CS Research on MOOCs and Online Education

• Amy Bruckman (organizer)
  – Professor of Interactive Computing, Georgia Tech
• Marti Hearst
  – Professor of Computer Science, School of Information, UC Berkeley
• Rob Miller
  – Professor of Computer Science and Engineering, Massachusetts Institute of Technology
• Scott Klemmer
  – Associate Professor, Cognitive Science and Computer Science & Engineering
Computer Science
Research on Higher Education?

• Future of online education is not just about trying things out
  – Research needed
  – Interdisciplinary
    • Industry
    • Schools of education
    • Schools of computer science

• How do we support this new interdisciplinary practice?
Intriguing Research Challenges

• Three strong examples of CS research on online education
• Discussion of the challenges
• Discuss both MOOCs and distance ed
  – Make clear which issues refer to which

• Thanks to Andy, Greg, and Brent
Design at Large
RESEARCH EXAMPLES

- peer assessment
- richer feedback
- small-group discussions
- machine+peer learning

Predict
Identify
Verify
Results
Peer assessment

1) Practice
2) Assess 5 Peers
3) Self-Assess

- Yes! Certificate: 92%
- Got it; didn’t deserve it: 6%
- Wrongly denied :(!: 2%

Peer and Self Assessment in Massive Online Classes, Kulkarni et al., TOCHI 2013
Peer assessment in 100+ classes

AT LARGE...

Human-computer Interaction Design
Programming in Python Code
Introduction to Philosophy Essays
Teaching character Management

Child Nutrition Recipes
Social Psychology Essays
Constitutional law Arguments
World Music Music

Peer and Self Assessment in Massive Online Classes, Kulkarni et al., TOCHI 2013
Qualitative, personalized feedback

- Peers can recognize errors from a list of patterns, even if they can’t articulate them
- Most errors are variations on a theme

FORTUNE COOKIES

Peer and Self Assessment in Massive Online Classes, Kulkarni et al., TOCHI 2013
Alone Together?
LEVERAGING DIVERSE EXPERIENCES

Small groups in massive classes

“It was like a mini-UN. We had an Australian currently residing in Dubai, an Afghan, a Romanian, an Indian & myself (a Pakistani).”
IDENTIFY-VERIFY
Creating Micro-Experts

- richer semantics increase quality

from scores to labels
- ______
- ______ 
- ______

Scaling Short-answer Grading by Combining Peer Assessment with Algorithmic Scoring, Kulkarni, Socher, Bernstein, & Klemmer, Learning at Scale, 2014
Machines modulate peer grading

Predict → Identify → Verify → Results

Identify-Verify

Scaling Short-answer Grading by Combining Peer Assessment with Algorithmic Scoring, Kulkarni, Socher, Bernstein, & Klemmer, Learning at Scale, 2014
CS RESEARCH OPPORTUNITY

• Build practical theory with real-world experiments
• Bake pedagogy into software that transforms learning

http://d.ucsd.edu/peer
“Nothing is as practical as a good theory”

“The best way to understand something is to try and change it”

-Kurt Lewin
• Build practical theory with real-world experiments
• Bake that theory into software that transforms <X>
Real experiments are critical
We Need to Do These 3 Things

• Insure that learners understand their role in experiments they opt in to
  
  Good design is key, and nuanced

• Insuring broad research access to conducting experiments, evaluating data, & open science
  
  Chairs: you have an important role here

• Few current CS curricula don’t teach experimental design. More should.
  
  Especially in data/HCI/learning tracks

http://cs303.stanford.edu
We Have Resources for You

• Open-source platforms with analytics, course materials, instructor resources, & graduating students :)
The Big Research Opportunity

• Tomorrow’s online class won’t look like today’s (I hope)

How might we...

scale personalized mastery-learning experiences?

http://d.ucsd.edu/peer
Why CS?

• The scientific opportunities are tremendous
• Concrete problems are a great forge for fundamental insights
• A proud history of lifelong learning
• The CS legacy: don’t just understand the world, make it a better place

Fred Brooks, *The Computer Scientist as Toolsmith*
with Chinmay Kulkarni + many collaborators

http://d.ucsd.edu/srk

@DesignAtLarge

follow student work at #hci5
Online Education with Learnersourcing

Rob Miller
User Interface Design Group
MIT CSAIL

Joint work with Juho Kim, Sarah Weir, Elena Glassman, Philip Guo, Carrie Cai, Max Goldman, Phu Nguyen, Rishabh Singh, Jeremy Scott
MOOCs: a New Scale for Learning

• Big problem
  – we’re very far from 1-on-1 mastery learning
  – little human feedback, mass production instead of personalization, high attrition rates

• Huge opportunity
  – much faster controlled experimentation & iterative improvement
  – big online crowds can do amazing things by themselves
Crowdsourcing vs. Learnersourcing

• Crowdsourcing
  – asking a crowd to do micro-work for problems we can’t solve with software
  – what does the crowd get in return? money, fun, social

• Learnersourcing
  – asking students to do micro-work for an online course
  – what do students get in return? learning (hopefully)

• Types of learnersourcing
  – active: asking people to do something
  – passive: watching what people do and inferring something

• Discussion forums are active learnersourcing
  – and without them, our current MOOCs would utterly fail
A Few Examples from My Group

Lecture video analytics
- find bugs and key parts in lecture videos
- passive learnersourcing

• Peer code review
  - students give feedback to each other
  - active learnersourcing

• Solution analytics
  - understand the range of solutions to a coding assignment
  - passive learnersourcing
LECTURE VIDEOS

Juho Kim
MOOC lecture videos
Challenge for instructors/editors

- Don’t know how students use lecture videos
  - Confusion
  - “Aha” moments
  - Bored
  - Re-watching important parts

- We analyzed video interaction data from the lectures in 4 edX courses
  - Clickstream (play, pause, scrub)

<table>
<thead>
<tr>
<th>Course</th>
<th>Subject</th>
<th>University</th>
<th>Students</th>
<th>Videos</th>
<th>Video Length</th>
<th>Processed Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.00x</td>
<td>Intro. CS &amp; Programming</td>
<td>MIT</td>
<td>59,126</td>
<td>141</td>
<td>7:40</td>
<td>4,491,648</td>
</tr>
<tr>
<td>PH207x</td>
<td>Statistics for Public Health</td>
<td>Harvard</td>
<td>30,742</td>
<td>301</td>
<td>10:48</td>
<td>15,832,069</td>
</tr>
<tr>
<td>CS188.1x</td>
<td>Artificial Intelligence</td>
<td>Berkeley</td>
<td>22,690</td>
<td>149</td>
<td>4:45</td>
<td>14,174,203</td>
</tr>
<tr>
<td>3.091x</td>
<td>Solid State Chemistry</td>
<td>MIT</td>
<td>15,281</td>
<td>271</td>
<td>6:19</td>
<td>4,821,837</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>127,839</td>
<td>862</td>
<td>7:46</td>
<td>39,319,757</td>
</tr>
</tbody>
</table>
Interaction Peaks
Example: Beginning of new material

before transition

Idea: Admissibility

- Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the fringe
- Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs

after transition

Admissible Heuristics

A heuristic \( h \) is admissible (optimistic) if:

\[ 0 \leq h(n) \leq h^*(n) \]

where \( h^*(n) \) is the true cost to a nearest goal

visual transition

peak
Example: Backing up

before transition

def fact(n):
    """assumes that n is an int > 0"
    res = 1
    while n > 1:
        res = res * n
        n -= 1
    return res

def factR(n):
    """assumes that n is an int > 0"
    if n == 1:
        return n
    return n * factR(n-1)

peak

after transition

visual transition
LectureScape: Enhancing lecture videos
SOLUTION ANALYTICS
A Typical Programming Assignment

Write an iterative function that computes $a^b$

What did our students do?

Oops, the autograder should have caught that.

Oh no! Never do that!

Clever--I didn't know you could do it that way.
OverCode

- OverCode allows teaching staff to see the similarity and variation among thousands of solutions.
Solution Cleaning & Clustering

- OverCode makes solutions easier to read and cluster
  - Reformat code for consistency
  - Rename variables with identical behavior
  - Ignore statement order when clustering solutions

```python
def iterPower(base, exp):
    result = 1
    while (exp > 0):
        result *= base
        exp -= 1
    return result
```

```python
def iterPower(base, exp):
    result=1
    while exp>0:
        result*=base
        exp-=1
    return result
```

```python
def iterPower(base, exp):
    wynik=1
    while exp>0:
        wynik*=base
        exp-=1
    return wynik
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Total Solutions</th>
<th>Largest Stack</th>
<th>2nd Largest</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterPower</td>
<td>~3800</td>
<td>~40%</td>
<td>~10%</td>
</tr>
</tbody>
</table>
| def iterPower(base, exp):
  result=1
  while exp>0:
    result*=base
    exp-=1
  return result |
| hangman          | ~1100           | ~9%           | ~9%         |
| def getGuessedWord(secretWord, result):
  if letter in lettersGuessed:
    result+=letter
  else:
    result+='_'
  return result |
| computeDeriv     | ~1400           | ~1%           | ~0.5%       |
| def computeDeriv(poly):
  result=[]
  if len(poly)==1:
    return[0.0]
  for i in range(1,len(poly)):
    result.append(float(poly[i]))
  result.append(float(poly[-1]))
  return result |
Performance

OverCode preprocessing pipeline is **linear** with number of solutions and runs on a **laptop**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Correct Solutions</th>
<th>Running Time</th>
<th>Initial Stacks</th>
<th>Common Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterPower</td>
<td>3875</td>
<td>15m 28s</td>
<td>862</td>
<td>38</td>
</tr>
<tr>
<td>hangman</td>
<td>1118</td>
<td>8m 6s</td>
<td>552</td>
<td>106</td>
</tr>
<tr>
<td>compDeriv</td>
<td>1433</td>
<td>10m 20s</td>
<td>1109</td>
<td>50</td>
</tr>
</tbody>
</table>

Other clustering approaches are quadratic in number of solutions and need a computer cluster.
Feedback Coverage

% solutions covered by the teacher’s post

users: 12 teaching assistants
control: all solutions concatenated in a page
task: write a discussion forum post

Baseline
OverCode

iterPower
hangman
computeDeriv

% raw solutions covered by feedback
Teacher Perceptions

Interface Satisfaction (Forum Post Study)

<table>
<thead>
<tr>
<th></th>
<th>Control</th>
<th>OverCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>was easy to use</td>
<td></td>
<td></td>
</tr>
<tr>
<td>helped me get a sense of my students' understanding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>was overwhelming</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Likert Scale Rating (1-7)

Strongly Disagree ... Strongly Agree: 1-7

19
PEER CODE REVIEW
Problem: Feedback about Coding Style

- MIT 6.005 Software Construction
  - foundation-level programming course (replaced 6.001/6.170)
  - 400 students per year, mostly sophomores
- Students write lots of code
  - roughly 10kloc in problem sets and projects
- Automatic grading is necessary but not sufficient
  - we need human readers, and we want line-by-line feedback

```c
// compute n! requires n >= 0
int factorial(int n) {
  int i, result=1;
  if (n == 0) return 1;
  else {
    for (i = 1; i < n; ++i) result *= i;
    result = result*n;
  }
  return result;
}
```

```c
int factorial(int n) {
  int i, result=1;
  if (n == 0) result = 1;
  else {
    for (i = 1; i < n; ++i) result *= i;
    result = result*n;
  }
  return result;
}
```

- correct and understandable
- correct but confusing
Approach: Crowd-Driven Code Review

• Chop up student programs into **chunks**
• Review the chunks by a **mixed crowd**: students, staff, alums

• Anticipated benefits
  – faster, cheaper, more diverse comments
  – give practice with code reviewing (a widespread industry practice)
  – expose to good and bad solutions
  – reduce workload on teaching staff
  – incorporate alumni back into the course

• Not using for grading... yet
Caesar: Divide & Conquer

```java
public class RulesOf6005 {

/**
 * Tests if the string is one of the items in the Course Elements section.
 * @param name - the element to be tested
 * @return true if <name> appears in bold in Course Elements section. Ignores case (capitalization).
 * Example: "Lectures" and "lectures" will both return true.
 */
public static boolean hasFeature(String name) {
    String[] elements = { "lectures", "recitations", "laptops required", "text", "problem sets", "";
    String test = name.toLowerCase();
    for (int i = 0; i < 9; i++) {
        if (elements[i].equals(test)) {
            return true;
        }
    }
    return false;
}

/**
 * Takes in the quiz, pset, project, and participation grades as values out of a hundred and returns the grade based on the course information as also as a value out of a hundred, rounded to the nearest integer.
 * Behavior is unspecified if the values are out of range.
 * @param quiz
 * @param pset
 * @param project
 * @param participation
 * @return the resulting grade out of a hundred
 */
public static int computeGrade(int quiz, int pset, int project, int participation) {
    return (int)Math.round((quiz*.2) + (pset*.4) + (project*.3) + (participation*.1));
}

/**
 * Based on the slack day policy, returns a date of when the assignment would be due, making sure not exceed the budget. In the case of the request being more than what's allowed, the latest possible due date is returned.
 * Hint: Take a look at http://download.oracle.com/javase/6/docs/api/java/util/GregorianCalendar.html
 * Behavior is unspecified if request is negative or due date is null.
 */
```
Social Reviewing

- automatic style checker comments
- reviewer comments
- upvotes & downvotes
- replies & discussion
- reviewers can see whole program (not just chunk) if needed
- reviewers have a reputation (#upvotes, + 100 if they're alums or staff of the course)
Experience

Fall 2011
- 180 students
- 54 alums
- 15 staff

Spring 2012
- 215 students
- 0 alums
- 17 staff

13 problem sets, 2200 submissions

- 21,500 comments
  - 5% alums
  - 8% staff
  - 87% students

- 9.6 comments per submission

- 16.2% upvoted
- 0.7% downvoted

average time students spent reviewing

PS0 PS1 PS2 PS3 PS4 PS5 PS6 PS7
Kinds of Comments

- Bug
- Clarity
- Performance
- Simplicity
- Style
- Positive Learning

- I don't really understand what you're trying to test for here - why the 20? Small comments would help tell what you're doing.
- This could be implemented within the next for loop for a faster algorithm.
- You don't necessarily need this variable.
- I think this code could have been more simpler without using so many if statements. For example, you could have divided the case in which the first operand is not a scalar or not. It is hard to read.
- You should put this comment above the line, otherwise it runs off the page.
- This is nice and concise. (I didn't know you could iterate through an array like this in a for loop)
- This is interesting. Why do you store all the messages you send/receive in a log?

**Code author:** For debugging. The log adds time stamps, which help a lot for debugging concurrency problems.
A LOOK AHEAD
MOOCs Have to Run Themselves

- Launching a MOOC is like authoring a textbook
  - But keeping it running currently requires sustained expert involvement
  - In the long run, we can teach the world for free only if we don’t have to staff the MOOCs

- Implications
  - Intelligent tutor systems
  - Peer help, feedback, assessment
  - Alumni or external staff help
MOOCs Have to Improve Themselves

• edX and Coursera will be littered with stale MOOCs
  – Because faculty have no time or incentive to revise them
  – In the long run, MOOCs have to revise and improve themselves, automatically

• Implications
  – Crowdsourced content: exercises, quizzes, textbook, videos
  – FrankenMOOCs that combine the best stuff out there
  – Video content that can be edited like Wikipedia
MOOCs Are Big Data for Education

• Google and Bing drive information retrieval research
  - because they own the data & control the interface
• Facebook and Twitter increasingly drive social network research
  - again: data + interface
• Universities could be driving learning science in CS
  - if we step up and take ownership of the data + the interface
Summary

Lecturescape (lecture videos)

OverCode (programming solutions)

Caesar (peer code review)

future

• self-running MOOCs
• self-improving MOOCs
• MOOCs are our big data

Thanks to support from NSF, Quanta Computer, Google, edX