Scopist: Building a Skill Ladder into Crowd Transcription

Jeffrey P. Bigham, Kristin Williams, Nila Banerjee, and John Zimmerman
Human-Computer Interaction Institute
Carnegie Mellon University
{jbigham, kmwillia, johnz}@cs.cmu.edu, {nilanjab}@andrew.cmu.edu

ABSTRACT
Audio transcription is an important task for making content accessible to people who are deaf or hard of hearing. Much of the transcription work is increasingly done by crowd workers, people online who pick up the work as it becomes available often in small bits at a time. Whereas work typically provides a ladder for skill development – a series of opportunities to acquire new skills that lead to advancement – crowd transcription work generally does not. To demonstrate how crowd work might create a skill ladder, we created Scopist, a JavaScript application for learning an efficient text-entry method known as stenotype while doing audio transcription tasks. Scopist facilitates on-the-job learning to prepare crowd workers for remote, real-time captioning by supporting both touch-typing and chording. Real-time captioning is a difficult skill to master but is important for making live events accessible. We conducted 3 crowd studies of Scopist focusing on Scopist's performance and support for learning. We show that Scopist can distinguish touch-typing from stenotyping with 94% accuracy. Our research demonstrates a new way for workers on crowd platforms to align their work and skill development with the accessibility domain while they work.

CCS Concepts
• Human-centered computing → Empirical studies in accessibility; Accessibility design and evaluation methods;

Keywords
Crowdsourcing; text-entry; crowd work; stenography; worker training; novice to expert transition

1. INTRODUCTION
For many people, work offers more than just an opportunity to earn money. It often provides other benefits such as social interaction, the opportunity to develop new skills leading to advancement, or even the chance to realize a personal passion. The same cannot be said for crowd work. Crowdwork typically places workers in front of a computer—one of the best devices for personalized learning—yet, crowdwork platforms have not invested in interfaces or systems that help workers advance their skills or realize other value through their work. While crowdwork has recently shown promise for supporting web accessibility [25], it currently depends on a labor ecosystem that neglects skill development and other valuable aspects of work.

Today, most crowdwork consists of low-paying, unskilled microtasks that result from decomposing complex jobs into subtasks to be worked on incrementally. Microtasks are typically easy for a person to do, but difficult for a computer [26]. One common example is audio transcription; the result of decomposing an audio file, like a recording or video of a speech, into a collection of short utterances that each get posted as a HIT (Human Intelligence Task) online. A recent survey reported audio transcription, a vital task in making audio content accessible, as the second most popular task on Amazon Mechanical Turk, comprising 26% of all microtasks [30]. Workers typically earn $0.01-0.02 USD per sentence they transcribe [28].

Yet unlike asynchronous audio transcription, real-time audio transcription is a highly valued and specialized skill that pays well; up to $300 per hour. Transcribers that work at the pace of human speech are needed to create written records of court proceedings (court reporters) and to close captioning live television, such as news, sports, and political speeches [12]. However, touch-typing skills are not sufficient for real-time captioning as it typically takes at least twice as long as the audio playback time (compare average speaking rates of 200 WPM [37] with good typing rates of 90 WPM [7]). In contrast, real-time transcription uses a chording method for text-entry called stenotype: a technique depressing several keys simultaneously to input whole words at a time. Using stenotype, well trained stenographers can achieve entry rates between 200-300 WPM, closing the gap between transcription time and audio time [29].

We investigate how crowdlabor platforms could support workers in learning a valuable new skill while they work. As a proof-of-concept of a skill ladder for crowdwork, we developed Scopist. Using Scopist, crowdworkers learn one new chord during each microtask. Scopist provides prompts for chords relevant to the microtask and accepts both touch-typing and stenotyping as valid inputs. This approach allows workers to gradually incorporate new chords into transcription without abandoning their familiar work practice.

We performed three formative evaluations of Scopist with crowd workers to demonstrate our approach. The first examines the ability of Scopist to distinguish between chording and touch-typing text-entry. The second and third examine whether crowdworkers can effectively incorporate a new chord into transcription microtasks. We found that Scopist could distinguish

W4A 2017, April 02-04, 2017, Perth, Western Australia, Australia
 ISBN 978-1-4503-4900-0/17/04 $15.00
DOI: http://dx.doi.org/10.1145/3058555.3058562

touch-typing from stenotype 94% of the time; a number that would likely improve with a more advanced language model. In our second and third studies, we found that using a new chord without interface support slowed workers. We then used these findings to design a new user interface which could support workers in learning the hand postures and aural mapping of stenotype.

Our work contributes a proof-of-concept application demonstrating how to extend crowdlabor platforms to support workers in gaining new skills supportive of accessibility. Further, we extend prior research reframing crowdlabor platforms as sites for redesigning work, and reflect on other opportunities for crowdlabor platforms to provide value to their workers.

2. RELATED WORK

Human-computer interaction (HCI) has radically reshaped crowdsourcing platforms. What began as a way for scientists to quickly accomplish large tasks (*e.g.*, protein folding or topological features of Mars) has become a living lab for social science research [2]. HCI has transformed how crowdsourcing platforms work by employing them in many stages of technology's design and build process before committing to the designs in commercial systems. This includes rapidly prototyping novel interaction techniques [3], testing user interfaces [20], and investigating how user goals and motivations can be used to solve difficult computing problems [26]. Researchers have turned crowdsourcing platforms into places for work, teamwork, real-time work, and even real-time teamwork [26]. Our work extends this previous research by recasting crowdlabor platforms as a site for designing crowdwork to provide value to workers beyond pay. We synthesize the related literature below on crowdlabor and stenography to inform our research.

2.1 On-the-job Learning for Real-time Captioning

Recently, researchers began examining how to design microtasks to support crowdworker learning. Related studies look at how workers can learn from examples [11], structure self and peer critique [40], use interactive tutorials [10], and treat the microtask as an internship [34]. However, these cases select the skills to be learned based on researcher interest, and they do not show how learning produces immediate or lasting value for the crowd worker, or even allow crowdworkers to make more effective use of the platform.

A few studies have used crowdsourcing to examine the platform itself to test input methods. These include researching Fitts Law [4, 18], bubble cursors [23], and even adaptive menus or interfaces [18, 23]. Yet again, these studies look at input techniques of interest to the researcher and do not look at the platform's support for skill development as we do here.

2.2 Worker Motivations

As both developer and requester, designers of crowdsourced captioning technologies risk cementing inequalities into the crowdlabor platform by not making sufficient functionality and information available to workers to make reasonable decisions about what skills would be most profitable to acquire [16]. To mitigate this risk, researchers began investigating the exchange model of crowdlabor platforms. Game theory has been proposed as a way to drive design decisions so that incentive properties of the platform are explicitly considered [19]. Strategies to align incentives so that they are intrinsic to the task include entertainment, altruism, and sharing information [26]. Yet, these techniques do not work for certain kinds of tasks in which demand outstrips supply [21], or when the tasks are inherently tedious and benefits accrue only to the requester [18]. That is, some tasks are susceptible to persistent asymmetries in exchange, and incentives for cooperative gain may not be intrinsic to the task. For example, a prior study found that novice audio captioners could collectively outperform professional real-time captioning speeds [25], but only the requester benefited from this task design. Similarly, Netflix depended on the altruism of volunteers from the Amara platform to bring their films in compliance with the American with Disabilities Act [36, 38]. Using crowdsourcing, requesters gain real-time captions at a fraction of the cost, but crowdworkers and volunteers simply gained more microtasks to complete.

Developing platform support for advancing skills in a domain like real-time captioning could connect workers to desirable, or at least, valuable work. Researchers report that crowdworkers engage crowdlabor platforms for many reasons beyond pay: to have greater self-direction and control over their schedules, to mentally challenge themselves, to socialize, and to work from home [5, 24, 41]. Similarly, real-time captionists identify these reasons as perks of their work [12]. Further, they describe the intrinsic value that comes from participating in certain professional communities, like being part of the legal profession, or providing needed services to others [12]. We created Scopist to support crowdworkers in preparing for real-time captioning to begin aligning their skill development with the professional communities they wish to be a part of.

2.3 Novice to Expert Transition

Any work domain must support novices entering the field, and stenography can serve as an instructive case for designing platforms to support workers developing expertise. Low/unskilled tasks may be the best entry level for individuals who have never worked in the domain before or who come from backgrounds that did not offer an opportunity to learn the demanded set of skills [24]. The stenography profession balances creating entry-level jobs with retaining claims to a wage supportive of seasoned workers [12, 33]. To support professional advancement, the profession evolved to provide jobs for a range of skill levels from offline audio transcription to real-time captioning of the nightly news [12]. We focus on stenography here because the profession supports such transitions between novice and expert captioning.

Expert stenographers develop complex associations between rhythm and motor planning while freeing up visual attention during text-entry. Chording provides efficiency gains over touch-typing by reducing the amount of time to search and reach intended keys [1, 15]. These gains are not direct since touch-typing does this at a sophisticated level: one finger is often down even as the next initializes the second keypress trajectory [1]. This touch-typing pattern motivated early designs of the qwerty layout [1, 15] and has led stenotypists to invest in specially designed keyboards–known as n-key roll-over keyboards–to address these limitations of some keyboard matrices. Typists' highly integrated typing patterns introduce key press detection challenges to a user interface supporting both touch-typing and stenotype, and so require careful design.

2.4 Stenotype and Text-Entry Studies

Shorthand languages, such as stenotype, evolved over several centuries and admit many spelling quirks. This introduces several challenges to technological support. Stenotype enables the typist to chunk the speech stream into candidate words using chorded text entry [14] and keeps pace with spoken language by supporting multiple key presses—or chords—to encode phonetic

syllables or abbreviated words and phrases [27]. Stenotype descends from a long history of shorthand methods going back as early as Mesopotamia [9], and as a result, adopts abbreviations which join consonants phonetically, drop vowels, and admit many homophones [9, 27]. Thus, stenotype is not completely defined, and typists are encouraged to use their own abbreviations [27]. This introduces problems such as identifying word boundaries and disambiguating homophones when translating stenotype to English [1, 27, 32]. It also places a significant burden on the learner to acquire the skill [1, 29].

Word frequency impacts the ability of the learner to acquire new chords. Shorthand for more frequent words are easier to learn [31] and even impacts touch-screen based shorthand [39]. An early system, called Rapid-type took advantage of frequency effects to modify a standard typewriter to facilitate learning chords for the most frequent words of the English language with positive results [29]. Rapid-type did not investigate this ability within the context of audio transcription nor generalization to crowd settings. We follow Rapid-type in supporting the most frequently used vocabulary, but our work differs by investigating the learning process on an at-home computer mediated by a crowdlabor platform.

3. THE DESIGN OF SCOPIST

The goal of Scopist is to teach workers stenotype while they perform an audio transcription task. Scopist's name is inspired by the professionals who historically translated the stenographer's shorthand to English. We created Scopist to facilitate learning stenotype rapidly enough to almost immediately benefit from the effort. To support this approach, Scopist accepts both chorded and touchtyping input. However, because stenotype is not a completely defined language, Scopist must first be able to disambiguate word boundaries and homophones arising in the resulting text-entry. In this section, we describe the Scopist algorithm for doing so.

3.1 Scopist's Algorithm

Scopist is an in-browser JavaScript application running on the Express framework for Node.js. So that Scopist could be easily incorporated into a typical transcription HIT, it is designed to intercept the web browser's key up and key down events as part of an Amazon Mechanical Turk webform. When the crowdworker types, Scopist saves keyboard events and their properties for the algorithm to use when translating pressed keys into English. Scopist tracks the entire set of key events because translation depends on the keys' sequence. We describe below how Scopist processes these keys to distinguish touch-typing from stenotype.

Suppose the crowdworker enters a set of key events, $K=k_1, k_2, k_3,\ldots k_i$ where i is the number of key events. Scopist processes K to determine which keys are from touch-typing and which keys are from stenotyping using properties of the events, $p(k_i)$, like their char, keycode, etc. For any key event, k_i, its' properties could indicate it is a key down or key up, so that $p(k_i) = d_1, d_2, u_1, u_2, d_3, u_3, d_4, u_4 \ldots d_n, u_m$ where d are down events and u are up events. Scopist uses these properties (up and down) and more to determine whether keys that are down at the same time are a chord (*e.g.* WO for would, Table 1) or qwerty (d_1 is a shift key indicating that d_2 should be capitalized as in 'Scopist'). The number of up events is always less than or equal to the number of down events, $m \leq n$ and $m+n=i$, where i is the number of key events, and n is the number of keys pressed. Scopist uses this feature to make inferences using both forward and backward passes through the keyset.

Scopist processes K in several passes and assigns each key, $f(k_i)$, to qwerty, stenotype, or unknown based off what is learned from the event properties at that stage of processing. In the first pass processing, all keys that are neither a member of the stenotype keyset, S, or the English alphabet and punctuation keys for the qwerty keyset, Q, are removed. Most of these keys are modifier keys, such as *ctrl, alt, tab, etc.* (see lines 5-6).

A second pass goes through the key event sequence and marks keys that are known to be qwerty according to properties of touch-typing (see lines 9-18). These include:

- if the shift key is pressed, then the key is qwerty
- if the key is not included among the keys used in steno
- if the key follows a key marked as qwerty
- if the key marks a word boundary, B_w (*e.g.*, space or punctuation mark)

Algorithm 1: Distinguishing QWERTY from Stenotype

```
1: procedure ProcessKeyEvents(K)
2:     input: array K of n key events
3:     h_highest ← □
4:     for k_i in K do
5:       if p(k_i) not Q and not S then
6:         remove k_i from K
7:     for k_i in K do
8:       if p(k_i) is type d then
9:         if p(k_i) is a shift key then
10:          qwerty sequence = true
11:          f(k_i) ← qwerty
12:        else if p(k_i) is B_w or not S then
13:          f(k_i) ← qwerty
14:          if p(k_i) is not B_w then
15:          qwerty sequence = false
16:        else if qwerty sequence then
17:          f(k_i) ← qwerty
18:          if p(ki) is not B_w then
19:            qwerty sequence = false
20:        else
21:          f(k_i) ← unknown
22:          add k_i to C
23:      else if p(k_i) is type u then
24:        if |C| > minimum chord length then
25:          for c_j in C do
26:            p(c_j) ← k_i
27:            p(k_j) ← C
28:        C ← {}
25:  for k_i in K do
26:    if k_i is type d then
27:    if k_i is a B_w then
28:      k_i ← qwerty
29:        qwerty sequence = true
29:      else if qwerty sequence then
30:        k_i ← qwerty
31:        qwerty sequence = false
32:  for k_i in K do
33:    if p(k_j) ← C and |C| > 0 then
34:      for c_j in C do
35:        if f(k_i) ← qwerty then
36:          for c_1 in C do
```

```
37:          p(c_l) ← remove chord assignment
38: H ← CREATEHYPOTHESES(K, 0, "", [], "")
39: for h_i in H do
40:    score ← H(h_i | D)
41: if score > score_best
42:    score_best ← score
```

As the second pass proceeds, keys that are observed to be down together without an intervening key up event are added to a sequence of possible steno keys, *C* (line 18). When a key up event is observed, that set, *C*, is considered as a possible steno chord (lines 23-28).

A third pass (a backward pass) then marks as qwerty any key preceding a known qwerty key (lines 25-31). For example, space keys are not used in stenotype and instead, are automatically inserted after a chord. So we use this feature to identify a keyset as qwerty when it occurs before a space. A fourth forward pass marks any potential chords as qwerty if any of the preceding passes marked any key in the possible chord as qwerty (32-37).

A final, recursive process generates a set of key translations that are possible candidates for the key sequence. This process considers a new translation at each key, k_i, where it remains unknown whether that key is qwerty or steno. In practice, the key sequences to which this applies tend to be short as it is difficult when typing qwerty to accidentally have more than 2 to 3 keys down at the same time (see line 38 and Algorithm 2). For keys that are qwerty, this process adds the key's qwerty letter, l_i, to a candidate word, w_q, and appends it to the current hypothesis, h_c (see lines 6-9 of Algorithm 2). For keys that have been marked as a possible chord, the keys are given both a stenotype translation, $s(\cdot)$, and a qwerty translation, $q(\cdot)$, (see lines 11-21 of Algorithm 2; and the next section for a description of the stenotype dictionary used).

Algorithm 2: Create Best Guesses for Key Sequence

```
1: procedure CREATEHYPOTHESES(K, i, h_c, W_q, w_q)
2:  input: K of n key events, index i, current hypothesis h_c, the set of keys W_q making up a possible chord, the possible word w_q
3:  for k_i in K do
4:    if k_i is type d
5:      if k_i is B_w then
6:        add l_i to w_q
7:        if |w_q| ≤ 1 then
8:          remove spaces from h_c
9:        add w_q to h_c
10:     reset w_q, W_q
11:     else if p(k_i) ← C
12:     if w_q is not ""
13:       add k_i to W_q
14:       add l_i to w_q
15:     else
16:       q ← q(C)
17:       h_1 ← CreateHypotheses(K, i, h_c + q, W_q, "")
18:       s ← s(C)
19:       if s
20:       h_2 ← CreateHypotheses(K, i, h_c + s, W_q, "")
21:       H ← h_1 and h_2
22:       return H
23:     else if f(k_i) ← unknown or qwerty
24:       add k_i to W_q
25:       add l_i to w_q
26:   add w_q to h_c
27:   return h_c
```

Each of the possible word sequences is then evaluated by a simple language model (lines 40-44 of Algorithm 1). We score each hypothesis, h_i, by counting how many of the words, w_i, in that hypothesis are English words. That is, given a dictionary, $g(\cdot)$, we add 1 for every word, w_i, found in the dictionary, and 0 otherwise. The total count is then divided by the total number of words.

$$H(h_i) = \frac{\sum_{i=0}^{n} g(w_i)}{n}$$

The scoring system distinguishes between stenotype and qwerty by exploiting the fact that the chorded qwerty keys are not often English words; either by themselves, or when combined with the subsequently pressed keys. The algorithm uses $H(h_i)$ to distinguish these cases because the boundary between chords and qwerty keys is not marked in the keyset since the spacebar is not used in stenotype to separate words.

4. EVALUATION OF SCOPIST

To support crowdworkers in developing their transcription skills, Scopist needs to facilitate audio transcription when a worker starts out using very few chords as well as when they develop further proficiency using several chords. To assess whether Scopist could offer this, we examined whether it supports 3 needs: skill transition, chord learning, and speed gains. In this section, we first describe our evaluation of Scopist's accuracy in distinguishing touch-typing from stenotype, as would be needed to support skill transition, before reporting on our study of Scopist's support for crowdworkers' learning and speed gains.

4.1 Study 1: Support for Mixed Skill Sets

Crowdworkers serious about developing their transcription skills will want to challenge themselves to learn more and incorporate new chords into their repertoire. To support this transition, we examined Scopist's ability to process phrases with a mixture of touch-typing and stenotyping.

4.1.1 Method and Analysis

We asked crowdworkers to use a mixture of touch-typing and stenotyping in a text-entry task. Then, we evaluated how well Scopist processed the entered keys by testing it against an English language dictionary. We randomly selected 10-word phrases from a corpus of transcribed TED talks to create phrases for the text-entry task that approximated spoken language [42]. In each phrase, we replaced between 0 and 4 of the words with the equivalent stenotype chord to simulate the number of chords we would expect crowdworkers with varying skill levels—ranging from novices to advanced-intermediate—would use. We only substituted chords with 2 to 3 keys because they are most likely to provide problems for Scopist's algorithm.

We made the microtask as simple as we could to encourage workers to type as quickly as possible so that we could test the algorithm on typing patterns where more than one key is down at a time (common in both moderately fast touch- typing and stenotyping). Since we were examining Scopist's algorithm—not crowdworker learning—we asked crowdworkers to type the requested key pattern from the 10 word phrase (Figure 1) using the qwerty keys for chords instead of the stenotype characters. We

required workers to start again if they made a mistake to ensure the collected keysets were as accurate as possible.

We recruited 150 crowdworkers from Amazon Mechanical Turk. This resulted in 150 keysets. We then processed each keyset through Scopist's algorithm to see how well it was able to reconstruct the correct phrase from the keys. To determine how accurately Scopist processed the keysets, we divided the total number of phrases Scopist processed correctly by the total number of phrases. This number estimates what percentage of mixed typing phrases from the TED corpus Scopist would be able to support.

Figure 1. The webpage shown above collected keysets from crowdworkers containing a mixture of touch-typing and stenotype. The example phrase 'so we need energy miracles now drni use op' shows how we substituted in the qwerty keys corresponding to the needed stenotype chords to collect the test keysets from crowdworkers. In this example, a worker would need to chord the key combinations d-r-n and o-p.

4.1.2 Results

Scopist was able to process 94.0% of the keysets accurately. This means that when Scopist processed the 150 collected keysets, 9 of the resulting phrases contained one or more errors.

4.2 Studies 2 & 3: The Impact of Learning to Chord on Transcription Tasks

Scopist distinguished touch-typing from stenotype with 94% accuracy. Thus, Scopist shows promise for supporting text-entry consisting of both stenotype and touch-typing. To further evaluate whether Scopist could support crowdworkers' learning with immediate benefit, we examined whether stenotype helped crowdworkers complete transcription tasks faster.

4.2.1 Studies 2 & 3 Procedure

We designed an Amazon Mechanical Turk HIT that asked workers to transcribe 30 short excerpts from speech described below. The HIT consisted of two conditions—a baseline, transcription task and a mixed, stenotype task—to allow us to assess and characterize how introducing a new chord into the transcription task impacted performance. Each condition consisted in a task block of 15 transcription microtasks. The order of conditions was counterbalanced across participants, and participants were randomly assigned to an order. We conducted separate studies for a text transcription version in which workers simply typed the sequence shown to them, and an audio transcription version in which workers listened to spoken phrase and transcribed it.

Each transcription task consisted in typing a phrase with the target vocabulary we wanted workers to learn to chord. We chose the target chord 'something' (S–G) because it is a commonly used word, would test predicted time gains from learning (the time to type 10 keys—including the space bar—would reduce to chording 2), and would prove a test case for the Scopist algorithm (the qwerty equivalent keys, "A" and "L", frequently appear together in English and are often typed rapidly in succession since they lie on the home row and require opposite hands to type).

Transcription tasks were selected from a corpus of TED talks used in a prior study and were subdivided as required to train an automatic speech recognition (ASR) model [42]. We used this corpus because it provided transcription text aligned with the audio files and consisted of common vocabulary characteristic of spoken English. Focusing on commonly spoken English allowed us to test microtasks that could introduce chords for the most frequently used vocabulary likely to be encountered in audio transcription tasks. We wrote a Python script using the output of the ASR model to identify 60 phrases which contained the target vocabulary, 'something'. We then chose two subsets of 15 phrases for each condition consisting of 6 speakers (both male and female), and different stages of their speech (the beginning, middle, and end). These phrases were also examined for occurrence of the target word appearing in the beginning, middle, and end.

To evaluate the representativeness of our files' phrase sets for any typical transcription task, we analyzed them using the AnalysePhrases Program from McKenzie and Soukoreff [33]. We did this to ensure that our assessment would generalize to other transcription tasks that did not use our specific speech corpus and ASR model. The minimum phrase length was 85 characters and the maximum 264, with an average phrase length of 160.8 characters (the set recommended by MacKenzie and Soukoreff ranges from 52-586, averaging 195.9 characters). The average word length was 4.31 characters and the correlation with English 0.95 (the Mackenzie and Soukoreff phrase sets average 4.39 and correlate 0.96 with English). We divided the phrases evenly between the two conditions, and the conditions did not differ substantially on the criteria detailed above.

4.2.2 Studies 2 & 3 Participants

We recruited 20 participants through Amazon Mechanical Turk for our text transcription task and 30 participants for our audio transcription task. Participants were compensated $3.00 for participating in the non-audio version of our study and $6.00 for participating in the audio version of our study. This worked out to approximately $10/hour. Since our evaluations depended on crowdworker performance across all tasks, participants were only compensated for completing all of the tasks.

4.2.3 Studies 2 & 3 Apparatus

We created a minimal user interface that allowed us to isolate the inclusion of a new chord from other potential confounds (Figure

2). The interface presented instructions for workers to type what they either saw or heard (depending on whether they were completing text transcription or audio transcription), either using the chord "al" for something, or typing all words using touch-typing. Workers did 15 transcriptions in sequence either using or not using the chord. While there are many ways the microtasks could be ordered and presented, we focused on the simplest case—presenting microtasks in a row—to evaluate how well workers could learn a new chord in this most basic scenario. For each crowdworker, we randomly shuffled the phrase sets in each condition using the Fisher-Yates shuffling algorithm so that each crowdworker progressed through a different phrase ordering and the order of phrases would not impact our performance measures. This allowed us to control for ordering effects.

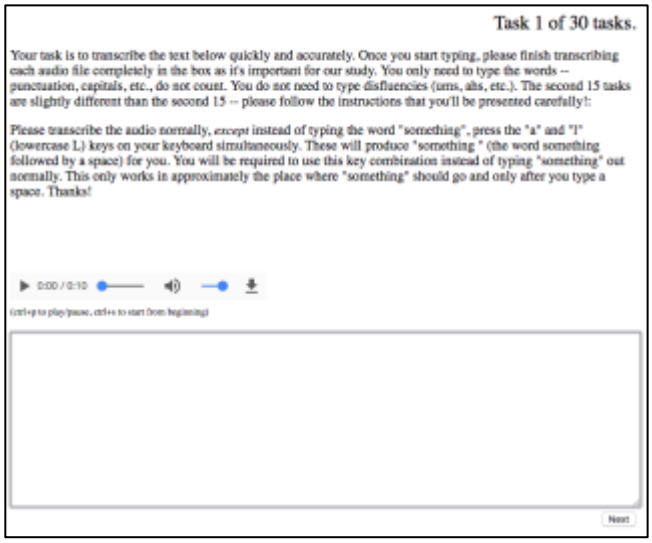

Figure 2: Figure 2 shows the task presented to workers in our second study. Workers received either an audio or text prompt, and were asked to transcribe it. Workers were asked to type 15 phrases in sequence using the prompted chord, and 15 in sequence using touch-typing only.

4.2.4 Studies 2 & 3 Design

We used a within subjects design with task condition (using the chord, or not using) as the independent variable. Dependent variables included time per task, number of times the backspace key was used, and keys pressed per task.

4.2.5 Studies 2 & 3 Analysis and Predictions

To analyze crowdworkers' performance, we used two-tailed paired t-tests to determine whether crowdworkers took less time per task, used a lesser number of keys, or used backspace less. For each of the dependent measures, we removed data points that were more than three standard deviations from the mean for that task.

4.2.6 Studies 2 Results

Overall, our hypothesis that Scopist would immediately benefit workers by supporting them in completing their transcription tasks can be rejected. Crowdworkers generally completed tasks faster, corrected errors less often, and used a lesser amount of keys during the baseline condition. We review these results below.

Time per task. Crowdworkers completed their tasks faster in the baseline condition (*M*=30.0s, *SD*=8.1, *Range*=15.7s-44.1s) than they did in the chording condition (*M*=36.8s, *SD*=9.7s, *Range*=20.7s-51s). A two-tailed, paired t-test revealed this difference to be significant: t_{19}=2.34, p=0.02.

Backspace Usage. Crowdworkers used the backspace key less in the baseline condition (*M*=4.9, *SD*=2.6, *Range*=0.5-11.5) than they did in the chording condition (*M*=7.0, *SD*=3.0, *Range*=1.5-12.2). A two-tailed, paired t-test revealed this difference to be significant: t_{19}=3.71, p<0.025.

Number of Keys Used. Crowdworkers used less keys in the baseline condition (*M*=167.4, *SD*=6.1, *Range*=158-181.6) than they did in the chording condition (*M*=175.2, *SD*=10.9, *Range*=159.4-200.4). A two-tailed, paired t-test revealed this difference to be significant: t_{19}=3.54, p<0.025.

Overall, results indicated that using the chord resulted in slightly worse performance than touch-typing only. This is surprising given that the chording condition requires fewer key presses, and so, should allow workers to transcribe faster. As we will discuss in Section 6, future work should consider how to lower this cost even more and investigate via longitudinal studies when and whether workers would eventually benefit from knowing and using a few stenotype chords.

4.2.7 Study 3 Results

When we examined how well workers did when using Scopist to transcribe an audio file, we found our earlier results further confirmed for audio transcription. Our results suggest that the addition of audio into the task required much more effort from crowdworkers. Crowdworkers generally completed tasks faster, corrected errors less often, and used a lesser amount of keys during the baseline condition for audio transcription. We review these results below.

Time per task. Crowdworkers completed their tasks faster in the baseline condition (*M*=53.4s, *SD*=16.1s, *Range*=26.1-92.2) than they did in the chording condition (*M*=80.0s, *SD*=26.3s, *Range*=44.4-151.4). A two-tailed, paired t-test revealed this difference to be significant: t_{29}=9.27, p<0.025.

Backspace Usage. Crowdworkers used the backspace key less in the baseline condition (*M*=9.2, *SD*=5.3, *Range*=1.7-25.7) than they did in the chording condition (*M*=17.8, *SD*=7.3, *Range*=4.1-33.0). A two-tailed, paired t-test revealed this difference to be significant: t_{29}=7.58, p<0.025.

Number of Keys Used. Crowdworkers used less keys in the baseline condition (*M*=177.4, *SD*=11.5, *Range*=159.5-207.3) than they did in the chording condition (*M*=197.3, *SD*=18.2, *Range*=161.7-231.7). A two-tailed, paired t-test revealed this difference to be significant: t_{29}=6.76, p<0.025.

5. UI Design for Learning Stenotype

Although the Scopist algorithm supports both touch-typing and stenotype to help workers transition to real-time captioning skills, it provides only part of the support workers would need for learning on-the-job. Workers also need guidance on how to use stenotype, such as finger placement to enter the desired word and mapping the sounds of aural speech to chords. To help guide future designs, we began by looking at best practices for learning keyboard use. Cockburn, *et al.* surveyed design approaches for learning keyboard shortcuts and offered four techniques for improving keyboard performance [10]:

1. Enable visibility of keyboard shortcuts without requiring a switch in input modality. Designs often require mousing to

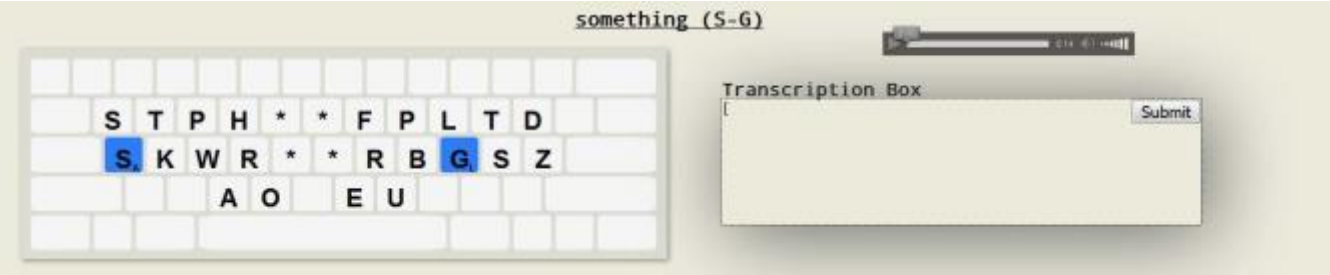

Figure 3. The user interface for Scopist highlights stenotype keys on a typical qwerty layout to teach a crowdworker the appropriate hand posture for chording. The highlighted keys are for the prompted word underlined at the top of the page, and a reminder of the link between the chord keys and the sounds of the prompted word are provided in parentheses.

switch from touch-typing to reveal shortcuts, and this significantly limits performance gains.

2. Support physical rehearsal such that a novice attempts to perform the exact same actions as an expert to develop muscle memory and automaticity.
3. Leverage spatial memory by providing visual cues that reduce visual search time and support intermediate skill level. Too many designs degrade performance, losing time to visual search.
4. Ensure shortcuts are stable (instead of rapidly changing through adaptation) so that they may be rapidly learned.

We incorporate these 4 techniques in our design of Scopist by leveraging crowdworkers' familiarity with the qwerty keyboard. Specifically, we focused on guiding the worker through the mapping principles for using the QWERTY keyboard to learn stenotype. Before describing how we did this, we provide some background on stenotype.

Stenotype keyboards have 22 keys; many fewer than a QWERTY keyboard. Rather than provide a one-to-one mapping for each letter of a word, the stenotype keyboard supports input of the salient sounds in a word, and so, truncates the number of key presses. Stenographers type chords to phonically represent a word: all of the relevant letters are pressed at the same time, making each word a single event instead of one event per letter. Because of this, stenographers gain more speed from long words. For example, the stenotypist would transcribe the word 'something' by pressing down 'S'-'G' at the same time, as a chord.

Sample English-Steno Mapping

English	Stenotype
would	WO
something	S-G
other	OER
where	W*R
there	THR*
through	THRU
considering	KRG

Table 1. The table shows example mappings between English words and their stenotype chords. The dash in the chord 'S-G' indicates that the S is on the left hand side of the keyboard and the 'G' on the right hand (Figure 3).

To provide visual guidance on what keys should be pressed to chord, we employed techniques 1 and 3 from Cockburn, et al. [8]. Since touch-typists are accustomed to focusing on the monitor in front of them and not their finger positions, we provided an onscreen depiction of a qwerty keyboard so that workers could visually orient their hand posture to the positions required for stenotype. We did this by modifying a CSS-drawn keyboard to show which stenotype keys map onto the qwerty layout and highlighted the target keys for the prompted chord with a contrasting color (Figure 3). We labeled the onscreen keyboard with the lettering used for a stenotype keyboard rather than the lettering workers would be accustomed to seeing on a qwerty keyboard to facilitate workers learning the stenotype layout. For the highlighted keys, we provided subscripts of the qwerty-equivalent letter to help workers orient themselves to the new layout.

Our approach further employed techniques 2 and 4 from Cockburn, et al. by keeping a stable mapping between the physical layout and the stenotype keys. This allowed us to support physical rehearsal by training novices on a stenotype vocabulary used by experts, and further helped them learn how stenotype phonetically represents aural speech (see Table 1). To do this, Scopist employs the JSON dictionaries of the Open Steno Project mapping stenochords to their English equivalents and adopting the same stenotype-qwerty layout [35]. While expert stenographers do use one key chords (like a -T for the word 'the'), Scopist only accepts chords with a minimum of two keys. Single keys are primarily used as an advanced technique for stacking chords—entering a sequence of chords to indicate one phrase verses another—and is already supported by other systems (e.g., [35]). We chose to not support stacking so that we can accept the touch-typing of novices transitioning to stenotype without requiring them to switch modes.

6. DISCUSSION

Scopist demonstrated how to extend the crowdlabor platform to support varied levels of skill development for workers learning a new skill while on-the-job. Learning to chord impeded performance during transcription microtasks while workers were experimenting with how to integrate chording into their touch-typing skills. Based on our findings, we designed a new user interface grounded in the literature on learning keyboard shortcuts and connected to the online community of stenographers. We discuss our findings in more detail below.

6.1 Skill Transition

Scopist's algorithm focused on supporting single word entry and left support for stacking chords to expert systems ('stacking' is when a sequence of chords is quickly entered in succession and the order of chords crucially distinguishes between one translation over another). This allows Scopist to provide support for transitioning from novice skill levels to adopting a wider

vocabulary range, but it also limits the algorithm's support for the upper ranges of skill level and disambiguating some cases. Scopist had a high processing accuracy (94.0%), but it was unable to disambiguate cases when a chord's keys serve as the beginning letters of an English word. For example, Scopist misprocessed one keyset as "doing alas simple" instead of the correct string, "doing something as simple". Both strings are composed of English words. It just so happens that 'al' are the qwerty keys for 'something', and so 'al' and 'as' get incorrectly combined together. These remaining errors could likely be corrected with a more sophisticated language model.

6.2 Chord Learning

Our results fit with related studies which find that attending to learning a new skill in the crowdsourcing context imposes a cost and can depend heavily on how the task is introduced in context. This may be mitigated by experimenting with alternative task orders and spacing, although would constitute its own area of study. Prior research has found that the microtask order impacts performance [6], spaced repetition impacts vocabulary acquisition [13], and interleaving tasks of varied difficulty and kind similarly shapes learning [22]. Changes to both the ASR model or the Python script's global search of the audio corpus could be used to both generate and schedule a set of microtasks for more nuanced support of learning.

We extended the microtasking platform to support microtasks crowdworkers were already working on: audio transcription. Yet, how skill development fits within crowdworkers' work practices and goals remains an open question. Our study confirms prior findings from the workplace showing that when a performance dip occurs while learning a new keyboard shortcut [10]. This may be enough to dissuade workers from adopting it even though it is more efficient. The greater longterm gains of learning a skill, like chording, may incentivize adopting new skills where the task category imposes its own significant inefficiencies. However, the way this information is presented to workers may impact worker performance and a skill's perceived value. Surfacing information about the predicted long-term gains of learning and making it visible to the crowdworker may support better-informed choices on whether the time and effort are worth investing or whether predicted gains are relevant to their goals. Based on our findings, we created a user interface that may better facilitate learning and decrease learning costs. Future work should test whether this interface is able to meet these goals.

6.3 Limitations and Future Work

6.3.1 Mapping Qwerty to Stenotype

Scopist offers a crowdworker opportunity to learn stenotype without investing in a specialized keyboard. However, this limits Scopist's current support for developing advanced stenotype skills. Some chords require key combinations that are ergonomically difficult, if not impossible, on a qwerty keyboard. A follow-up study to ours should examine how to generate and schedule tasks for skill development and optimize chord learning for the ergonomically feasible chords. This might be done by developing an alternative set of steno chords. Although, this would break with the community supported vocabulary (see the Open Steno Project [35]). However, personalized vocabulary, known as briefs, are common practice in the stenography profession [12]. Secondly, we did not examine how well Scopist facilitates crowdworkers developing a phonic representation of steno chords. It is possible that using both the qwerty and stenotype keyboard may hinder developing a steno- supported phonetic model. A follow-up study to ours would need to assess how well crowdworkers learn stenotype's mapping of aural speech to keys.

6.3.2 Career Trajectories

We introduced Scopist to support crowdworkers developing their skills while on-the-job in preparation for work of greater complexity and expertise. However, we have not observed workers actually go through the transition from offline transcription to real-time captioning. This is very much a longitudinal question, as traditionally, learning stenography requires 1-3 years. Future work would explore how and whether that transition actually happens and what would be needed of the crowdlabor platform to support it.

In the nearer term, we believe Scopist could be adapted to other domains. What skills might workers acquire by doing other sorts of common crowdsourcing tasks? A promising approach could be to use audio files from a foreign language for developing foreign language listening skills as a twist on the model of Duolingo [26]. Alternatively, platforms could recommend another skillset to obtain a more complex job. For example, crowdworkers who develop skills in audio captioning might also develop skills in image transcription. Images are often tagged in order to provide alternative descriptions for accessibility, and workers might be trained to appreciate more of the context surrounding captioning work (both audio and visual) to gradually become accessibility professionals.

Another path that could be developed would be to support workers in developing their skillsets to create their own emerging job where they would like to work. For example, there is a growing area of research on crowdsourcing creative tasks which may be amenable to captioning films. Creativity tools like web-based typography could support workers in using expressive animation techniques like kinetic typography to begin captioning verbal nuance [17, 38]. Crowdlabor platforms could be extended to help connect audio captioners with filmmakers to create films that are compliant with disability rights laws before hitting the market and offer crowdworkers the opportunity to express creativity in microtasks.

7. CONCLUSION

We presented Scopist to support crowd transcriptionists learning a new skill while on-the-job as a proof-of-concept application. Through Scopist we show that the crowdlabor platform can be modified to facilitate skill development at varying levels of expertise. We showed that Scopist is able to distinguish touch-typing from stenotype with 94% accuracy. Further, Scopist demonstrates how crowdworkers' skill development can be made central to task design so that the crowdlabor platform provides a skill ladder for valuable careers like real-time captioning. Our findings motivate research on supporting crowdworkers to transition to more complex skills and connecting them with valuable career paths.

8. ACKNOWLEDGEMENTS

The contents of this paper were funded by the National Science Foundation and the National Institute on Disability, Independent Living, and Rehabilitation Research. We thank the workers on Amazon Mechanical Turk who participated in our studies.

9. REFERENCES

[1] Beddoes, M. and Hu, Z. 1994. A chord stenograph keyboard: a possible solution to the learning problem in stenography. *IEEE Transcations on Systems, Man and*

Cybernetics. 24, 7 (1994), 953–960.

[2] Benkler, Y. 2006. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press.

[3] Bigham, J.P., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R.C., Miller, R., Tatarowicz, A., White, B., White, S. and Yeh, T. 2010. VizWiz: Nearly Real-Time Answers to Visual Questions. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'10)* (2010), 333–342.

[4] Bigham, J.P., Wobbrock, J.O. and Lasecki, W.S. 2015. Target Acquisition and the Crowd Actor. *Human Computation*. 2, 2 (2015), 135–154.

[5] Brewer, R., Morris, M.R. and Piper, A.M. 2016. "Why would anybody do this?": Understanding Older Adults' Motivations and Challenges in Crowd Work. *Proceedings of the 2016 SIGCHI Conference on Human Factors in Computing Systems (CHI '16)* (2016), 2246–2257.

[6] Cai, C.J., Iqbal, S.T. and Teevan, J. 2016. Chain Reactions: The Impact of Order on Microtask Chains. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'16)* (2016), 3143–3154.

[7] Card, S.K., Moran, T.P. and Newel, A. 1980. The Keystroke-Level Model for User Performace Time with Interactive Systems. *Communications of the ACM*.

[8] Cockburn, A., Gutwin, C., Scarr, J. and Malacria, S. 2014. Supporting Novice to Expert Transitions in User Interfaces. *ACM Computing Surveys*. 47, 2 (2014), 1–36.

[9] Daniels, P., Bright, W. and Editors 1996. *The World's Writing Systems*.

[10] Dontcheva, M., Morris, R.R., Brandt, J.R. and Gerber, E.M. 2014. Combining crowdsourcing and learning to improve engagement and performance. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'14)*. (2014), 3379–3388.

[11] Doroudi, S., Kamar, E., Brunskill, E. and Horvitz, E. 2016. Toward a Learning Science for Complex Crowdsourcing Tasks. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'16)* (2016), 2623–2634.

[12] Downey, G.J. 2008. *Closed Captioning: Subtitling, Stenography, and the Digital Convergence of Text with Television*. The Johns Hopkins University Press.

[13] Edge, D., Searle, E. and Chiu, K. 2011. MicroMandarin: mobile language learning in context. *Proceedings SIGCHI Conference on Human Factors in Computing Systems (CHI'11)* (2011), 3169–3178.

[14] Galli, E.J. 1962. The Stenowriter A System for the Lexical Processing of Stenotypy. *IRE Transactions on Electronic Computers*. 11, 2 (1962), 187–199.

[15] Gopher, D. and Raij, D. 1988. Typing With a Two-Hand Chord Keyboard: Will the QWERTY Become Obsolete? *IEEE Transactions on Systems, Man and Cybernetics*. 18, 4 (1988), 601–609.

[16] Hanrahan, B. V., Willamowski, J.K., Swaminathan, S. and Martin, D.B. 2015. TurkBench : Rendering the Market for Turkers. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'15)* (2015), 0–3.

[17] HCID's Kinetic Sentences: *https://github.com/hcid-snu/kinetic-sentence*. Accessed: 2016-09-18.

[18] Horton, J. and Chilton, L. 2010. The Labor Economics of Paid Crowdsourcing. *Proceedings of the 11th ACM Conference on Electronic Commerce*. 1 (2010), 209–218.

[19] Jain, S. and Parkes, D.C. 2009. The role of game theory in human computation systems. *Proceedings of the ACM SIGKDD Workshop on Human Computation - HCOMP '09*. (2009), 58.

[20] Kittur, A., Chi, E.H. and Suh, B. 2008. Crowdsourcing User Studies with Mechanical Turk. *Conference of the SIGCHI Conference on Human Factors in Computing (CHI'08)* (2008), 453–456.

[21] Kittur, A., Nickerson, J., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M. and Horton, J. 2013. The Future of Crowd Work. *Proceedings of the ACM Conference on Computer Supported Collaborative Work (CSCW'13)* (2013), 1–17.

[22] Koedinger, K.R., Corbett, A.T. and Perfetti, C. 2012. The Knowledge-Learning-Instruction Framework: Bridging the Science-Practice Chasm to Enhance Robust Student Learning. *Cognitive Science*. 36, 5 (2012), 757–798.

[23] Komarov, S., Reinecke, K. and Gajos, K.Z. 2013. Crowdsourcing performance evaluations of user interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)* (2013), 207.

[24] Kuek, S.C., Paradi-Guilford, C., Fayomi, T., Imaizumi, S. and Ipeirotis, P. 2015. *The Global Opportunity in Online Outsourcing*.

[25] Lasecki, W., Miller, C., Sadilek, A., Abumoussa, A., Borrello, D., Kushalnagar, R. and Bigham, J. 2012. Real-time Captioning by Groups of Non-experts. *Proceedings of the ACM symposium on User interface software and technology (UIST '12)* (2012), 23.

[26] Law, E. and Ahn, L. von 2011. *Human Computation*. Morgan & Claypool Publishers.

[27] Newitt, J.W. and Odarchenko, A. 1970. A Structure for Real-time Stenotype Transcription. *IBM Systems Journal*. 9, 1 (Mar. 1970), 24–35.

[28] Novotney, S. and Callison-Burch, C. 2010. Cheap, Fast and Good Enough: Automatic Speech Recognition with Non-Expert Transcription. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. June (2010), 207–215.

[29] Seibel, R. 1964. Data Entry through Chord , Parallel Entry Devices. *The Journal of Human Factors and Ergonomics Society*. 6, 2 (1964), 6–9.

[30] Smith, A. 2016. *Shared, Collaborative and On Demand: the New Digital Economy*.

[31] Smith, P.T. and Kelliher, S. 1992. Frequency Effects in Writing Shorthand. *Language and Cognitive Processes*. 7, 1 (Feb. 1992), 67–84.

[32] Smith, R.N. 1973. Automatic Steno translation.

Proceedings of the annual conference on - ACM'73 (1973), 92–96.

[33] Springer, V., Russell, J., Dahir, V., Dressel, W., Sawyer, J. and Brunson, W. 2011. *The Future of Court Reporting: 2011 National Survey - Judges & Court Reporters*.

[34] Suzuki, R., Salehi, N., Lam, M.S., Marroquin, J.C. and Bernstein, M.S. 2016. Atelier : Repurposing Expert Crowdsourcing Tasks as Micro-internships. *Proceedings of the SIGCHI Conference on Human Factors in Computing (CHI'16)* (2016), 2645–2656.

[35] The Open Steno Project: *http://www.openstenoproject.org/*. Accessed: 2016-09-17.

[36] The Sorry State of Closed Captioning: 2014. *http://www.theatlantic.com/entertainment/archive/2014/06/why-tv-captions-are-so-terrible/373283/*. Accessed: 2016-09-17.

[37] Turn, R. 1974. Speech as a Man-Computer Communication Channel. *Proceedings of the May 6-10, 1974, national computer conference and exposition* (1974), 139–144.

[38] Zdenek, S. 2015. *Reading Sounds: Closed Captioned Media and Popular Culture*. The University of Chicago Press.

[39] Zhai, S. and Kristensson, P.-O. 2003. Shorthand Writing on Stylus Keyboard. *Proceedings of the SIGCHI Conference on Human Factors in Computing (CHI'03)* (2003), 97–104.

[40] Zhu, H., Dow, S.P., Kraut, R.E. and Kittur, A. 2014. Reviewing Versus Doing : Learning and Performance in Crowd Assessment. *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'14)* (2014), 1445–1455.

[41] Zyskowski, K., Morris, M.R., Bigham, J.P., Gray, M.L. and Kane, S.K. 2015. Accessible Crowdwork? Understanding the Value in and Challenge of Microtask Employment for People with Disabilities. *Proceedings of the ACM Conference on Computer Supported Collaborative Work (CSCW'15)* (2015), 1682–1693.