Soft Skills and Where to Find Them: Crowd and Expert Analysis of Conversations on GitHub

Ariel Weingarten
University of California, San Diego
San Diego, USA
aweingar@ucsd.edu

ABSTRACT
With the ever-increasing footprint of individuals on the web, googling prospective hires has become a norm. Recruiters have taken this a step further by screening prospective hires’ GitHub profiles to get a sense of the hard skills people possess, but it remains elusive to get a read on a person’s so-called “soft skills”. This paper presents an exploration of what technical and non-technical individuals extracted from conversations held on the open source software collaboration site GitHub. Representative samples from the observations made by these individuals are presented to support a discussion on defining and extracting software project soft skills from GitHub.

Author Keywords
crowdsourcing; open source; social analysis; GitHub

ACM Classification Keywords
K.4.2. Computing Milieux: Computers and Society; Organizational Impacts: Employment

INTRODUCTION
Since its inception in 2007, GitHub has served as a place for coders to host their work and collaborate with others on software projects. Being for and about open source software (OSS), there was little reason for non-technical folk to browse the site. However, now the search for software engineering talent has become important in many industries and recruiters are starting to make use of sites like GitHub to learn about prospective hires [5].

Previous works have demonstrated the ease with which hard skills can be scraped from GitHub profiles and how this can be enough to secure an interview and potentially a job [2]. However, there is much more to a job than the technical skills needed, one needs to be able to get along with the team, back up their decisions, and lead initiatives.

Can we leverage the conversations around open source projects to gain insight into these aspects of individuals?

This investigation consisted of two studies: a task formation study and an exploratory study. In the task formation study, four GitHub threads were analyzed by the author to ascertain whether there was useful information in them and how to frame the task to enable others to pull out this information. In the exploratory study, crowd workers and professional software engineers performed an analysis of six threads across three open source projects.

An analysis of the results revealed three interesting groups of observations: situations where soft skills manifest, a difference between expert and novice interpretations, and the perception of crowd workers and known experts on the experience analyzing of these threads.

Taken together, this paper projects a path towards a concrete understanding of desirable skills and behaviors around software projects and how to discover them.

RELATED WORK
This work builds on previous investigations into the way users of GitHub interact with and form opinions on one another, how recruiters use GitHub to learn about prospective hires, and attempts that have been made to automate information extraction from user profiles.

Project Maintainers Seek Out Potential Contributors on GitHub Using their Activity History
Through interviews with a cross-section of GitHub users Dabbish et al. uncovered patterns in how activity traces are used to learn about the ecosystem. Project maintainers understand how their work is being used by looking at what projects include it and what modifications are being made by users. When a project develops a community, it usually develops a backlog of issues that need to be addressed. It is not guaranteed that the project maintainer is working full time on the project and even if they are there can still be an interest in bringing on additional contributors to ensure the project can continue to serve the needs of the community. Project maintainers will look at the histories of the people who have interacted with their project to scope out individuals that they would like to turn in to contributors. In the open source software world, this is somewhat analogous to hiring. This is a concrete example of how information on GitHub influences collaborative relationships [1].
The Optimization Problem of Software Developer Recruiters

Marlowe and Dabbish performed interviews with six software recruiters to understand how they incorporate GitHub into their screening process. What they uncovered is that recruiters appreciate how every action taken is timestamped and in some sense verified by GitHub. That being said, recruiters recognized that some signals were more easily manipulated than others. For this reason, not much faith is placed in the number of people watching a project. In contrast, having a patch accepted and merged into a large open source project is a more reliable signal because it is difficult to fake and the evaluation cost is not large. Learning about a candidate is constrained by how much time a recruiter can afford to spend on each candidate which shifts their focus onto artifacts with a low evaluation cost. Marlowe remarks that it was interesting that none of the interviewees mentioned looking through prospective hires’ interactions with others to gauge their attitude and personality. They postulated that this could be a result of the high evaluation cost of going through issue and pull request threads and piecing together the social context around a project [5]. This suggests that making activity traces of conversations more readily accessible to recruiters would be incorporated into their screening process as conversations among disparate project contributors are hard to surreptitiously manipulate given GitHub’s audit trail.

Mining Technical Skills from GitHub

There has been work done to automate the process of scraping information from GitHub now that its usefulness has been validated. All work found focused on extracting a profile of technical skills. Greene and Fischer created CVExplorer, a tool that uses various equations to turn activities on GitHub into a measure of technical expertise. They recommended candidates for open positions at two companies to validate its efficacy. Many of the algorithmically identified candidates were brought into the interview funnel and the recruitment teams provided positive feedback on the good match between what they were looking for and who was recommended [2].

Choosing Among Technically Qualified Individuals

If one had a program that could provide a list of individuals who had the pre-requisite technical experience, how would they narrow the selection of individuals they would consider working with? That investigation is what that this paper starts.

TASK FORMATION STUDY

Four threads [7–10] in Facebook’s React (https://facebook.github.io/react/) project were analyzed by the author. Two of the threads were in Issues, GitHub’s issue tracker and the other two were pull requests. Here are the definitions of Issues and Pull Requests as provided in guides and help page on GitHub:

“Issues are a great way to keep track of tasks, enhancements, and bugs for your projects. They’re kind of like email—except they can be shared and discussed with the rest of your team.” (https://guides.github.com/features/issues/)

“Pull requests let you tell others about changes you’ve pushed to a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before the changes are merged into the repository.” (https://help.github.com/articles/about-pull-requests/)

Method

A comment-by-comment summary was written that characterized the interactions between participants. Afterwards, a meta-analysis was done to discuss the thread holistically and to reflect on how to turn create a repeatable structure for the task so that others could reliably and usefully perform it.

Results

It was discovered that these analyses are time consuming. It consistently took longer than twenty-five minutes to analyze threads. There was an of 9 comments per thread. A task template for the exploratory study was created and a task price of $5 was chosen to provide roughly the equivalent of a wage of $10/hour for crowd workers.

EXPLORATORY STUDY

The task consisted of three parts: a Likert scale where participants self-reported their experience with software projects and programming, an event-by-event commentary of the GitHub thread, and answers to two questions about the thread:

1. Are you satisfied by the outcomes and interactions of the thread and why?
2. What guidelines would you recommend for a future thread on a similar topic

Participants were asked to assume the role of a manager of the software project and the overview

One Issue thread and one pull request thread were chosen from three well-known open source projects: Facebook’s react (https://facebook.github.io/react/) , npm (https://www.npmjs.com/), and Oh-My-Zsh (http://ohmyz.sh/) for a total of six conversation threads [7–14].

Participants

The author, six software engineers, and nine crowd workers were given the task for a total of two expert analyses and two crowd analyses of each thread. The crowd workers were recruited via Amazon Mechanical Turk (https://www.mturk.com/mturk/welcome) and the experts were friendsourced on Facebook.

Each participant was provided a Google Doc that contained the Likert scale, instructions for the task, and link to the
thread to be analyzed. Crowd workers were given a time limit of two hours which is more than twice as long of a window as the longest initial review in the previous study.

**Results**
Five out of nine crowd workers had no experience with either working on software projects or programming in general.

<table>
<thead>
<tr>
<th>Coding Experience</th>
<th>Software Project Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C: 5 C: 1</td>
</tr>
<tr>
<td>2</td>
<td>C: 1 C: 1</td>
</tr>
<tr>
<td>3</td>
<td>C: 3 C: 1</td>
</tr>
<tr>
<td>4</td>
<td>P: 1 P: 2</td>
</tr>
<tr>
<td>5</td>
<td>P: 1 P: 1</td>
</tr>
</tbody>
</table>

*Table 1. Summary of Crowd (C) and Professional (P) Self-reported Software Project and Coding Experience (1 = not experienced and 5 = extremely experienced)*

**Observations**
Crowd workers will be quoted with the abbreviation CW and the professional software engineer and the author will be referred to as known experts (KE).

*Introducing Changes and Problems*
At least one analysis for every thread made a judgement on the opening poster for the quality of their post. Most positive comments mentioned that the motivation for the change or problem had been made clear. Most negative comments mentioned that the tone was less than friendly or impatient. Below is one example of a thread introduction and the accompanying analyses:

“I have tried many times to install firebase-tools but not succeed. and after i have updated my npm version from 3.10.6 to 4.0.2 but this time also failed the error is “error fetch failed https://registry.npmjs.org/faye-websocket/-/faye-websocket-0.9.3.tgz”. please help me asap. It’s urgent.”

CW1: “he was very friendly in his request for help”

CW2: “would have been helpful if the person provided more details. Seems to be impatient.”

KE1: “Asking for help “asap” does not help anything, nobody has any particular reason to help him... He says “so please help me” which is incredibly self-centered. The fact that anyone is participating in this thread means that they are attempting to help him.”

KE2: “He also asks for help “asap” which I find unhelpful on an internet forum, everyone has an important problem, that’s why they are asking on the internet.”

CW1 self-reported being “very experienced” working on software projects and “somewhat experienced” with writing code. CW2 self-reported being “not experienced” with either working on software projects nor writing code.

**Making a Claim**
Claim making was scrutinized throughout the analyses. Workers would often praise individuals that cited their sources. Outright condemnation of lack of sources was not common, but many would indicate it as an area where there is room for improvement. The first quote is about someone who did not provide documentation to support a claim and the second is about a user in the same thread who provided the missing documentation.

“User could have done additional research and provided supporting documentation in order to ensure that all parties were convinced and assured that this command was part of the program itself.”

The same worker had this to say about the following post’s user:

“links to supporting documentation from the program’s ... This is a very helpful response that goes above and beyond.”

Linking to documentation was not the only contribution made in that post. The two times that the phrase “above and beyond” were used by workers in their analyses was about users that provided documentation of their claims.

**Dealing with Disagreement**
Disagreement elicited commentary from participants. In one thread, the original posted (OP) started a conversation about whether a particular default behavior should be opt-in rather than opt-out. Several others joined the thread to provide their opinion on the matter and the OP engaged each of them directly. There were several disagreements in the thread and the way that OP handled them left an impression on two of the analysts.

KE1: “@OP’s behavior is fantastic in this thread. I would hold it up as an example of how to conduct oneself in conversations on software project issues.”

KE2: “I’m starting to quite like this user, who seems to be very friendly even when in a disagreement.”

One crowd worker did not comment on the nature of the disagreements and the other mentioned it briefly:

CW2: “Is agreeable and listens to others views”

**Inferences on Thread-Level Actions**
In addition to writing comments in GitHub Issue threads, users can add tags to an issue. These tags help group issues as well as make it easier to understand what is going on in a conversation. In one of the threads, the known-experts read
into the adding of a “Good first bug” tag, whereas the crowd workers did not make any inferences.

KE1: “adds a “good first bug” label to the Issue. This acknowledges that the original poster seems to have lost interest and that someone else could come in and contribute the fix which has more or less already been outlined in the preceding conversation.”

KE2: “I infer that @gaeron is frustrated by @uxitten because @uxitten clearly is uninterested in understanding the root of the problem and therefore fixing the problem.”

The crowd workers did not comment on the labeling other than stating that it occurred. In this case, the crowd workers were both self-reported “not experienced” with either working on software projects or writing code.

Crowd Perception of the Task
One worker sent a message about their experience with the task, “I really enjoyed this hit and I hope I did it well enough. It made me think a bit which is a nice change of pace from other tasks here on turk. :)”

Professional Perception of the Task
One software engineer sent a message about their experience with the task, “I could write a book about this one comment thread. There's so much happening under the surface here”

DISCUSSION
From these observations, we can start to answer some questions about extracting soft skills from open source software repositories: What are soft skills and where do they appear? Can the crowd contribute to the analysis of these threads as effectively as professional software engineers? In addition, we can imagine future work to apply the discoveries of this exploration.

Soft Skills and Where to Find Them
So far this paper has not defined what is meant by soft skills. One could cite sources such as Executive Perceptions of the Top 10 Soft Skills Needed in Today’s Workplace and define them as: communication, courtesy, flexibility, integrity, interpersonal skills, positive attitude, professionalism, responsibility, teamwork, and work ethic [6]. However, these don’t come with a mapping onto actions taken by software developers around their projects. This investigation uncovered three regularly occurring events in GitHub threads where this mapping can be explored. Keeping these buckets in mind while engaging in a bottom-up design process should yield an approachable and concrete collection of soft skills that are valuable in a software team environment.

Future work can design a task to produce a deeper analysis of an individual by examining many of their introductions, claims, and disagreements and present this information in a useful manner. In this case, useful is meant to mean that the goal is not to say some individuals are “bad” and others are “good”, but rather to uncover people’s communication styles. This would enable individuals to better understand their own communication style and companies to make an informed decision on whether someone would be an appropriate “culture fit”.

Efficacy of the Crowd
Unexpectedly, only about half of crowd workers had no experience with software projects or writing code. This suggests that it may be possible to screen workers based on a base level of coding and software project knowledge and still have access to a large enough pool of workers to accomplish a meaningful amount of work.

The disparity between novices and experts in the inferences made on the usage of tags in threads suggests that training on the features available on GitHub threads should be incorporated into the task. It makes sense that individuals who have not created and tagged issues on GitHub would not have an intuitive understanding of it. Schema sharing [4] among workers could be an effective way to fill the experience gaps of individual workers.

Well-trained workers will only go so far. The task itself needs to be refined to produce data that is directly useful. The current task’s results took hours to analyze and are not immediately useful outside of this work. Need finding is required to discover what information recruiters and software developers would consider useful.

Leveraging people for their social computing power rather than their ability to perform menial tasks stood out to at least one crowd worker. Finding meaningful, enjoyable, and distributable work is an important step towards advancing crowd work out of the age of menial, mechanical tasks [3]. However, employing the crowd to evaluate individuals in the wild could easily take a dystopian turn. It is important that a user centered design process guides the development of this application of the crowd to ensure it benefits the people’s whose digital footprints are being scrutinized.

CONCLUSION
This work examined how known-expert and crowd workers understand and describe conversations in open source software repositories. We found three regularly occurring events that solicited soft skill commentary: problem and change introduction, claim making, and dealing with disagreement. This work also uncovered a difference in how expert and novice workers understand actions taken on the threads themselves, such as thread labelling. The results inform the design of an approach for defining and extracting software project soft skills from GitHub.

ACKNOWLEDGMENTS
Thank you to all the students in COGS 260 for providing feedback over the course of the semester. Special mention to Steven Dow and Tricia Ngoon for numerous conversations about the project. This work is made possible by the 1st year CSE fellowship.
REFERENCES


7. how remove debug info from react.min.js file when publish project. Retrieved November 28, 2016 from https://github.com/facebook/react/issues/7990


