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
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 to improve an online course
  - what do students get in return? **learning**

- **Active learnersourcing**
  - **asking** learners to do something for us

- **Passive learnersourcing**
  - **watching** what learners are doing to infer improvements
Outline

- How-to videos (active learnersourcing)
  - Lecture video analytics (passive learnersourcing)
  - Solution visualization
  - Online Python tutor
  - Wait-learning
How-to videos are everywhere
Problems in Watching How-To Videos

You can’t tell what happens in the video.

It’s difficult to navigate to specific parts you need.
ToolScape Player

With ToolScape:

Learners feel more confident about their design skills.
- self-efficacy gain

Learners believe they produced better designs.
- self-rating on designs produced

Learners actually produce better designs.
- rating by external judges
How to get annotations?

Keep humans in the loop: learnersourcing
Are “steps” enough?

- Students follow verbatim without thinking
- Limited retention and transfer
Subgoal Labeling supports learning new knowledge by clustering a group of steps into a high-level conceptual unit.

Goal: How to Bake a Cake

Mix dry ingredients
1. Add 2 c. flour
2. Add 1/2 tsp salt
3. Mix ingredients together

Mix wet ingredients
4. Beat 1/2 c. butter in a separate bowl
5. Add 2 eggs
6. Combine ingredients until smooth

subgoals
individual steps
Adding Style Rules Using Google Chrome Developer Tools

Wiki Video Outline

1. Right click link element to inspect and bring up developer tools
2. Click below existing properties of <a> element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
14. Use Developer Tools to change styles dynamically
15. Copy created code in Developer Tools to file to change styles on live site

Periodically throughout this video, you’ll be asked to summarize what you just watched.

alternative YouTube
Adding Style Rules Using Google Chrome Developer Tools

Wiki Video Outline

1. Right click link element to inspect and bring up developer tools
2. Click below existing properties of <a> element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'a:hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
14. Use Developer Tools to change styles dynamically
15. Copy created code in Developer Tools to file to change styles on live site

Periodically throughout this video, you’ll be asked to summarize what you just watched.

individual steps
stage 1
 generation

stage 2
 evaluation

stage 3
 proofreading

Adding Style Rules Using Google Chrome Developer Tools

Wiki Video Outline
1. Right click link element to inspect and bring up developer tools
2. Click below existing properties of \(<a>\) element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'a:hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
14. Use Developer Tools to change styles dynamically
15. Copy created code in Developer Tools to file to change styles on live site

What was the overall goal of the video section you just watched?
e.g., Create event handlers
Note: Other users will see your outline to help them better understand the steps in the video.
stage 1
 generation

stage 2
 evaluation

stage 3
 proofreading

Adding Style Rules Using Google Chrome Developer Tools

Wiki Video Outline

1. Right click link element to inspect and bring up developer tools
2. Click below existing properties of <a> element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'a:hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
14. Use Developer Tools to change styles dynamically
15. Copy created code in Developer Tools to file to change styles on live site

Which of the following best describes the video section you just watched?

Choose the best answer (submitted by other users) or add your own.

- set up developer tools
- begin using developer tools
- developer tools
- I have a better answer:

Submit  Cancel
Adding Style Rules Using Google Chrome Developer Tools

Wiki Video Outline

1. Right click link element to inspect and bring up developer tools
2. Click below existing properties of \(<a>\) element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'a:hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
14. Use Developer Tools to change styles dynamically
15. Copy created code in Developer Tools to file to change styles on live site

Does the below statement (submitted by other users) accurately summarize the steps?

Statement:
Set up developer tools

Steps:
1. Right click link element to inspect and bring up developer tools

- Yes, this statement applies
- No, these steps don't require summarization
- No, and I want to revise the statement:
  
  Set up developer tools

Submit  Cancel
Pilot Deployment

MIT UI design class
~280 students
21 videos
Pilot Deployment

76 students/video, σ=39
20 subgoals/video, σ=16
6:18 min./video, σ=2:49

learner-generated subgoals
Future Work

Smarter interval choice based on audio and/or content

Large-scale deployment and evaluation of learner-generated subgoals

Study of pedagogical benefit of learnersourcing tasks
LECTURE VIDEOS
Motivation: Online lecture videos are widespread, but standard video players are not optimized for learning.
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

Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs

Admissible (pessimistic) heuristics break optimality by trapping good plans on the fringe

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

peak

visual transition
Example: Backing up

before transition

```
def fac(n):
  # assumes that n is an int > 0
  res = 1
  while n > 1:
    res *= n
    n -= 1
  return res
```

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

after transition

peak

visual transition
LectureScape: Enhancing lecture videos
SOLUTION MINING
Motivation: It’s hard for CS instructors in large classes and MOOCs to give meaningful feedback on coding style.
**OverCode**: Visualize student code variation

```
def iterPower(base, exp):
    result = 1
    while exp > 0:
        result *= base
        exp -= 1
    return result
```
ONLINE PYTHON TUTOR
Online Python Tutor

Motivation: Execution diagrams are essential for learning programming, but manual drawings are messy and tedious.
Online Python Tutor

1. \( x = [1, 2, 3] \)
2. \( y = [4, 5, 6] \)
3. \( z = y \)
4. \( y = x \)
5. \( x = z \)

Help future learners by describing what you just learned.

Start a shared session

I just learned that

Submit

Global frame

\( x \)
\( y \)
\( z \)

Frame

Objects

list

1
2
3

list

4
5
6

To share this visualization, click the ‘Generate URL’ button above and share that URL. To report a bug, paste the URL along with a brief error description in an email addressed to philip@pgbovine.net
Online Python Tutor

Real-time shared sessions for remote tutoring and collaborative learning
Online Python Tutor

www.pythontutor.com

500,000+ users from 165+ countries, popular in MOOCs
WAIT LEARNING
Motivation

Learning a second language requires **significant time and effort** to practice on a recurring basis.

The **busyness of daily life** makes it difficult for those with a casual interest in learning to set aside time for regular practice.
Micro-Waiting

browser loading

check-out line

video ad

escalator
Wait-Learning

Use “wait time” during IM conversations to help people learn vocabulary

While I’m waiting for Carrie to reply...

...I get a language exercise!
Rationales Guiding System Design

Encoding Specificity & Situated Cognition: if we learn language in its context of use, it should help us recall it when/where we need to use it.

Spaced Repetition: learning is best when content is spaced over time rather than in quick succession.

Feedback: learning is more effective when feedback is delivered frequently and close in time to learner action.
WaitChatter Modes

Study Mode (first exposure)

Quiz Mode (subsequent exposures)
Difficulty Levels

Correctly answered cards

1 2 3 4 5

Incorrectly answered cards

Spaced Repetition

Easier
(translate to English)

Harder
(translate to Spanish)
Conversational Relevance

Non-Contextual

Contextual

Yellow background & italic text signal that this word is taken from the conversation
Tracking Knowledge and Progress
User Study / Deployment

- 20 beginner/intermediate learners (age 19-33) used WaitChatter for 14 days on their personal computers inside GChat
- 46% of vocabulary presented was contextual
- on average, learners were exposed to 88 new vocabulary words that they didn’t already know
- post-study evaluation quiz:
  1) production quiz (translate to Spanish/French)
     57 words correct
  2) recognition quiz (translate to English)
     77 words correct
Preliminary Results

- Users engage with a learning exercise faster if it is presented:
  - shortly after he sends a chat message, or
  - as he is typing the last few characters of the message

- Because users tend to multi-task while awaiting a response, grabbing their attention **before** they context-switch is key

- Even when exercises appeared during inopportune times, users reported that they did not feel interrupted because the exercise does not occlude IM-ing, and fades away after several seconds, so they always had the option to ignore it.
Learner Perception of Time
“The key thing is is that I didn't feel like I was taking extra time out of my day to dedicate to learning vocabulary words”
“With Duolingo you definitely have to take your time to devote to it. I think it requires your main focus, you have to actively seek and go and learn it. It feels like you actually have to set aside time to use duolingo, but it doesn't feel like it’s that way with [WaitChatter].”

Learner Perception of Effort
“To go on my phone and open Duolingo is more work for me than if I already have gmail open and I’m always chatting with people anyway.”
“I think it being so close to the chatbox made me do it more, cuz it was a separate thing it wouldn’t be as easy to do some words while I’m waiting for a response.”
Preliminary Results

Learner Perception of Structure vs. Pressure

WaitChatter Disadvantages:
“A disadvantage is that we could do a lot of stuff in class where we would memorize a blob of thematically related vocabulary so that way you could give it a little bit of context in your head, in [WaitChatter] there was no real connection between the words.”

WaitChatter Advantages:
“This is more entertaining and almost more low-key. I would just type it in, low pressure, just a simple thing.”
“With other apps there are modules, so on the one hand its very structured, but it also feels like you know what’s ahead of you, so it also feels like more work -- there another module next and next, whereas with [WaitChatter] there are no modules.”
“With other apps i tend to have this mental thing that oh i should do it for 5 minutes straight”
Domain
- How-to videos
- MOOC videos
- Programming assignments
- Second-language learning

Theory
- Role of interactivity
- Learner control
- Subgoal labeling
- Situated cognition
- Spaced repetition

Method
- Crowdsourcing
- Learnersourcing
- Visualization
- Collaborative systems
Vision in learnersourcing

• Feedback loop between
  – Learners: natural, pedagogically useful activities
  – System: improve interaction using learner data

• Visualize and analyze large-scale learning activities
• Use data to inform learning platform design
Conclusion

Toolscape
(how-to videos)

OverCode
(programming solutions)

Lecturescape
(MOOC videos)

Online Python Tutor
(programming help)

WaitChatter
(language learning)