PROGRAM BOOSTING:
PROGRAM SYNTHESIS VIA CROWD-SOURCING

Robert Cochran
Loris D’Antoni
Benjamin Livshits
David Molnar
Margus Veanes
In Search of the Perfect URL Validation Regex

<table>
<thead>
<tr>
<th>URL</th>
<th>Spoon Library</th>
<th>@krijnhoetmer</th>
<th>@gruber</th>
<th>@grubor v2</th>
<th>@cowboy</th>
<th>Joffrey Friedl</th>
<th>@mattfarina</th>
<th>@stephenhay</th>
<th>@scottgonzales</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://www.example.com/foo/">https://www.example.com/foo/</a>?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bar=aza&amp;inga=42&amp;quux</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="http://Odf.ws/123">http://Odf.ws/123</a></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><a href="http://userid:password@example.com:8080">http://userid:password@example.com:8080</a></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

These URLs should match (1 → correct)

<table>
<thead>
<tr>
<th>URL</th>
<th>Spoon Library</th>
<th>@krijnhoetmer</th>
<th>@gruber</th>
<th>@grubor v2</th>
<th>@cowboy</th>
<th>Joffrey Friedl</th>
<th>@mattfarina</th>
<th>@stephenhay</th>
<th>@scottgonzales</th>
</tr>
</thead>
<tbody>
<tr>
<td>http://</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>http://.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><a href="http://a.b--c.de/">http://a.b--c.de/</a></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><a href="http://a.b.co">http://a.b.co</a></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><a href="http://a.b-.co">http://a.b-.co</a></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>http://??</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

These URLs should fail (0 → correct)
Key Insight for Crowd-Sourcing of Programs

Regular expressions

- Most people get **easy cases** right
- People are good with positive examples
- ...but bad at rejecting negative examples – more permissive than they should be
- However, **piecing together** different solutions will produce a good score on the examples

CrowdBoost

- In this project we apply this intuition to programs
- CrowdBoost
  - Crowd-source initial programs
  - “Blend” them together
  - Refine the result
- We call this **program boosting**
Overview of Program Boosting

- Specification is not formal, and often elusive and incomplete
- Broad space of inputs difficult to get full test coverage for
- Easy to get started, tough to get good precision

**Specification**
- Textual description
- Open to interpretation

**Training set**
- Provided by whoever defines the task
- Positive and negative examples

**Initial programs**
- Get something right
- But usually get something wrong
Outline

• Vision and motivation
• Our approach: CrowdBoost
• Technical details: regular expressions and SFAs
• Experiments
CrowdBoost in a nutshell

Specification
CrowdBoost Outline

• Crowd-source initial programs
• We use genetic programming approach for blending
• Needed program operations:
  1. Shuffles (2 programs => program)
  2. Mutations (program => program)
  3. Training Set Generation and Refinement (program => new labeled examples)

<table>
<thead>
<tr>
<th>ID</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex1</td>
<td>+</td>
</tr>
<tr>
<td>Ex2</td>
<td>-</td>
</tr>
<tr>
<td>Ex3</td>
<td>+</td>
</tr>
<tr>
<td>Ex4</td>
<td>-</td>
</tr>
</tbody>
</table>
Example of Program Blending

Input 1 (0.53)

Mutation (0.60)

Input 2 (0.58)

Mutation (0.60)

Mutation (0.50)

Mutation (0.69)

Shuffle (0.62)

Shuffle (0.63)

Winner! (0.85)

Need to prevent over-fitting
How Do We Measure Quality?

Initial Examples “Gold Set”

Possible Input Space
Skilled and Unskilled Crowds

**Skilled**
- More expensive, longer units of work (hours)
- May require multiple rounds of interaction
- *Provide initial programs*

**Unskilled**
- Cheaper, smaller units of work (seconds or minutes)
- Automated process for hiring, vetting and retrieving work
- *Used to grow/evolve training examples*
CrowdBoost

Specification

+ Initial Examples "Gold Set"

.accuracy = +/−/Total
Outline

• Vision and motivation
• Our approach: CrowdBoost
• Technical details: regular expressions and SFAs
• Experiments
Shuffle: Example

- Regular expressions for phone numbers
  A. \^[0-9]\{3\}-[0-9]*-[0-9]\{4\}\$
  B. \^[0-9]\{3\}-[0-9]\{3\}-[0-9]*\$

Shuffle:
\^[0-9]\{3\}-[0-9]\{3\}-[0-9]\{4\}\$
Training Set Generation

• Compute automaton $D$ of strings passing through uncovered states
• Choose string $s$ in $D$ at random
  • https://f.o/..Q/
  • ftp://1.bd:9/:44ZW1
  • http://h:68576/:X
  • https://f68.ug.dk.it.no.fm
  • ftp://hz8.bh8.fzpd85.frn7..
  • ftp://i4.ncm2.lkxp.r9...:5811
  • ftp://bi.mt..:349/
  • http://n.ytnsw.yt.ee8o.w.fos.o

• Given a string $e$, choose find the closest string to $e$ in $D$
  • $e =$ “http://youtube.com”
    • Whttp://youtube.com
    • http://y_outube.com
    • h_ttp://youtube.com
    • WWWhttp://youtube.co/m
    • http://yout.pe.com
    • ftp://yo.tube.com
    • http://y.foutube.com
Outline

• Vision and motivation
• Our approach: CrowdBoost
• Technical details: regular expressions and SFAs
• Experiments
Four Crowd-Sourcing Tasks

• We consider 4 task specifications
  • Phone numbers
  • Dates
  • Emails
  • URLs

• For Bountify sourcing we used a handful of + and - examples

Date Specification:
• Please write a regular expression that validates dates in different formats. Note that we are asking for original work. Please do not copy your answer from other sites.
• + (9 total)
  • June 7, 2013
  • 7/7/2013
  • June-7-2013
• - (10 total)
  • Junu 7, 2013
  • 7/77/2013
  • Jul-7-2013
• Please provide the regular expression in the form ^YOUR ANSWER IS HERE $ as part of your answer. Please test your regex on the samples provided before submitting. You may want to use http://regexpal.com for testing.
Final Fitness After Boosting

Final fitness upwards of 90%
## Other experimental results (per pair)

<table>
<thead>
<tr>
<th>Task</th>
<th>Mechanical Turk Latency (avg)</th>
<th>Total running time (avg)</th>
<th>Mechanical Turk Cost (avg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone</td>
<td>8 minutes</td>
<td>25 minutes</td>
<td>0.41 $</td>
</tr>
<tr>
<td>Date</td>
<td>30 minutes</td>
<td>55 minutes</td>
<td>2.59 $</td>
</tr>
<tr>
<td>Email</td>
<td>11 minutes</td>
<td>17 minutes</td>
<td>0.50 $</td>
</tr>
<tr>
<td>URL</td>
<td>30 minutes</td>
<td>70 minutes</td>
<td>3.00 $</td>
</tr>
</tbody>
</table>

- We run up to 10 generations
- Often 5 or 6 generations are enough to hit plateau
- Classification tasks given in batches
- We hire 5 workers per batch
Conclusions

- Programs that implement non-trivial tasks can be crowd-sourced effectively.
- We focus on tasks that defy easy specification and involve controversy.
- **CrowdBoost**: use genetic programming, to produce an improvement in quality of crowdsourced programs.

- Experiments with regular expressions.
- Tested on 4 complex tasks:
  - Phone numbers, Dates, Emails, URLs.
- Considered pairs of regexes from Bountify, RegexLib, etc.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSISTENT BOOSTS</td>
<td>0.12 - 0.28 median increase in fitness</td>
</tr>
<tr>
<td>MTURK LATENCY</td>
<td>8 – 37 minutes per iteration</td>
</tr>
<tr>
<td>RUNNING TIME</td>
<td>10 minutes to 2 hours</td>
</tr>
<tr>
<td>MTURK COSTS</td>
<td>$0.41 to $3.00</td>
</tr>
</tbody>
</table>