Trials and Tribulations of Developers of Intelligent Systems: A Field Study

Charles Hill[1,2], Rachel Bellamy[2], Thomas Erickson[2], and Margaret Burnett[1]

[1]Oregon State University
Corvallis, OR, USA
{hillc, burnett}@eecs.oregonstate.edu

[2]IBM TJ Watson Research Center
Yorktown Heights, NY, USA
{rachel, snowfall}@us.ibm.com

***Abstract*— Intelligent systems are gaining in popularity and receiving increased media attention, but little is known about how people actually go about developing them. In this paper, we attempt to fill this gap through a set of field interviews that investigate how people develop intelligent systems that incorporate machine learning algorithms. The developers we interviewed were experienced at working with machine learning algorithms and dealing with the large amounts of data needed to develop intelligent systems. Despite their level of experience, we learned that they struggle to establish a repeatable process. They described problems with each step of the processes they perform, as well as cross-cutting issues that pervade multiple steps of their processes. The unique difficulties that developers like these face seem to point to a need for software engineering advances that address such machine learning systems, and we conclude by discussing this need and some of its implications.**

***Keywords*— Empirical; applying machine learning; machine learning**

I. INTRODUCTION

The convergence of machine learning, big data and cloud computing has been receiving increasing attention. More and more people are aware of intelligent systems such as Siri, Google Now, Cortana and Watson. Such systems give the impression of being able to understand natural language, reason about abstract concepts, take historical and contextual factors into account, and learn from experience. These gains come from combining computing systems' analytical power with massive amounts of data. It is no surprise, then, that enterprises are scrambling to develop big data strategies that will transform their stores of data into actionable insights, and enable them to engage customers in ways tailored to each customer's needs and situation.

At the same time, popular media is rife with accounts of seemingly intelligent computers. According to more imaginative accounts, such systems are well on their way to being autonomous: able to understand language, reason about abstract concepts, and learn from experience. While some of the popular accounts strike us as hyperbolic, the convergence does indeed offer a promising range of new applications. However, we are struck by the absence from many accounts of a key player: people.

People play a variety of roles in developing intelligent systems, from gathering and curating data to developing and tuning machine learning models. Therefore, in this paper, we focus on how people develop intelligent systems in practice. We believe that by understanding the processes by which people go about developing intelligent systems, we as a research community can better understand how to support their efforts at developing such systems. What skills do developers of intelligent systems need? What tools do they use? What problems and pains do they face? To what extent is there even a consensus on the answers to these questions? Answers to these questions can point the way toward supporting more effective tooling for intelligent systems development.

This study is a first step in addressing these questions. In this paper, we focus on professionals who use machine learning (ML) techniques to develop intelligent systems in the context of a large enterprise. After discussing prior work, we describe the study's approach: interviews with 11 informants from multiple groups involved in the development of different intelligent systems. Next we qualitatively analyze the data, describing a general process and its characteristic problems and pains, and a set of themes that emerged from the interviews. Finally, we consider the unique difficulties of intelligent systems development as a software engineering practice, possible sources of those difficulties, and a few possible directions forward.

II. BACKGROUND AND RELATED WORK

Difficulties with ML, especially for end users, have been well established. For example, Amershi et al.'s survey and other works point to users' feedback leading to wild fluctuations in ML systems' reasoning, users' need for more transparent reasoning by ML systems, users' inability to accurately serve as oracles to the system, and their lack of trust in such systems' reasoning [1, 7, 11, 19, 34, 35]. In response to such issues, an increasing body of research has begun to explore what users can do when their ML systems make mistakes (e.g., [1, 2, 15, 17, 18, 35, 37]). The aim of much of this work is to enable users to effectively and efficiently personalize the predictions or recommendations these intelligent systems make on their behalf.

Still, given such populations of end users, one might argue that it is not surprising that those outside the computing profession often have trouble using or trusting such complex systems. But while the literature has less to say about professional developers' attempts to use ML, there has been some work indicating that they too face difficulties. For example, in 2008 Patel et al. interviewed researchers who applied ML to HCI research problems. These researchers reported numerous obstacles. Patel et al. then watched computer science graduate students build an ML model to see how these obstacles manifested [25]. Among the obstacles were problems with inappropriate training data, inability to understand whether the results were valid or not, and

978-1-5090-0252-8/16/$31.00 ©2016 IEEE

protracted periods of exploration without any useful result. In contrast to Patel et al.'s participants, our participants were professional ML developers using ML for a broader range of applications. Our participants also had a broader range of experience: from creating ML algorithms to making use of those algorithms in applications.

More recently, Guo reported on "research programming" [13] –a term he uses to refer to programming when the goal is to obtain insights from data. All of Guo's examples are data-intensive programming, and often involve machine learning to process that data. His examples include web marketing analysts writing programs to analyze clickstream data, computational scientists writing programs to analyze data to make scientific discoveries, algorithmic traders writing programs to simulate experimental trading strategies given existing financial data, and public policy analysts mining U.S. Census and Labor statistics to predict the outcomes of proposed government policies [5, 14, 23, 27, 33]. As with Patel et al.'s study, Guo recounts numerous obstacles faced by these developers relating to the data, the analysis, and evaluation of output's correctness [13].

In essence, this evidence suggests that even experienced programmers have trouble applying ML. This is disturbing because for everyday applications and user interfaces to become more intelligent, software developers will have to be able to somehow incorporate intelligence.

A variety of tools have been devised to try to address these issues using the idea of interactive machine learning [8], so named for its tight feedback loop between a system's reasoning and some kind of feedback from a human (e.g., the end user or developer). Some of these tools treat the ML algorithms as "black boxes," with the idea that non-experts should not need to know the inner workings of ML.

Popular black box approaches are instance labeling, in which a user trains instance-based classifiers [3, 10], active learning [4, 30], and interactive changes of reinforcements in reinforcement learning [16]. The black box notion at first seems attractive from both a usability and an economic perspective, because its success would suggest that ordinary developers, without extensive ML backgrounds, could select, connect with simple API calls, evaluate, and parameterize ML algorithms and thus incorporate ML into their systems.

Participant	Position	Machine Learning Experience
P1	Researcher	4.5 years
P2	Intern	3 years
P3	Software Architect	2 years
P4	Researcher	5 years
P5	Engineer	20 years
P6	Software developer	10 years
P7	Researcher	15 years
P8	Researcher	3 years
P9	Researcher	6 years
P10	Project Lead	10 years
P11	Researcher	25 years

Table 1: Participants had a range of experience and positions.

However, it has been argued that, given the complexity of ML algorithms, ML algorithms as black boxes cannot work. The argument rests on non-specialists' mental models: humans develop mental algorithms that they believe reflect the system's reasoning, but these models can be very flawed in the absence of transparency, and are not easily altered once established [36]. With flawed mental models, users tend to lead ML systems astray, providing feedback that causes an algorithm to perform worse instead of better [19, 20, 35].

To solve this problem, some researchers advocate for white box (or at least whit*er* box) approaches that unveil an ML system's reasoning. Some examples are explaining ML with a variety of intelligibility types [17, 22], feature labeling [6, 28], and constraining or critiquing the ML system's search space [12, 15, 24, 37]. However, white box approaches are also far from a panacea: they face many unsolved challenges, such as the difficulty in telling the whole truth to non-expert users without overwhelming them with the complexity of many of today's ML algorithms [17].

Still, non-experts are a worst-case situation. One way of understanding whether white box approaches have even the potential to succeed is to consider a best-case situation—the whitest of white boxes, namely a white box viewed by experienced developers of intelligent systems. If we can establish what experienced ML developers bring to the development of systems incorporating ML and how (and if) they succeed, we can better understand what kinds of support or automated intelligence must be brought to inexperienced or novice ML developers or users. However, until now, there has not been a comprehensive look at how experienced ML developers go about performing machine learning tasks. That is the gap this paper aims to fill.

III. Methodology

A. Participants and Procedures

The study site was a large global enterprise that is well known for its work in intelligent systems. Participants were recruited by an email that solicited experienced ML developers for participation in an interview (which we'll refer to as Interview #1). Ultimately, we interviewed 12 participants. One was excluded from the study because s/he had only four months experience with ML. Our remaining 11 participants were experienced ML developers from different parts of the enterprise who represented a variety of different projects.

All participants' primary role in their profession was machine learning, but they had a wide range of experience and skill sets. Participants had one of two types of experience with ML: they either worked on developing ML algorithms or used ML algorithms in the development of intelligent systems. The 11 participants (10 male, 1 female) had a minimum of two

`

years of ML experience. The median number of years of experience was 6, the mean was 9.4, and the maximum was 25 years. Table 1 shows the job title and experience level of each.

Interview #1 was a semi-structured interview, in which the interviewer arrives with a required list of questions, and then drills down dynamically with further questions that follow up on the participant's specific response. We interviewed the participants in person when possible; otherwise we did so via conference calls. Interviews ranged from twenty to forty minutes. All interviews were audio recorded with permission from participants.

We started Interview #1 with demographic questions. These included the participant's position, years worked at the company, and years of experience with machine learning. We then began the body of the interview with the following "main" prompt:

Please describe the last time you worked on any component of an application or Machine Learning model where you had to diagnose or solve problems with the model.

For the remainder of the interview we asked participants to reflect on the project they had just described, using the following topics to guide discussion:

Data collection: How did you gather data? If you didn't collect the data yourself, where did it come from? How did you decide which data was relevant for your task?

Feature selection: How did you go about the process of feature selection?

Ground truth: Did you establish ground truth yourself? Did you ever have to change ground truth? How did you keep track of changes to ground truth?

Process: What was your role in the project? What steps did you follow over time?

Algorithm selection, implementation, improvement: How did you choose the algorithm you used? How did you integrate it into your work process? How did you go about improving the performance of the resulting model?

Version control: When and for what purposes did you use version control?

For each topic we also asked participants whether or not they'd encountered any problems along the way.

B. Analysis

After transcribing the Interview #1 responses, one coder analyzed them by listening to and reading the interviews repeatedly to find recurring themes and patterns across interviews. Participants then validated the analysis of Interview #1 via verification interviews, which we'll refer to as Interview #2.

In these verification interviews (Interview #2), we asked participants questions about the themes and patterns we had derived, namely whether they (1) had *personally* experienced them (to validate our analysis) and (2) had *observed* them in the field (which, added to question 1, measured the extent of each phenomenon). Seven of the 11 original participants participated in Interview #2. (We were unable to reach the other four original participants for participation in Interview #2.)

Participant validation levels ranged from 67% to 91%. Overall, participants agreed with our analysis of *their* Interview #1 content 82% of the time (i.e., question (1))—a validation level that exceeds Stemler's [32] guidelines for consensus measures being at least 70%.

In the following sections, we aggregate the results of Interview #2 questions together, to measure the extent to which these participants actually experienced or witnessed the phenomenon in the field. In essence, aggregated Interview #2 responses will represent how widespread a phenomenon was.

IV. Results: The Process

Figure 1 shows participants' high-level description of the process they follow, but in its ideal, trouble-free form. Thus, we present our results in the sequence of Figure 1. Note, however, that these steps were not always performed in this exact order, and not all participants performed all steps. Also note that our interviews did not cover the first step shown in Figure 1, labeled "Define Problem". This step in itself is a potentially interesting one, but our study's scope begins after the problem has already been defined.

Table 2 shows the results of recurring themes and patterns we derived from the Interview #1 data for each step in Figure 1 (starting with Collecting Data), and how widespread each phenomenon was as per Interview #2. We detail each in the next subsections.

A. Collecting the Data

Data for an application based on statistical machine learning must come from somewhere. For example, one

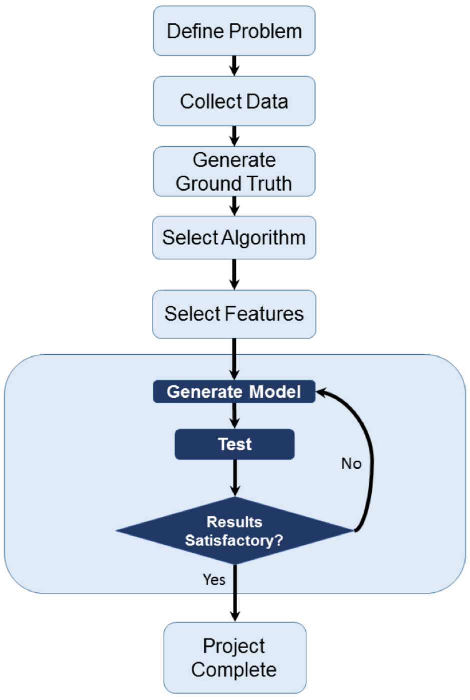

Figure 1: A simplified, "ideal" version of the machine learning process described by our participants.

`

participant needed text snippets from business descriptions to classify, and another participant needed radio protocols to predict the source of radio waves.

When we asked participants how they collected their data, we learned that most of our participants had managed to avoid collecting their own data. Instead, many of our participants received data from third party vendors or from machine learning competitions where the data was provided in a pre-packaged format; in Interview #2, 6 of the 7 respondents reported having seen or done this (Table 2).

A key reason for this high rate of avoiding collecting data was that data collection is very difficult. As P3 cautioned, *"It's a very involved process."* One problem was that (P4): "*... there's some data that is not easily obtainable."* Another problem was that, even given access, the data required special tools to process. As P4 explained:

P4: *"... custom tools that we don't have access to ... we still, years after this product is being sold ... <have> fundamental data that's hard to get at."*

Phenomenon (from Interview #1)	# Participants who saw or experienced it (Interview #2)
Collecting the Data	
Obtaining data in some way other than collecting yourself	6/7 (85%)
Establishing Ground Truth	
Difficulty establishing ground truth, due to errors and faulty data	7/7 (100%)
Selecting the Algorithm	
"Obvious" choice as to which ML algorithm(s) to use	4/6 (67%)
Using "common sense" to select algorithms	4/6 (67%)
Starting out with simple ML algorithms and then moving to complex algorithms.	6/6 (100%)
Relying on your past experience to select ML algorithms	5/6 (83%)
Selecting the Features	
Importance of understanding the algorithm and its limitations in being able to select features	6/6 (100%)
Evaluating the Model	
Testing taking extensive time and effort	7/7 (100%)
Errors caused by lack of experience or knowledge	3/7 (42%)
Needing to consult with an expert to debug the results	3/7 (42%)

Table 2: Process results overview. 11 participants answered Interview #1, and 7 of them later answered at least some questions in Interview #2; the denominator shows how many answered each Interview #2 question.

`

P3's team, who collected data from an internet source manually, even had to build a system just to keep track of their data:

P3: *"... so we built a ... <special-purpose> system ... a very complex system to manage this process."*

Even when participants were provided with data, there were difficulties. For example, Figure 2 shows how P8's process deviated from Figure 1's ideal path. Issues with the provided data set P8's project back by an entire month:

P8*: " the data was stored in a [type of] system, and we wanted to use [another system]... all these migrations <were> ... painful. It took us ... more than one month."*

P11 had also been given data, but still had to analyze its usefulness to their needs:

P11: *"This may be a very rich, good source of data or it may be junky.... hard to sift through... There may be so little uniformity in what we find ... that non-uniformity will render the data basically useless."*

At the time of the interview, P11 was still trying to decide whether and how to supplement the provided data.

B. Establishing Ground Truth

For some particular set of test data, what are the right answers? For example, a machine learning system might need to classify statements such as "Babe Ruth hit one out of the park" with the appropriate category label, choosing among possibilities like Baseball, Candy, or Misdemeanors. Ground truth is the "correct" answer. Data sources provided by machine learning competitions included ground truth (i.e., answers were provided to them), but in other situations, the participants had to establish it (i.e., they generated the answers themselves). All 7 Interview #2 respondents had seen or experienced difficulties establishing ground truth (Table 2).

Ideally, ground truth should be a perfect answer key against which to test a model. Unfortunately, this often wasn't the case. For example:

P2: *"even the canonical benchmark sets ... have label noise or garbage data ..."*

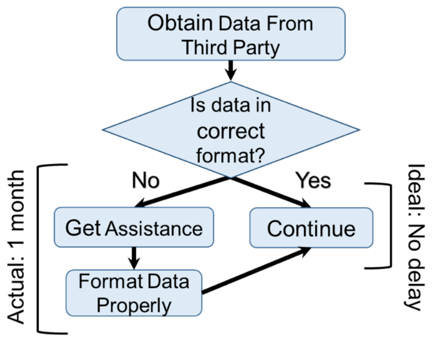

Figure 2: P8's data collection process. P8: "*a really painful stage in which we couldn't move forward*" until someone helped them format their data.

P11: *"In the end, I think ground truth is: Are we reflecting what the human expert would have done. ...anything we do is probably based on an imperfect ground truth."*
P9: *"some cases ... <were> more qualitative - so it was like I subjectively estimated if it's good or bad"*

Not only was ground truth often imperfect, it was costly to establish. For example, Figure 3 shows how P2's process deviated from Figure 1's ideal path. P5 emphasized the extent of such deviations from the ideal path:

P5: *"So ground truth is a huge problem, very labor intensive, ... we're doing a lot of this very manually."*

This labor intensiveness was especially reflected by P3's team—about 40 members of that team were working to establish ground truth.

C. *Selecting the Algorithm*

Different ML algorithms can function differently on the same data, and scholarly papers abound on which ML algorithm to use for different situations and ways to choose among them (e.g., [6, 30]). However, most participants described relatively ad hoc ways of making their selections.

For example, some participants simply selected algorithms that they liked:

P8: *"it's not that important ... you fit using whatever algorithm you like ... "*

Others experimented with a set of algorithms they had used in prior projects. For example, when P7's team did not know which algorithm to use:

P7: *"Past experience in other projects led us to the choice ... we tried several ... and saw which ones went better."*

A key element of the participants' algorithm selection process was to keep the algorithm as simple as possible. All 6 Interview #2 responses reflected this (Table 2). As P10 put it:

P10: *"I always start with the simplest algorithm, linear classifier, as a baseline, and then ... try .. more complicated algorithms."*
P11 agreed: *"My rule of thumb is ... always to start with the simplest ..."*

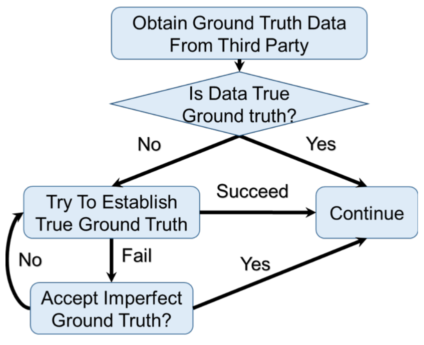
Obtain Ground Truth Data From Third Party
Is Data True Ground truth?
No
Yes
Try To Establish True Ground Truth
Succeed
Continue
No
Fail
Yes
Accept Imperfect Ground Truth?

Figure 3: P2's ground truth process, which P2 summed up as: *"True ground truth eludes us."*

`

D. *Selecting the ML Features*

Wikipedia offers an informal definition of an ML *feature* as "an individual, measurable property of a phenomenon being observed" [38]. But choosing which features to use can be challenging. All 6 Interview #2 respondents reported the importance of understanding the algorithm when selecting features (Table 2). Further, although choosing informative, discriminating and independent features is key for algorithms' effectiveness, as P5 aptly pointed out, it is not yet scientifically well understood:

P5: *"...it's an ongoing science. <ML researchers> are still trying to figure out what features are useful for different types of problems."*

Sometimes participants were able to circumvent the challenge by simply using automatic feature selection:

P6: *"I just used the feature selector that ... came out of the box."*
But when P6 could not do this, a set of experiments would ensue: *"We <used> a very simple feature selection process where we just looked at the statistical ... and only those that have some relatively low threshold would be kept."*

However, most participants relied mostly on domain knowledge, past experience, and opinions. For example

P10: *"usually ... using common sense and prior knowledge about the domain ... and features that we believe are indicative of the labels we're looking for."*
P11 (summing up): *"Well, there's always some art to it."*

E. *Evaluating the Model*

An algorithm works with features and training data, and ultimately produces a model. The *model*, in turn, is what is run to produce the answers. At this point, participants described how they evaluated whether these models' answers were correct.

All 7 Interview #2 respondents (Table 2) reported evaluating the model to be a long, arduous process, in which participants iteratively refined the model through changes to parameters, training data, and so on. They did these iterations using an evaluate-fix-evaluate cycle that often required many changes, sometimes even sending the participants back to earlier steps in the process:

P7: *"a lot of repeating trials: add new features run it again, try replacing some features, run it again, try some different parameters run it again..."*
P8: *"<We> try various models on the cross validation set and then pick the one that gives you the best performance..."*

But it was often difficult to understand the results of the model, and about half of the Interview #2 responses (Table 2) pointed to a need for significant ML expertise. P2 described the experience shown in Figure 4, and summarized it this way:

P2: *"Normally ... we run something <to> see if it works at all. Most of the time, it doesn't..."*
P2 (continuing): *"...If we figure out why, then ... try to tackle that problem. If we don't then ... back to the drawing board."*

P6 didn't feel it was even worth his time to try to improve their model, because his colleagues didn't understand the nuances well enough:

P6: *"I started with beating the baseline, and then just said 'good enough.'"*
P6 went on to explain why: *"It was so difficult to explain to people what evaluating a model means ... It would probably be a small effect, and no one would understand what it meant... "*

V. Results: The Cross-Cutting Issues

Table 3 shows a set of cross-cutting issues we derived from the Interview #1 data for each step in Figure 1 (starting with Collecting Data). We detail these issues in the next subsections.

A. Cross-Cutting Issues with Environments and Tools

1) Tools for doing the work

Participants often had to rely on in-house solutions purpose-built for their situation; 5 out of the 7 Interview #2 respondents reported this issue (Table 3). These solutions ranged from scripts written by one person and passed along through the years to websites purpose-built by teams to help them manage data. Common reasons were that there was no other way to access the data they needed, or there was no previous way to do what they needed to do:

P7: *"At some point we developed a web interface ... to display data ..."*
P7 continues, lamenting: "*... <but> we didn't really have the right skill."*

Rather than build anew, P6 tried to rely on older (custom) scripts to interact with the machine learning software he needed. Unfortunately, he did not understand these scripts, so he had to meet often with the original scripts' authors. This was a situation he'd found himself in often in his 10 years of ML development experience:

P6 (about reusing custom tools): "*Ideally you push a button ... in practice it would always fall down ... it wasn't documented, you just had to be sitting there ... face to face with people who wrote it."*

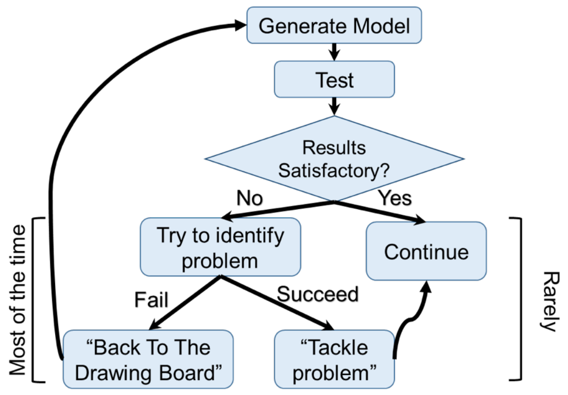

Figure 4: There were significant issues with P2's machine learning model: *"Most of the time, it doesn't work at all."*

2) Tracking versions and experiments

As experienced developers like our participants know, keeping track of the evolution of their software is important. Indeed, all 7 of them (Table 3) used standard version control systems to track the source code in their projects. Unfortunately however, they found themselves without options when it came to tools for tracking their models.

P3: *"From the software perspective, there is version control, but from the side of ML, there is no tool for that."*

To solve these problems, some participants used text files to keep track of old commands they had run on their model, along with results. One participant used a database to record old commands and results. Overall, participants tended to track workflow via informal methods, such as emails among colleagues or notes to themselves.

Especially problematic was the lack of tools to keep track of tests and experiments on their models, as indicated by 6 out of the 7 Interview #2 respondents. This led to wasted time, and tests and conditions had to be repeated unnecessarily. Another problem was an inability to reproduce the exact conditions under which a machine learning model was created. As P2 explained:

P2: *"If you're not careful ... you have something that's not reproducible. You have this magic binary ... then you try to recreate it and it doesn't work ..."*
P2 continues: *"Congratulations, you have something that can't be ... duplicated. That's bad. Very bad."*

Cross-cutting Issues (from Interview #1)	**# Participants who saw or experienced it (Interview #2)**
Issues with Environments and Tools	
Tools: Had to rely on and/or build in-house tools	5/7 (71%)
Version control tools: can/did use for code (algorithm, application, ...), but not available for models and model experiments	7/7 (100%)
Tracking experiments: only through emails, notes, or other informal methods	6/7 (85%)
Procedural Issues	
Trial and error: Had to resort to basic trial and error	7/7 (100%)
Assistance: Needed to turn to subject matter experts for assistance.	6/7 (85%)
Magic: Mysterious results/outputs of models	4/5 (80%)

Table 3: Crosscutting issue results overview.

`

When asked how he kept track of his data to help solve this problem, P2 responded:

P2: *"Currently, I am just very, very, very careful."*

B. *Cross-Cutting Procedural Issues*

Participants were largely dissatisfied with the ad hoc collection of procedures they found themselves using—an abundance of trial and error, judgment calls requiring extensive prior experience, and reliance on "magic."

1) Trial & error, judgment calls, and assistance

Every step of the process had participants referring to some measure of uncertainty: from feature selection being an "ongoing science" and an "art" to having to test a machine learning model over and over again due to being unable to diagnose problems with the model. Evaluation and testing were particularly described as a "long, labor-intensive process" (all 7 Interview #2 respondents). The only consistent testing methodology among the participants was to run the model over and over again.

Judgment calls based on prior ML experience also abounded. Participants relied on rules of thumb (e.g., "start with the simplest algorithm") (all 6 Interview #2 respondents in Table 2), past experience (5 of 6 Interview #2 respondents in Table 2), and having to turn to subject matter experts for assistance (5 of 7 Interview #2 respondents in Table 3). The common theme was a heavy reliance on prior ML knowledge in order to be successful:

P10: *"... usually engineered using common sense and prior knowledge."*

Unfortunately, this need to draw heavily upon prior ML experience suggests a rather large chasm for ordinary software developers to cross before they can reasonably incorporate ML into their applications.

2) Magic, black art and voodoo

Finally, several of these experienced ML developers referred to machine learning as something akin to magic:

P4: *"... inside this black box ... all the magic of machine learning that happened on the inside is off-limits to us."*
P5: *"Too many of our cases, the machine learning people are ... in the background doing this like a black art."*
P2's description of problem-solving: *"a process of going from obvious tricks to one level away from, like, voodoo."*

Participants went on to describe a special inner circle of "high priests" whose help was necessary in order to obtain the right incantations. For example:

P5: *"A lot ... is done manually by the high priests of machine learning."*
P5: *"Too much of this is in the hand of machine learning guys ... what we need to be able to figure out is a way to ... train a ML model without one of those guys in the loop ...*

Even P4, who was something of an ML expert himself in that he interpreted the results of machine learning algorithms for others, referred to himself and his team as being "mere mortals" whose expertise was not enough:

\`

P4, on the way the process would ideally be: *"Look at the questions to see: do they appear to us as mere mortals, do they look like they do have the same intent?"*
P5: *"[instead it should be possible to succeed by] ask mere mortals a set of questions."*

VI. Discussion and Concluding Remarks

Our participants identified a set of steps that together make up the process of developing an intelligent systems application—and had difficulties with all of these steps.

A. *The Need for a Software Engineering for Machine Learning*

The issues our participants described suggest an important need for explicit attention to software lifecycle phases—they show that developing intelligent applications is not simply a matter of creating an application and occasionally calling ML library routines. For intelligent applications, "regular" software engineering activities and skills apply, of course, but our participants revealed that the skills needed for nurturing an intelligent application through its birth and lifecycle go far beyond this set.

As just one example, consider the testing and debugging activities of software development. In traditional software development, the thing that needs testing and debugging is the application. But in intelligent system development, our participants described onerous processes of testing and debugging on up to four different artifacts—the ML algorithm, the training data and its relationship to the "real" data, and the parameters to send to the algorithm, in addition to the actual application. Compounding these problems, these experienced ML developers described many cases in which developing ML-based systems required skills held only by certain "high priests". Given such complexity, perhaps it is not so surprising that these experienced developers described debugging such systems as something akin to magic and even voodoo.

It is interesting to consider why developing ML applications appears to be so different from developing other kinds of applications. Sculley et al. reflected upon their own experiences in a position paper recounting their views of the difficulties of developing ML applications [29]. One point they raise is the "change anything, changes everything" nature of ML models [29]. As they explain, the dependencies among all the parts of the ML application (application code, "glue code", ML libraries, and external data) prevent use of standard techniques for reducing coupling such as abstraction and information hiding. Not being able to isolate the impact of a specific change anywhere in the system of dependencies could be a reason that our practitioners so often resorted to ad hoc practices like trial and error and rules of thumb.

Some participants' reports of having to fall back on assistance from "high priests" of machine learning are reminiscent of craft trades, with apprentices learning from master craftsmen/women. To go beyond this craftsmanship state, systematic methods usable by "ordinary" practitioners (like our participants) are needed. Perhaps deriving these systematic methods from ML experts' practices would provide a starting place for an ML-specialized software engineering to meet this need.

B. Tools Today and Tomorrow

The participants also suggested a basic mismatch between the tools available vs. their practical needs. For example, they reported a lack of usable tools for their specialized data transformation needs and data-set integration. The tools they described using targeted only one step in their development processes, but did not easily interoperate with other tools they needed to use.

Our participants also said they needed tools and methods that helped them track the data they have chosen to use, the choices they have made as to algorithm and features, and the experiments they had already run. These were important so that they could rerun experiments with the same data and model settings. When such tools were not available, they needed them so badly, they created their own.

One kind of tool they did not seem to have that has helped in other software engineering situations is a "foraging" tool. Information foraging theory (IFT) has been helpful not only in explaining some of the problems developers have when developing and debugging code, but also in supporting them in these activities [9, 21, 26, 31]. Studies have shown that developers follow "scent" in a code base to track down the places in their programs that are causing bugs. One possible interpretation of our results is that the ML tools used by these developers did not help developers follow scent when trying to understand a model and its associated interdependencies.

IFT hasn't yet been applied to debugging ML models. Future work is needed to see if this theoretical approach, which has been found useful for thinking about how to support debugging in traditional programming, has explanatory power for how ML developers program. If so, we will need to establish what types of information scent ML developers need to follow in order to improve performance, and fix errors in their ML models.

C. A Matter of Ever-Rising Technical Debt

ML practitioners are struggling. Their reports of the inaccessibility of the necessary skills and tools for "*mere mortals*" (P4 and P5's terminology) echoes Sculley et al.'s position paper on ML's rising technical debt [29]. As they argued, "technical debt does tend to compound," such as in "ratcheting up maintenance costs"—which predicted exactly the kinds of maintenance experiences our participants described. As participant P6 summarized his experiences with such maintenance costs:

"The whole thing is very complicated and rickety."

ACKNOWLEDGMENTS

We thank our participants for their help. This work was supported in part by the National Science Foundation under 1302113 and 1314384. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

`

REFERENCES

[1] Amershi, S., Cakmak, M., Knox, W. B., and Kulesza, T. Power to the people: The role of humans in interactive machine learning. *AI Magazine,* 2014. vol. 35,4, pp. 105-120.

[2] Amershi, S., Fogarty, J., Kapoor, A., and Tan. Examining multiple potential models in end-user interactive concept learning. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '10)*, 2010, pp. 1357-1360.

[3] Bryan, N. J., Mysore, G. J., and Wang, G. ISSE: an interactive source separation editor. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '14), 2014, 257-266.

[4] Cakmak, M., and Thomaz, A. L. Designing robot learners that ask good questions. In *Proceedings of the ACM/IEEE international conference on Human-Robot Interaction* (HRI '12), 2012, pp. 17-24.

[5] Carver, J. C., Kendall, R. P., Squires, S. E., and Post, D. E. Software development environments for scientific and engineering software: A series of case studies. In *Proceedings of the ACM/IEEE International Conference on Software Engineering* (ICSE '07), 2007, pp. 550–559.

[6] Das S., Moore, T., Wong, W., Stumpf, Simone, Oberst, I., McIntosh, K., and Burnett, M. End-user feature labeling: Supervised and semi-supervised approaches based on locally-weighted logistic regression. *Artificial Intelligence,* 2013, vol. 204, pp. 56-74.

[7] Dzindolet, M. T., Peterson, S. A., Pomranky, R. A., Pierce, L. G., and Beck, H.P. The role of trust in automation reliance. *Int. J. Hum.-Comput. Stud.* 2003, vol. 58, 6, pp. 697-718.

[8] Fails, J. A., and Olsen, D. R., Jr. Interactive machine learning. In *Proceedings of the ACM International Conference on Intelligent User Interfaces (IUI '03)*, 2003, pp. 39-45.

[9] Fleming, S. D., Scaffidi. C., Piorkowski, D., Burnett, M. M., Bellamy, R. K. E., Lawrance, J., and Kwan, I. An information foraging theory perspective on tools for debugging, refactoring, and reuse tasks. In *ACM Trans. Software Engineering Methodology,* 2013, vol. 22,2.

[10] Fogarty, J., Tan, D., Kapoor, A., and Winder, S. CueFlik: interactive concept learning in image search. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '08), 2008, pp. 29-38.

[11] Glass, A., McGuinness, D. L., and Wolverton, M. Toward establishing trust in adaptive agents. In *Proceedings of the ACM International Conference on Intelligent User Interfaces* (IUI '08), 2008, pp. 227-236.

[12] Glowacka, D., Ruotsalo, T., Konuyshkova, K., Athukorala, K., Kaski, S., and Jacucci, G. Directing exploratory search: reinforcement learning from user interactions with keywords. In *Proceedings of the ACM International Conference on Intelligent User Interfaces* (IUI '13), 2013, pp. 117-128.

[13] Guo, P. *Software Tools to Facilitate Research Programming*. PhD dissertation. Stanford University, Stanford, CA, 2012.

[14] Hannay, J. E., MacLeod, C., Singer, J., Langtangen, H. P., Pfahl, D., and Wilson, G. How do scientists develop and use scientific software? In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering* (SECSE '09), 2009, pp. 1–8.

[15] Kapoor, A., Lee, B., Tan, D., and Horvitz, E. Interactive optimization for steering machine classification. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '10), 2010, pp. 1343-1352.

[16] Knox, W. B., and Stone, P. Reinforcement learning from human reward: Discounting in episodic tasks. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication*, 2012, pp. 878-885.

[17] Kulesza, T., Burnett, M., Wong, W., and Stumpf, S. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the ACM International Conference on Intelligent User Interfaces* (IUI '15), 2015, pp. 126-137.

[18] Kulesza, T., Stumpf, S., Wong, W., Burnett, M., Perona, S., Ko, A., and Oberst, I. Why-oriented end-user debugging of naive Bayes text classification. *ACM Transactions on Interactive Intelligent Systems*, 2011, vol. 1, 1.

[19] Kulesza, T., Stumpf, S., Burnett, M, and Kwan, I. Tell Me More?: The effects of mental model soundness on personalizing an intelligent agent.

In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '12), 2012, pp. 1–10.

[20] Kulesza, T., Amershi, S., Caruana, R., Fisher, D., and Charles, D. Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (*CHI '14*)*, 2014, pp. 3075-3084.

[21] Lawrance, J., Bogart, C., Burnett, M., Bellamy, R., Rector, K., and Fleming, S. D. How programmers debug, revisited: An information foraging theory perspective, *IEEE Transactions on Software Engineering*, 2013, vol. 39(2), pp. 197-215.

[22] Lim, B. Y., and Dey, A. K. Assessing demand for intelligibility in context-aware applications. In *Proceedings of the ACM International Conference on Ubiquitous Computing* (UbiComp '09), 2009, pp.195-204.

[23] MacLean, D. Provenance, PASS & People: a Research Report. Technical report, Harvard University, 2007.

[24] Parra, D., Brusilovsky, P., and Trattner, C. See what you want to see: Visual user-driven approach for hybrid recommendation. In *Proceedings of the ACM International Conference on Intelligent User Interfaces* (IUI '14), 2014, pp. 235-240.

[25] Patel, K., Fogarty, J., Landay, J. A., and Harrison, B. Investigating statistical machine learning as a tool for software development. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '08), 2008, pp. 667-676.

[26] Piorkowski, D., Fleming, S. D., Scaffidi, C., Burnett, M., Kwan, I., Henley, A. Z., Macbeth, J., Hill, C., and Horvath, A. To fix or to learn? How production bias affects developers' information foraging during debugging, *IEEE International Conference on Software Maintenance and Evolution*, 2015.

[27] Prabhu, P., Jablin, T. B., Raman, A., Zhang, Y., Huang, J., Kim, H., Johnson, N. P., Liu, F., Ghosh, S., Beard, S., Oh, T., Zoufaly, M., Walker, D., and August, D. I. A survey of the practice of computational science. In State of the Practice Reports (SC '11), 2011.

[28] Raghavan. H., Madani, O., and Jones, R. Active learning with feedback on features and instances. *J. Mach. Learn. Res.,* 2006, vol. 7, pp. 1655-1686.

[29] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., and Young, M. Machine Learning: The high-interest credit card of technical debt. In *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, 2014.

[30] Settles, B. Active learning literature survey. Tech. Rep. 1648, University of Wisconsin-Madison, 2010.

[31] Srinivasa Ragavan, S., Kuttal, S. K., Hill, C., Sarma, A., Piorkowski, D., and Burnett, M. Foraging among an overabundance of similar variants, *ACM Conference on Human Factors in Computing Systems (CHI)*, 2016 (to appear).

[32] Stemler, S. and Tsai, J. 3 Best Practices in Interrater Reliability: Three Common Approaches. In Best Practices in Quantitative Methods, SAGE Publications, 2008.

[33] Strong, C., Jones, S., Parker-Wood, A., Holloway, A., and Long, D. D. E. Los Alamos National Laboratory Interviews. Technical Report UCSC-SSRC-11-06, UC Santa Cruz, 2011.

[34] Stumpf, S., Rajaram, V., Li, L., Burnett, M., Dietterich T., Sullivan, E, Drummond, R., and Herlocker, J. Toward harnessing user feedback for machine learning. In *Proceedings of the ACM International Conference on Intelligent User Interfaces* (IUI '07), 2007, pp. 82-91.

[35] Stumpf, S., Rajaram, V., Li, L., Wong, W., Burnett, M., Dietterich, T., Sullivan, E, and Herlocker, J. Interacting meaningfully with machine learningsystems: Three experiments. *International Journal of Human-Computer Studies,* 2009, vol. 67, 8, pp. 639–662.

[36] Tullio, J., Dey, A. K., Chalecki, J., and Fogarty, J. How it works: A field study of non-technical users interacting with an intelligent system. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '07), 2007, pp. 31–40.

[37] Vig, J., Sen, S., and Riedl, J. Navigating the tag genome. In *Proceedings of the ACM International Conference on Intelligent User Interfaces* (IUI '11), 2011, pp. 93–102.

[38] Wikipedia, Feature (machine learning). https://en.wikipedia.org/wiki/Feature_(machine_learning). Accessed Feb. 13, 2016.

`