A Comparative Analysis of the Usability of Stack Overflow Code Snippets Across Languages

Cristina Videira Lopes
Di Yang
Aftab Hussain

Previous

Characteristics of Answers
Nasehi et al. 2012

Programming Knowledge & Age
Morrison and Murphy-Hill 2013

Developer Q&A Habits
Wang et al. 2013

Now

Previous

Usability: i.e. Execution Behaviour of Snippets

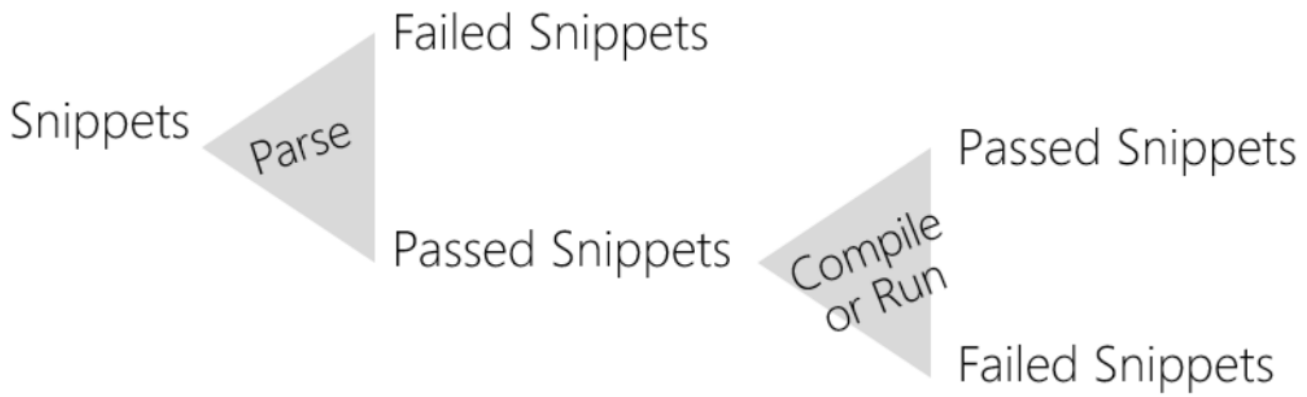
Characteristics of Answers
Nasehi et al. 2012

Programming Knowledge & Age
Morrison and Murphy-Hill 2013

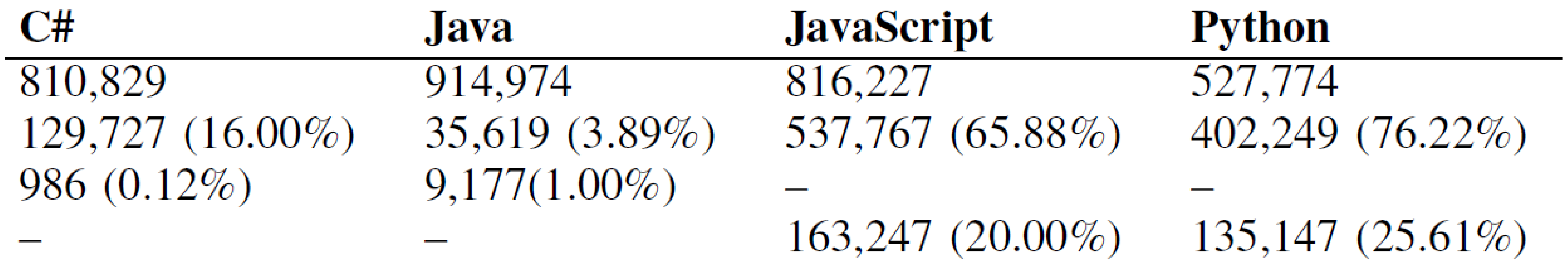
Developer Q&A Habits
Wang et al. 2013

Extracted Snippets from Accepted Answers

Static Languages	
Java	914,974
C#	810,829
Dynamic Languages	
Python	527,774
JavaScript	816,227



Results

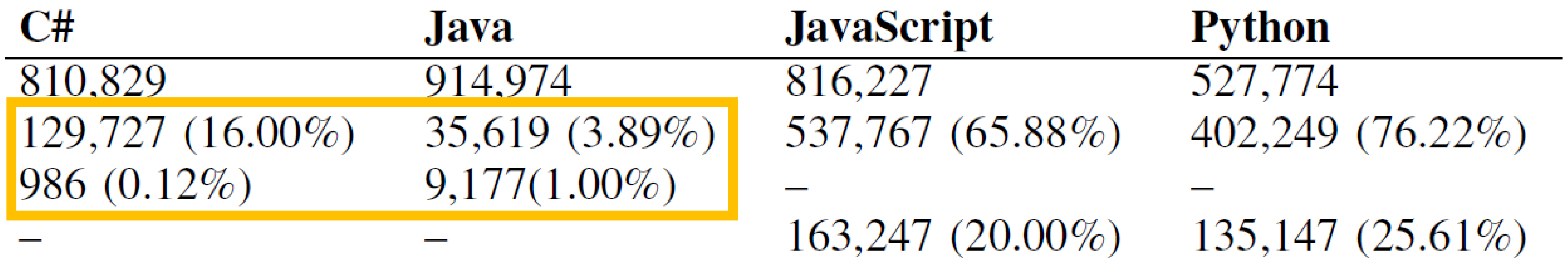


	C#	Java	JavaScript	Python
Total	810,829	914,974	816,227	527,774
Parsable	129,727 (16.00%)	35,619 (3.89%)	537,767 (65.88%)	402,249 (76.22%)
Compilable	986 (0.12%)	9,177(1.00%)	–	–
Runnable	–	–	163,247 (20.00%)	135,147 (25.61%)

Results

	C#	Java	JavaScript	Python
Total	810,829	914,974	816,227	527,774
Parsable	129,727 (16.00%)	35,619 (3.89%)	537,767 (65.88%)	402,249 (76.22%)
Compilable	986 (0.12%)	9,177(1.00%)	–	–
Runnable	–	–	163,247 (20.00%)	135,147 (25.61%)

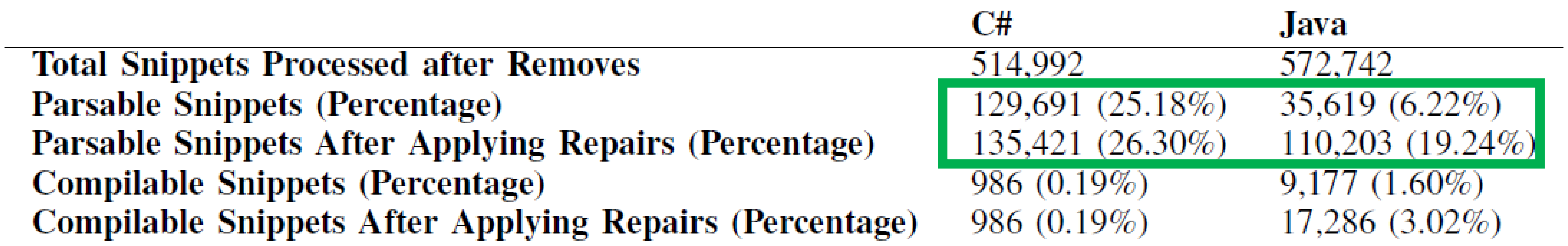
Results



	C#	Java	JavaScript	Python
Total	810,829	914,974	816,227	527,774
Parsable	129,727 (16.00%)	35,619 (3.89%)	537,767 (65.88%)	402,249 (76.22%)
Compilable	986 (0.12%)	9,177(1.00%)	–	–
Runnable	–	–	163,247 (20.00%)	135,147 (25.61%)

- Removed single word snippets
+ Added heuristic repairs:
+ semi-colons against each statement
+ class structure (for Java only)

Results



	C#	Java
Total Snippets Processed after Removes	514,992	572,742
Parsable Snippets (Percentage)	129,691 (25.18%)	35,619 (6.22%)
Parsable Snippets After Applying Repairs (Percentage)	135,421 (26.30%)	110,203 (19.24%)
Compilable Snippets (Percentage)	986 (0.19%)	9,177 (1.60%)
Compilable Snippets After Applying Repairs (Percentage)	986 (0.19%)	17,286 (3.02%)

Overarching Goal?

Generate programs from smaller existing programs with minimal effort in your IDE.

Easy to search
Easy to integrate the code

Generate programs from smaller existing programs with minimal effort in your IDE.

SOF
GitHub
SourceForge

Easy to search
Easy to integrate the code

Generate programs from smaller existing programs with minimal effort in your IDE.

SOF
GitHub
SourceForge

Program Synthesis by Recombination and Learning

What is Program Synthesis?

Compiler v Synthesizer

Same goals:

Given a source program (the specification), generate a target program.

Find a program P that meets the specification φ(i/p, P)

Approach:

1. Generate AST of the program.
2. Rewrite the nodes/cluster of nodes in the AST in to the statements of the target language, until entire program is in the target language.

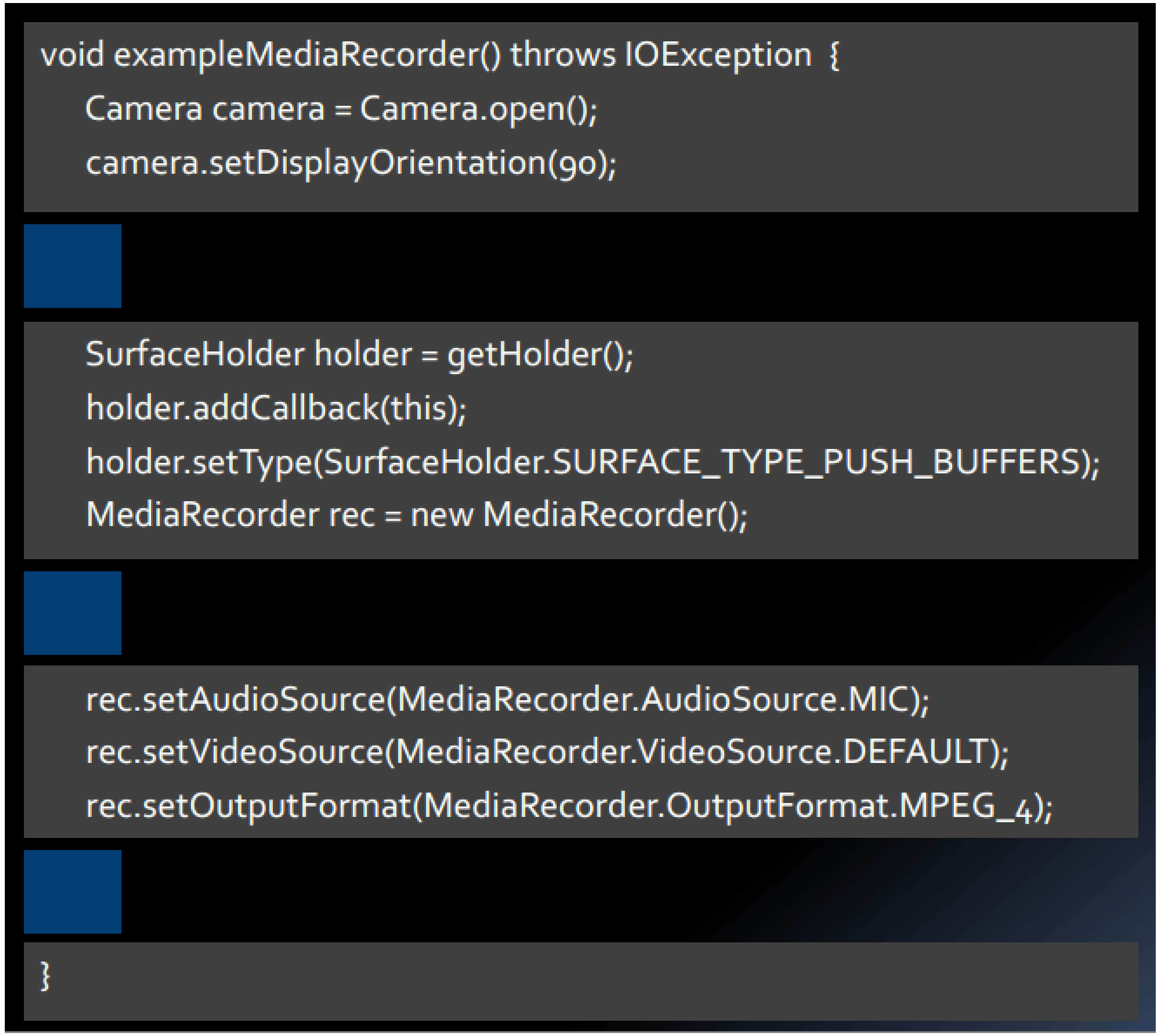
Ras Bodik Lecture:
http://www.cs.berkeley.edu/~bodik/cs294/fa12/Screencasts/L1/L1.html

The Difference?

Begin from a specification of source program
Rewrite the rules into language of desired program
The rewrite rules can be non-deterministic
There can be many sequences for a specification
Also does not stop if program cannot be lowered
Backtrack and analyze the program in other ways & generate sequence

Ras Bodik Lecture:
http://www.cs.berkeley.edu/~bodik/cs294/fa12/Screencasts/L1/L1.html

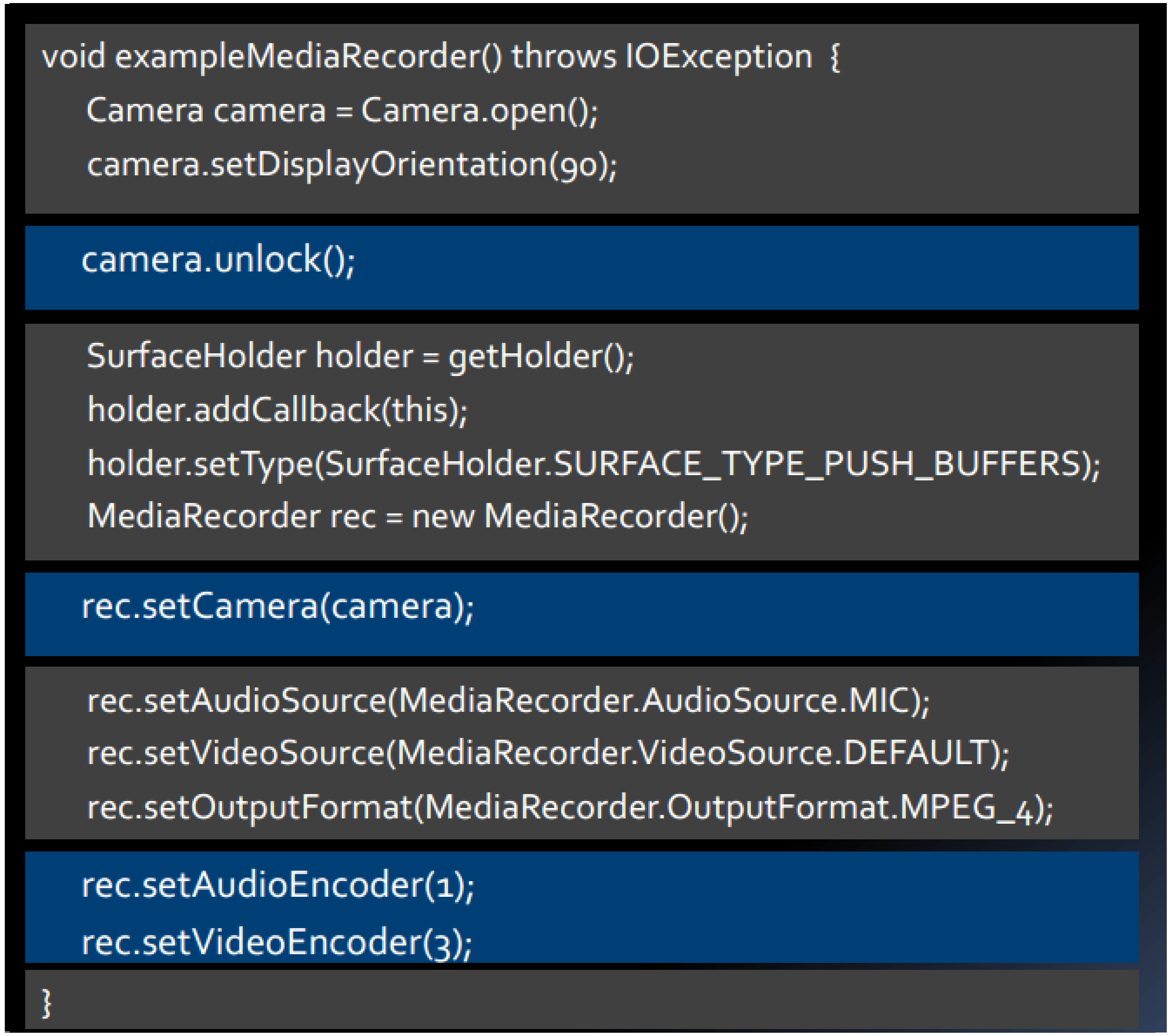
Code Completion with Statistical Language Models *Raychev et al. 2014*



```
void exampleMediaRecorder() throws IOException {
    Camera camera = Camera.open();
    camera.setDisplayOrientation(90);

    SurfaceHolder holder = getHolder();
    holder.addCallback(this);
    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    MediaRecorder rec = new MediaRecorder();

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

}
```

Martin Vechev Lecture:
http://www.srl.inf.ethz.ch/workshop2014/eth-vechev.pdf

Code Completion with Statistical Language Models *Raychev et al. 2014*



```
void exampleMediaRecorder() throws IOException {
    Camera camera = Camera.open();
    camera.setDisplayOrientation(90);

    camera.unlock();

    SurfaceHolder holder = getHolder();
    holder.addCallback(this);
    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    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);
}
```

Martin Vechev Lecture:
http://www.srl.inf.ethz.ch/workshop2014/eth-vechev.pdf

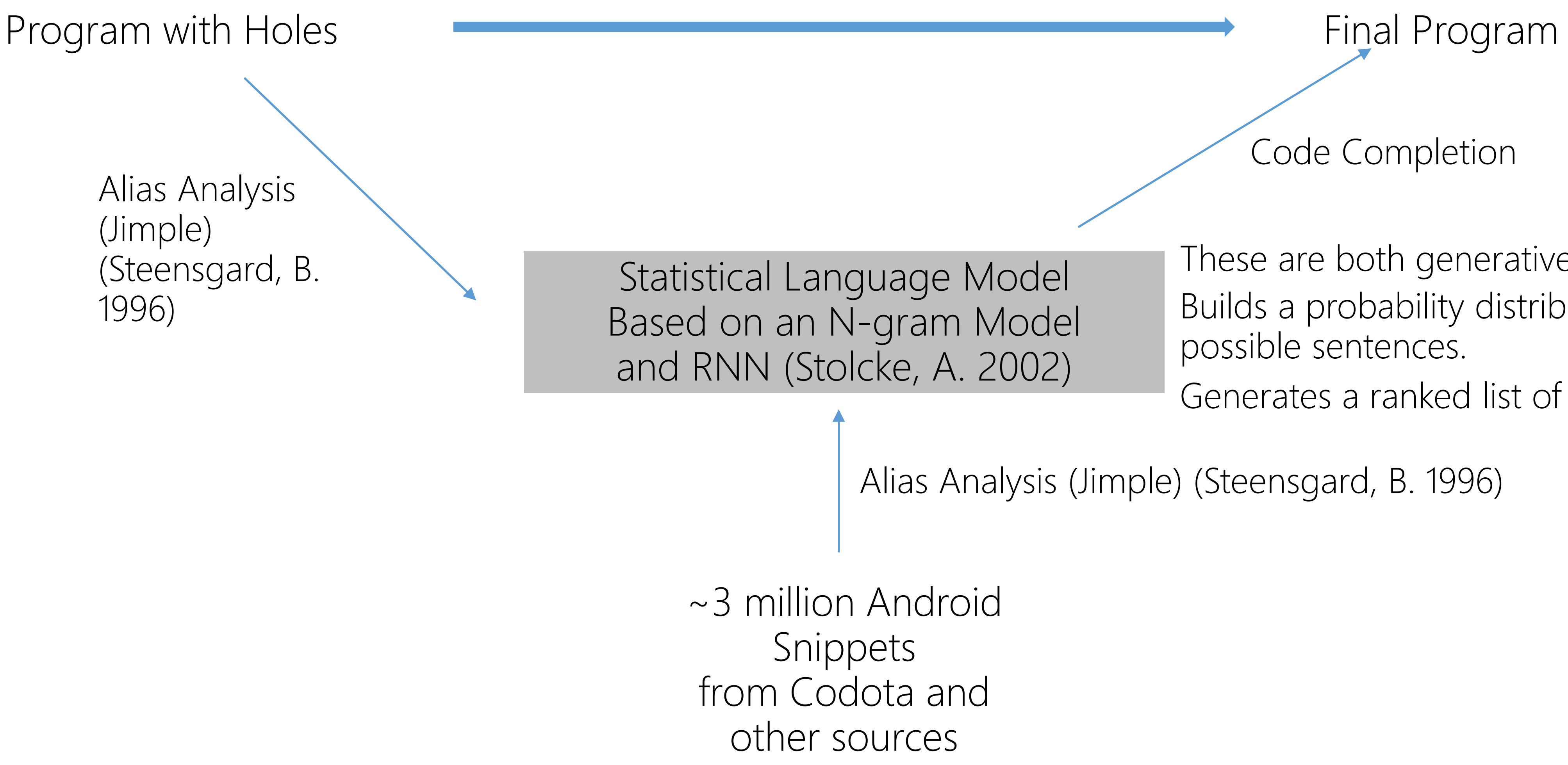
Code Completion with Statistical Language Models *Raychev et al. 2014*

```
----------------------------------------
----------------------------------------
----------------------------------------
?{params}// H1
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
?{params}// H2
----------------------------------------
```

$$\longrightarrow$$

```
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
```

Code Completion with Statistical Language Models *Raychev et al. 2014*

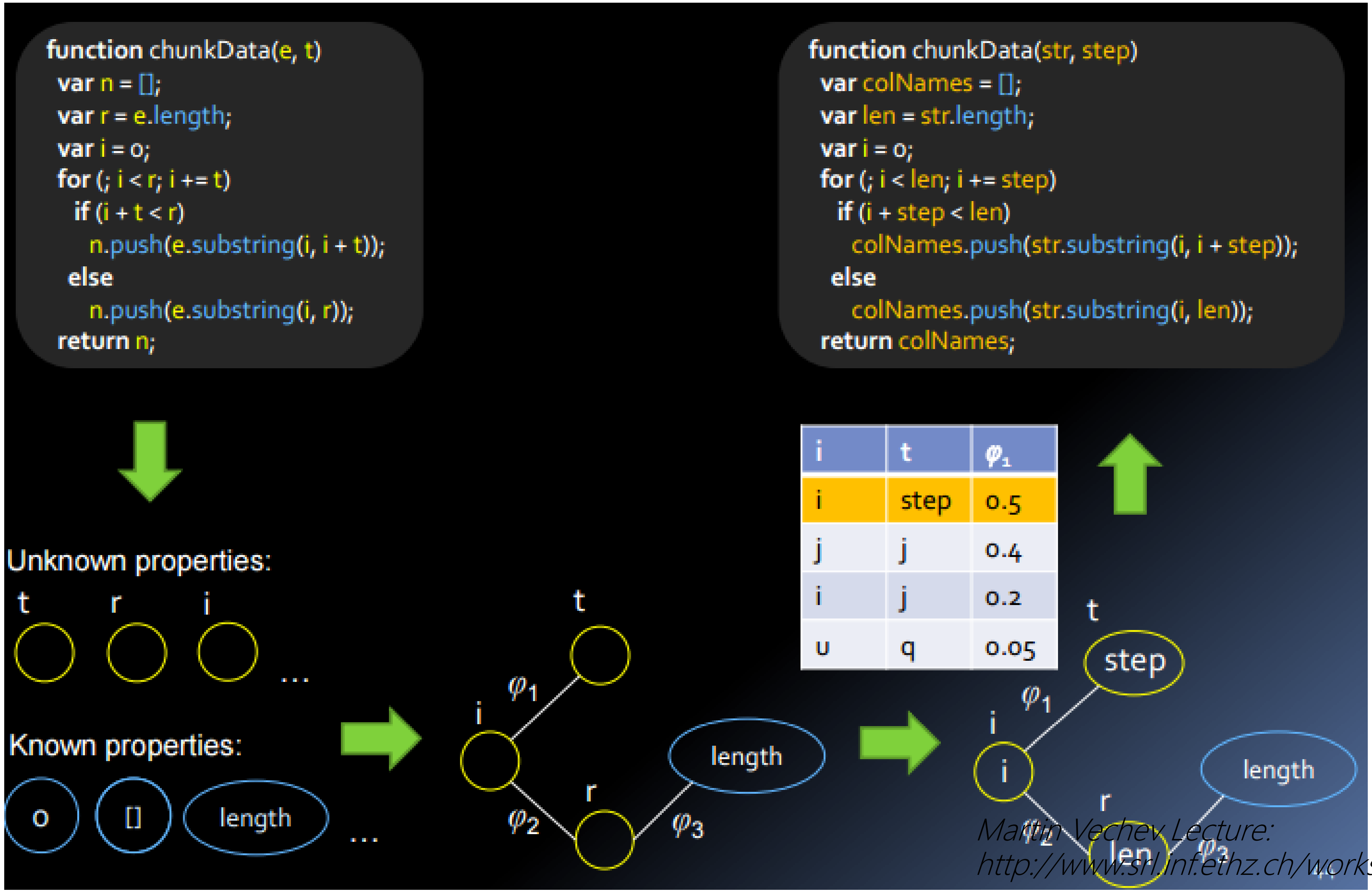

Program with Holes
Final Program
Alias Analysis (Jimple) (Steensgard, B. 1996)
Code Completion
Statistical Language Model Based on an N-gram Model and RNN (Stolcke, A. 2002)
Alias Analysis (Jimple) (Steensgard, B. 1996)
~3 million Android Snippets from Codota and other sources

These are both generative models.

Builds a probability distribution over all possible sentences.

Generates a ranked list of candidate.

JSNice: Raychev et al. – Predicting Program Properties from "Big Code"



```
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;
```

i	t	φ_1
i	step	0.5
j	j	0.4
i	j	0.2
u	q	0.05

Martin Vechev Lecture: http://www.srl.inf.ethz.ch/workshop2014/eth-vechev.pdf

Our Long Distance Goal:

Synthesize code into your IDE from Stack Overflow

Realities:

1. Not All Answers are meant to be synthesized.
E.g.   "What is the difference between a C# Reference and a Pointer?"
Strategy: Identify and reject some classes of questions from consideration.

2. Answers split in the form of code chunks, with explanations in between.
Strategy: Try to combine those code snippets, using help from explanations, other SOF snippets, open source projects.

Refined Goal:

To make SOF Code usable.

Thank you!