Today's programming tools and languages typically allow for interactions that go in one direction: a programmer expresses their intent, and a program executes. My research in human-computer interaction (HCI) explores what happens when we make these interactions bidirectional, creating tighter feedback loops between languages and developers. For example, a language might engage in dialog with its programmer to determine the function they want to call, or proactively warn a user when a function they are using has encountered a bug in someone else's code. By combining such agency with new kinds of intelligence—drawn from crowds and communities—my research enables a new class of programming systems that work with users interactively to solve their problems.

CONVERSATIONAL AGENTS FOR PROGRAMMING

My work with Iris demonstrates how users can construct and execute programs through conversation, interacting with an agent to perform complex data science tasks [1]. Back-and-forth dialog allows a system to better understand user intent through two main ideas. First, it allows programmers to ask for what they want at a high level, bridging the vocabulary problem to connect with underlying technical specifications. And second, it allows agents to directly ask back for any missing or ambiguous or details.

Modeling useful programs through conversation is challenging, however, as you must support complex operations such as composition. For example, consider how a conversational agent should decompose, "I want to do a t-test on the log-transform of the education data". While semantic parsing or deep learning offer the potential to construct a parse tree for this statement directly, these techniques require large amounts of training data and are computationally expensive. Iris instead offers an insight inspired by conversation analysis theory: that we can replace such automatic decomposition with sub-conversation to more scalably enable the same set of compositions. For example, a user can first say, "I want to do a t-test", and when Iris asks, "what are the values you want to analyze?", the user can follow up a nested command for Iris to "take the log-transform of the education data".

Iris is loaded with several hundred primitive functions that enable thousands of composed commands. In a user study, we found that programmers were able to build and evaluate a predictive model more than two times faster using Iris than a Jupyter Notebook pre-loaded with the necessary functions. Today, Iris has entered private deployment across Stanford University, and is being used to develop new bioinformatics tools in the Altman Lab at the Stanford School of Medicine.

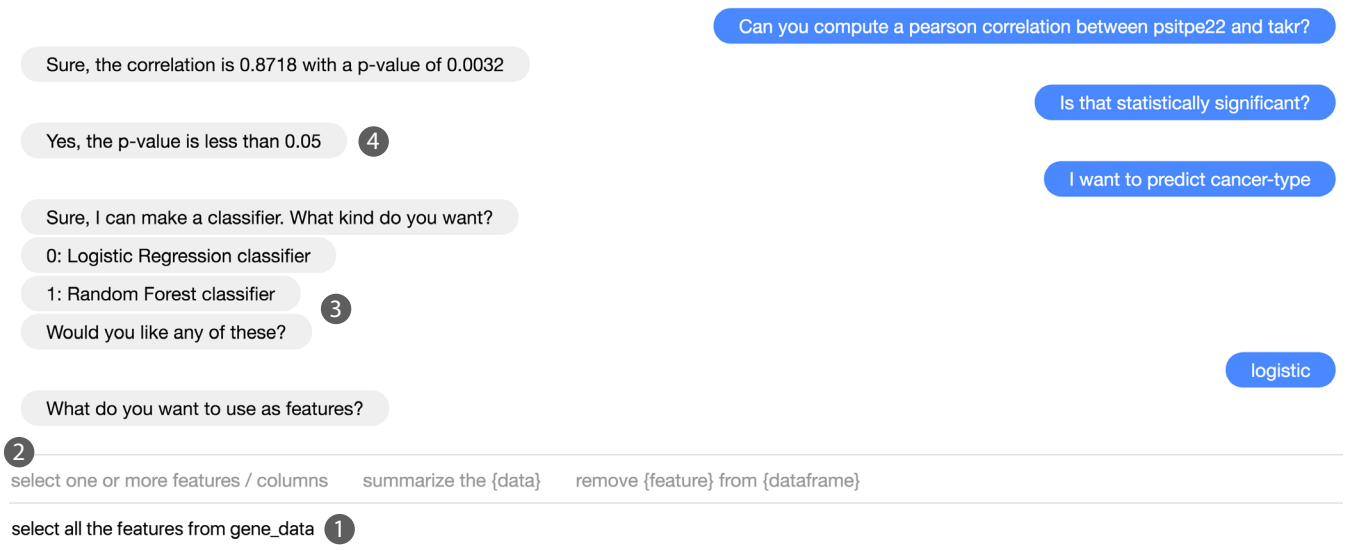

Figure 1: Iris allows users to combine commands through nested conversations to accomplish open-ended data science tasks. (1) Users interact with Iris through natural language requests and (2) the system responds with real-time feedback on the command the request will trigger. Once a command is triggered, Iris (3) converses with users to resolve arguments. When resolving arguments, users can initiate a nested conversation via a new command, or reference the result of previous conversation.

COMMUNITY-POWERED PROGRAMMING LANGUAGES

An enormous amount of community knowledge remains uncodified and inaccessible to programming tools and languages. I have built systems that explore how we can draw on this knowledge—via static and dynamic analyses—to provide programmers with more nuanced and proactive feedback about their code.

My work with Codex shows how static analysis of open source code can inform a statistical linting tool that warns users about code that deviates from the community [2]. For example, Codex can determine a variable named `my_hash` should probably not be assigned a list value, or that a method that ends with the bang operator should not be on the right side of an assignment statement (in Ruby, the bang operator indicates a method that performs an in-pace operation and always returns `None`). These analyses are enabled by a model of surprisingly unlikely code: by analyzing which lexical tokens or abstract syntax tree nodes are commonly combined together over millions of lines of code, it is possible to identify combinations that are surprisingly rare given the commonality of their components (and so likely worth reconsidering). An investigation of this approach across thousands of lines of gold standard test code found a conservative false positive rate of 1.5%, which can be lowered arbitrarily via a threshold that reflects a user's preference for error. Beyond simply finding these patterns, Codex further demonstrates how they can be automatically annotated by expert crowdworkers to enable a self-documenting knowledge base.

My work with Meta further examines how dynamic analyses can be applied to help a community of programmers [3]. By instrumenting a programming language such that every function is indexed in a centralized database, along with a growing set of run time data, Meta demonstrates how functions can proactively notify users when a more optimized version is written by someone else in the community, or warn users when a function that one person is executing encounters a bug in someone else's code. Meta has found 44 real code optimizations and 5 bug fixes by instrumenting and analyzing 4000 popular functions on Stackoverflow. One technical challenge when collecting so much run time data, however, is managing the overhead introduced by the instrumentation. In a user study, we show how a probabilistic approach to sampling run time data can limit this overhead to 0.31 milliseconds per function call.

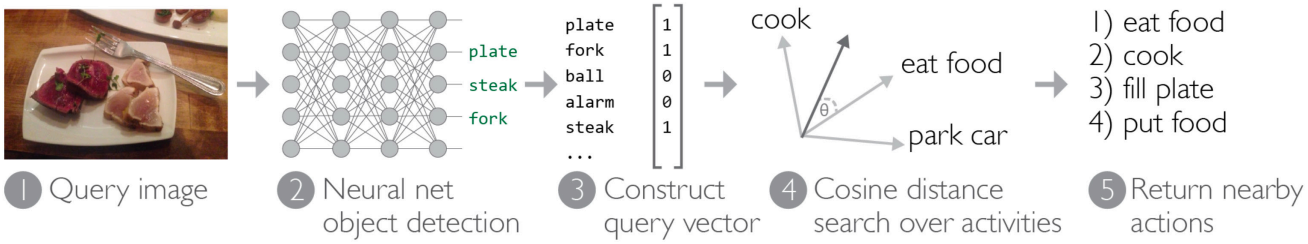

Figure 2: Augur mines a large vector space of human behaviors from fiction. Here, Augur's APIs map input images through a deep learning object detector, then initializes the returned objects into a query vector. Augur then compares that vector to the vectors representing each activity in its vector space and returns those with lowest cosine distance.

MINING HUMAN BEHAVIOR FOR INTERACTIVE SYSTEMS

From smart homes that prepare coffee when we wake, to phones that know not to interrupt us during important conversations, our foundational HCI visions present a future in which computers understand a broad range of human behaviors. Today's systems fall short of these visions, however, because this range of behaviors is too large for designers or programmers to capture manually. My work with Augur, which was awarded an Honorable Mention at CHI, constructs a knowledge base of human behavior by analyzing more than one billion words of modern fiction [4]. The resulting knowledge base enables an API that can predict human behavior, for example `predict_next("wake up") => "make coffee"`, and react to such behaviors, for example `if_predict("spend money"){ show_bank_balance() }`.

Augur trains vector space models that can predict thousands of user activities from surrounding objects in modern contexts: for example, whether a user may be eating food, meeting with a friend, or taking a selfie. These models are based on smoothed co-occurrence statistics between object and activity phrases

within paragraphs of hundreds of thousands of fiction stories. For example, such statistics produce an object-to-activity space which can be searched for activity predictions by constructing a query vector using the dimensions of related objects. Using a similar approach, we can construct an activity-to-activity space that predicts future activities. Augur uses these predictions to identify actions that people commonly take on objects in the world and estimate a user's future activities given their current situation. In evaluation, we demonstrate Augur-powered, activity-based systems such as a phone that silences itself when the odds of you answering it are low, and a dynamic music player that adjusts to your present activity. A field deployment of an Augur-powered wearable camera resulted in 96% recall and 71% precision on its unsupervised predictions of common daily activities.

COMPUTATIONAL SOCIAL SCIENCE

I also publish tools and analyses in computational social science that reflect my interest in broader social questions. For example, my work on Empath, a tool that empowers other researchers to create new lexicons for text analysis, won Best Paper at CHI, is currently in use by Facebook Data Science, and has been cited more than 30 times since it was published last year [5]. I leverage this tool-building work to inform other, more targeted analyses. For example, I have investigated gender bias and dogmatism in online communities, with an eye towards understanding how we can construct interfaces that correct for such effects [6, 7]. I have also focused on questions that may become important driver's of public policy, such as trends in the public perception of AI, as reflected though decades of articles in the New York Times [8].

BLOCKCHAIN DEVELOPMENT WITH CROWDSOURCING

Crowdsourced and decentralized organizations can be a powerful tool for achieving large-scale projects and ecosystem development. In the summer of 2017, I co-founded an open source community, CoZ, incentivized by monetary rewards on the NEO blockchain [9]. This community has since harnessed the contributions of hundreds of developers across the world to build open source tools and software used by tens of thousands of users: for example, a wallet for managing NEO assets, a community block explorer, and a compiler that allows programmers to write NEO smart contracts in Python. I am currently writing an account of my experiences bootstrapping this decentralized organization, which I plan to publish in CSCW 2018. As a platform for ongoing research, CoZ provides a large pool of users and developers that I can leverage for future work in blockchain technology and decentralized organizations.

RESEARCH AGENDA

I will continue to work at the intersection of human-computer interaction, programming languages, and natural language processing. In this section, I outline several ideas that I am interested in pursuing:

Can interactive systems learn and react to higher level user goals? Iris provides a platform for mapping user language onto low level goals such as individual function calls. However, a user still needs to understand the structure of the task they are attempting to complete. For example, to build a statistical model, first you call a function to load some data, then train the model, and so on. Can a conversational agent map a user's interactions onto some higher level goal, and use that understanding to inform what they are likely to want to do next? Reinforcement learning offers one approach to investigating this idea.

What are the limits of existing lexicons? Empath shows that tools that generate lexicons can have an enormous impact on computational social science, but today's lexicons are still not capable of capturing word order or combination effects that can dramatically change the meaning of text (e.g., "he was not angry" signals *anger* in Empath and LIWC). More sophisticated sequence embedding models such as LSTMs can capture and correct for these ordering effects, but it is an open question how to best apply these models to lexical analysis, and enhance their interpretability for users familiar with simple word lists.

How will we program our future financial infrastructure? Smart contracts on a blockchain offer a new way of encoding complex financial transactions, both among public users and within private organizations. However, current languages for programming such contracts are error-prone and have led to mistakes that have cost users and organizations hundreds of million of dollars. How can we design programming languages and APIs that maintain usability, while providing much stronger safety guarantees against unwanted behavior? Dependent type systems provide one way of formally reasoning about smart contract

functionality and would help to reduce such errors, but such powerful type systems have traditionally been very difficult for non-experts to use. Is it possible to create a DSL that exposes this functionality in a way more interpretable by end users?

REFERENCES

[1] **Ethan Fast**, Binbin Chen, Julia Mendelsohn, Jon Bassen, Michael Bernstein. *Iris: A Conversational Agent for Complex Tasks*. CHI: Conference on Human Factors in Computing Systems, 2018 (to appear)

[2] **Ethan Fast**, Daniel Steffee, Lucy Wang, Joel Brandt, Michael Bernstein. *Emergent, Crowd-scale Programming Practice in the IDE*. CHI: Conference on Human Factors in Computing Systems, 2014

[3] **Ethan Fast**, Michael Bernstein. *Meta: Enabling Programming Languages to Learn from the Crowd*. UIST: User Interface Software and Technology, 2016

[4] **Ethan Fast**, Will McGrath, Pranav Rajpurkar, Michael Bernstein. *Augur: Mining Human Behaviors from Fiction to Power Interactive Applications*. CHI: Conference on Human Factors in Computing Systems, 2016

[5] **Ethan Fast**, Binbin Chen, Michael Bernstein. Empath: *Understanding Topic Signals in Large-Scale Text*. CHI: Conference on Human Factors in Computing Systems, 2016

[6] **Ethan Fast**, Eric Horvitz. *Identifying Dogmatism in Social Media: Signals and Models*. EMNLP: Empirical Methods in Natural Language Processing, 2016

[7] **Ethan Fast**, Tina Vachovsky, Michael Bernstein. *Shirtless and Dangerous: Quantifying Linguistic Signals of Gender Bias in an Online Fiction Writing Community*. ICWSM: International AAAI Conference on Web and Social Media, 2016

[8] **Ethan Fast**, Eric Horvitz. *Long-Term Trends in the Public Perception of Artificial Intelligence*. AAAI: Conference on Artificial Intelligence, 2017

[9] CoZ Github Organization. https://github.com/CityOfZion