

Let's Read!

Junsoo Park

20164320

Youngbo Shim

20164350

Sanggyun Ahn

20164352

What have you been doing the most
for this semester?

Schedule

| Week | Date | Topic | Reading (response indicates a reading response is required for the article.) | Due |
|------|------|---|---|---------------------------------|
| 1 | 9/1 | Introduction & Course Info (PDF) | | |
| 1 | 9/6 | Introduction to crowdsourcing and human computation (PDF) Discussion by Juho (PDF) | (1) Howe, Jeff. "The rise of crowdsourcing." Wired magazine 14.6 (2006): 1-4. (2) Quinn, Alexander J., and Bederson, Benjamin B. "Human computation: a survey and taxonomy of a growing field." CHI 2011. | |
| 2 | 9/8 | Crowdsourcing platforms (PDF) Discussion by Oisen (PDF) | (1) response Ipeirotis, Panagiotis G. "Analyzing the amazon mechanical turk marketplace." XRDS: Crossroads 17.2 (2010): 16-21. (2) response Geiger, David, et al. "Managing the Crowd: Towards a Taxonomy of Crowdsourcing Processes." AMCIS. 2011. (3) Vakharia, Donna, and Matthew Lease. "Beyond AMT: An analysis of crowd work platforms." arXiv preprint arXiv:1310.1672 (2013). | |
| 2 | 9/13 | Worker Issues in Crowdsourcing (PDF) Discussion by Sang-gyun (PDF) | (1) response Irani, Lilly C., and M. Silberman. "Turkooption: interrupting worker invisibility in amazon mechanical turk." CHI 2013. (2) response Martin, David, et al. "Being a turker." CSCW 2014. (3) Bigham, Jeffrey P. "My MTurk (half) Workday." (4) Gray, Mary L., et al. "The crowd is a collaborative network." CSCW 2016. | Assignment 1: Be a crowd worker |

Reading Response

You'll **READ** and **CRITIQUE** influential research papers and articles in crowdsourcing.

Motivation



Problem Statement

Novice researchers face **difficulties**
in reading scientific papers of *unfamiliar fields*
with a *good level of understanding*
in a *short period of time*.

Tasks we want to support

- Task1: Grasping the trend of conferences
- Task2: Writing critique of a paper
- Task3: Reimplementing the paper



Depth of
understanding

Task 1 : Grasping the trend of conference from dozens of papers

- Sean is a 25-year-old **graduate student** who has newly joined KIXLAB.



CSCW subject areas

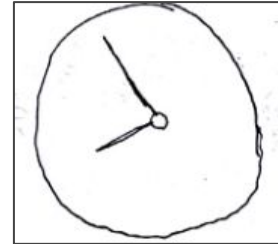
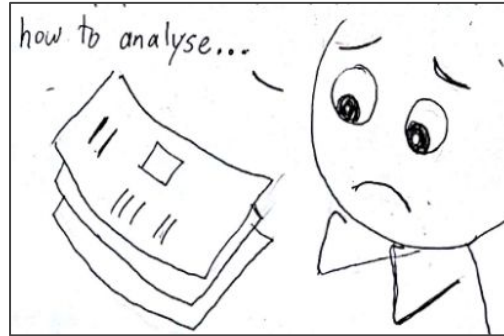
Collaborative and social computing design and evaluation methods
Collaborative and social computing theory, concepts and paradigms Collaborative and social computing Computer supported cooperative work Computing / technology policy Computing and business Enterprise information systems Human computer interaction Human-centered computing Information systems Internet communications tools Social and professional topics Synchronous editors Web conferencing Web-based interaction

Task 1 : Grasping the trend of conference from dozens of papers

- Sean has **little background knowledge** of crowdsourcing discipline. Thus, while reading each paper, he feels **hard to understand the concept** of paper or even misunderstands it. To avoid it, he should read the paper in detail, which is very **time consuming**.
- Thus, through our solution,
 - i. he could get **quick overview** of the paper.
 - ii. **skimming comes easy** with the service, reducing Sean's reading time.
 - iii. **summary** of the paper **in several dimensions** is provided.

Task 2 : Writing critique of a paper

- Junsoo a 25-year-old **graduate student** and has little experience with paper reading.
- Professor gives a **paper critique assignment**, which will help my research capability.
- As the deadline approaches, he doesn't have enough time for a good critical analysis.



Task 2 : Writing critique of a paper

- He goes to our solution and sees that the paper is analyzed by others.
- Through the solution he can:
 - i. obtain a solid **list of strengths and weaknesses** of this paper,
 - ii. as a result **understand the content** of the paper in such depth that he can perform his own critical analysis,
 - iii. achieve that in less than 30 minutes.



Task 3: Reimplementing the paper

- Sam, 23-year old **first year graduate student**
- Find out a **key paper** about crowdsourcing application. After scanning the contents, he think it would be a good starting point to construct his own experiment so try to **re-implement** the paper's experiment program.

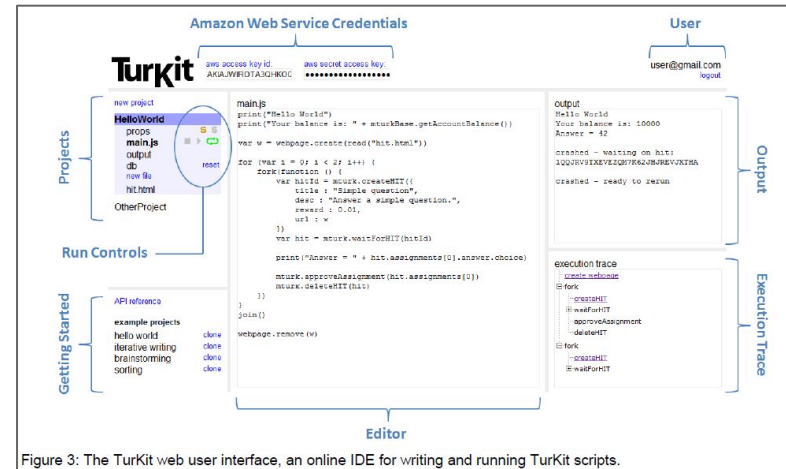
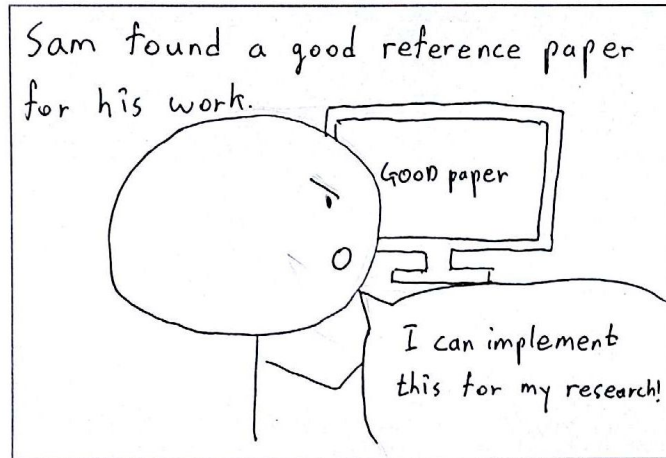


Figure 3: The TurKit web user interface, an online IDE for writing and running TurKit scripts.

Task 3: Reimplementing the paper

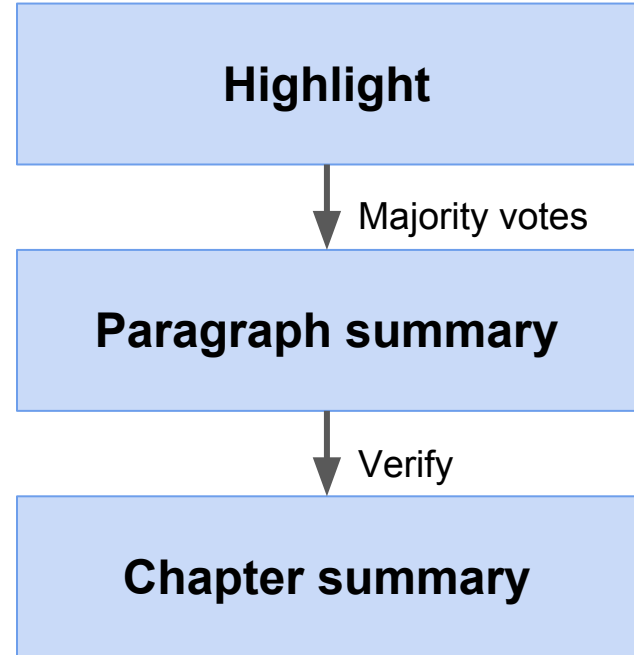
- The paper shows only a bare pseudocode and a sample class. There is **no specific implementation details** such as web circumstances, or meanings of all program functions. He **has lots of questions**, but can't ask mentor or senior students all the time when he stuck.
- Using our solution, Sam can
 - a. Find **directed annotations** of a pseudocode which leads to actual explanation in the paper
 - b. Watch **other users' coding trials** in various programming platform
 - c. Read **step-by-step coding guide** written and verified by other readers.

As a result, Sam easily understand the program structure and complete his task.

Solution1: Co-summary

Do summary tasks while reading the paper

Target crowd: Lab members



Solution1

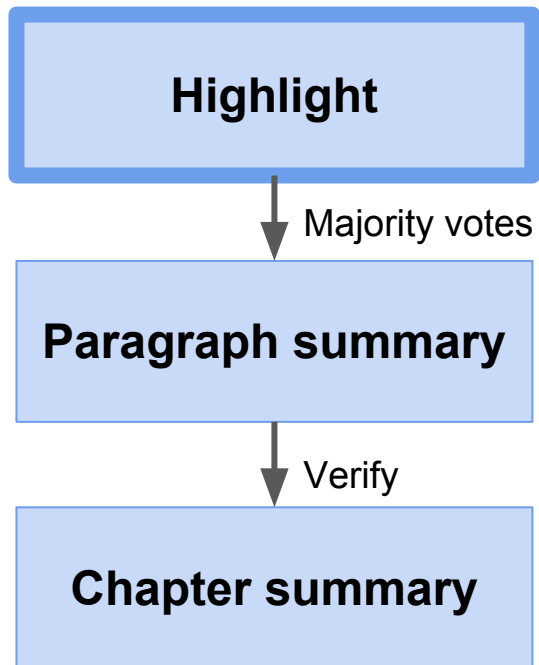
Start reading

The Human Macro: Natural Language Crowd Scripting

Embedding crowd workers in an interface allows us to reconsider designs for short end-user programming tasks. Typically, users need to translate their intentions into algorithmic thinking explicitly via a scripting language or implicitly through learned activity [6]. But tasks conveyed to humans can be written in a much more natural way. While natural language command interfaces continue to struggle with unconstrained input over a large search space, humans are good at understanding written instructions.

The Human Macro is Soylent's natural language command interface. Soylent users can use it to request arbitrary work quickly in human language. Launching the Human Macro opens a request form (Figure 3). The design challenge here is to ensure that the user creates tasks that are scoped correctly for a Mechanical Turk worker. We wish to prevent the user from spending money on a buggy command.

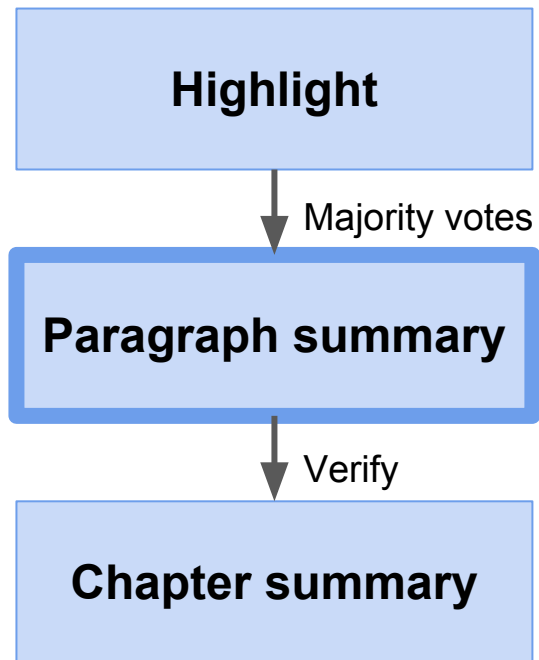
Solution1



The Human Macro: Natural Language Crowd Scripting
Embedding crowd workers in an interface allows us to reconsider designs for short end-user programming tasks. Typically, users need to translate their intentions into algorithmic thinking explicitly via a scripting language or implicitly through learned activity [6]. But tasks conveyed to humans can be written in a much more natural way. While natural language command interfaces continue to struggle with unconstrained input over a large search space, humans are good at understanding written instructions.

The Human Macro is Soylent's natural language command interface. Soylent users can use it to request arbitrary work quickly in human language. Launching the Human Macro opens a request form (Figure 3). The design challenge here is to ensure that the user creates tasks that are scoped correctly for a Mechanical Turk worker. We wish to prevent the user from spending money on a buggy command.

Solution1



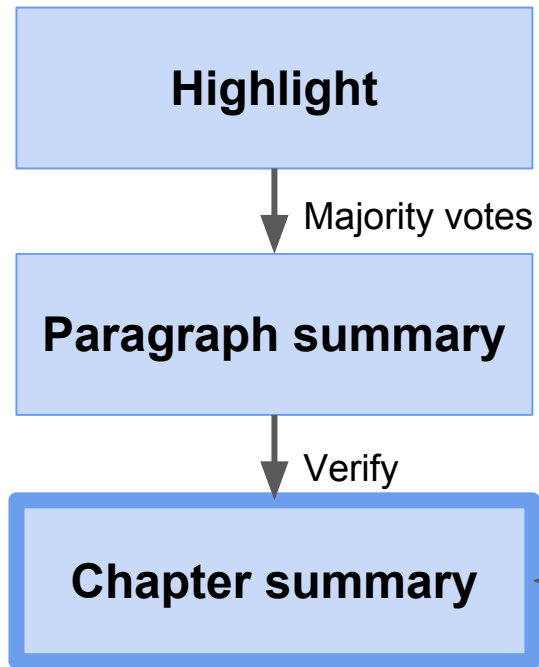
A

The Human Macro: Natural Language Crowd Scripting
Embedding crowd workers in an interface allows us to reconsider designs for short end-user programming tasks. Typically, users need to translate their intentions into algorithmic thinking explicitly via a scripting language or implicitly through learned activity [6]. But tasks conveyed to humans can be written in a much more natural way. While natural language command interfaces continue to struggle with unconstrained input over a large search space, humans are good at understanding written instructions.

B

The Human Macro is Soylent's natural language command interface. Soylent users can use it to request arbitrary work quickly in human language. Launching the Human Macro opens a request form (Figure 3). The design challenge here is to ensure that the user creates tasks that are scoped correctly for a Mechanical Turk worker. We wish to prevent the user from spending money on a buggy command.

Solution1



The Human Macro: Natural Language Crowd Scripting

Embedding crowd workers in an interface allows us to reconsider designs for short end-user programming tasks. Typically, users need to translate their intentions into algorithmic thinking explicitly via a scripting language or implicitly through learned activity [6]. But tasks conveyed to humans can be written in a much more natural way. While natural language command interfaces continue to struggle with unconstrained input over a large search space, humans are good at understanding written instructions.

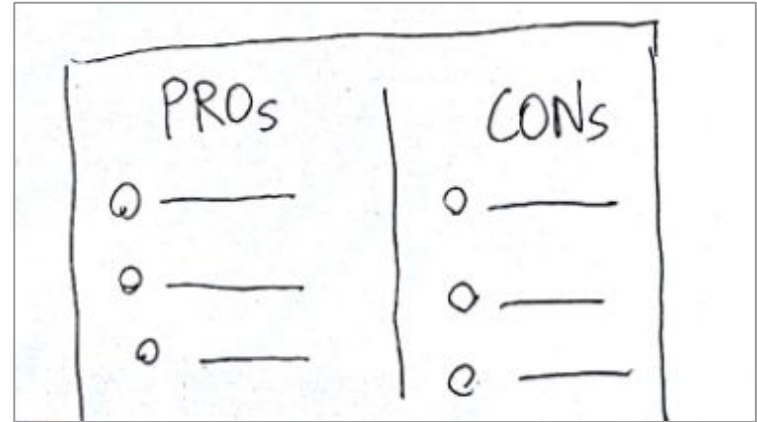
The Human Macro is Soylent's natural language command interface. Soylent users can use it to request arbitrary work quickly in human language. Launching the Human Macro opens a request form (Figure 3). The design challenge here is to ensure that the user creates tasks that are scoped correctly for a Mechanical Turk worker. We wish to prevent the user from spending money on a buggy command.

Solution2: Pros & Cons

Produce a solid list of strengths and weaknesses of the paper

Target crowd: class, or study groups

The screenshot shows a PDF viewer window titled 'JTE324554.qxd - Okular'. The main content is a page from a research paper, 'Figure 5 Domains of Mathematical Knowledge for Teaching'. The diagram shows two overlapping circles: 'SUBJECT MATTER KNOWLEDGE' on the left and 'PEDAGOGICAL CONTENT KNOWLEDGE' on the right. The intersection is labeled 'Pedagogical Content Knowledge (PCK)'. The left circle also includes 'Content Knowledge (CK)' and 'Pedagogical Knowledge (PK)'. The right circle includes 'Pedagogical Knowledge (PK)' and 'Content Knowledge (CK)'. The text on the page is partially highlighted in orange and blue. The viewer interface includes a menu bar (File, Edit, View, Go, Bookmarks, Tools, Settings, Help), a toolbar with navigation and zoom controls, and a sidebar with thumbnails and bookmarks. The status bar at the bottom shows '16 of 20' pages and a resolution of '8.50393 x 11 in'.



Solution2: Pros & Cons



Detecting precise timing of a step. We observed that Turkers add new steps with higher latency than trained annotators, resulting in Turker-labeled time points being slightly later than those by annotators for the same step. **The trained annotators often rewinded a few seconds to mark the exact timing of a step after seeing the step, whereas most Turkers completed their tasks in a single pass.** While this might be a limitation of the workflow, our results show that a reasonable window of differences. We will explore time-shifting if timing accuracy improves.

The trained annotators often rewinded a few seconds to mark the exact timing of a step after seeing the step, whereas most Turkers completed their tasks in a single pass.

Any additional limits?

A: This didn't consider this point.

Which analyses are invalid?

A1: This didn't consider this point.



A2: This is stupid.



A3: This assumes that this is the case.

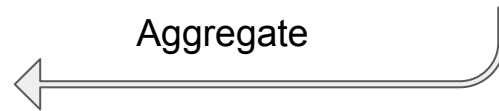


Limitation

Final output:

| PROs | CONs |
|---------|---------|
| ○ _____ | ○ _____ |
| ○ _____ | ○ _____ |
| ○ _____ | ○ _____ |

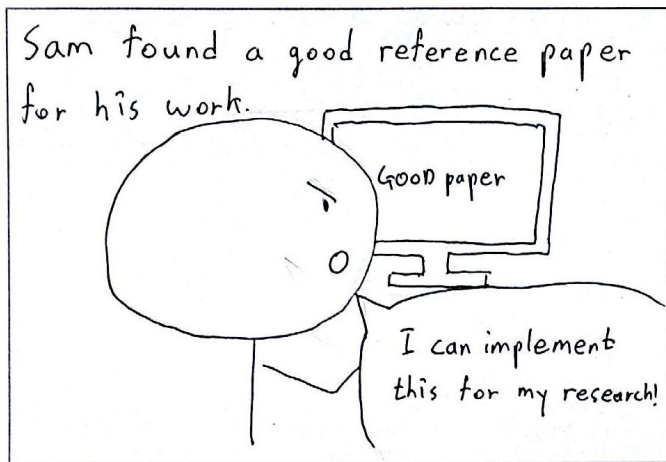
Aggregate



Solution3: Micro-stackoverflow

In-paper Q&A service

Crowd target: global



stackoverflow

Questions Jobs Documentation ^{Beta}

Crowdsourced Translation

Get personalized job matches now

stackoverflow **JOBS**

Get started

▲ I once saw a free translation management software that was used to publish the original software texts and allow translation to other languages in a collaborative way. Something similar to <http://www.mtranslator.com/> but also allowed community driven additional translations.

5

▼ I just cannot seem to find it again so I was wondering if anyone knows of it? Cheers

★ internationalization translation community-translations

6 share improve this question

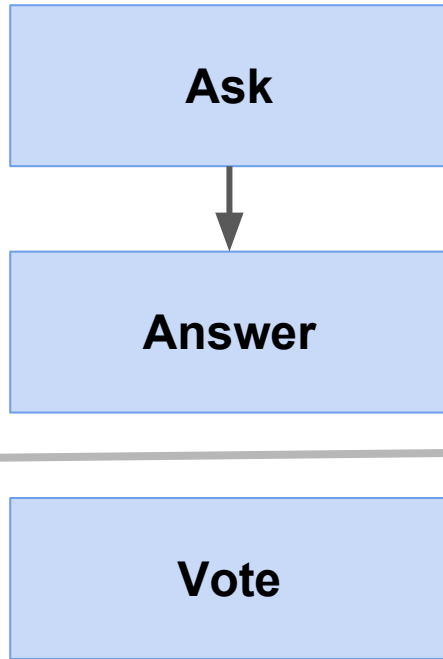
edited Sep 21 '10 at 11:40

sorin 48.2k ● 73 ● 246 ● 405

asked Dec 8 '09 at 12:29

Rok 527 ● 1 ● 10 ● 27

Solution3: Micro-stackoverflow



The screenshot displays the TurKit web user interface, which is an online IDE for writing and running TurKit scripts. The interface is organized into several sections:

- Amazon Web Service Credentials:** Fields for AWS access key ID and AWS secret access key.
- Projects:** A list of projects including "HelloWorld", "prog", "main.js", "output", "db", "new file", "ht/html", and "OtherProject".
- Run Controls:** Buttons for "reset" and "run".
- Getting Started:** A list of links for "API reference", "example projects", "hello world", "iterative writing", "brainstorming", and "sorting".
- Editor:** A code editor showing JavaScript code for a "Hello World" application that interacts with an AWS account.
- Execution Trace:** A section showing the execution flow of the code, including "fork", "waitForHT", "approveAssignment", and "deleteHT".

A callout box on the right side of the screenshot shows a Stack Overflow question titled "Crowdsourced Translation". The question asks for help finding a free translation management software that allows community-driven translations. The question has 5 answers and 6 votes.

Figure 3: The TurKit web user interface, an online IDE for writing and running TurKit scripts.

Compounded solution

Co-summary

Micro-stackoverflow

Pros & Cons

- Paper augmented platform
- Readers = Requester+Worker
- Support interactive reading
- Aims to benefit readers (workers) at any point of task

Plan

- Main roles
 - Project Manager, Programmer - Junsoo
 - Platform Design, Programmer - Youngbo
 - Task Design, UI/UX Designer - Sanggyun

- Crowd gathering plan?
 - test it with **reading response** assignment in class
 - host a public website to gather volunteers from various **labs and study groups**

Thank you for listening...

Q&A