CSAIL

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

MIT HUMAN-COMPUTER INTERACTION

MOOCs: a New Scale for Learning

classroom 1:10

lecture 1:100

stadium 1:10,000

- 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

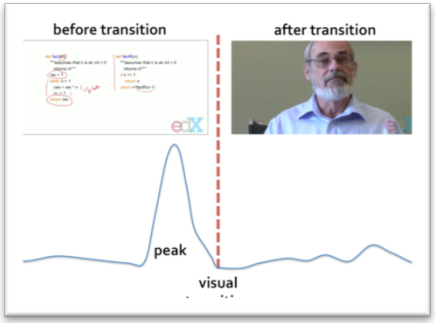
Crowdsourcing vs. Learnersourcing

- 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

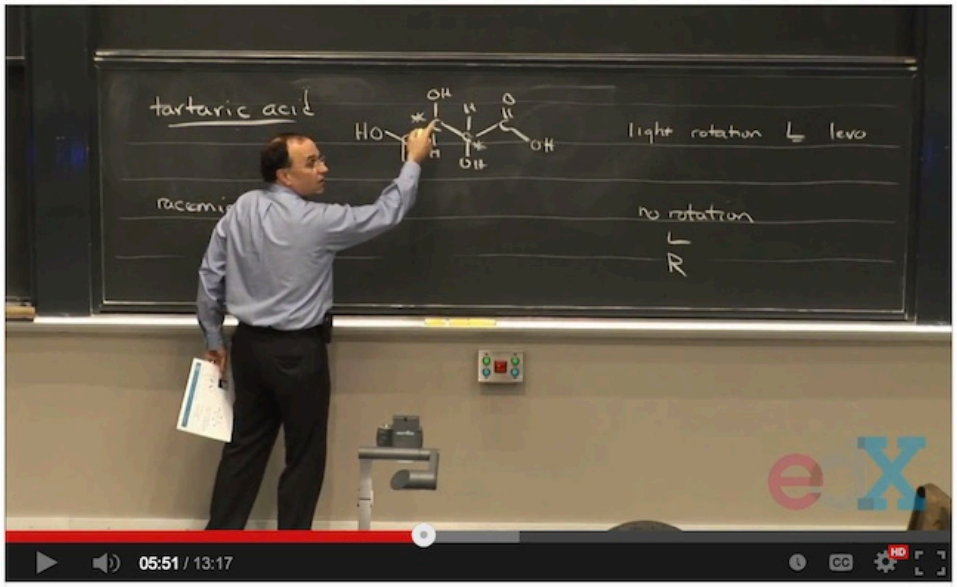
- Solution analytics
  - understand the range of solutions to a coding assignment
  - passive learnersourcing

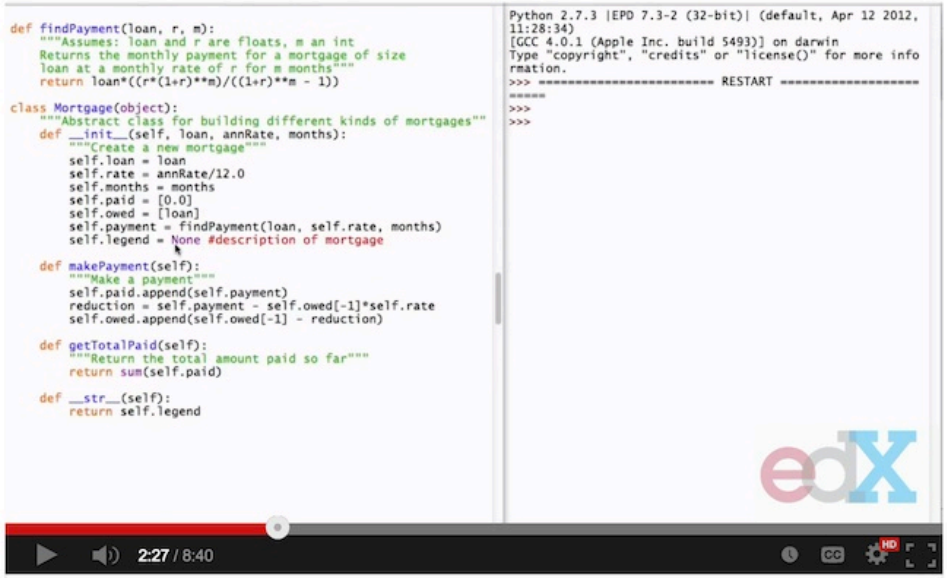
LECTURE VIDEOS

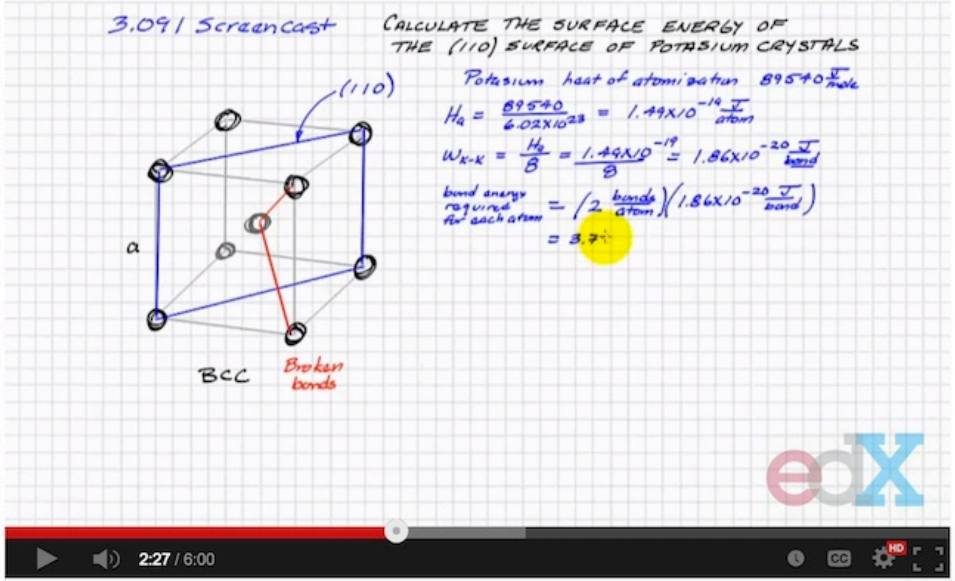
Juho Kim

MIT HUMAN-COMPUTER INTERACTION

MOOC lecture videos







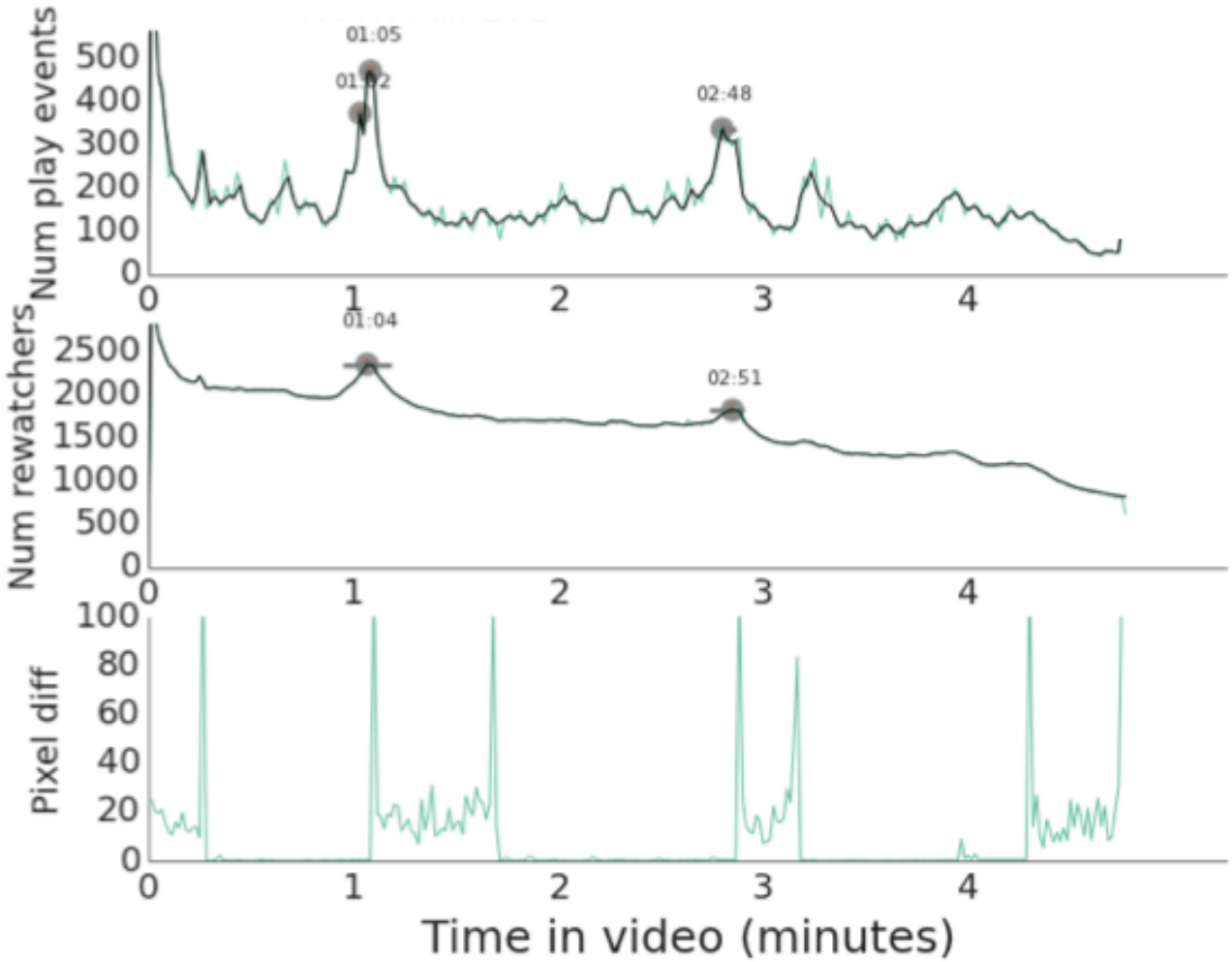
Challenge for instructors/editors

- 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)

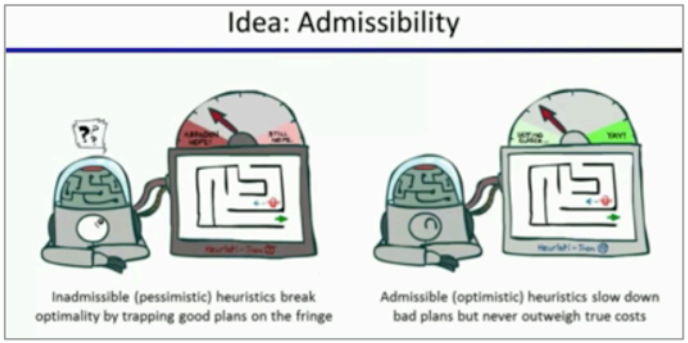
Course	Subject	University	Students	Videos	Video Length	Processed Events
6.00x	Intro. CS & Programming	MIT	59,126	141	7:40	4,491,648
PH207x	Statistics for Public Health	Harvard	30,742	301	10:48	15,832,069
CS188.1x	Artificial Intelligence	Berkeley	22,690	149	4:45	14,174,203
3.091x	Solid State Chemistry	MIT	15,281	271	6:19	4,821,837
Total			**127,839**	**862**	**7:46**	**39,319,757**

Interaction Peaks

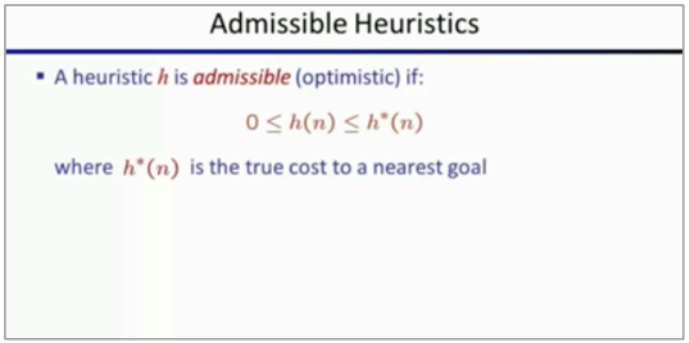


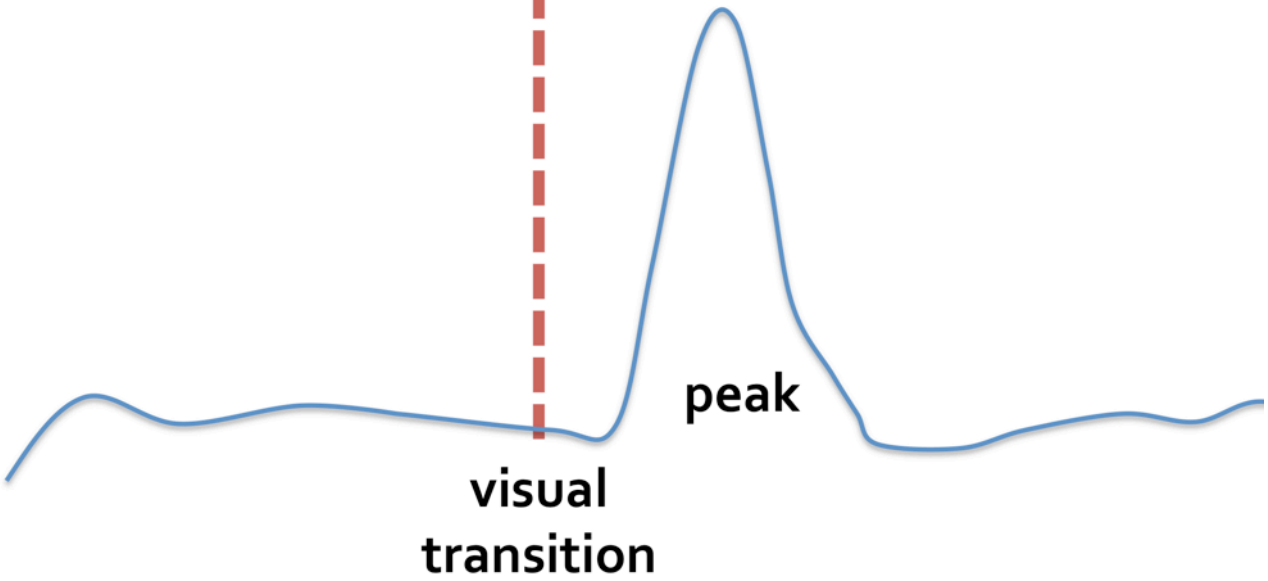
Example: Beginning of new material

before transition



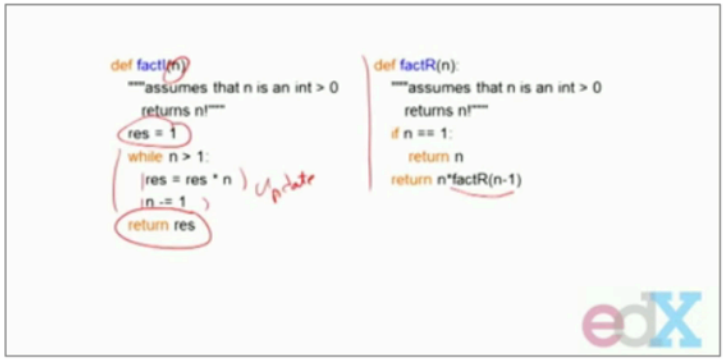
after transition



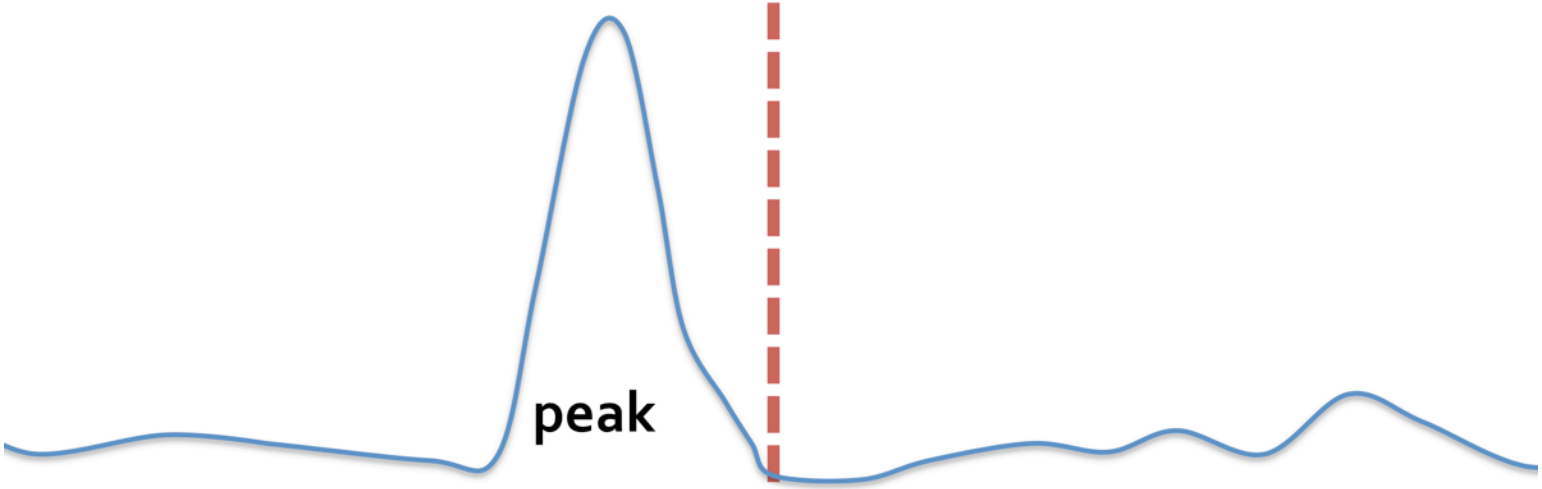


Example: Backing up

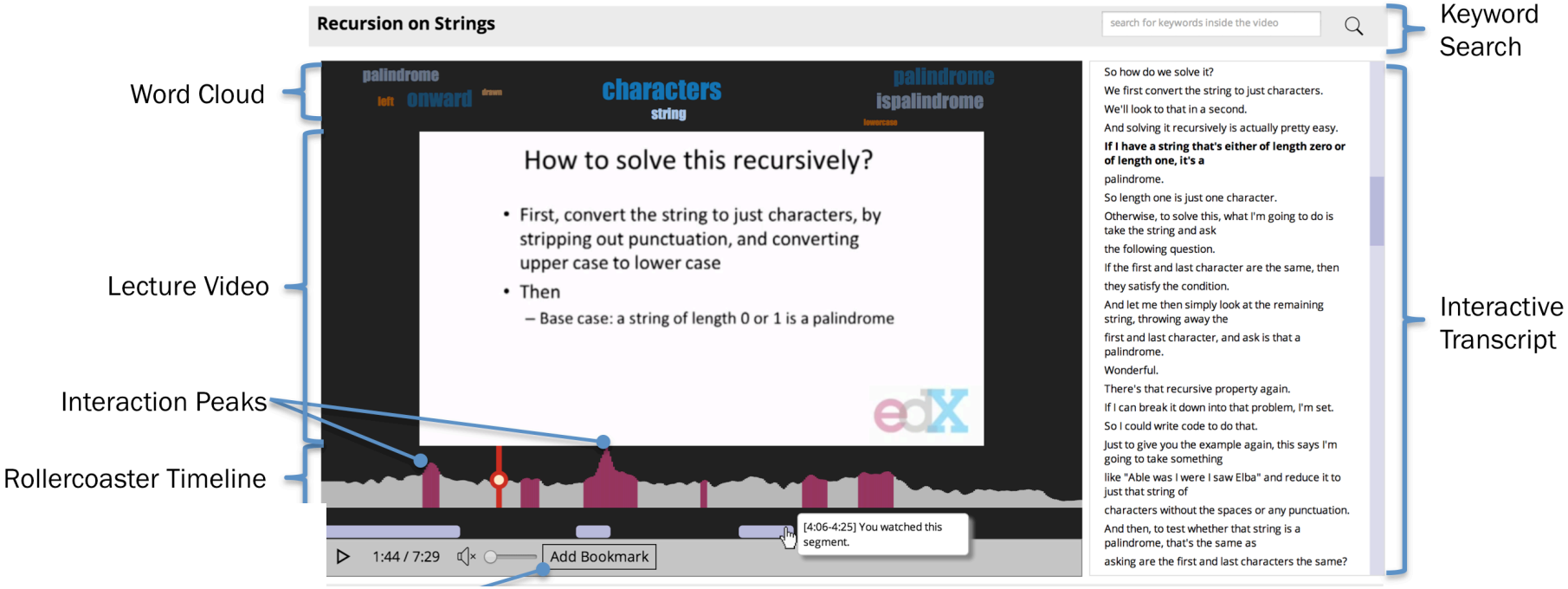
before transition



after transition



LectureScape: Enhancing lecture videos



SOLUTION ANALYTICS

Elena Glassman

A Typical Programming Assignment

Time

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.

iterPower

done

showing 862 total stacks

that represent 3842 total submissions

filtering by:

nothing yet

Filter | Rewrite | Legend

lines that appear in at least 50 submissions

77 base=resultB
2592 def iterPower(base,exp):
701 def iterPower(base,expB):
349 def iterPower(base,expC):
51 def iterPower(base,expD):
51 def iterPower(resultB,expC):
55 elif expC==1:
527 else:
2466 exp-=1
279 exp=exp-1
135 exp=expB
366 expC-=1
65 expC=expC-1
63 for i in range(0,expB):
174 for i in range(expB):
52 iC+=1

Largest stack (matching filters)

1534 id: 1

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

Remaining stacks (matching filters)

374 id: 3

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

153 id: 727

```
def iterPower(base,exp):
    result=1
    while exp>0:
```

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

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

```
def iterPower(base, exp):
    # Your code here
    res =1
    while exp> 0:
        exp-=1
        res *= base
    return res
```

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

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

	Total Solutions	Largest Stack		2nd Largest	

iterPower	~3800	1534
		`def iterPower(base,exp):`
		`  result=1`
		`  while exp>0:`
		`  result*=base`
		`  exp-=1`
		`  return result`
			~40%	374
				`def iterPower(base,exp):`
				`  result=1`
				`  while exp>0:`
				`  result=result*base`
				`  exp-=1`
				`  return result`
					~10%

hangman	~1100	97
		`def getGuessedWord(secretWord,`
		`  result=''`
		`  for letter in secretWord:`
		`  if letter in lettersGue`
		`  result+=letter`
		`  else:`
		`  result+='_ '`
		`  return result`
			~9%	92
				`def getGuessedWord(secret`
				`  resultB=''`
				`  for letter in secretWo`
				`  if letter in lette`
				`  resultB+=lette`
				`  else:`
				`  resultB+='_'`
				`  return resultB`
					~9%

computeDeriv	~1400	22
		`def computeDeriv(poly):`
		`  result=[]`
		`  if len(poly)==1:`
		`  return[0.0]`
		`  for i in range(1,len(poly))`
		`  result.append(float(pol`
		`  return result`
			~1%	13
				`def computeDeriv(poly):`
				`  result=[]`
				`  if len(poly)<2:`
				`  return[0.0]`
				`  for i in xrange(1,len(`
				`  result.append(floa`
				`  return result`
					~0.5%

Performance

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

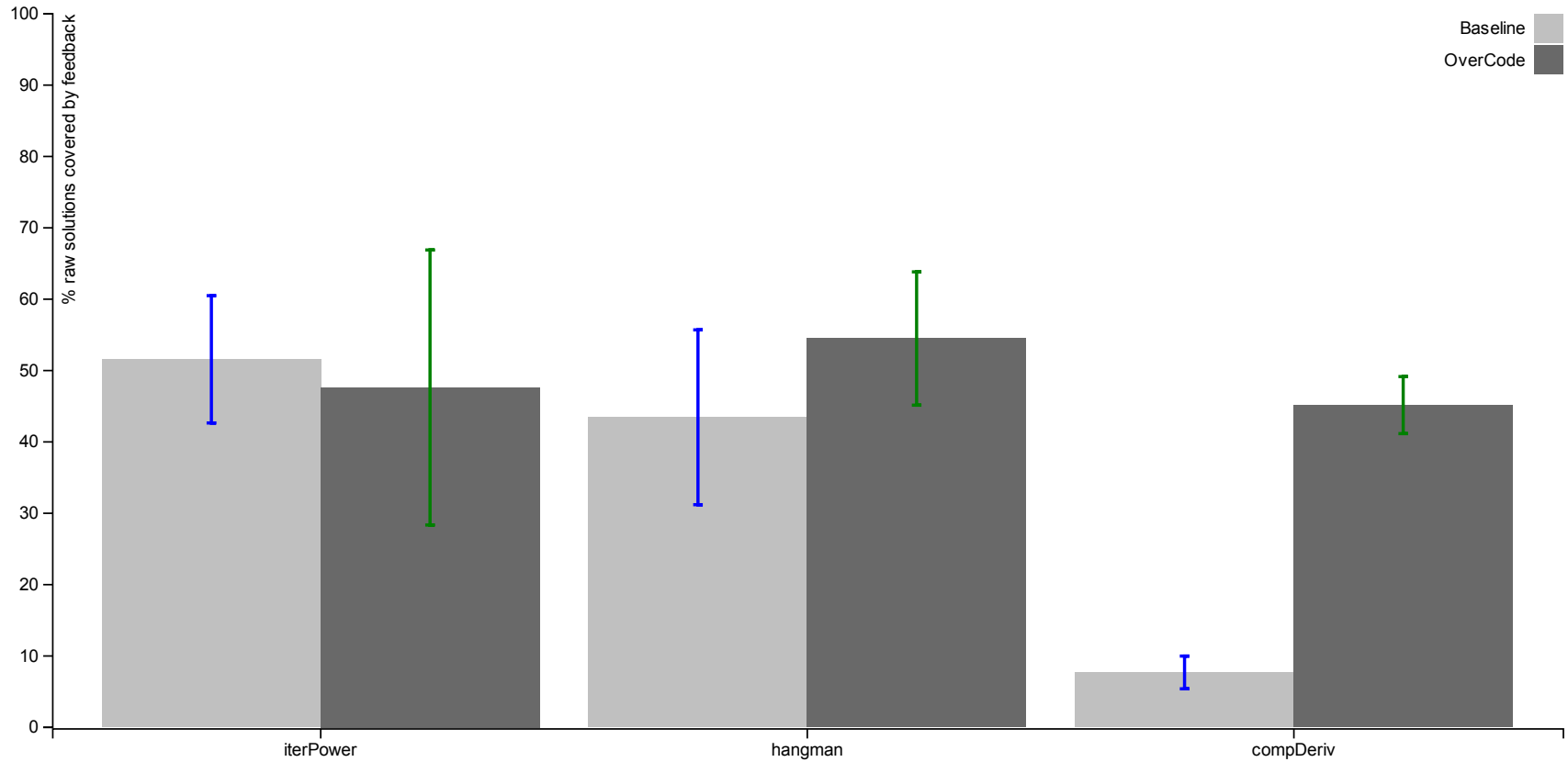
Problem	Correct Solutions	Running Time	Initial Stacks	Common Variables
`iterPower`	3875	15m 28s	862	38
`hangman`	1118	8m 6s	552	106
`compDeriv`	1433	10m 20s	1109	50

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

Feedback Coverage

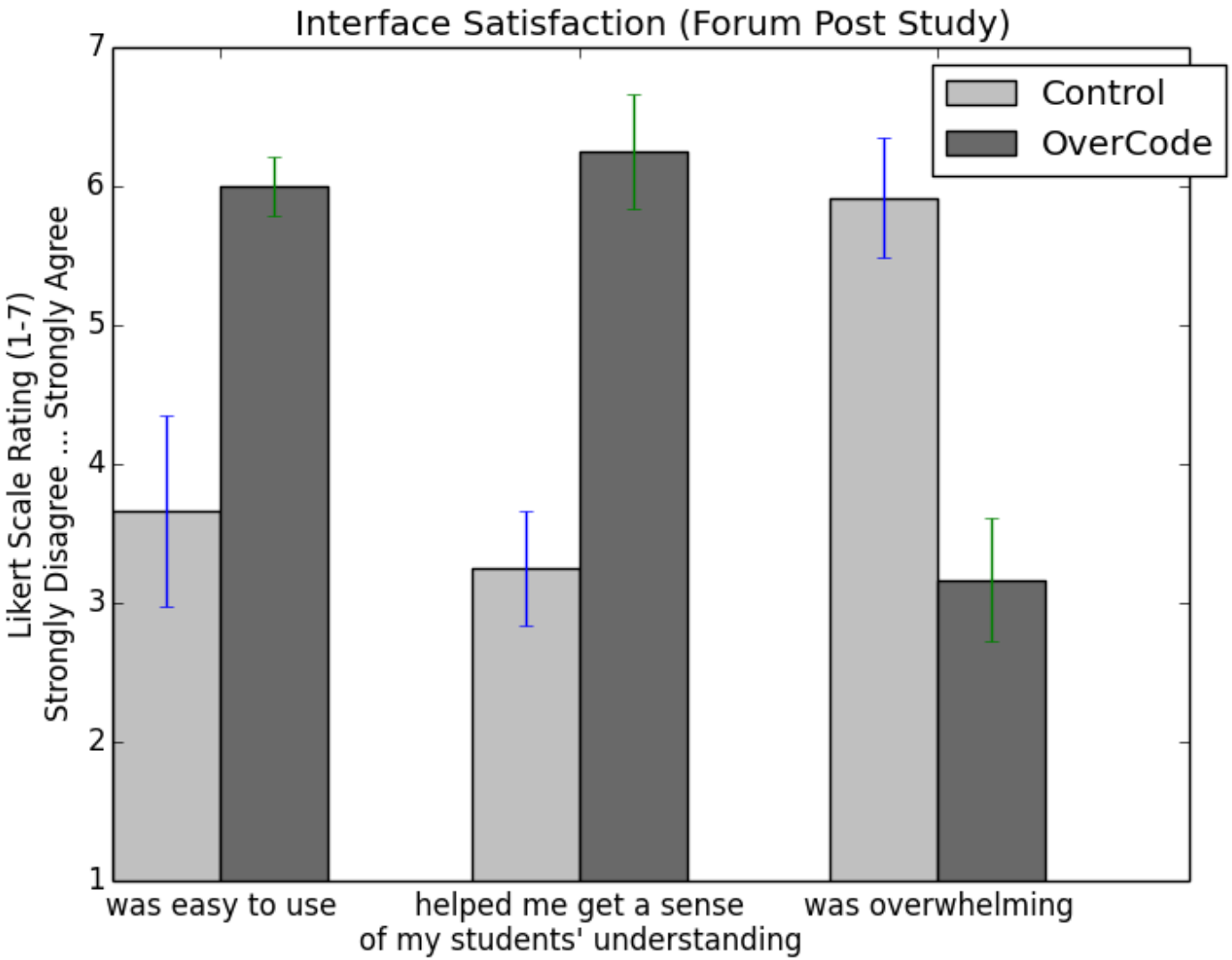
% solutions covered
by the teacher's post

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



iterPower hangman computeDeriv

Teacher Perceptions



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

```
// compute n!  requires n >= 0
int factorial(int n) {
    if (n == 0) return 1;
    else return n * factorial(n-1);
}
```

correct and understandable

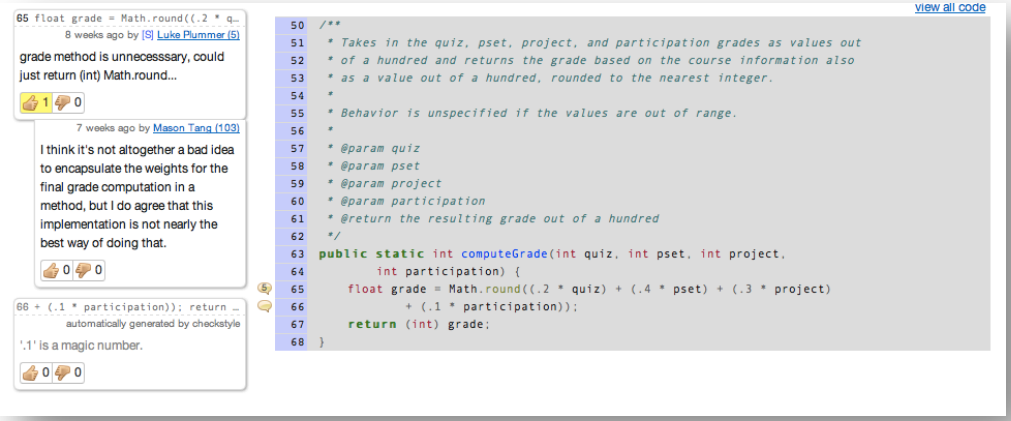
```
int factorial(int n) {
    int i, result=1;
    if (n == 0) result = 1;
    else {
        for (i = 1; i < n; ++i) result *= i;
        result = result*n;
        return result;
    }
    return 1;
}
```

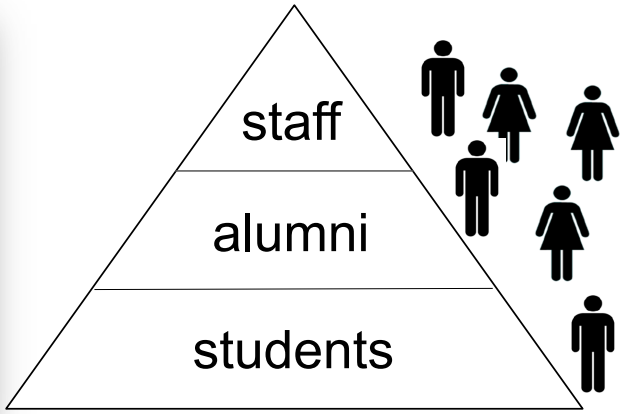
correct but confusing

  - we need human readers, and we want line-by-line feedback

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

```
14 public class RulesOf6005 {
15
16
17  /**
18   * Tests if the string is one of the items in the Course Elements section.
19   *
20   * @param name - the element to be tested
21   * @return true if <name> appears in bold in Course Elements section. Ignores case (capitalization).
22   * Example: "Lectures" and "lectures" will both return true.
23   */
24  public static boolean hasFeature(String name){
25          // TODO: Fill in this method, then remove the RuntimeException
26          String[] elements = { "lectures", "recitations", "laptops required", "text", "problem sets", "
27          String test = name.toLowerCase();
28          for (int ii = 0; ii < 9; ii++) {
29                  if (elements[ii].equals(test)) {
30                          return true;
31                  }
32          }
33          return false;
34  }
35
36
37  /**
38   * Takes in the quiz, pset, project, and participation grades as values out of a
39   * hundred and returns the grade based on the course information also as a value out
40   * of a hundred, rounded to the nearest integer.
41   *
42   * Behavior is unspecified if the values are out of range.
43   *
44   * @param quiz
45   * @param pset
46   * @param project
47   * @param participation
48   * @return the resulting grade out of a hundred
49   */
50  public static int computeGrade(int quiz, int pset, int project, int participation){
51          return (int)Math.round((quiz*.2) + (pset*.4) + (project*.3) + (participation*.1));
52  }
53
54
55  /**
56   * Based on the slack day policy, returns a date of when the assignment would be due, making sure not
57   * exceed the budget. In the case of the request being more than what's allowed, the latest possible
58   * due date is returned.
59   *
60   * Hint: Take a look at http://download.oracle.com/javase/6/docs/api/java/util/GregorianCalendar.html
61   *
62   * Behavior is unspecified if request is negative or duedate is null.
```

programs chopped into **chunks**

each chunk assigned to multiple reviewers

code to review

PrimeFactorsServer	5 1
PrimeFactorsServer	5 1
EchoClient	1 1
EchoClient	5 1
EchoClient	3 1
EchoServer	3 1
PrimeFactorsClient	6 1
PrimeFactorsServer	6 1
EchoClient	2 1
EchoClient	2 1

code recently reviewed

RulesOf6005.extendDeadline(..)	18 2
RulesOf6005.extendDeadline(..)	4 3
RulesOf6005.computeGrade(..)	9 3
RulesOf6005.extendDeadline(..)	6 2
RulesOf6005.computeGrade(..)	6 3
RulesOf6005.hasFeature(..)	3 2
RulesOf6005.hasFeature(..)	6 2
RulesOf6005.hasFeature(..)	4 2
RulesOf6005.hasFeature(..)	5 2
RulesOf6005.hasFeature(..)	4 2

Social Reviewing

CSAIL

12 /** * @param expression * a Stri…

automatically generated by checkstyle

ile contains tab characters (this is the first instance).

1 0

13 * @param expression * a String r…

5 weeks ago by Jason Juang (104)

Specify here that expression must not be null. You're throwing an unchecked exception when that happens, so you owe it to whoever is calling your method to explain that you're going to crash their program if they do that.

1 1

4 weeks ago by [T] Robert C Miller (107)

But a NullPointerException is the typical result when passing null anyway, and that's unchecked too.

view all code

```
12  /**
13   * @param expressio
14   *            a Str
15   *            the p
16   * @return the valu
17   *         units, e
18   */
19  public String evaluate(String
20      if (expression == null) {
21
22      Lexer lexer = new Lexer(ex
23      Parser p = new Parser(lexe
24      return p.evaluate().toStri
25  }
```

automatic style checker comments

reviewer comments

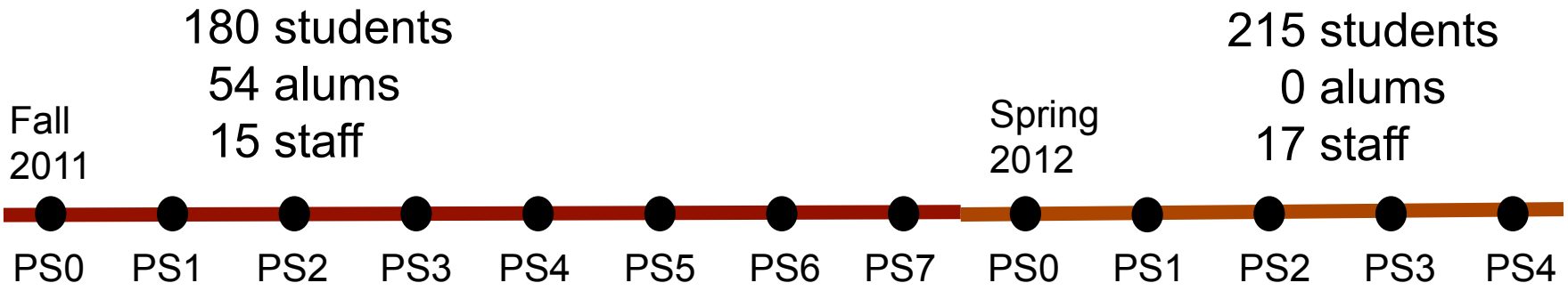
upvotes & downvotes

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

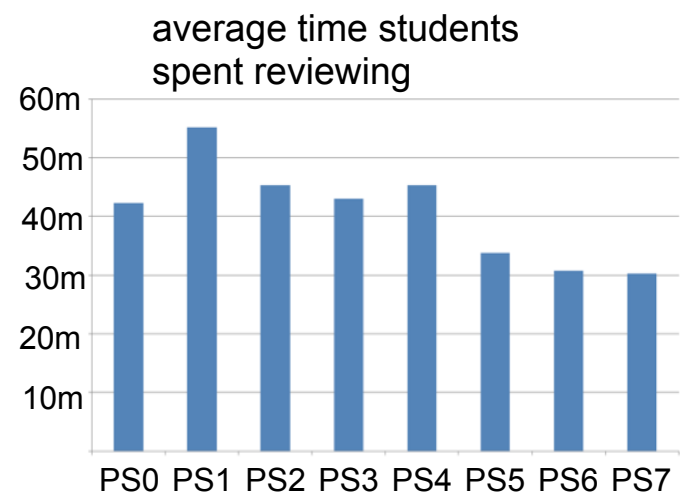


13 problem sets, 2200 submissions

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

16.2% upvoted
0.7% downvoted

9.6 comments per submission



Kinds of Comments

Bug
Clarity
Performance
Simplicity
Style
Positive
Learning

- I don't really understand what you're

- 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?

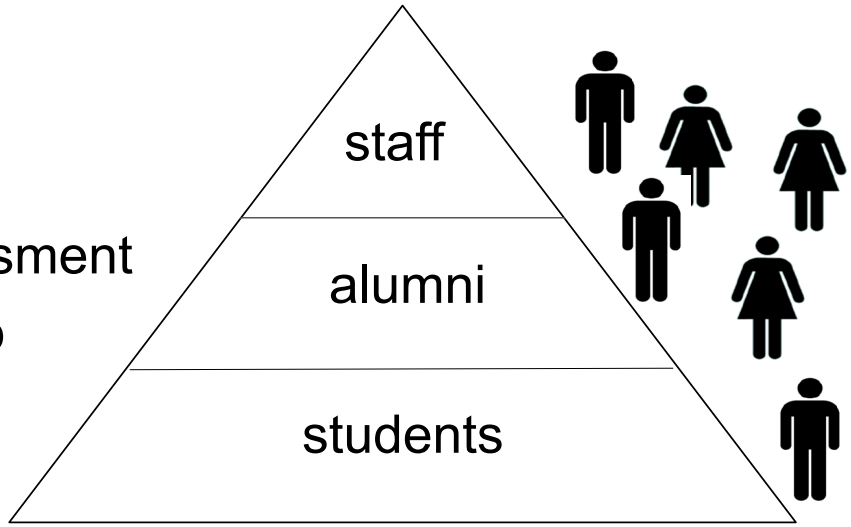
  Code author: For debugging. The log adds time stamps, which help a lot for debugging concurrency problems.

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

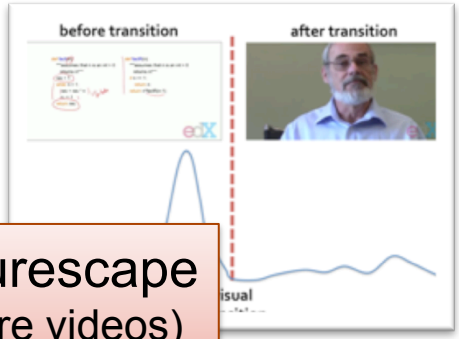
- edX and Coursera will be littered with stale MOOCs
  - Because faculty have no time or incentive to revise them
  - In the long run, MOOCs have to revise and improve themselves, automatically

- Implications
  - Crowdsourced content: exercises, quizzes, textbook, videos
  - FrankenMOOCs that combine the best stuff out there
  - Video content that can be edited like Wikipedia

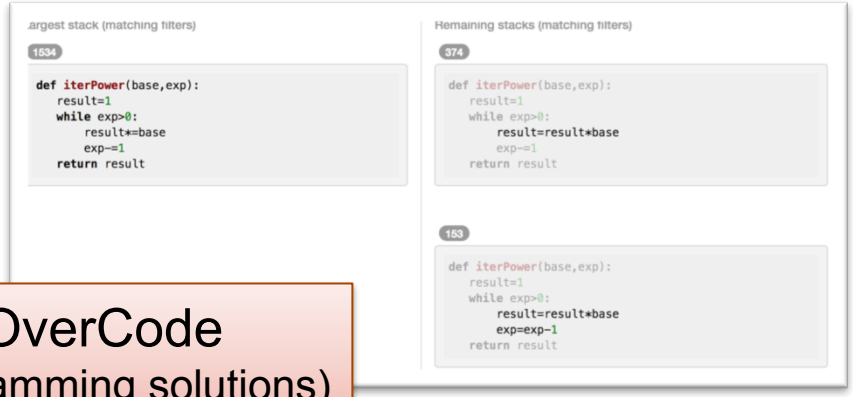
MOOCs Are Big Data for Education

- 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

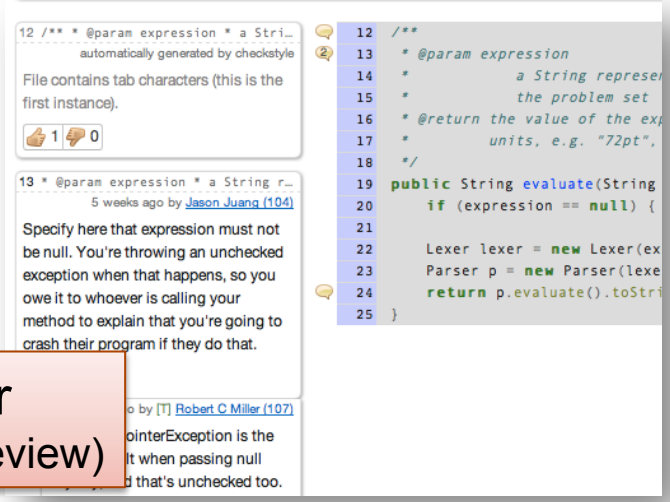
Summary



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

MIT HUMAN-COMPUTER INTERACTION