Let \( y = i, r \) and \( x = t \)
Conditional Random Field

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

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

Example: Let \( y = i, r \) and \( x = t \)

<table>
<thead>
<tr>
<th></th>
<th>( i )</th>
<th>( t )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>Eran</td>
<td>IL</td>
<td>0.1</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>Mooly</td>
<td>IL</td>
<td>0.3</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
P(y \mid x) & = \frac{1}{Z(x)} \exp(w^T f(y, x)) \\
\text{Example: Let } y & = i, r \text{ and } x = t \\
\end{align*}
\]
Conditional Random Field
(J. Lafferty, A. McCallum, F. Pereira, ICML 2001)

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

Example: Let \( y = i, r \) and \( x = t \)

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>t</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>Eran</td>
<td>IL</td>
<td>0.1</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>Mooly</td>
<td>IL</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>r</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_3 )</td>
<td>Eran</td>
<td>Yahav</td>
<td>0.7</td>
</tr>
<tr>
<td>( f_4 )</td>
<td>Mooly</td>
<td>Sagiv</td>
<td>0.4</td>
</tr>
</tbody>
</table>

\[ P(i, r \mid t) = \frac{\exp(0.1 \cdot f_1 + 0.3 \cdot f_2 + 0.7 \cdot f_3 + 0.4 \cdot f_4)}{Z(t)} \]
Structured SVM Training
(N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007)

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

Given a data set: \( D = \{ x^i, y^i \}_{j=1..n} \) learn weights \( w^T \)
Structured SVM Training
(N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007)

\[ P(y \mid x) = \frac{1}{Z(x)} \exp(\mathbf{w}^T \mathbf{f}(y, x)) \]

Given a data set: \( D = \{ x^i, y^i \}_{j=1..n} \) learn weights \( \mathbf{w} \)

Optimization objective (max-margin training):
\[ \forall j \ \forall y \ \Sigma w_i f_i(x^{(i)}, y^{(i)}) \geq \Sigma w_i f_i(x^{(i)}, y) + \Delta(y, y^{(i)}) \]
Structured SVM Training
(N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007)

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

Given a data set: \( D = \{ x^i, y^i \}_{j=1..n} \) learn weights \( w^T \)

Optimization objective (max-margin training):

\[ \forall j \forall y \sum w_i f_i(x^{(j)}, y^{(j)}) \geq \sum w_i f_i(x^{(j)}, y) + \Delta(y, y^{(j)}) \]

for all samples

Given prediction is better than any other prediction by a margin
Structured SVM Training
(N. Ratliff, J. Bagnell, M. Zinkevich, AISTATS 2007)

\[
P(y | x) = \frac{1}{Z(x)} \exp(w^T f(y, x))
\]

Given a data set: \( D = \{ x^i, y^i \}_{j=1..n} \) learn weights \( w^T \)

Optimization objective (max-margin training):
\[
\forall j \forall y \Sigma w_i f_i(x^{(j)}, y^{(j)}) \geq \Sigma w_i f_i(x^{(j)}, y) + \Delta(y, y^{(j)})
\]

for all samples

Given prediction is better than any other prediction by a margin

Avoids expensive computation of the partition function \( Z(x) \)
Querying the model: MAP Inference

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

Given \( x \), we would like to predict \( y = y_1, y_2, \ldots, y_n \) that maximizes \( P(y \mid x) \).

This requires us to make a joint prediction, together for all \( y_1, y_2, \ldots, y_n \).
Querying the model: MAP Inference

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

Given \( x \), we would like to predict \( y = y_1, y_2, \ldots, y_n \) that maximizes \( P(y \mid x) \).

This requires us to make a joint prediction, together for all \( y_1, y_2, \ldots, y_n \)

\[ y^\text{best} = \arg\max_{y \in \Omega_x} P(y \mid x) = \arg\max_{y \in \Omega_x} w^T f(y, x) \]
Querying the model: MAP Inference

\[ P(y | x) = \frac{1}{Z(x)} \exp(w^T f(y, x)) \]

Given \( x \), we would like to predict \( y = y_1, y_2, \ldots, y_n \) that maximizes \( P(y | x) \). This requires us to make a joint prediction, together for all \( y_1, y_2, \ldots, y_n \)

\[ y^{\text{best}} = \arg\max_{y \in \Omega_x} P(y | x) = \arg\max_{y \in \Omega_x} w^T f(y, x) \]
Querying the model: Max-marginals

\[ y_1^{\text{best}} = \text{argmax } P(y_1 | x) \quad \ldots \ldots \quad y_n^{\text{best}} = \text{argmax } P(y_n | x) \]

\[ y^{\text{best}} = (y_1^{\text{best}}, \ldots, y_n^{\text{best}}) \]

\( \Sigma \Pi \) belief propagation algorithms answer max-marginal queries
Querying the model: Max-marginals

\[ y_{1}^{\text{best}} = \arg \max P(y_1 \mid x) \]

\[ y^{\text{best}} = \arg \max P(y_n \mid x) \]

\[ \sum \Pi \text{ best} \]
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(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(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;

Unknown facts: t, r, i, ...
Known facts: length ...

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

<table>
<thead>
<tr>
<th>i</th>
<th>t</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>step</td>
<td>0.5</td>
</tr>
<tr>
<td>j</td>
<td>step</td>
<td>0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>i</th>
<th>r</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>len</td>
<td>0.6</td>
</tr>
<tr>
<td>i</td>
<td>length</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r</th>
<th>length</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>length</td>
<td>0.5</td>
</tr>
<tr>
<td>len</td>
<td>length</td>
<td>0.3</td>
</tr>
</tbody>
</table>
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;

Unknown facts: \( t, r, i, \ldots \)

Known facts: \( \text{length} \) \( \ldots \)

\[
\text{argmax } w^T f(i, t, r, \text{length})
\]

<table>
<thead>
<tr>
<th>i</th>
<th>t</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>step</td>
<td>0.5</td>
</tr>
<tr>
<td>j</td>
<td>step</td>
<td>0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>i</th>
<th>r</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>len</td>
<td>0.6</td>
</tr>
<tr>
<td>i</td>
<td>length</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r</th>
<th>length</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{length}</td>
<td>\text{length}</td>
<td>0.5</td>
</tr>
<tr>
<td>\text{len}</td>
<td>\text{length}</td>
<td>0.3</td>
</tr>
</tbody>
</table>
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;

argmax w^T f(i, t, r, length)
Structured Prediction for Programs
(V. Raychev, M. V., A. Krause, ACM POPL’15)

First connection between Programs and Conditional Random Fields
Structured Prediction for Programs

(V. Raychev, M. V., A. Krause, ACM POPL’15)
Structured Prediction for Programs
(V. Raychev, M. V., A. Krause, ACM POPL’15)
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;

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;

Structured Prediction for Programs
(V. Raychev, M. V., A. Krause, ACM POPL’15)
Structured Prediction for Programs
(V. Raychev, M. V., A. Krause, ACM POPL’15)
Structured Prediction for Programs
(V. Raychev, M. V., A. Krause, ACM POPL'15)

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;

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;
Structured Prediction for Programs
(V. Raychev, M. V., A. Krause, ACM POPL’15)

Prediction Phase

```
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;
```

```
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;
```

Learning Phase

```
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;
```

```
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;
```

30 nodes, 400 edges

150MB

Program analysis

MAP inference

Transform

Prediction Phase

Program analysis

MAP inference

Transform

150MB

Learning Phase

Program analysis

SSVM learning

max-margin training

Conditional Random Field

\[ P(y | x) \]

Alias, call analysis

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

Atlassian

Bitbucket

GitHub

150MB

30 nodes, 400 edges
Structured Prediction for Programs
(V. Raychev, M. V., A. Krause, ACM POPL’15)

30 nodes, 400 edges

```
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;
```

```
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;
```

Time: milliseconds

Prediction Phase

Program analysis ➔ MAP inference ➔ transform

Learning Phase

Program analysis ➔ SSVM learning ➔ max-margin training

Conditional Random Field

```
P(y | x)
```

150MB

150MB

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```

```
150MB
```
Structured Prediction for Programs
(V. Raychev, M. V., A. Krause, ACM POPL’15)

```
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;
```

```
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;
```

30 nodes, 400 edges

Program analysis → MAP inference → transform

```
Prediction Phase
```

```
Learning Phase
```

```
Conditional Random Field
P(y | x)
```

Names: 63% Types: 81% (helps typechecking)

150MB
# Machine Learning for Programming

<table>
<thead>
<tr>
<th>Applications</th>
<th>Code completion</th>
<th>Program synthesis</th>
<th>Feedback generation</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate Representation</td>
<td>Code completion</td>
<td>Program synthesis</td>
<td>Feedback generation</td>
<td>Translation</td>
</tr>
<tr>
<td>Analysis Program (PL)</td>
<td>Sequences (sentences)</td>
<td>Translation Table</td>
<td>Trees</td>
<td>Graphical Models (CRFs)</td>
</tr>
<tr>
<td>Train Model (ML)</td>
<td>typestate analysis</td>
<td>SVM</td>
<td>alias analysis</td>
<td></td>
</tr>
<tr>
<td>Query Model (ML)</td>
<td>N-gram language model</td>
<td>Greedy MAP inference</td>
<td>Structured SVM</td>
<td></td>
</tr>
</tbody>
</table>

Next 10 years

Knowledge

Time

Applications

SLANG

PRIME (Technion)
Next 10 years

Knowledge

Frameworks

Nice2Predict.org

Applications

SLANG

PRIME (Technion)

Time

2014 2017 2020 2024
Next 10 years

Knowledge

Methods

Frameworks

Applications

Potts model of programs?

Nice2Predict.org

SLANG

PRIME (Technion)

2014

2017

2020

2024

Time
Machine Learning for Programming

TREND

DIMENSIONS

Applications
- Code completion
- Deobfuscation
- Program synthesis
- Feedback generation
- Translation

Intermediate Representation
- Sequences (sentences)
- Translation Table
- Trees
- Graphical Models (CRFs)
- Feature Vectors

Analyze Program [PL]
- Typestate analysis
- Scope analysis
- Control-flow analysis
- Alias analysis

Train Model [IML]
- Neural Networks
- SVM
- Structured SVM
- N-gram language model

Query Model [IML]
- \( \arg \max_{f \in \mathcal{F}} \end{aligned} \)
- \( y \in \mathcal{Y} \)
- Greedy
- MAP inference

IMPACT

FUTURE

More information: http://www.srl.inf.ethz.ch/