CSC200h Spring 2016 Guest Lecture

Mitchell Gordon

mgordon.me

m.gordon@rochester.edu
Your Speaker

● I am not a professor
● Senior URCS undergrad
● I do HCI systems building research
● Recently admitted to PhD programs, now choosing which offer to accept
● Internships at Google (Google Keyboard, YouTube) and CMU (HCII)
● Several publications. Worked with Profs Jeff Bigham, Walter Lasecki, Philip Guo
● CRA Outstanding UGrad winner
Agenda

First, grad school.

Second, presenting 2-3 unrelated projects:

1. WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding
2. Codepourri: Creating Visual Coding Tutorials Using A Volunteer Crowd Of Learners

IF TIME REMAINS

What is grad school (PhD)

- 5 - 7 years
- Take a few classes first 2 years, then all research (but lower ranked schools often have more coursework requirements)
- Free tuition, paid small stipend (enough to live off of)
- Goal is to publish papers at top conferences
Why I’m going to grad school

- Ease of switching. Can drop out with MS after 2 years.
- It’s hard (if you do it well). Way harder than being a SWE at most companies.
- Exclusivity (these days it’s harder to get into Stanford than Google).
- Intellectual freedom. Pursue my own interests, within some constraints.
- Surrounded by crazy smart people.
- Learn diverse set of skills: coding, writing, giving talks, framing/selling your work, networking, advising, teaching.
- BE ON, AND CREATE, THE FOREFRONT OF HUMAN KNOWLEDGE (area dependant).
- Chance at really cool jobs afterwards.
Why NOT go to grad school

• More money
  ○ Salary ("stipend") of ~$40k/yr
  ○ Compare that to new grad total comp of ~$150k/yr at top SV companies. And way higher after 5-6 years. You’ll never make that up.

• Guaranteed a better job
  ○ Only ~10% become faculty
  ○ Industrial research labs (MSR) are also cool but still hard to get into and as a concept slowly dying
  ○ Many PhDs, even at top schools, end up doing the same SWE job they could have done straight out of undergrad

• Academia and publishing is kind of screwed up
How PhD admissions work (at top schools)

- Decided by a committee of professors.
- They are looking for RESEARCHERS. That's it. This isn't law/med/business school.
- What matters, in order: letters, top publications (but most applicants don't have any), research experience, SOP, grades, GRE, everything else.
- Acceptance rates are ~5%
How I got in to grad school

- Accepted everywhere I applied
- Excellent letters from 4 well-known researchers
- Track record of success (in the form of top publications) in several groups. Side note: this also helped me get awards and get people to want work with me
- Average GPA (probably below average for top schools)
- SOP that clearly spelled out my experience and vision
- Key: most professors knew of me before I even applied
Codepouri: Creating Visual Coding Tutorials Using A Volunteer Crowd Of Learners

Mitchell Gordon and Philip Guo
University of Rochester
Learning To Code Is Hard

- Cognitively complex task that requires one’s mind to manipulate abstract and dynamic state.
- Struggle to develop mental models of code execution.
What’s Hard About CS Tutorials?

- Tutorials can help.
- They’re tedious to create.
- Static text-based format cannot visualize what happens as code executes.
Python Tutor - (pythontutor.com)

a.)

```python
1 x = ['a', 'b', 'c']
2 y = x
3 z = [y, 'd', 'e']
4 y = z
5 print x
6 print y
7 print z
```

b.)

Edit code

```
line that has just executed
```

```
next line to execute
```

Program output:

```
['a', 'b', 'c']
```

c.)

[Guo SIGCSE 2013]
Codepourri

- Displays a piece of code and allows the user to step through its execution, add annotations, view annotations left by other users, and vote on what they think are the best ones.
def inner(string, width):
    return string + " " * (width - len(string))

def outer(string1, string2):
    w = max(len(string1), len(string2))
    print("*" * (w + 4))
    print("* " + inner(string1, w) + " *")
    print("* " + inner(string2, w) + " *")
    print("*" * (w + 4))

outer("Box number one", "is very fun")
outer("Box #2", "(not False) is " + str(not False))
Creating Tutorials Using A Crowd Of Learners

- More scalable than just using experts.
- Overcomes “expert blind-spot.”
- We are the first to leverage a crowd of anonymous volunteer learners to create coding tutorials.
Crowdsourcing Recruitment

pythontutor.com website top banner.

```
1  def inner(string, width):
2      return string + " " * (width - len(string))
3  
4  def outer(string1, string2):
```

What does this part of the code do?

Want to help our research? Click here!
Crowdsourcing Workflow

```python
1 def inner(string, width):
2     return string + " " * (width - len(string))
3
def outer(string1, string2):
4     w = max(len(string1), len(string2))
5     print("*" * (w + 4))
6     print("* " + inner(string1, w) + " ")
7     print("* " + inner(string2, w) + " ")
8     print("*" * (w + 4))
9
eo = inner("Box number one", "is very fun")
o = outer("Box #2", "(not False) is " + str(not False))
```

What does this part of the code do?
Crowdsourcing Problems: Density

- When dropped at 1st step, most workers heavily annotated first few steps and then left.
- Weighted routing.
Crowdsourcing Problems: Asking The Right Questions

- Initial instructions just asked workers to “annotate the code”. Led to disparate annotation content.
- Tried analyzing code and asking specific questions. E.g. “What is this variable being assigned to?” This limited creativity.
- Ended up with “What does this part of the code do?”
Evaluation: Raw Annotations

- Ran a study that used Codepourri to annotate 2 pieces of code -- easy (7 lines) and medium difficulty (12 lines).
  - 145 crowd-contributed annotations.
- Annotations evaluated by 4 experts (3 professors and a TA).
- Experts created tutorials themselves to get a sense for the code, then rated annotations.
Annotation Quality: Results

- Experts used a scale ranging from *BS* to “great and surprising.”
- 65% of comments rated at least to have accurate content.
- 17% of correct annotations also had an *Surprising* label: they provided insights that even our experts did not think to provide.
Sample Annotations

BS - “what”

Correct - “checks if the age is over 65”

GS - “This checks whether age is greater than 65. If so, it will execute the print statements located in the indented block below it.”

Expert - “check if ready for retirement”
Creating The Final Tutorial

- Want to create a single tutorial from all the annotations gather from crowd.
- Each step has at most one annotation.
- Should always use the best annotation we have available.
- Used the crowd to decide. Experiment with 2 aggregation methods.
Comparison Voting

20% of votes were not for a worker’s own annotation.
"Choose The Best"

```python
1  birthYear = 1928
2  currentYear = 2013
3  age = currentYear - birthYear
4  if age > 65:
5      print('Enjoy retirement!')
6      print('You turn 100 in', 100+birthYear)
7  print('You are', age, 'years old')
```

- Initializes variable "birthYear" with integer value "1928"
- >>> 50 - 5*6
- Variable string your birth year
- Create a variable birthYear as an int, and initialize it with default value of 1928
Overall Quality Of Tutorials From Codepourri

- Crowd-created tutorials comparable to those created by experts.
- Some annotations less precise than experts, but others more detailed and better explained concepts.
- Experts suffered from fatigue, and it showed in their tutorials.
- Drawback: crowd-created tutorials sometimes had inconsistent writing style, could be jarring.
Codepourri: Creating Visual Coding Tutorials Using A Volunteer Crowd Of Learners
Mitchell Gordon and Philip Guo

- Codepourri enables visual coding tutorials by annotating execution steps within an automatically-generated program visualization.
- Novel crowdsourcing workflow where learners volunteer to make annotations within Codepourri and then vote on the best ones to use in a tutorial.
- First to leverage a crowd of anonymous volunteer learners to create coding tutorials.
- Comparable to expert-created ones for intro-level code.

Thank you. Questions?
Web Accessibility Evaluation with the Crowd: Using Glance to Rapidly Code User Testing Video

Mitchell Gordon
University of Rochester | CMU

With thanks to: Walter Lasecki, Erin Brady, Jeff Bigham
User Tests

- User test: testing your site with real users with disabilities
- W3C recommended
User Tests

• Most web developers don’t do user testing
Behavioral Video Coding

• Insight: when a developer watches video of their user tests, they are doing informal video coding
Behavioral Video Coding

[Image courtesy of Malte Jung]
What to code for?

• WCAG2 guidelines
• E.g. “user encounters unreadable or difficult to understand text.”
Behavioral Video Coding

Video coding

5-10x

Video playback

Time
Behavioral Video Coding

crowd of video coders
Behavioral Video Coding
Glance

Hours of Video

Glance Server

Response in seconds

Combine Answers

Crowd
Accuracy: Aggregation
Glance: Web developers UI
Feedback: Helping Web developers

• Agreement predicts quality
• WCAG 2 language is vague
  – Analysts may need to re-phrase
  – Workers may need examples
Evaluation

- 20 MTurk workers, 3.5 minute video of visually-impaired users using a website on a desktop computer
- 90% accuracy
- Full Glance eval in UIST 2014 Glance paper
Envisioned Interaction

1. Web dev records videos of users interacting with website (possibly in parallel)
2. Use Glance to automatically evaluate videos for violations of WCAG2 guidelines.
3. Use Glance’s results to fix trouble spots, improve accessibility.