Program Boosting: Program Synthesis via Crowd-Sourcing

Robert A. Cochran, Loris D. Antoni, Benjamin Livshits, David Molnar, and Margus Veanes

slide author names omitted for FERPA compliance

Problem

How can we use crowd-sourcing to boost program accuracy where the program's initial specification may be open to interpretation?

Key Insight

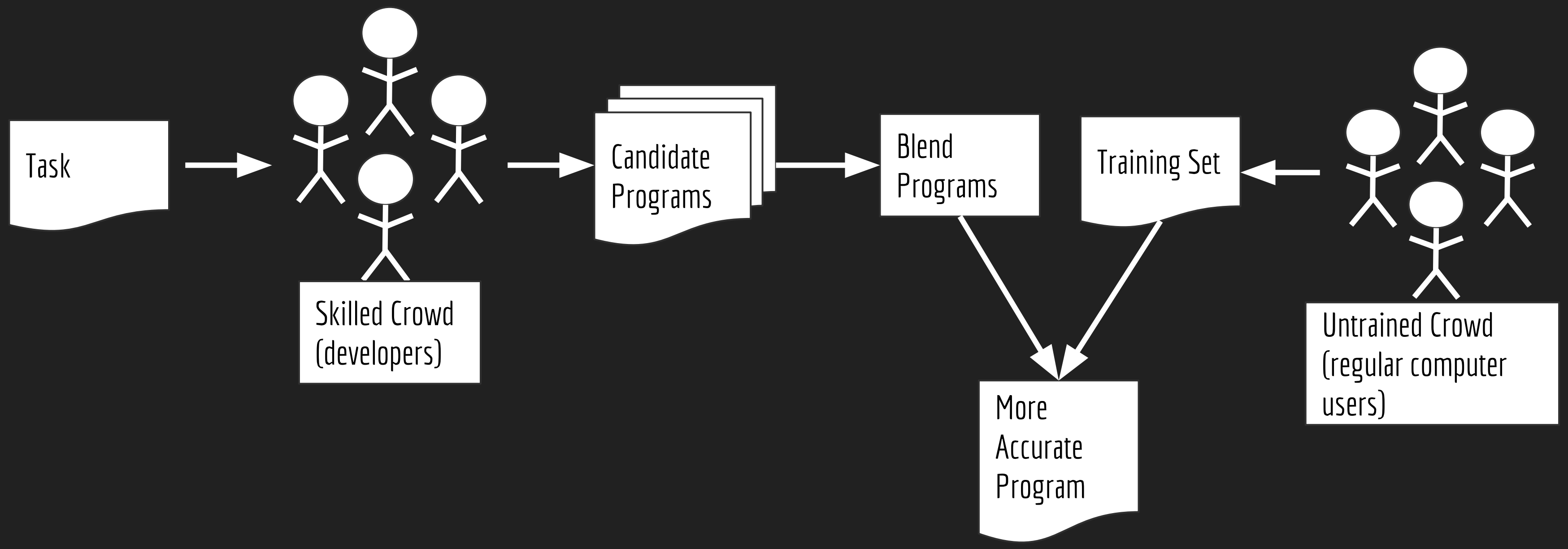
Regular Expressions

- ➔ Lack an easy-to-formalize specification
- ➔ Different regexes cover different cases
- ➔ Surprisingly difficult to implement addressing all the tricky corner cases
- ➔ Plenty of room for ambiguity

CrowdBoost

- ➔ Pose a tricky programming task as a crowd-sourcing challenge
- ➔ Describe the task in question in a very loose form of specification
- ➔ Provide positive and negative examples ("the golden set"), giving a partial specification
- ➔ Blend imperfect solutions together to yield a solution of higher quality using a genetic programming approach
- ➔ Refine the result using a two-crowd approach

Approach

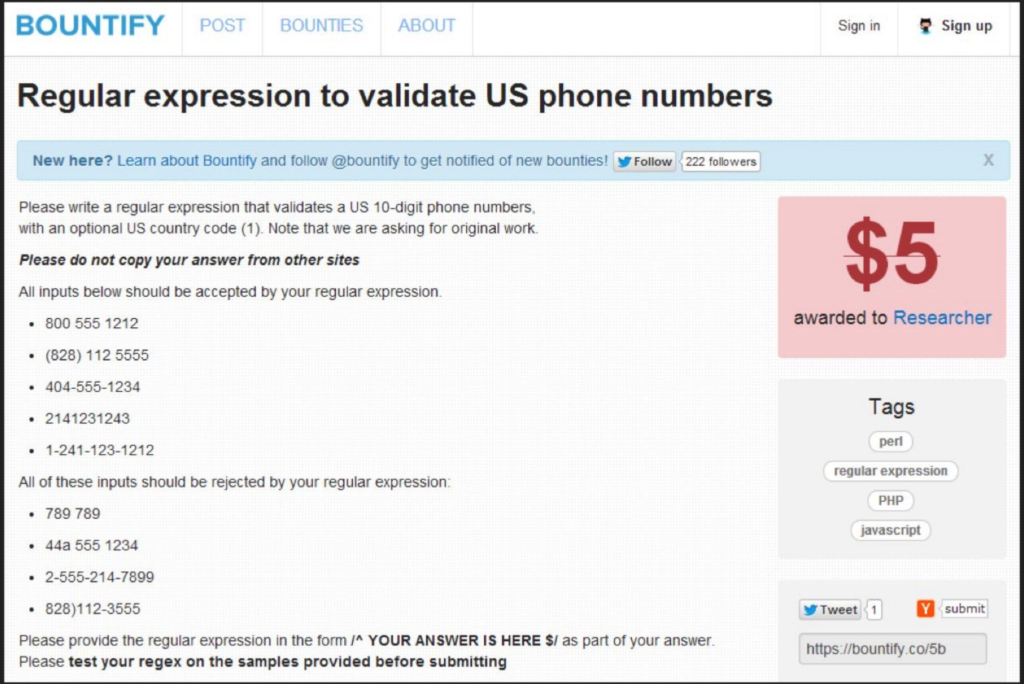


Task
Skilled Crowd (developers)
Candidate Programs
Blend Programs
Training Set
Untrained Crowd (regular computer users)
More Accurate Program

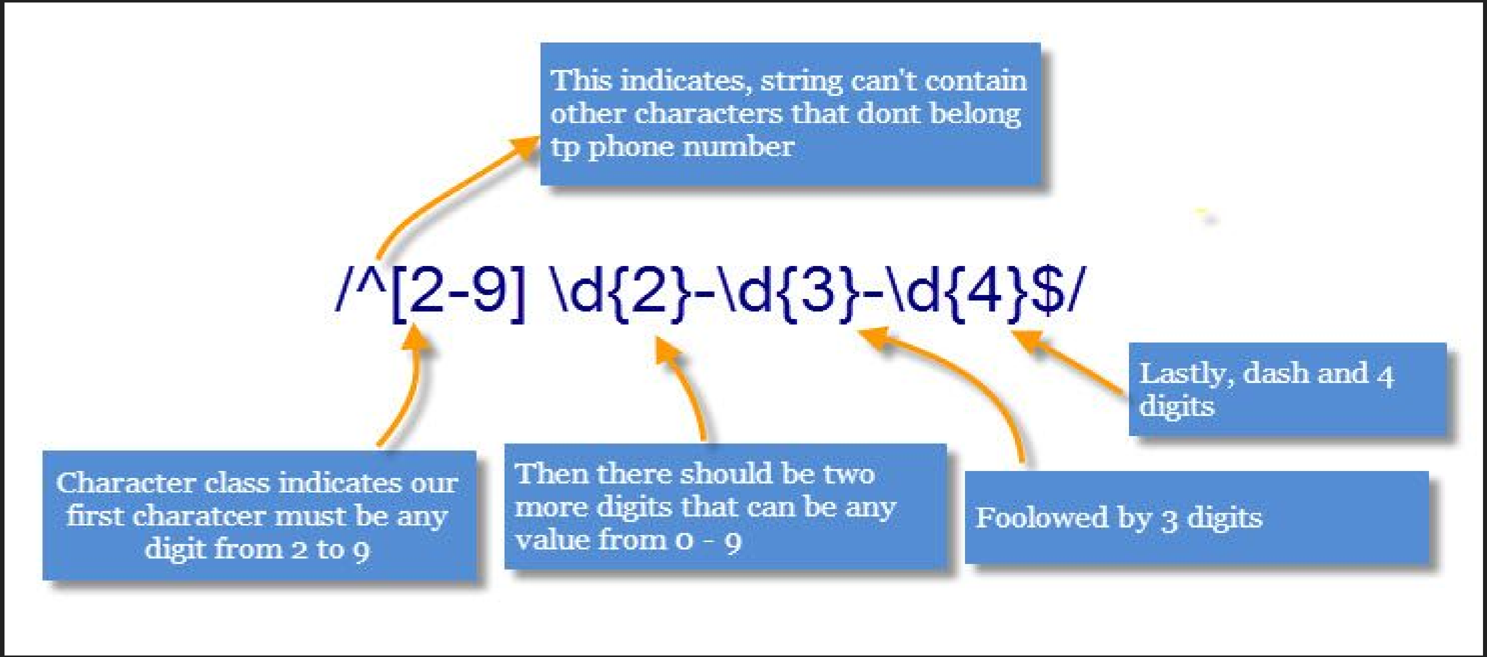
Example

- ➔ Post a programming task to Bountify: “Generate a regular expression to validate phone numbers”

- ➔ Take the first 3 submissions



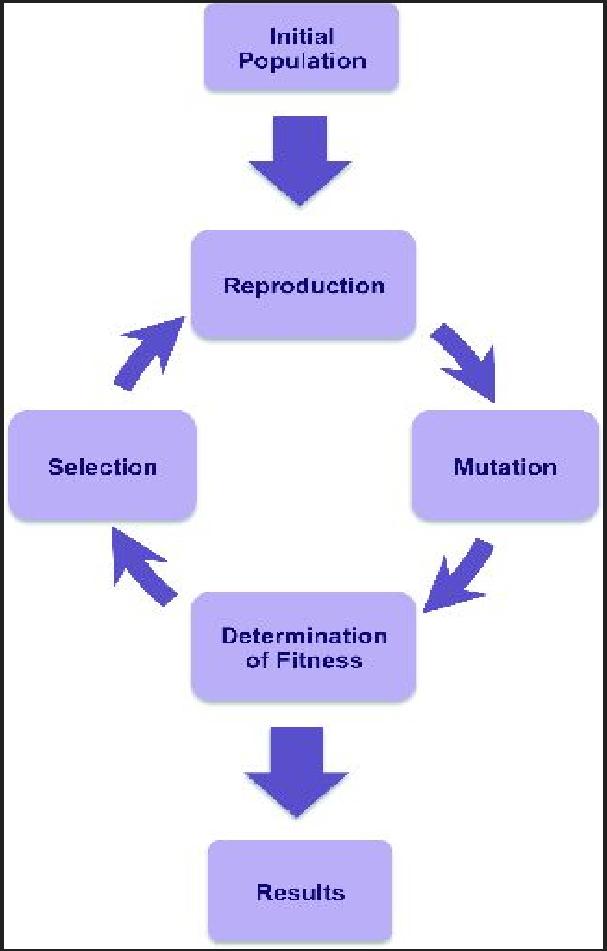
Example



A regular expression for a phone number

Example

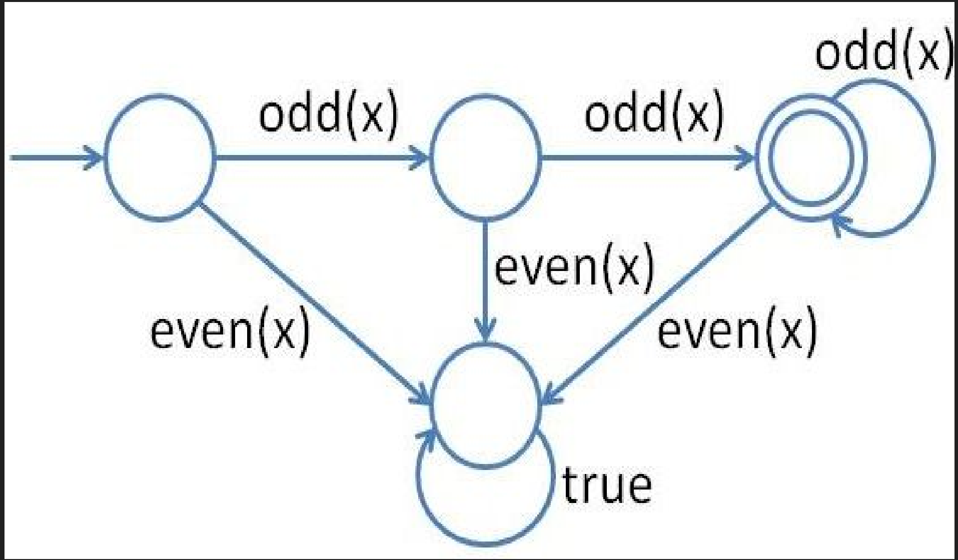
Genetic Programming Algorithm



Example

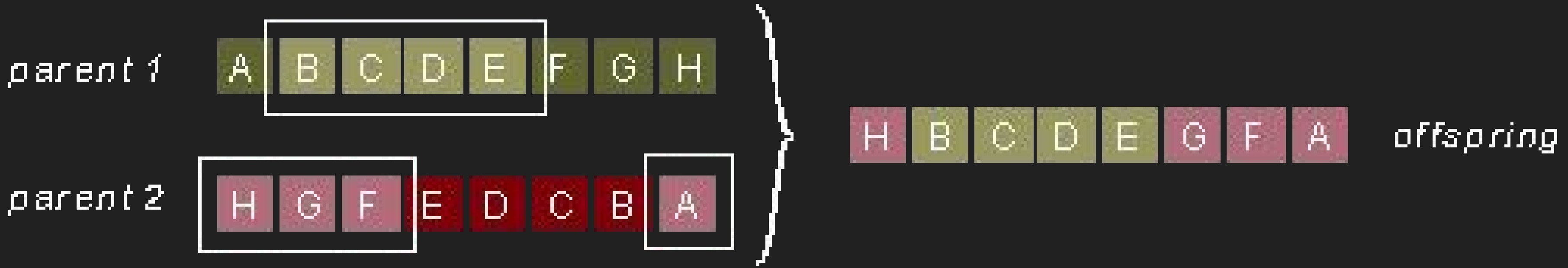
- Represent each regular expression as a Symbolic Finite Automaton

- Manipulate each SFA within a genetic programming algorithm



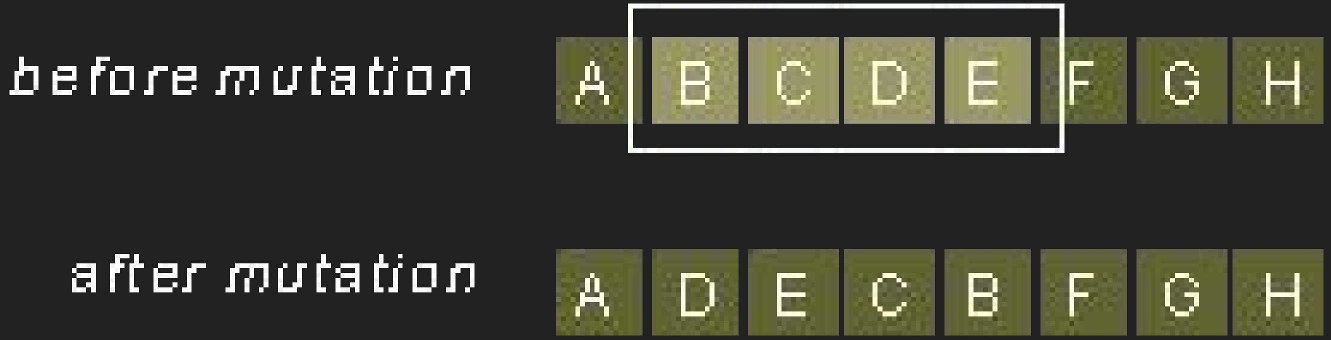
Example

➔ Perform crossover operations on the candidates



Example

→ Perform mutation operations on the candidates



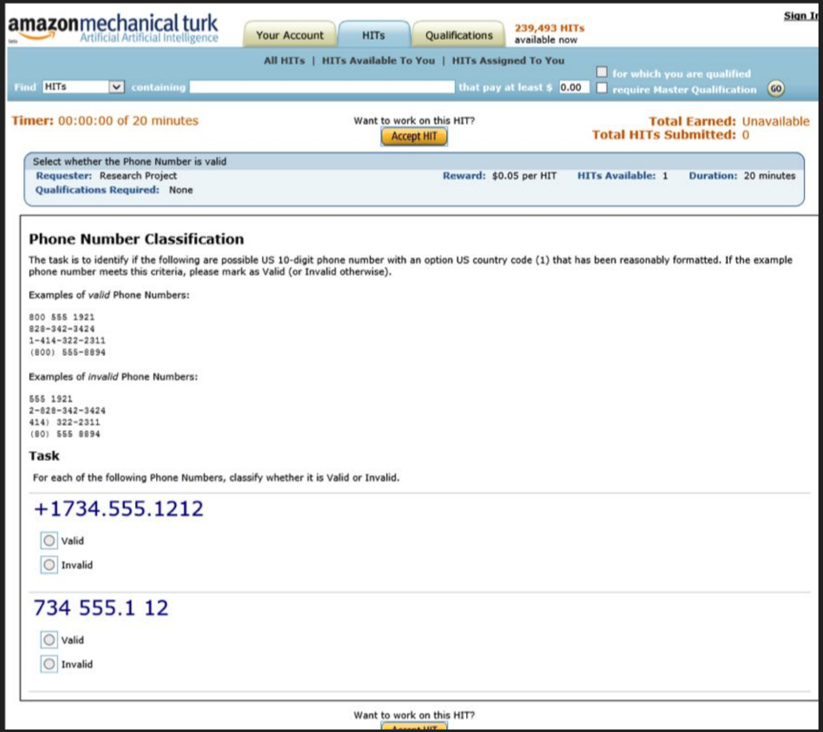
Example

→ Filter the new examples by human evaluation via Mechanical Turk, and update our candidates.

→ Run fitness tests on the candidates and select the best.

→ Submit our resulting population back to the head of the main algorithm.

→ Repeat the process until we find a perfect solution, or we run out of money .



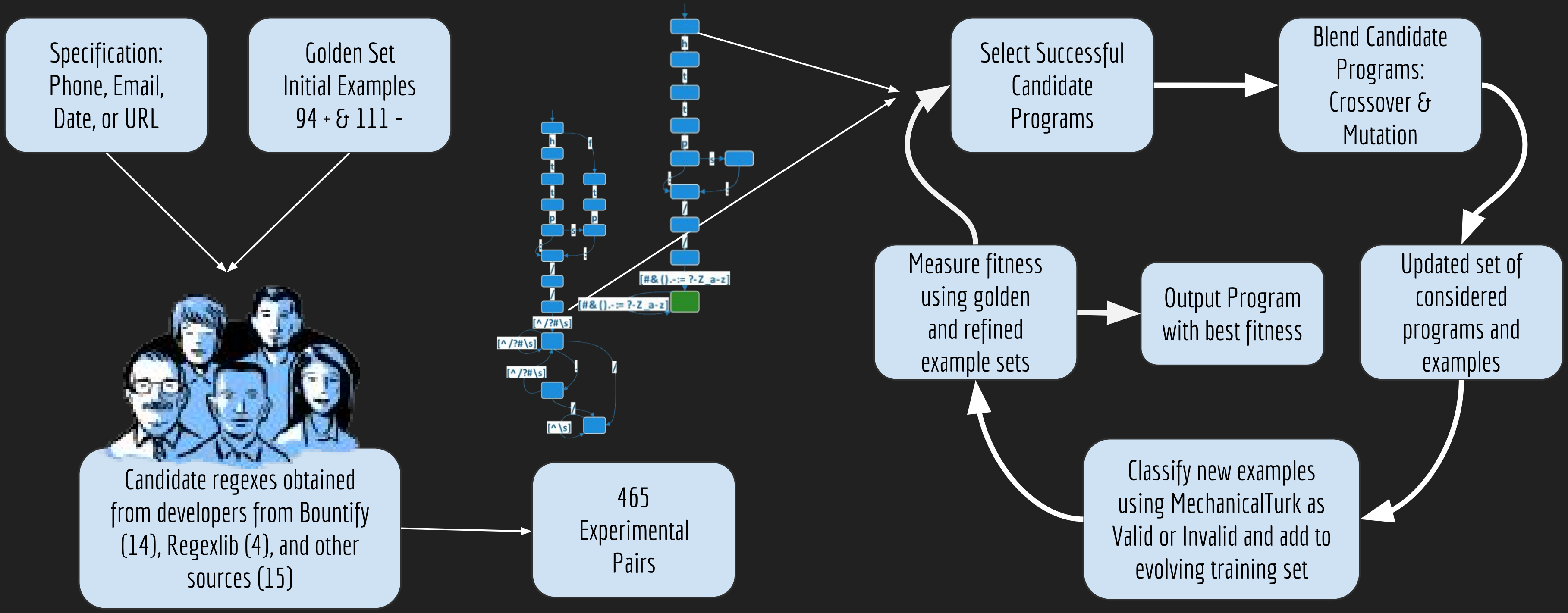
Example

Task: Phone numbers

Fitness: 0.897959183673469

```
^(((((((([02-9]|+1([0-9]|[\x20-.][0-9]))|((|+1((|[\x20-.]())[0-9])([0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9])|((([02-9]|+1([0-
9]|[\x20-.][0-9]))|((|+1((|[\x20-.]())[0-9])[0-9][0-9][\x20-.]([0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9]))|(([02-9]|+1([0-9]|[\x20-.][0-9]))|((|+1
((|[\x20-.]())[0-9])[0-9][0-9])(([0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9])|[\x20-.]([0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9]
[\x20-.][0-9][0-9][0-9][0-9]))))|1(_)*((([0-9][0-9][0-9]([0-9][0-9][0-9][0-9][0-9][0-9]([0-9])?|([0-9][0-9][\x20-.]|[0-9][0-9][0-9][\x20-.])[0-9][0-9][0-9][0-9])|([0-9]([0-9][0-9]
[0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9]))|((([0-9][0-9][\x20-.]|[0-9][0-9][0-9][\x20-.])|([0-9][0-9][0-9][\x20-.])([0-9][0-9][0-9]
[0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9]))|((([0-9][0-9])|[0-9][0-9][0-9]))|([0-9][0-9][0-9]))(([0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][\x20-.]
[0-9][0-9][0-9][0-9])|[\x20-.]([0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9]))))|1(_)*[\x20-.](1_)*(((((((([0-9][0-9]([0-9][0-9][0-9][0-9][0-9]
[0-9]([0-9])?|([0-9][0-9][\x20-.]|[0-9][0-9][0-9][\x20-.])[0-9][0-9][0-9][0-9])|((([0-9][\x20-.]|([0-9][0-9][\x20-.])([0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][\x20-.][0-
9][0-9][0-9][0-9]))|1[0-9][0-9][0-9]([0-9][0-9][0-9][0-9][0-9][0-9]([0-9])?|([0-9][0-9][\x20-.]|[0-9][0-9][0-9][\x20-.])[0-9][0-9][0-9][0-9]))|(([02-9]|+1([0-9]|([0-9]))|1([0-9])
([0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9]))|(1([0-9][0-9][\x20-.]|[0-9][0-9][0-9][\x20-.])|(([02-9]|+1([0-9]|([0-9]))|1
([0-9])[0-9][0-9][\x20-.])([0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9]))|(+1[\x20-.]|1[\x20-.])(([0-9]|([0-9])([0-9][0-9][0-9][0-9][0-9][0-9][0-
9][0-9][0-9]|[0-9][0-9][0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9])|([0-9]|([0-9])[0-9][0-9][\x20-.]([0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-
9]))))|((([0-9])|([0-9][0-9]))(([0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9])|[\x20-.]([0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][\x20-.]
[0-9][0-9][0-9][0-9])))|((1([0-9][0-9])|[0-9][0-9][0-9]))|(([02-9]|+1([0-9]|([0-9]))|1([0-9])[0-9][0-9]))|(+1[\x20-.]|1[\x20-.])([0-9]|([0-9])[0-9][0-9]))(([0-9][0-9][0-9][0-9][0-9]
[0-9][0-9]|[0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9])|[\x20-.]([0-9][0-9][0-9][0-9][0-9][0-9][0-9]|[0-9][0-9][0-9][\x20-.][0-9][0-9][0-9][0-9]))))))$
```

Evaluation: Experimental Analysis



Evaluation: Initial Data

Task	Specification	Examples +	Examples −
Phone numbers	https://bountify.co/5b	5	4
Dates	https://bountify.co/5v	9	9
Emails	https://bountify.co/5c	10	7
URLs	https://bountify.co/5f	14	9

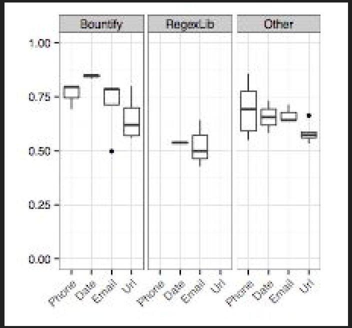
Specifications provided to Bountify developers. The last two columns capture the number of positive and negative examples (a subset of the golden set) given to workers in the task specifications.

	Golden set +	Golden set -	Candidate regexes	Candidate regex source: Bountify	Regexlib	Other
Phone numbers	20	29	8	3	0	5
Dates	31	36	6	3	1	2
Emails	7	7	10	4	3	3
URLs	36	39	9	4	0	5

The number of examples in the golden set and the number of candidate regexes in each case study.

	Regex character length				SFA state count			
	25%	50%	75%	Max	25%	50%	75%	Max
Phone numbers	44.75	54	67.75	96	14.75	27	28	30
Dates	154	288	352.25	434	19	39.5	72	78
Emails	33.5	68.5	86.75	357	7.25	8.5	10	20
URLs	70	115	240	973	12	25	30	80

Summarized size and complexity of the candidate regexes by length and by number of states in each resulting SFA.



Distribution of initial accuracy (fitness) of candidate regular expressions by source. Overall, initial fitness values hover between .5 and .75, with none of the regexes being either "too good" or "too bad".

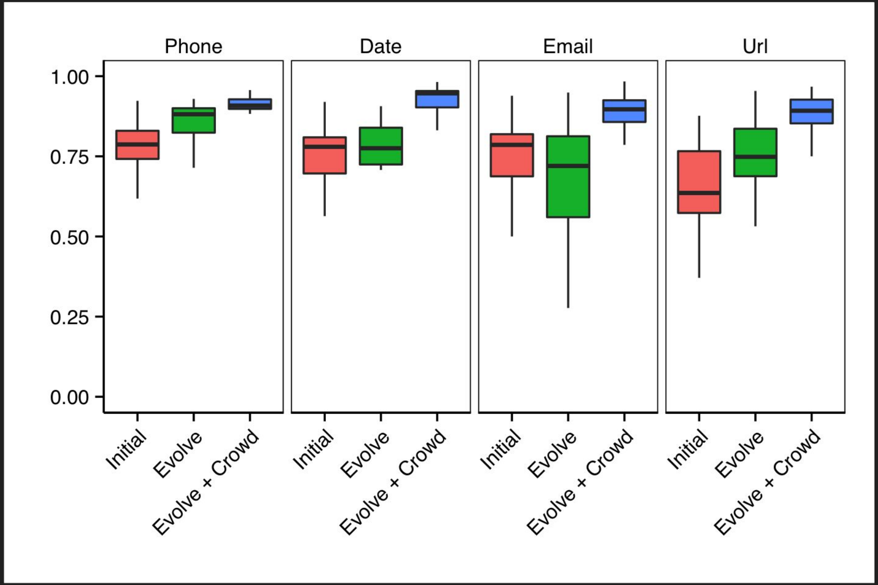
Evaluation: Findings

The regular expressions for each of the tasks were tested for accuracy on positive and negative examples in two sets, the golden set and the evolved set.

Golden set can be manipulated by adding and removing examples to influence accuracy measurements.

Evolved set is more representative since it evolves naturally through refinement and crowd consensus.

High-level results obtained from the boosting process are consistent across all tasks showing an average boost of 16.25%.



Evaluation: Findings

Evaluated on...	Golden set			Evolved set		
		Boosted			Boosted	
Task	initial	no crowd	crowd	initial	no crowd	crowd
Phone numbers	0.80	0.90	0.90	0.79	0.88	0.91
Dates	0.85	0.99	0.97	0.78	0.78	0.95
Emails	0.71	0.86	0.86	0.79	0.72	0.90
URLs	0.67	0.91	0.88	0.64	0.75	0.89

In each task category, boosting results (mean) are shown via fitness values measured on either the golden set or the evolved set for three separate regexes; initial, "no crowd" and "crowd".

Task	Generations				Generated strings				Consensus			
	25%	50%	75%	Max	25%	50%	75%	Max	25%	50%	75%	Max
Phone numbers	7	8	10	10	0	6.5	20.25	83	1	1	1	1
Dates	10	10	10	10	29	45	136	207	1	1	1	1
Emails	5	5	6.5	10	2	7	17	117	1	1	1	1
URLs	10	10	10	10	54	72	107	198	0.99	1	1	1

Characterizing the boosting process in three dimensions: the number of generations, the number of generated strings, and the measured consensus for classification tasks.

Task	Crossovers (thousands)				% Successful crossovers				Mutations (thousands)				% Successful mutations			
	25%	50%	75%	Max	25%	50%	75	Max%	25%	50%	75%	Max	25%	50%	75%	Max
Phone numbers	73	98	113	140	0.002	0.071	1.888	17.854	5	6	8	13	3.8	5.5	11.6	34.0
Dates	14	108	162	171	0.21	1.51	7.22	38.92	8	12	17	37	16	31	35	53
Emails	3	8	22	165	0.45	1.62	5.11	15.04	0	0	2	15	41	54	78	100
URLs	116	178	180	180	0.88	6.62	34.29	50.15	9	20	52	114	30	35	41	64

Statistics for the crossover and mutation process across the tasks. The number of crossovers produced during boosting is in ten of thousands, but only a small percentage of them survive to the next generation. The number of mutations is smaller (single thousands), and their survival rate is somewhat higher. This can be explained by the fact that mutations are relatively local transformations and are not nearly as drastic as crossovers.

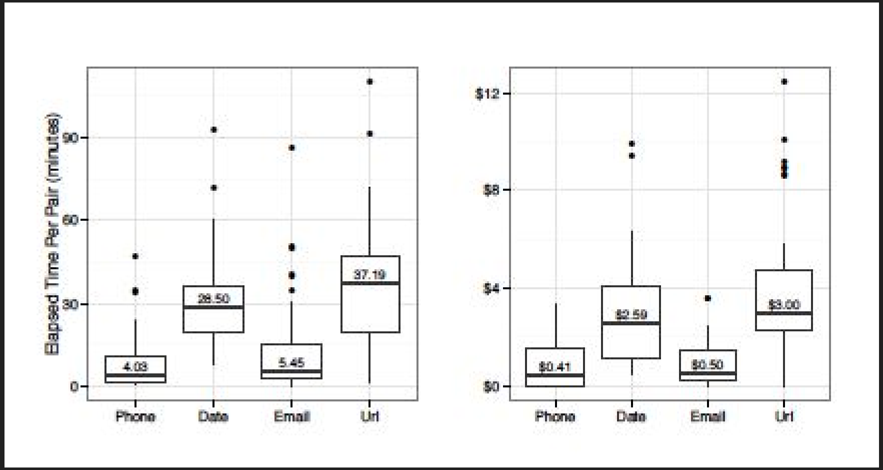
Evaluation: Findings

Running times:

Pair-wise boosting for each task averaged from about 4 minutes and 37 minutes per pair.

Overall cost:

Performing program boosting across all four tasks ranged between 41 cents and 3 dollars per pair.



Left: Running times for each task
Right: Costs for Mechanical Turk

In Summary

- A semi-automatic program synthesis technique using a set of initial crowd-sourced programs that finds the best result by crowd-sourcing a training set for a measure of fitness
- An implementation for program boosting algorithm involving a genetic programming technique with crossover and mutation algorithms
- CrowdBoost represents regular expressions using Symbolic Finite Automata (SFAs). This is most likely the first work to use genetic programming on automata over a complex alphabet, UTF-16 in this case
- An evaluation of this program boosting technique over four case studies, which yielded an average program boost of 16.25% over 465 pairs of regular expressions. The results also showed consistency across the tasks and sources of regular expressions, giving support to the generality of their approach

Discussion

How can crowd-sourcing programs and examples go wrong and affect program boosting?

Discussion

How will this technique scale on pieces of code?

Discussion

How do you know when to stop crowd-sourcing?

Discussion

Is this approach worth the amount of time it takes to get the results?

Discussion

Do we know that the final program is the most fit?

References

Program Boosting Powerpoint Presentation -> google.com

Symbolic Automata -> cs.wisc.edu

Regular Expressions -> code.tutsplus.com

Genetic Programming -> geneticprogramming.com, wikipedia.org

Pairwise Testing -> tutorialspoint.com

Genetic Algorithms vs Genetic Programming -> stackoverflow.com