

COMMUNICATIONS

CACM.ACM.ORG

OF THE

08/2019 VOL.62 NO.08

ACM

Embedded EthiCS Integrating Ethics Broadly Across CS Education

Activity-Centric
Computing Systems

Scaling Static Analyses
at Facebook

What Makes a Robot Likeable?

Association for
Computing Machinery



Hardness of Approximation Between P and NP

Nash equilibrium is the central solution concept in Game Theory. Since Nash's original paper in 1951, it has found countless applications in modeling strategic behavior of traders in markets, (human) drivers and (electronic) routers in congested networks, nations in nuclear disarmament negotiations, and more. A decade ago, the relevance of this solution concept was called into question by computer scientists, who proved (under appropriate complexity assumptions) that computing a Nash equilibrium is an intractable problem. And if centralized, specially designed algorithms cannot find Nash equilibria, why should we expect distributed, selfish agents to converge to one? The remaining hope was that at least approximate Nash equilibria can be efficiently computed.

Understanding whether there is an efficient algorithm for approximate Nash equilibrium has been the central open problem in this field for the past decade. In this book, we provide strong evidence that even finding an approximate Nash equilibrium is intractable. We prove several intractability theorems for different settings (two-player games and many-player games) and models (computational complexity, query complexity, and communication complexity). In particular, our main result is that under a plausible and natural complexity assumption ("Exponential Time Hypothesis for PPAD"), there is no polynomial-time algorithm for finding an approximate Nash equilibrium in two-player games.

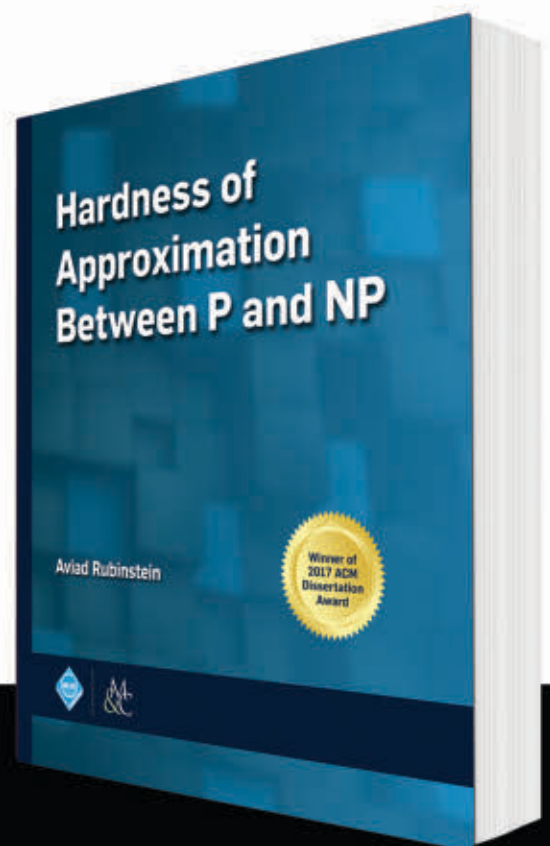
2017 ACM Dissertation Award Winner

Aviad Rubinstein

ISBN: 978-1-947487-20-8

DOI: 10.1145/3241304

<http://books.acm.org>



ACM BOOKS
Collection 1



**SIGGRAPH
ASIA 2019
BRISBANE**

The 12th ACM SIGGRAPH Conference
and Exhibition on Computer Graphics
and Interactive Techniques in Asia

DREAM ZONE!

Conference 17 - 20 November 2019
Exhibition 18 - 20 November 2019

Brisbane Convention & Exhibition Centre (BCEC),
Brisbane, Australia

Sponsored by:



Organized by



we energize your business | since 1924

Departments

- 5 **From the President**
Dispelling Common Myths About ACM Awards and Honors
By Cherri M. Pancake
-
- 7 **Cerf's Up**
Undo, Redo, and Regrets
By Vinton G. Cerf
-
- 9 **Letters to the Editor**
A Case Against Mission-Critical Applications of Machine Learning
-
- 12 **BLOG@CACM**
Cutting the Wait For CS Advice
 Mark Guzdial suggests ways to cut the long lines for college students seeking to meet with their computer science advisors.
-
- 29 **Calendar**
-
- 101 **Careers**

Last Byte

- 104 **Future Tense**
Fluid Democracy
 In trying to “drown” the opposition with daily online elections, I didn’t realize they could wash me away.
By William Sims Bainbridge

News



- 15 **The Algorithm that Changed Quantum Machine Learning**
 A college student discovered a classical computing algorithm that experts overlooked. It promises to change both classical and quantum machine learning.
By Samuel Greengard
-
- 18 **I Don’t Understand My Car**
 Self-driving cars will need good communication skills.
By Don Monroe
-
- 20 **What Makes a Robot Likable?**
 Interactions with robotics teach us more about people.
By Gregory Mone

Viewpoints



- 22 **Education**
Block-based Programming in Computer Science Education
 Considering how block-based programming environments and tools might be used at the introductory level and beyond.
By David Weintrop
-
- 26 **Economic and Business Dimensions**
A Response to Fake News as a Response to *Citizens United*
 How boundaries on speech could free the market for speech.
By Marshall W. Van Alstyne
-
- 30 **Kode Vicious**
MUST and MUST NOT
 On writing documentation.
By George V. Neville-Neil
-
- 32 **Viewpoint**
The Success of the Web: A Triumph of the Amateurs
 Connecting the unique factors that influenced the origination and subsequent development of the World Wide Web.
By Marco Aiello

Practice

36 **Industry-Scale Knowledge Graphs: Lessons and Challenges**

Five diverse technology companies show how it's done.

By *Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor*

44 **Research for Practice: The DevOps Phenomenon**

An executive crash course.

By *Anna Wiedemann, Nicole Forsgren, Manuel Wiesche, Heiko Gewalt, and Helmut Krcmar*

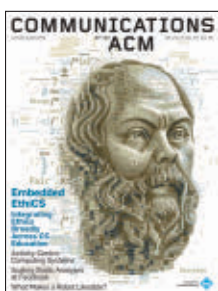
50 **Overly Attached**

Know when to let go of emotional attachment to your work.

By *Kate Matsudaira*



Articles' development led by **acmqueue**
queue.acm.org



About the Cover:
The illustration of Socrates, considered a founding father of ethics philosophy, was inspired by this month's cover story (p. 54) about a Harvard-based pilot program that integrates class sessions on ethical reasoning throughout its CS curriculum. Cover illustration by Charis Tsevis.

Contributed Articles

54 **Embedded Ethics: Integrating Ethics Across CS Education**

A Harvard-based pilot program integrates class sessions on ethical reasoning into courses throughout its computer science curriculum.

By *Barbara J. Grosz, David Gray Grant, Kate Vredenburg, Jeff Behrends, Lily Hu, Alison Simmons, and Jim Waldo*



Watch the authors discuss this work in the exclusive *Communications* video.
<https://cacm.acm.org/videos/embedded-ethics>

62 **Scaling Static Analyses at Facebook**
Key lessons for designing static analyses tools deployed to find bugs in hundreds of millions of lines of code.

By *Dino Distefano, Manuel Fähndrich, Francesco Logozzo, and Peter W. O'Hearn*

Review Articles

72 **Activity-Centric Computing Systems**
The ability to build a construct that organizes work from different devices and information resources is as complex as it is invaluable.

By *Jakob E. Bardram, Steven Jeuris, Paolo Tell, Steven Houben, and Stephen Volda*



Watch the authors discuss this work in the exclusive *Communications* video.
<https://cacm.acm.org/videos/activity-centric-computing-systems>

82 **The History of Digital Spam**

Tracing the tangled web of unsolicited and undesired email and possible strategies for its demise.

By *Emilio Ferrara*

Research Highlights

94 **Technical Perspective****The True Cost of Popularity**

By *Graham Cormode*

95 **Heavy Hitters via****Cluster-Preserving Clustering**

By *Kasper Green Larsen, Jelani Nelson, Huy L. Nguyễn, and Mikkel Thorup*



Association for Computing Machinery
Advancing Computing as a Science & Profession



ACM, the world's largest educational and scientific computing society, delivers resources that advance computing as a science and profession. ACM provides the computing field's premier Digital Library and serves its members and the computing profession with leading-edge publications, conferences, and career resources.

Executive Director and CEO

Vicki L. Hanson

Deputy Executive Director and COO

Patricia Ryan

Director, Office of Information Systems

Wayne Graves

Director, Office of Financial Services

Darren Ramdin

Director, Office of SIG Services

Donna Cappel

Director, Office of Publications

Scott E. Delman

ACM COUNCIL

President

Cherri M. Pancake

Vice-President

Elizabeth Churchill

Secretary/Treasurer

Yannis Ioannidis

Past President

Alexander L. Wolf

Chair, SGB Board

Jeff Jortner

Co-Chairs, Publications Board

Jack Davidson and Joseph Konstan

Members-at-Large

Gabriele Anderst-Kotis; Susan Dumais;

Renée McCauley; Claudia Bauzer Medeiros;

Elizabeth D. Mynatt; Pamela Samuelson;

Theo Schlossnagle; Eugene H. Spafford

SGB Council Representatives

Sarita Adve and Jeanna Neefe Matthews

BOARD CHAIRS

Education Board

Mehran Sahami and Jane Chu Prey

Practitioners Board

Terry Coatta

REGIONAL COUNCIL CHAIRS

ACM Europe Council

Chris Hankin

ACM India Council

Abhiram Ranade

ACM China Council

Wenguang Chen

PUBLICATIONS BOARD

Co-Chairs

Jack Davidson and Joseph Konstan

Board Members

Phoebe Ayers; Chris Hankin; Xiang-Yang Li;

Nenad Medvidovic; Tulika Mitra;

Sue Moon; Michael L. Nelson;

Sharon Oviatt; Eugene H. Spafford;

Stephen N. Spencer; Divesh Srivastava;

Robert Walker; Julie R. Williamson

ACM U.S. Technology Policy Office

Adam Eisgrau,

Director of Global Policy and Public Affairs

1701 Pennsylvania Ave NW, Suite 200,

Washington, DC 20006 USA

T (202) 580-6555; acmpo@acm.org

Computer Science Teachers Association

Jake Baskin

Executive Director

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

Communications of the ACM is the leading monthly print and online magazine for the computing and information technology fields. *Communications* is recognized as the most trusted and knowledgeable source of industry information for today's computing professional. *Communications* brings its readership in-depth coverage of emerging areas of computer science, new trends in information technology, and practical applications. Industry leaders use *Communications* as a platform to present and debate various technology implications, public policies, engineering challenges, and market trends. The prestige and unmatched reputation that *Communications of the ACM* enjoys today is built upon a 50-year commitment to high-quality editorial content and a steadfast dedication to advancing the arts, sciences, and applications of information technology.

STAFF

DIRECTOR OF PUBLICATIONS

Scott E. Delman

cacm-publisher@cacm.acm.org

Executive Editor

Diane Crawford

Managing Editor

Thomas E. Lambert

Senior Editor

Andrew Rosenbloom

Senior Editor/News

Lawrence M. Fisher

Web Editor

David Roman

Editorial Assistant

Danbi Yu

Art Director

Andrij Borys

Associate Art Director

Margaret Gray

Assistant Art Director

Mia Angelica Balaquiot

Production Manager

Bernadette Shade

Intellectual Property Rights Coordinator

Barbara Ryan

Advertising Sales Account Manager

Ilia Rodriguez

Columnists

David Anderson; Michael Cusumano;

Peter J. Denning; Mark Guzdial;

Thomas Haigh; Leah Hoffmann; Mari Sako;

Pamela Samuelson; Marshall Van Alstyne

CONTACT POINTS

Copyright permission

permissions@hq.acm.org

Calendar items

calendar@cacm.acm.org

Change of address

acmhelp@acm.org

Letters to the Editor

letters@cacm.acm.org

WEBSITE

<http://cacm.acm.org>

WEB BOARD

Chair

James Landay

Board Members

Marti Hearst; Jason I. Hong;

Jeff Johnson; Wendy E. MacKay

AUTHOR GUIDELINES

<http://cacm.acm.org/about-communications/author-center>

ACM ADVERTISING DEPARTMENT

1601 Broadway, 10th Floor

New York, NY 10019-7434 USA

T (212) 626-0686

F (212) 869-0481

Advertising Sales Account Manager

Ilia Rodriguez

ilia.rodriguez@hq.acm.org

Media Kit acmm mediasales@acm.org

Association for Computing Machinery (ACM)

1601 Broadway, 10th Floor

New York, NY 10019-7434 USA

T (212) 869-7440; F (212) 869-0481

EDITORIAL BOARD

EDITOR-IN-CHIEF

Andrew A. Chien

eic@cacm.acm.org

Deputy to the Editor-in-Chief

Lihan Chen

cacm.deputy.to.eic@gmail.com

SENIOR EDITOR

Moshe Y. Vardi

NEWS

Co-Chairs

Marc Snir and Alain Chesnais

Board Members

Monica Divitini; Mei Kobayashi;

Rajeev Rastogi; François Sillion

VIEWPOINTS

Co-Chairs

Tim Finin; Susanne E. Hambrusch;

John Leslie King; Paul Rosenbloom

Board Members

Michael L. Best; Judith Bishop;

James Grimmelmann; Mark Guzdial;

Haym B. Hirsch; Richard Ladner;

Carl Landwehr; Beng Chin Ooi;

Francesca Rossi; Len Shustek; Loren Terveen;

Marshall Van Alstyne; Jeannette Wing;

Susan J. Winter

PRACTICE

Co-Chairs

Stephen Bourne and Theo Schlossnagle

Board Members

Eric Allman; Samy Bahra; Peter Bailis;

Betsy Beyer; Terry Coatta; Stuart Feldman;

Nicole Forsgren; Camille Fournier;

Jessie Frazelle; Benjamin Fried; Tom Killalea;

Tom Limoncelli; Kate Matsudaira;

Marshall Kirk McKusick; Erik Meijer;

George Neville-Neil; Jim Waldo;

Meredith Whittaker

CONTRIBUTED ARTICLES

Co-Chairs

James Larus and Gail Murphy

Board Members

William Aiello; Robert Austin; Kim Bruce;

Alan Bundy; Peter Buneman; Jeff Chase;

Andrew W. Cross; Yannis Ioannidis;

Gal A. Kaminka; Ben C. Lee; Igor Markov;

Lionel M. Ni; Adrian Perrig; Doina Precup;

Marie-Christine Rousset; Krishan Sabnani;

m.c. schraefel; Ron Shamir; Alex Smola;

Sebastian Uchitel; Hannes Werthner;

Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs

Azer Bestavros and Shriram Krishnamurthi

Board Members

Martin Abadi; Amr El Abbadi;

Animashree Anandkumar; Sanjeev Arora;

Michael Backes; Maria-Florina Balcan;

David Brooks; Stuart K. Card; Jon Crowcroft;

Alexei Efros; Bryan Ford; Alon Halevy;

Gernot Heiser; Takeo Igarashi; Sven Koenig;

Greg Morrisett; Tim Roughgarden;

Guy Steele, Jr.; Robert Williamson;

Margaret H. Wright; Nicolai Zeldovich;

Andreas Zeller

SPECIAL SECTIONS

Co-Chairs

Sriram Rajamani, Jakob Rehof,

and Haibo Chen

Board Members

Tao Xie; Kenjiro Taura; David Padua

ACM Copyright Notice

Copyright © 2019 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@hq.acm.org or fax (212) 869-0481.

For other copying of articles that carry a code at the bottom of the first or last page or screen display, copying is permitted provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center; www.copyright.com.

Subscriptions

An annual subscription cost is included in ACM member dues of \$99 (\$40 of which is allocated to a subscription to *Communications*); for students, cost is included in \$42 dues (\$20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is \$269.

ACM Media Advertising Policy

Communications of the ACM and other ACM Media publications accept advertising in both print and electronic formats. All advertising in ACM Media publications is at the discretion of ACM and is intended to provide financial support for the various activities and services for ACM members. Current advertising rates can be found by visiting <http://www.acm-media.org> or by contacting ACM Media Sales at (212) 626-0686.

Single Copies

Single copies of *Communications of the ACM* are available for purchase. Please contact acmhelp@acm.org.

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 1601 Broadway, 10th Floor New York, NY 10019-7434 USA. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to *Communications of the ACM* 1601 Broadway, 10th Floor New York, NY 10019-7434 USA

Printed in the USA.



Association for
Computing Machinery



DOI:10.1145/3339362

Cherri M. Pancake

Dispelling Common Myths About ACM Awards and Honors

HAVE YOU WONDERED why a person you admire has not received an ACM award, or been made a Distinguished Member or Fellow? We all want to see our colleagues recognized for their achievements. When an outstanding candidate has not received recognition, the reason is usually one of three things: they were not nominated at all, the nomination did not adequately address the committee's criteria, or this particular honor was not a good match for the candidate's background.

As a former ACM Awards Chair, I'd like to share some insights to help you understand what makes nominations effective. You will find similar—and additional—information on individual award Web pages, under “How to Nominate” and “Frequently Asked Questions.”

Myth: The committee knows X's work already, so why hasn't she/he gotten the award? *Reality:* Even if the entire committee knows how great X is (which is probably a stretch, given that ACM committees must span such a broad range of expertise), that isn't enough. ACM requires a winner be chosen on the basis of the nomination packet that was submitted. If X was not formally nominated, or if the packet did not address the selection criteria, they cannot be selected.

Myth: I work with X, so I am the best person to nominate her/him. *Reality:* Nominations/endorsements are much stronger when they come from people outside the candidate's own organization. Not only can they speak to the broader impact of the work, they are also considered to be more objective since the writer won't derive any benefit if the candidate is successful. Why not help organize it instead by

finding a nominator, providing background information, and perhaps helping to ensure the endorsements span a range of perspectives?

Myth: The more famous the nominator/endorsers are, the better. *Reality:* What the person says is more important than who the person is. It's also important to have endorsers address the selection criteria in different ways, rather than just reiterating the nominator's comments. That said, the same words will carry more weight when they are from a person with stature in the community, as opposed to someone who may not have as much experience to draw on.

Myth: Nominations from ACM Fellows are always successful, but I don't know any to ask. *Reality:* There's no requirement that a nominator/endorser be a Fellow or even an ACM member. It's often best to approach people who have cited or commented publicly on the quality of the candidate's work, even if you don't know them.

Myth: They never give this honor to people working outside North America. *Reality:* On the contrary, nothing makes our committees happier than to recognize achievements of people from around the globe. The sad truth is there are very few nominations from

outside North America, and some of those suffer from the problems described in the other myths.

Myth: It was too competitive this year; we will just resubmit the same packet again next year. *Reality:* Think of it this way: leaving the packet unchanged implies the candidate didn't achieve anything new in the intervening year.

Myth: X does wonderful work, so let's go for the biggest award. *Reality:* The reason there are so many different awards/honors is that excellence occurs in different ways as one's career progresses. Choose a target that not only reflects the candidate's accomplishments, but also their career stage. There are few hard-and-fast rules, but the table here might be helpful to you.

Remember that ACM awards only exist because people like you identify and nominate outstanding individuals. For more details on specific awards and selection criteria, visit <https://awards.acm.org/award-nominations>. □

Cherri M. Pancake is President of ACM, professor emeritus of electrical engineering and computer science, and director of a research center at Oregon State University, Corvallis, OR, USA.

Copyright held by author/owner.

ACM offers a wide range of awards.

Students	HPC Fellowships, Cutler-Bell Prize
Early Career	Hopper Award, ACM and SIG dissertation awards (recent graduates), SIG “rising star” awards; also Senior Member
Mid Career	ACM Prize; also Distinguished Member
Late Career	Turing Award, Distinguished Service Award; also Fellow
Area-Specific, typically Mid to Late Career	Thacker Award, Software System Award, Athena Lecturer, Newell Award, Lawler Award, Eckert-Mauchly Award, Kennedy Award, Bell Prize, Karlstrom Award, Kannellakis Award, Policy Award, SIAM/ACM Prize—plus dozens of awards from individual SIGs

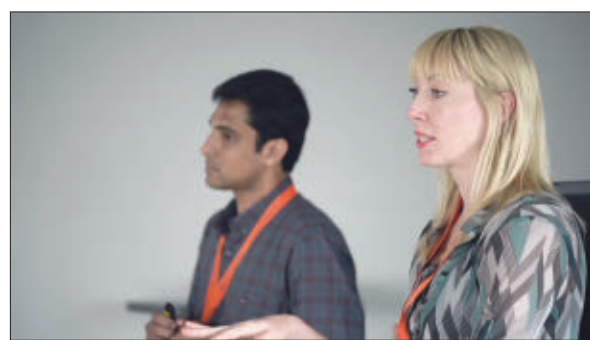


APPLY FOR MEMBERSHIP IN THE ACM FUTURE OF COMPUTING ACADEMY

Are you early in your computing career, with a passion for improving the field? Are you excited about the joy, sacrifice, and professional benefits of becoming a leader in the computing community? If so, you should apply to join the ACM Future of Computing Academy (ACM FCA) — a new initiative where early-career researchers, practitioners, educators, and entrepreneurs can develop a strong and influential collective voice to help shape the future of ACM and the computing community.

ACM FCA was created to foster the next generation of ACM leaders and help shape the future directions of ACM. As part of that leadership development, it defines and implements pilot projects that address challenging issues facing the organization, the field, and society in general. The focus is on harnessing collective volunteer action to pilot new initiatives that can carry ACM into the future. At the same time, FCA members interact with ACM leaders and learn about or participate in a range of existing initiatives. Thus, FCA members have the opportunity to perform valuable services to the community while expanding their professional networks and leadership experience.

The deadline for applications is 23 August 2019.



APPLY NOW!

<https://www.acm.org/fca>



Association for
Computing Machinery



DOI:10.1145/3342707

Vinton G. Cerf

Undo, Redo, and Regrets

THIS MONTH I have been mulling over a number of conversations and experiences that I would like to share with you. They all revolve around the notions that some actions or decisions are irreversible and thus deserve considerably more attention than choices or actions that can be undone. I am a regular Apple MacBook Pro user and one of the features I like the most is the Command-Z key that, most of the time, can undo something I just did and wish I hadn't. Unfortunately, not all things in life have an attached Command-Z button.

Ismail Serageldin, the founding director of the new Library of Alexandria^a explained it this way. "There are some solutions I call 'toothpaste solutions' because once you put them into operation, like toothpaste, you can't put them back in the tube." This reminds me of the story of the magic lamp and its resident genie. Once you rub the lamp, the genie escapes and you can't put him back again. Perhaps Pandora's Box^b has similar features—once opened, the ills of the world escape and resist recapture into the box.

This suggests to me that our increasingly digital, software-driven world has many opportunities to exhibit these features. There are software updates that, once done, cannot be undone. As much as I wish all actions were reversible, they aren't. I suppose there is something about entropy that drives this one-way direc-

tion. Unscrambling an egg provides a concrete example. For sufficiently powerful cryptographic methods, the loss of a cryptographic key spells doom for any attempt to decrypt the content. This should be a major concern for users who rely on cryptography to support confidentiality. Key management becomes a crucial matter thanks to the consequences of losing the key(s).

Judy Estrin draws another example of things you might not be able to undo. "Sorry, we broke democracy." I think of artificial intelligence and machine learning: "Sorry, the AI-based self-flying airplane did not work and we can't revive the fatalities."

In our increasingly complex digital world, these features (hazards, phenomena) deserve our utmost attention to minimize the irretrievable consequences our software systems might introduce. More than once, I have experienced the "software update" that has run amuck and proves resistant to being put back into a known and safe state.

There are software updates that, once done, cannot be undone. As much as I wish all actions were reversible, they aren't.

Software backup practices can easily fall into this category. Suppose you religiously back up your primary disk to a backup disk—but the backup fails and you have partially overwritten the previous backup. Then the primary disk fails. You have no complete backup. Ready. Aim at big toe. Fire.

It occurs to me that we would do ourselves a great favor if we were to design our digital systems to the maximum extent possible to avoid irreversible traps to fall into. Software upgrades offer examples for good practices: Backup the system disk *before* doing an upgrade so you can recover if the upgrade fails or produces some unacceptable consequence. Of course, this raises a fundamental question about backups. Are they complete? Do you ever test the backup system before it is actually needed to ensure the backup is complete? I have heard many horror stories of people who have religiously performed the backup incantation only to discover that—in reality—the backup system was not doing a thing, or was only saving some but not all of everything that might be needed to recover from a serious disk failure. Testing the backup system is just as important as carrying out the procedure itself.

Ronald Reagan's famous quote seems appropriate here: "доверяй, но проверяй."^c Reagan said this with regard to strategic arms limitations and reductions, but it seems to apply equally well to software systems. 

^c Translation: "Trust but verify."

Vinton G. Cerf is vice president and Chief Internet Evangelist at Google. He served as ACM president from 2012–2014.

Copyright held by author.

^a <http://bibalex.org/en/default>

^b In classical mythology, Zeus gave Pandora, the first woman, a box with strict instructions that she not open it. Pandora's curiosity soon got the better of her, and she opened the box. All the evils and miseries of the world flew out to afflict mankind.

ACM Transactions on Computing for Healthcare (HEALTH)

Open for
Submissions

A multidisciplinary journal for
high-quality original work on how
computing is improving healthcare



Computing for Healthcare has emerged as an important and growing research area. By using smart devices, the Internet of Things for health, mobile computing, machine learning, cloud computing and other computing based technologies, computing for healthcare can improve the effectiveness, efficiency, privacy, safety, and security of healthcare (e.g., personalized healthcare, preventive healthcare, ICU without walls, and home hospitals).

ACM Transactions on Computing for Healthcare (HEALTH) is the premier journal for the publication of high-quality original research papers, survey papers, and challenge papers that have scientific and technological results pertaining to how computing is improving healthcare. This journal is multidisciplinary, intersecting CS, ECE, mechanical engineering, bio-medical engineering, behavioral and social science, psychology, and the health field, in general. All submissions must show evidence of their contributions to the computing field as informed by healthcare. We do not publish papers on large pilot studies, diseases, or other medical assessments/results that do not have novel computing research results. Datasets and other artifacts needed to support reproducibility of results are highly encouraged. Proposals for special issues are encouraged.

For more
information
and to submit
your work,
please visit:

health.acm.org



Association for
Computing Machinery

DOI:10.1145/3332409

A Case Against Mission-Critical Applications of Machine Learning

IN THEIR COLUMN “Learning Machine Learning” (Dec. 2018), Ted G. Lewis and Peter J. Denning raised a crucial question about machine learning systems: “These [neural] networks are now used for critical functions such as medical diagnosis ... fire-control systems. How can we trust the networks?” They answered: “We know that a network is quite reliable when its inputs come from its training set. But these critical systems will have inputs corresponding to new, often unanticipated situations. There are numerous examples where a network gives poor responses for untrained inputs.”

David Lorge Parnas followed up on this discussion in his Letter to the Editor (Feb. 2019), highlighting “the trained network may fail unexpectedly when it encounters data radically different from its training set.”

We wish to point out that machine learning-based systems, including commercial ones performing safety critical tasks, can fail not only under “unanticipated situations” (noted by Lewis and Denning) or “when it encounters data radically different from its training set” (noted by Parnas), but also under normal situations, even on data that is extremely similar to its training set.

In our article “Metamorphic Testing of Driverless Cars” (Mar. 2019), we tested the real-life LiDAR obstacle perception system of Baidu’s Apollo self-driving software, and reported surprising findings that as few as 10 sheer random points scattered outside the driving area could cause an autonomous vehicle to fail to detect an obstacle on the roadway, with 2.7% probability—of the 1,000 tests, 27 failed (see Figure 3a in our article). The Apollo self-driving team confirmed “it might happen” because the system was “deep learning trained.”

Now, after a further investigation, we have found that in 24 of these 27 failed tests, the 10 random points can actually be reduced to just one single point, on which the system still fails. For the remaining three failed tests, the 10 ran-

dom points can be reduced to two points on which the system still fails. Such a random point can represent a tiny particle in the air or sheer noise commonly found in real-life data from sensors. In all our studies, the original data (before adding the random points) was training data downloaded from Apollo’s official website, where each data frame normally contained more than 100,000 data points (therefore, adding one or two additional points to the frame is trivial).

These findings mean the existence of just one single tiny particle in the air outside the driving area can cause the LiDAR system of a real-life deep-neural-network-driven autonomous vehicle to fail to detect an obstacle on the roadway. This result reveals a much more serious problem than those pointed out by Lewis and Denning, and by Parnas, and hence provides a case against mission critical applications of machine learning techniques.

Zhi Quan Zhou and Liquan Sun,
Wollongong, Australia

Authors’ Response

We agree completely. Examples of fragility of trained neural networks keep popping up and casting doubt on whether deep learning networks can be trusted in safety-critical applications. We saw a recent demonstration in which a network trained to read road signs correctly identified a stop sign when the image was clean, and incorrectly called it a speed limit sign when just a few pixels of the image were altered. This is one of the reasons Dave Parnas called for skilled programmers instead of neural networks to find solutions to problems because their programs could be verified and bugs fixed.

Ted G. Lewis and Peter J. Denning

The relevance of Zhou and Sun’s important point is not limited to neural networks or machine learning technology. They illustrate dangers that can exist whenever a program’s precise behavior is not known to its developers.

I have heard neural network researchers say, with apparent pride, that devices they

have built sometimes surprise them. A good engineer would feel shame not pride. In safety-critical applications, it is the obligation of the developers to know exactly what their product will do in all possible circumstances. Sadly, we build systems so complex and badly structured that this is rarely the case.

David Lorge Parnas

Never Too Late to Share Computational Thinking

I commend Judy Robertson’s wonderful blog post “What Children Want to Know About Computers” (Oct. 19, 2018) for illustrating the challenges children face understanding computers, and for the challenges CS educators face helping them. It reminded me of a moment in 2018 when I was explaining programming to my daughter, a recent college graduate who majored in the humanities and was never very interested in computers. She asked, “How does the computer actually work? How does it add two numbers?” It turned out to be the perfect opportunity to whip out my copy of Digital Equipment Corporation’s 1981 *VAX Architecture Handbook*. She found the details of the instruction sets, opcodes, registers, and memory fascinating and helped her begin to understand what computers and programs actually do behind the scenes.

She has since been studying computers and software development online, as she considers a career in technology. As Robertson said, we can do a much better job teaching our children how computers work. I think it is important to add that young adults—everyone, really—can benefit from a greater understanding of computers and computational thinking.

Geoffrey A. Lowney,
Issaquah, WA, USA

Communications welcomes your opinion. To submit a Letter to the Editor, please limit your contribution to 500 words or less, and send to letters@cacm.acm.org.

© 2019 ACM 0001-0782/19/8

ACM ON A MISSION TO SOLVE TOMORROW.

Dear Colleague,

Without computing professionals like you, the world might not know the modern operating system, digital cryptography, or smartphone technology to name an obvious few.

For over 70 years, ACM has helped computing professionals be their most creative, connect to peers, and see what's next, and inspired them to advance the profession and make a positive impact.

We believe in constantly redefining what computing can and should do.

ACM offers the resources, access and tools to invent the future. No one has a larger global network of professional peers. No one has more exclusive content. No one presents more forward-looking events. Or confers more prestigious awards. Or provides a more comprehensive learning center.

Here are just some of the ways ACM Membership will support your professional growth and keep you informed of emerging trends and technologies:

- Subscription to ACM's flagship publication ***Communications of the ACM***
- Online books, courses, and videos through the **ACM Learning Center**
- Discounts on registration fees to ACM Special Interest Group conferences
- Subscription savings on specialty magazines and research journals
- The opportunity to subscribe to the **ACM Digital Library**, the world's largest and most respected computing resource

Joining ACM means you dare to be the best computing professional you can be. It means you believe in advancing the computing profession as a force for good. And it means joining your peers in your commitment to solving tomorrow's challenges.

Sincerely,



Cherri M. Pancake
President
Association for Computing Machinery



Association for
Computing Machinery

Advancing Computing as a Science & Profession

SHAPE THE FUTURE OF COMPUTING. JOIN ACM TODAY.

www.acm.org/join/CAPP

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

- ☐ Professional Membership: \$99 USD
- ☐ Professional Membership plus
ACM Digital Library: \$198 USD
(\$99 dues + \$99 DL)

ACM STUDENT MEMBERSHIP:

- ☐ Student Membership: \$19 USD
- ☐ Student Membership plus ACM Digital Library: \$42 USD
- ☐ Student Membership plus Print *CACM* Magazine: \$42 USD
- ☐ Student Membership with ACM Digital Library plus
Print *CACM* Magazine: \$62 USD

- ☐ **Join ACM-W:** ACM-W supports, celebrates, and advocates internationally for the full engagement of women in computing. Membership in ACM-W is open to all ACM members and is free of charge.

PAYMENT INFORMATION

Name

Mailing Address

City/State/Province

ZIP/Postal Code/Country

- ☐ Please do not release my postal address to third parties

Email Address

- ☐ Yes, please send me ACM Announcements via email
- ☐ No, please do not send me ACM Announcements via email

- ☐ AMEX ☐ VISA/MasterCard ☐ Check/money order

Credit Card #

Exp. Date

Signature

Purposes of ACM

ACM is dedicated to:

- 1) Advancing the art, science, engineering, and application of information technology
- 2) Fostering the open interchange of information to serve both professionals and the public
- 3) Promoting the highest professional and ethics standards

By joining ACM, I agree to abide by ACM's Code of Ethics (www.acm.org/code-of-ethics) and ACM's Policy Against Harassment (www.acm.org/about-acm/policy-against-harassment).

I acknowledge ACM's Policy Against Harassment and agree that behavior such as the following will constitute grounds for actions against me:

- Abusive action directed at an individual, such as threats, intimidation, or bullying
- Racism, homophobia, or other behavior that discriminates against a group or class of people
- Sexual harassment of any kind, such as unwelcome sexual advances or words/actions of a sexual nature

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.



Association for
Computing Machinery

ACM General Post Office
P.O. Box 30777
New York, NY 10087-0777

1-800-342-6626 (US & Canada)
1-212-626-0500 (Global)
Hours: 8:30AM - 4:30PM (US EST)

Fax: 212-944-1318
acmhelp@acm.org
www.acm.org/join/CAPP

The *Communications* Web site, <http://cacm.acm.org>, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.



Follow us on Twitter at <http://twitter.com/blogCACM>

DOI:10.1145/3339456

<http://cacm.acm.org/blogs/blog-cacm>

Cutting the Wait For CS Advice

Mark Guzdial suggests ways to cut the long lines for college students seeking to meet with their computer science advisors.



Mark Guzdial
How to Reduce Long Lines at CS Office Hours in Five Tweets

May 2, 2019

<http://bit.ly/2KoUyFN>

My Ph.D. student, Katie Cunningham, tweeted the possibility that we could reduce long lines at computer science (CS) class office hours with improved teaching (see <http://bit.ly/2ESfr8N>). I agreed (<http://bit.ly/2HTpIDv>), and received a significant and somewhat surprising response.

There are some well-supported methods that might reduce the long lines at office hours, but few people use them. Based on the latest research at SIGCSE 2019 (<https://doi.org/10.1145/3287324.3287363>), the adoption rate of these methods is likely less than 5%. I decided to write five tweets with these methods (which you can see at <http://bit.ly/2Z1cev8>). Since a tweet is limited to only a few characters, I am expanding on those tweets here.

I'll admit up front that the title is a bit of click-bait. I can't *guarantee* that I can reduce the long lines at office hours. I have seen these methods work. I recommend them. Of course,

there may be issues in your classes that I haven't seen or learned about from the research literature.

My suggestions here do not in any way suggest that students visiting office hours is a bad thing. Students *should* interact with instructors. However, I don't believe that the long queues do anyone any good. I suggest every student should interact with teachers and teaching assistants one-to-one (1:1) at several points in every class, but if *most* students need 1:1 help on *most* assignments or to revisit *most* topics, then maybe there are inefficiencies elsewhere in the system. How can we meet student learning needs without those long lines?

Here are the five tweets, with commentary:

1. Use Peer Instruction. It is the most effective in-class teaching method I've ever used. If students learn more in-class, maybe they'll need less 1:1 help. <https://www.peerinstruction4cs.org/>

Beth Simon, Leo Porter, and Cynthia Lee taught me how to do peer instruction. They had to convince me. I blogged about my first experiences in 2011 (see example post at <http://bit.ly/2wyoCGW>). It changed how I taught, and it changed me into a better teacher.

I have always taught with peer instruction and other forms of active learning (like inverse live coding, <http://bit.ly/2Wda9L1>) ever since.

Let's be really clear: The research supports the Peer Instruction protocol (see a nice description of it at <http://bit.ly/2HTBxti>). Hiring lots of teaching assistants is *NOT* Peer Instruction (though that might be part of Peer Mentoring or Peer-Led Team Learning). Having students work in small groups is *NOT* Peer Instruction. There are other active learning methods in CS classes. Peer Instruction in CS has the strongest research evidence in support of its effectiveness.

2. Organize and require pair programming. Students do need 1:1 help. Pair programming can lead to better learning and improved attitudes about CS. But actually organize it—form pairs, expect it. <http://bit.ly/2ESi7TQ>

Just saying “You can collaborate” or “I encourage Pair Programming” is not the same as (a) organizing how pairs are formed, (b) teaching how to do pair programming, and (c) expecting it in grading.

Students often need personalized help. It doesn't always have to

come from the teacher or teaching assistant. Pair programming also has some significant positive impacts on diversity. There were so many links that I could have put in this tweet, and I struggled to decide which one to put in. The ETR link I included is a great overview. If you want a detailed explanation of how to do pair programming in the classroom, I highly recommend the NCWIT *Pair Programming in a Box* resources (at <http://bit.ly/2EQBX1I>).

Pair Programming doesn't always work. Pairs can go awry. But do the cost-benefit analysis. What percentage of pairs are ineffective, vs. how much time is wasted with students waiting in queues for office hours and what percentage of office hour 1:1 time is ineffective? In the long run, pair programming is a bigger win.

3. Backward Design. Look at your assignments. Did you teach everything needed to do them? No, it doesn't help learning to have students "figure it out on their own." That's what leads to long lines. If you didn't teach it, don't require it in the assignment. <http://bit.ly/2Msh5nH>

In CS, there is a pervasive attitude summarized in four letters: RTFM (read a great post about those four letters at <http://bit.ly/2KsgkZ6>). We tend to believe students should learn to figure things out on their own and seek out resources to guide them. The problem is that it's really inefficient and encourages long lines at office hours. Students would rather wait hours for a *specific answer* than spend hours (maybe more, maybe less) for the *chance* at an answer.

Here's the point of the tweet in an even shorter form: If you didn't teach something, don't require students to use it in an assignment.

Seriously. Direct instruction beats out students wandering around through resources trying to find an answer (see my post from last November making that same point at <http://bit.ly/2ARuUTA>). **There is no learning benefit for making students figure it out on their own.**

4. Change classroom climate. CS classes tend to have a defensive climate: critical, impersonal, with informal student hierarchies. That discourages diversity, and discourages coming to lectures. <http://bit.ly/2wxZbFm>

This last semester, I taught my first CS Education Research class at the University of Michigan (<http://bit.ly/31avw2Y>). A significant percentage of my students did their research work around the idea of "Defensive Climate." Computer science classes tend to have a defensive climate; that is, the class is impersonal, unfriendly, often critical or combative, and there are informal student hierarchies (for example, the students with lots of experience ask questions that other students don't even understand). We need to teach in a way that discourages defensive climate (see NCWIT article on that at <http://bit.ly/2Xp8X8w>). If we taught in a way that was more inclusive and welcoming, it might lead to students coming to lectures, engaging, and learning more—and they might need less 1:1 teaching in office hours.

Please don't use grade caps to limit enrollment. I make that argument at <http://bit.ly/2Z4RaUx>. You will very likely reduce your diversity if you do. If you have to limit enrollment, make it a lottery so that both privileged and underprivileged students have a fair shot at getting in.

5. Be explicit in your expectations that every student can learn CS. Many CS instructors believe that some students "got it" and other students can't. There's no evidence that's true, but we teach and design our classes that way. Teach to the bottom third of the distribution. <http://bit.ly/2KqZ7iN>

I have written a lot about the Geek Gene (<http://bit.ly/2IhP5xL>), the belief that some students have innate ability and others do not. It's a pervasive belief in CS education. If you believe that only some students are going to succeed, you will likely teach for their success. You will teach to the top few percent of the class. If you explicitly expect that everyone can succeed (that is, saying in class "You can all pass this class"), and you teach to those students who have less preparation but *can succeed*, then they will succeed, and fewer students will be waiting in the hallway for hours to get help. 

Mark Guzdial is a professor in the Computer Science & Engineering Division of the University of Michigan involved in the Engineering Education and Research program.

© 2019 ACM 0001-0782/19/8 \$15.00



Association for
Computing Machinery

Digital Government: Research and Practice

Digital Government: Research and Practice (DGOV) is an interdisciplinary journal on the potential and impact of technology on governance innovations and its transformation of public institutions. It promotes applied and empirical research from academics, practitioners, designers, and technologists, using political, policy, social, computer, and data sciences methodologies.



For further information
or to submit your
manuscript,
visit dgov.acm.org

Computing Reviews

Connect with our Community of Reviewers

“I like CR because it covers the full spectrum of computing research, beyond the comfort zone of one’s specialty. I always look forward to the next Editor’s Pick to get a new perspective.”

- Alessandro Berni



Association for
Computing Machinery

ThinkLoud

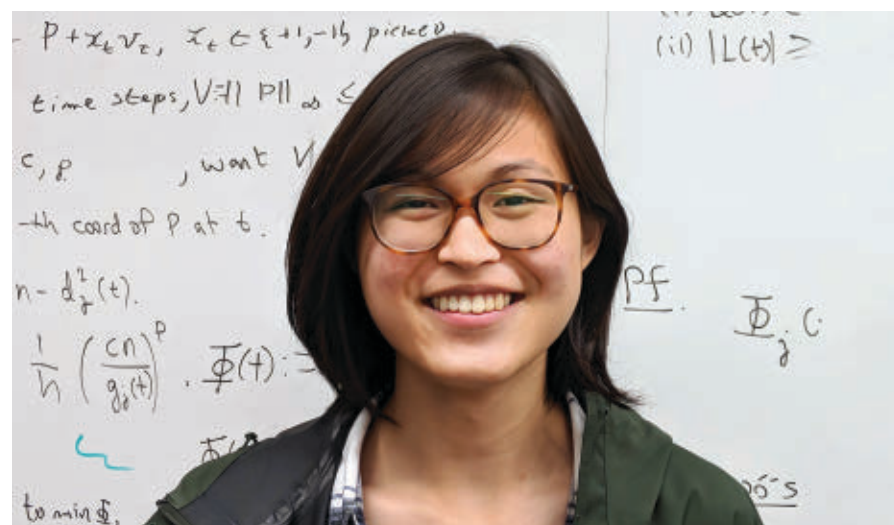
www.computingreviews.com

The Algorithm that Changed Quantum Machine Learning

A college student discovered a classical computing algorithm that experts overlooked. It promises to change both classical and quantum machine learning.

IT'S NOT EVERY day that an 18-year-old college student catches the eye of the computing world, but when Ewin Tang took aim at recommendation algorithms similar to those commonly used by the likes of Amazon and Netflix, the University of Texas at Austin mathematics and computer science undergraduate blew up an established belief: that classical computers cannot perform these types of calculations at the speed of quantum computers.

In a July 2018 paper, which Tang wrote for a senior honors thesis under the supervision of computer science professor Scott Aaronson, a leading researcher in quantum computing algorithms, she discovered an algorithm that showed classical computers can indeed tackle predictive recommendations at a speed previously thought possible only with quantum computers. "I actually set out to demonstrate that quantum machine learning algorithms are faster," she explains. "But, along the way, I realized this was not the case."



Ewin Tang set out to show that quantum machine learning algorithms are faster than classical algorithms, "but ... I realized this was not the case."

The ripples of Tang's research have reached far and wide. Not surprisingly, the press fawned over her discovery, in some cases implying that it had made quantum computing advances obsolete. Mathematicians and computer scientists in the machine learning field took notice, too. Some acknowl-

edged that Tang found a blind spot in previous research—an algorithm that had somehow escaped the top minds in the field. Her discovery has forced computer scientists to rethink basic assumptions about quantum machine learning, at a time when quantum computing is advancing rapidly.

Make no mistake, Tang's research is remarkable, and it will have a significant impact on both classical and quantum machine learning. "What Ewin's paper demonstrates is that the properties exploited by the quantum algorithm can also be used by a classical algorithm," explains James R. Lee, a professor of computer science at the University of Washington and Tang's Ph.D. advisor. "Ultimately, this offers a new framework for designing conventional algorithms on classical computers."

A New Formula

The algorithm in question is one that computer scientists Iordanis Kerenidis and Anupam Prakash introduced in 2016. Their quantum computing algorithm solved theoretical recommendation problems exponentially faster than any previously known classical algorithm. It achieved a speed increase by eliminating the need to access all components in a decision-based matrix; instead, the algorithm accesses only those components most relevant for a specific person. In the case of a movie recommendation, for example, this might mean a proclivity toward French films, or a desire to watch action films or romantic comedies.

At the time, "The algorithm delivered an exponential improvement compared to the best-known classical algorithms," says Kerenidis, a senior researcher in the Algorithms and Complexity Group at Université Paris Diderot. Essentially, the pair demonstrated that a quantum computer could process recommendation challenges exponentially faster than any known algorithm, though it didn't eliminate the possibility of an equally fast classical algorithm. When, in 2017, Tang selected the topic for a senior honors thesis in a class Aaronson taught about quantum information, she simply hoped to confirm such an outcome. "We thought that the analysis would back the validity of existing research," she says.

A famous example of this approach is Simon's Problem, which was created by computer scientist David Simon in 1994. It demonstrated, at least on a theoretical level, that this computational problem can be solved with exponentially fewer queries on a quantum computer. It served as the foundation for Shor's Factoring Algorithm, which appeared shortly thereafter and remains a highly significant contribution to quantum the-

Kerenidis and Prakash's quantum computing algorithm solved theoretical recommendation problems faster than any previously known classical algorithm.

ory. Peter Shor delivered further evidence that Simon's concept was correct by developing a quantum algorithm for integer factorization. The algorithm finds the prime factors of an integer N in time polynomial in the number of digits of N .

Yet as Tang studied the algorithm further, things began to get a lot more interesting. For months, she worked to rule out the possibility that a classical algorithm could address the Kerenidis-Prakash problem. Aaronson refers to this as phase 1. "At the time, nothing about that failure seemed odd or anomalous to us: most likely, we thought, there was indeed no classical algorithm, but ruling it out was just a very hard problem," he says. "At some point, though, Ewin had the key insight about why an efficient classical algorithm should exist after all and set out to develop it." Aaronson refers to this stage as phase 2.

All along, the algorithm Kerenidis and Prakash had discovered and the validity of their calculations were never in doubt. What eventually became apparent was that there was more to the story. "It appeared that there was something other people had missed," she states.

During the early stages of the project, Aaronson worked with Tang to further explore the mathematical underpinnings—despite the fact that he was on sabbatical during most of this period. During the latter stages, Tang worked almost entirely on her own. Finally, "We went over it and over the research because we wanted to be absolutely sure that it was sound before it was published," Aaronson explains.

Soon after graduating, Aaronson arranged for Tang to appear at a quantum computing workshop at the Simons

Institute for the Theory of Computing, part of the University of California, Berkeley (UC Berkeley). In attendance were a number of luminaries from the quantum machine learning field, including Ronald de Wolf from the University of Amsterdam, Mario Szegedy from Rutgers University, and Iordanis Kerenidis who, with Prakash, a graduate student at UC Berkeley, had written the original quantum machine learning recommendations paper.

At the symposium, the luminaries watched and listened closely for about four hours, spread across two days. They asked questions and probed, but none of them could find anything wrong with Tang's research and calculations. Aaronson encouraged Tang to publish the paper, *A Quantum-inspired Classical Algorithm for Recommendation Systems*. It appeared in July 2018 at the *Electronic Colloquium on Computational Complexity* (ECCC) and it was later accepted to the ACM Symposium on Theory of Computing (STOC 2019). Says Kerenidis, "It is an impressive piece of theoretical work. It is very interesting to see that a quantum algorithm can inspire new classical algorithms."

Finding the Proof

Tang's research focused on a specific piece of the machine learning puzzle. Recommendation systems suggest products or choices based on data about user preferences. The algorithm typically revolves around a model that is based on a specific task: completing an $m \times n$ matrix of small rank k . However, Tang presented a classical algorithm that produces a recommendation in $O(\text{poly}(k) \text{ polylog}(m, n))$ time. This represents "an exponential improvement on previous algorithms that run in time linear in m and n ," she noted in the paper.

In plain terms, Tang achieved a speedup in classical computing using the same general approach Kerenidis and Prakash tapped. She directed the algorithm to narrow the matrix and focus only on the most important criteria to generate a useful recommendation.

The belief that quantum computing produces exponential improvements over classical computing algorithms for recommendation systems had been shattered. "The best we can hope for is a polynomial improvement, but in practice this can still be of great value,"

Kerenidis explains. Lee says that it's important to view the findings in context, and not allow the short-term ramifications to distort the long-term uses and benefits of quantum machine learning. "There are two important components to the research," he says. "First, it's an exciting direction in algorithms research because it offers a new approach to the design of classical algorithms. Second, it's an important step in mapping out the types of problems where we could expect quantum computers to be exponentially more efficient than classical computers."

In fact, the recommendation problem is only a tiny sliver of the quantum computing universe. Says Bob Sutor, vice president for IBM Q Strategy and Ecosystem at IBM Research (which unveiled the first commercially available quantum computer in January 2019): "While it might be fun to set up some sort of competition between classical and quantum algorithm creators, the truth is that they work with and inspire each other... there is a real separation in some cases between classical and quantum."

By the Numbers

Beyond the obvious appeal of an 18-year-old deflating the quantum machine learning balloon lies a basic fact. "It would be very shortsighted to view this research as any sort of setback for quantum computing," Lee points out. "This is essentially how scientific research works and, over the long run, this research helps us better understand how and where to use classical computing methods and quantum computing methods. This doesn't fundamentally change the viability of quantum computing. It hasn't made the field any less promising."

Aaronson concurs. "Ewin's breakthrough rules out the hope of an exponential quantum speedup for certain types of machine learning problems. But there are many other applications of quantum computers that are not affected by it—including breaking cryptographic codes, getting modest speedups for optimization problems, and probably most important of all, simulating quantum physics and chemistry themselves."

Amid all the claims and counter-claims, it is important to keep a few things in mind. There is strong evidence that quantum computing can

significantly accelerate certain computations, but there is no definitive proof of such. The only known provable separation theorem between quantum and classical is $\sqrt{n}$ vs. n . Proofs of stronger separation between classical and quantum are relative to an oracle.

There is also a long way to go before quantum computing becomes a reality. The IBM Q has only 20 qubits, meaning that at this point, it is more of a demonstration machine than a viable quantum computing device. Finally, whether quantum or classical computing devices are used, machine learning requires very large input sets.

Tang is taking things in stride. She is now pursuing her doctorate in theoretical computer science from the University of Washington, where she continues to explore quantum and classical algorithms. She has since collaborated on a second academic paper focusing on classical sub-linear-time algorithms that are designed to solve low-rank linear systems of equations. And she finds comments from a handful of detractors, upset that she chose to focus on a problem that helps companies make money, "amusing."

Concludes Tang, "At this point, it's not clear whether a broader group of quantum algorithms or techniques can be dequantized and applied to make classical algorithms faster." ■

Further Reading

Tang, E.

A Quantum-inspired Classical Algorithm for Recommendation Systems, *Electronic Colloquium on Computational Complexity (ECCC)*, July 2018. <https://arxiv.org/abs/1807.04271>

Kerenidis, I., and Prakash, A.

Quantum Recommendation Systems, September 2016. <https://arxiv.org/abs/1603.08675>

Aaronson, S.

Quantum Computing Since Democritus, Cambridge University Press. 2013. <https://www.scottaaronson.com/democritus/>

Ciliberto, C., Herbster, M., Ialongo, A.D., Pontil, M., Rocchetto, A., Severini, S., and Wossnig, L. Quantum Machine Learning: a Classical Perspective, *Quantum Physics*, Volume 474, Issue 2209, page 20170551. January 1, 2018. <https://royalsocietypublishing.org/doi/full/10.1098/rspa.2017.0551>

Samuel Greengard is an author and journalist based in West Linn, OR, USA.

© 2019 ACM 0001-0782/19/8 \$15.00

ACM Member News

GETTING COMPUTERS TO HELP CONSTRUCT SOFTWARE PROGRAMS



Rastislav Bodik is a professor at the Paul G. Allen School of Computer Science and Engineering at

the University of Washington (UW), in Seattle. Before joining UW, Bodik spent more than a decade on the faculty at University of California, Berkeley. Before that, he was an assistant professor at the University of Wisconsin-Madison.

After receiving an undergraduate degree in computer engineering from the Technical University of Košice, Slovakia, Bodik earned his master's degree and doctorate in computer science from the University of Pittsburgh.

His research is focused primarily on automatic programming, also known as program synthesis, a technique for the computer-aided construction of software that simplifies the process of writing computer programs.

"In the past few years, reaching human parity in programming tasks has been a huge achievement," Bodik says. Program synthesis has achieved parity with human programmers on at least a dozen tasks that usually require months of training, he says, adding that "Automatic programming can now program as well as humans, and sometimes faster."

Another direction more research-focused, Bodik says, "is to look at programming puzzles that only experts can solve, and make them solvable by new automatic programming algorithms." That would include applications such as the design of computer systems, programming for robots, and dialogue systems enabling conversations with artificial intelligence.

Bodik looks forward to extending program synthesis to other problems, like creating tools to make it easier to build automatic programming tools, such as language editors and coding assistants.

"This is a big challenge in the next decade," he says.

—John Delaney

I Don't Understand My Car

Self-driving cars will need good communication skills.

SOMEDAY, PERHAPS, STREETS and highways will host only fully autonomous vehicles, wirelessly communicating and following algorithms that let them handle any situation they encounter. For now, though, city streets are filled with pedestrians, bicyclists, delivery trucks, double-parked cars, emergency vehicles, and construction crews, as well as human-operated cars with issues of their own.

In this chaotic setting, self-driving cars face additional challenges beyond rapidly analyzing the complex environment and navigating through it. They also must keep their distracted occupants informed of issues potentially requiring attention. Equally important, they must continually coordinate their actions with humans, whether in other cars or on the street.

Researchers are exploring novel ways for cars to communicate, such as displays that confirm that a pedestrian has been seen, but these ideas will take a long time to become standardized. Moreover, even as new methods are introduced, there will be many older cars with no such tools, so people won't know what to expect.

Human-Machine Interface

The industry has adopted a five-level classification scheme for vehicle automation, from simple steering or speed assistance to fully autonomous driving. Even the lowest levels require new behaviors from drivers, like not pumping the brake pedal when an anti-lock system engages. At the higher levels, at least short of complete autonomy, ensuring that humans engage properly with complex, algorithmically driven behaviors is challenging.

Human-machine interfaces for cars can draw on experience with airplanes, said Christopher Hart, who previously chaired the National Transportation Safety Board and was deputy director of the Federal Aviation Administration. In particular, Hart said, designs should



Self-driving cars need new ways to communicate.

be “human-centered,” rather than driven by automation for its own sake. It is important to keep the interface simple, Hart said, providing only the information that the human needs. As in some versions of airline “glass cockpits,” he said, “you could really overwhelm them with information.”

However, although simplicity is good, as a system withholds more and more information from users, they are likely to disengage from the decision-making process.

An important difference from cars is that airplane pilots are intensively trained on the specific system they are operating. In contrast, “Drivers don’t even read the owner’s manual,” Hart said. “That’s one of the reasons that it has to be so user-friendly.”

Pilots also undergo periodic assessments of their knowledge, physical health, and operational skills, said Mary “Missy” Cummings, a professor in the Department of Mechanical Engineering and Materials Science of Duke University, who spent a decade as a U.S. Navy pilot. “Pilots are highly trained, with regular certification testing, as opposed to drivers, which have none of those attributes.”

A major challenge occurs if the automation encounters unanticipated

circumstances and tries to transfer control to a person. There is a fundamental conflict, Hart said, because “Automation needs to be very reliable before you put it on the street,” but “humans are not good monitors of very reliable systems,” making a “graceful exit” almost impossible.

“If it’s going to take a minute or more for the person to get back in the loop, and it only takes 10 seconds to pull to the side of the road and stop the vehicle,” then stopping is the smarter course, said Philip Koopman, an associate professor in the department of electrical and computer engineering of Carnegie Mellon University and co-founder of startup Edge Case Research. Although pulling over may disrupt traffic, “That’s a whole lot safer than a human that doesn’t know what’s going on.” He warns, however, that “Building a car that realizes it doesn’t know what’s going on is almost the same difficulty as building a car that can handle it,” so some companies are trying to skip this stage.

Koopman also worries whether designers have enough of a safety mindset to prepare for multiple, surprising failures. “When you have a million vehicles on the road, weird stuff happens every day,” he said. For example, an ac-

cident could endanger first responders by disabling sensors that should have immobilized the car. “That actually happened in some prototypes I knew about,” Koopman said.

Playing Well with Others

Beyond the operator interface, a critical part of good driving is communicating with others on the road. Some researchers envision a world where vehicles exchange detailed information, for example, by radio. In addition to reducing accidents, such communication would allow vehicles to coordinate with each other, and even with the road infrastructure, to reduce traffic delays and save fuel. For now, however, cars must use more primitive channels to communicate with other automobiles, bicycles, and pedestrians.

Car manufacturers, subject to approval by government regulators, have adopted many standardized signaling mechanisms. Some, such as turn signals and horns, are dedicated for signaling. Other devices are also recruited to send messages. Flashed headlights, for example, might indicate anything from “speed trap ahead,” to “I am yielding to you.” Often, these crude messages are ambiguous, like a turn signal that may indicate an intent to turn at an upcoming intersection or into a closer driveway, or to stop, reverse, and parallel park.

Vehicles also implicitly signal to others on the road though details of their motion. For example, “If you come up to a stop sign and you roll forward, it means you’re eager, and if you stop five feet back from the stop line, everyone knows that you saw them,” Koopman said. Similarly, a merge on a highway sometimes involves a game of “chicken,” with cars speeding or slowing to negotiate who goes first.

Self-driving vehicles will need to both interpret and send these subtle cues and clues, in addition to the more-explicit signals. Ultimately, they need something like a “theory of mind” like that human drivers employ to guess how other drivers might react. Indeed, although the self-driving cars currently being tested have good safety records, in many of the accidents they have had, they were hit from behind. “One very plausible reason for all of those rear-end collisions is that the cars aren’t behaving the

way people expect them to,” Koopman said. To address this problem, “self-driving cars might be constrained to drive the way typical people do, even if that’s not optimal.”

This issue might get better as self-driving cars become more commonplace, however, because people should get familiar with their idiosyncrasies and compensate for them, as they do with erratic drivers who may be drunk or otherwise impaired. On the other hand, Rodney Brooks, emeritus professor at the Massachusetts Institute of Technology and a founder of iRobot and Rethink Robotics, warns that some people will maliciously exploit the quirky, conservative behaviors of autonomous vehicles.

The challenges for self-driving cars are particularly profound when they interact with pedestrians, both in interpreting human body language and in communicating intentions. For example, a prudent pedestrian will not walk in front of a stopped car until they have met the driver’s eyes, or even gotten a wave of the driver’s hand. To replace this signaling, researchers have explored strategies for communicating awareness of pedestrians, such as endowing cars with eye-like displays or projecting crosswalks onto the street to invite safe passage.

Cummings and others, however, have found that today’s pedestrians are not likely to work very hard to interpret such signals. Koopman also worries that these generic displays don’t really let someone know that it sees them specifically, only that it sees something nearby. For human drivers, “humans are remarkably good at gaze tracking,” which makes eye contact highly specific.

A Long Road

Adoption of self-driving cars is likely to be a messy, slow process. Compared to airplanes, “Automation on the ground is going to be much more complicated,” Hart said. “Nevertheless, I am convinced that, when automation matures enough to be used on the streets, it will save more lives,” he said, although “It will never be perfect.”

Indeed, like the other imperfections of autonomous vehicles, the communications weaknesses of autonomous vehicles should be compared with those we tolerate from distracted and

imperfect human drivers. With respect to turn signals, for example, “We have best practices now in terms of signaling, and nobody uses them,” Cummings said.

A similar transition occurred when automobiles replaced horse-drawn vehicles a century ago, Brooks said. That transitional period included large investments in road infrastructure, as well as passing laws against pedestrians walking wherever they want. One reliable strategy for pedestrian safety would be total separation of people from traffic with physical barriers, as are often used for the small number of existing driverless trains. Such isolation could be practical, say, for dedicated long-distance trucking lanes, but it seems hard to imagine in heavily populated areas.

The specific expectations for cars and pedestrians also vary widely with local culture, making a generic solution more difficult. Even in a single country like the U.S., pedestrians are much more likely to cross a street against a traffic light in Boston than in Los Angeles. Within a single city, expectations differ between neighborhoods or even individual intersections.

“In the fullness of time, self-driving will come,” Brooks said, but he warns that autonomous vehicles won’t just be dropped into the existing framework. “It’s going to go hand in hand with changes in the environment,” he said. “We don’t know what that’s going to look like.”

Further Reading

Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, SAE International Standard J3016, revised June 2018
https://www.sae.org/standards/content/j3016_201806/

The Big Problem with Self-Driving Cars Is People, Rodney Brooks, *IEEE Spectrum*, July 2017
<http://bit.ly/2Nho04T>

Is a Robot about to take my Job?, Mary “Missy” Cummings, <https://www.youtube.com/watch?v=DU1rzjSMdkU>

Don Monroe is a science and technology writer based in Boston, MA, USA.

© 2019 ACM 0001-0782/19/8 \$15.00

What Makes a Robot Likable?

Interactions with robotics teach us more about people.

ON SCREEN, THE virtual character sits in a comfortable purple chair. She wears plain pants, a turquoise shirt, and a slim jacket with the sleeves rolled up past her elbows. Her short dark hair is swept to one side and her ethnicity is intentionally ambiguous, according to her developers, a team of researchers with the University of Southern California (USC) Institute for Creative Technologies. Some of the people who have interacted with her assume she is Asian; others conclude she has a completely different ethnicity. “People have come up and said that they’re so thankful we paired them with someone of their race because it helped them connect,” recalls Gale Lucas, a research assistant professor at USC.

The platform, SimSensei, is designed for one-on-one sessions with individuals, and uses visual and audio feedback to tailor its responses. In one study, veterans who submitted to counseling sessions with SimSensei shared personal and mental health concerns they would have withheld from actual human therapists. The system is designed to encourage this kind of open interaction, engaging in active listening by offering affirming or comforting responses or noting when the subject pauses or hesitates—and asking why. Human therapists carry out these techniques intuitively, yet Lucas and her colleagues found the participants were still more open with the virtual platform.

Northeastern University computer scientist Timothy Bickmore and his team have found similarly surprising connections between people and virtual agents across numerous studies. Typically, Bickmore and his team will try to simulate a counseling or information-sharing session between a patient and a healthcare professional, then measure the effectiveness of virtual agents against their human counter-



“And how did that make you feel?” is a question the SimSensei virtual therapist, shown above, might ask. Image courtesy of USC Institute for Creative Technologies.

parts. “We try to simulate face-to-face counseling,” Bickmore explains. “We have found over the years that many disadvantaged groups prefer and do much better with agents and robots compared to those with high-level tech literacy. They can get the information better. The people don’t feel they’re being talked down to.”

As robots become an increasingly present and powerful force in our lives, from healthcare to home maintenance to the workplace, researchers are hard at work exploring different ways to strengthen the bonds between people and both virtual and physical agents. Some of the lessons learned in developing virtual platforms apply to embodied robots; in other cases, the rules appear to be different. What has become clear, researchers say, is that there is no simple recipe for developing likable robots.

The Cult of Personality

Before Bickmore and his team develop a new virtual agent for a specific exper-

iment, they first study the nurses, general practitioners, or other professionals who typically conduct the session they are trying to simulate, then base their agents on these individuals. All of the agent’s responses are approved by health professionals in advance. Natural language exchanges would be too risky in a healthcare environment, according to Bickmore. Instead, participants in the studies typically interact with the virtual agent through an iPad or some other large touchscreen device, and the conversation follows a narrative tree. The virtual agent asks a question or delivers a spoken prompt. The participant selects a response from a multiple-choice list. The conversation moves on.

The USC platform, SimSensei, monitors pauses in conversations and tracks changes in tone, gaze aversion, and other social cues. SimSensei then processes all these indicators and autonomously determines the appropriate response. This is a risky interaction, Lucas explains. “What if the system told you

‘that’s great’ when you told it your father was dying? That would completely ruin the conversation,” she notes. “So we are really careful. Only if the system is really certain that it understands correctly are those kinds of responses used. It does not use them if there is too much uncertainty in the algorithm.”

When the goal is to have someone interact with a virtual agent over a long period of time, researchers say it is helpful to build in additional layers of interaction, such as exchanging social pleasantries or remembering basic facts about the person, then recalling those in a future exchange.

Bickmore and his colleagues have even experimented with giving their agents biographical details to share, such as where the robot grew up or where it has traveled. “People want to know more about who these agents are,” he says. “Even though they know these stories are fake, the stories increase engagement. There’s nothing that helps people perceive a robotic entity more than if it says the same thing over and over.”

Stories can also help people establish bonds with physical robots, not just virtual ones. Kate Darling, a research specialist at the Massachusetts Institute of Technology Media Lab, and her then-robotics student Palash Nandy set up an experiment in which participants were introduced to a simple toy robot, the insect-like Hexbug. In one case, individuals in the study were shown the robot, then given a mallet and told to strike the toy. Other participants were presented with a slightly different scenario: the robot was accompanied by a snippet of text that provided a story about the Hexbug and its recent behavior in the lab. Then these individuals, too, were asked to hit the toy. The results were clear: Participants hesitated longer, and were less willing to harm the Hexbug, when it had a backstory.

Physical vs. Virtual

While there are some similarities between what makes a physical or virtual robot likeable, the rules often vary. Embodied robots that appear too human-like can make people feel uneasy, an effect known as the Uncanny Valley. Yet this does not always apply to virtual agents. SimSensei has an

android-like appearance, and people are open and willing to share with the platform.

Bilge Mutlu, a computer scientist at the University of Wisconsin-Madison, has been exploring these differences in what makes physical and virtual agents appealing.

When two people converse, certain cues, such as breaking eye contact, tend to promote intimacy. So Mutlu conducted an experiment to see whether this would hold true for virtual and embodied robots. In one case, it worked; people disclosed more information when the virtual human in his study broke eye contact during an exchange. But when the physical robot used the same technique, the effect was completely different: people were put off. “They thought the physical robot had intent,” Mutlu says. “They thought it was this intentional, thoughtful being, and they didn’t want to disclose as much.”

Mutlu suggests people might be hardwired to respond differently to virtual and physical entities. Interacting with a virtual robot, in his view, might be more like theater; people experience it almost as they would an interactive play. They assume there is a creator and that the interaction has been designed. Meeting with an embodied robot might be closer to an encounter with an unfamiliar animal: the creature has the ability to violate your space, and you don’t know initially whether it is friendly or dangerous.

“With a virtual application, you’re the one who initiates the interaction,” Mutlu says. “You might be willing to engage and even be vulnerable, but you can walk away and be done. With a physical robot, that’s not clear. There’s you and the robot, and it can cross the divide at any time. You think it’s more of an independent agent, so you are going to have less trust.”

Building Bonds

So how do designers make these platforms more appealing and less threatening?

The simplest route to establishing a bond might be functionality; people like robots when they are useful. Matthias Scheutz, director of the Human-Robot Interaction Laboratory at Tufts University, cites Roomba

vacuum cleaners as evidence. These autonomous machines are not particularly cute. They don’t have large eyes or make appealing noises, yet owners become attached to them, because they work. “People are so grateful and think the robot works so hard that they actually clean for the robot,” Scheutz notes. “That’s a big danger. I don’t want people to feel gratitude toward the machine. You’re not thankful to your microwave for heating the meal: that’s what it’s there for. But a robot is perceived as something with goals, and intentions, and an inner life.”

The enormous variety and differing results from the many human-robot interactions being studied today have made it clear that what makes a robot likeable varies from case to case. But another emerging theme is that the field is not just about understanding virtual or physical platforms.

“We put these systems in front of people and have people respond to them, and in those responses we see such interesting things,” says Mutlu. “We’re understanding more about how to build these systems, but we’re also learning so much more about people.” **C**

Further Reading

- Bickmore, T. and Picard, R. Establishing and Maintaining Long-Term Human-Computer Relationships, *ACM Transactions on Computer Human Interaction (ToCHI)*, 2005, 12 (2) : 293-327.
- Deng, E., Mutlu, B., and Mataric, M. Embodiment in Socially Interactive Robots, *Foundations and Trends in Robotics*, 2019, Vol. 7: No. 4, pp 251-356.
- Lucas, G.M., Gratch, J., King, A., Morency, L.P. It’s Only a Computer: Virtual Humans Increase Willingness to Disclose, *Computers in Human Behavior*, August 2014, Volume 37, pp 94-100.
- Darling, K., Nandy, P., and Breazeal, C. Empathic Concern and the Effect of Stories in Human-Robot Interaction, *Proceedings of the IEEE International Workshop on Robot and Human Communication (ROMAN)*, 2015.
- Arnold, T., and Scheutz, M. Observing Robot Touch in Context: How Does Touch and Attitude Affect Perception of a Robot’s Social Qualities?, *Proceedings of the 13th ACM/IEEE International Conference on Human-Robot Interaction*, 2018.

Gregory Mone is the co-author, with Bill Nye, of the *Jack and the Geniuses* series of novels.

© 2019 ACM 0001-0782/19/8 \$15.00

Education

Block-based Programming in Computer Science Education

Considering how block-based programming environments and tools might be used at the introductory level and beyond.

BLOCK-BASED PROGRAMMING IS increasingly the way that learners are being introduced to the practice of programming and the field of computer science more broadly. Led by the success of environments like Scratch (see the figure appearing later in this column) and initiatives like Code.org's Hour of Code, block-based programming is now an established part of the computer science education landscape. While not a recent innovation (for example, LogoBlocks has been around since the mid-1990s), the last decade has seen a blossoming of new toys, games, programming environments, and curricula that incorporate block-based programming features. Given this growing presence, it is important that we as a community look critically at the block-based programming modality to understand its affordances, drawbacks, and identify

how best to use it as a means to welcome people into the discipline of computer science and support them as they grow and learn.

What Is Block-based Programming?

Block-based programming has a number of key features that make it distinct from conventional text-based programming and other visual programming approaches. Block-based programming uses a programming-primitive-as-puzzle-piece metaphor as a means of providing visual cues to the user as to how and where commands may be used. Figure 1b shows a block-based program written in Scratch. Block-based programming environments have been designed for children as young as five years old but most environments are designed for kids ages eight to 16. Writing a program in a block-based environment takes the

form of dragging-and-dropping programming instructions together. If two instructions cannot be joined to produce a valid statement, then the environment prevents them from snapping together. In this way, block-based programming environments can prevent syntax errors while still retaining the practice of authoring programs by assembling statements one-by-one.

While the visual cues and mitigation of syntax errors are key ingredients in supporting novices in having early programming success, there are additional features of block-based programming that support beginners. For example, block-based programming environments present the available set of commands to the user in the form of an easily browsed blocks palette from which the user can drag the command into their program (left-hand side of Figure 1a). Within the palette, the blocks



Students working on Scratch projects during a summer algebra camp in Irvine, CA, USA.

are conceptually organized and color coded. This allows users to browse the set of available commands to see what is possible rather than needing to know before-hand what can be done in the language. At the same time, the drag-and-drop composition approach removes the challenge of typing and finding uncommon punctuation marks on the keyboard, making programming more accessible for people who struggle with typing. Another notable feature of block-based programming environments is that the graphical presentation of each programming statement makes it possible to use natural language to describe the behavior of the command. For example, incrementing the value of a variable, which in a programming language like Java would look like this: $x = x + 1$;, can be accomplished with a command that reads: change x by 1. Given the myriad of supports present in block-based environments, it is important to understand if, how, and why this approach is an effective way to introduce novices to programming.

The Case for Block-based Programming

A good place to start the discussion of the benefits of block-based programming is with the most successful (to date) block-based programming environment: Scratch.² The goal of Scratch was to create a programming environment with sufficient scaffolds for novices to start to program with little or no formal instruction (low threshold) while also being able to support sophisticated programs (high ceiling). At the same time, it was important that the environment support a variety of types of programmers and programs (wide walls) and provide a means for programmers to share the programs they authored and participate in a larger community of programmers. Since its launch, over 35 million users have created accounts on the Scratch website and almost 40 million projects have been shared with a majority of users being under the age of 14. Further, research has shown block-based tools like Scratch and Looking Glass Alice can be an effective environment for

welcoming learners from populations historically underrepresented in computing fields.¹ These numbers reinforce the idea that block-based programming is playing a significant and important role in introducing youth to programming.

To understand how learners make sense of the block-based modality and understand the scaffolds that novice programmers find useful, I conducted a series of studies in high-school computer science classrooms. As part of this work, I observed novices writing programs in block-based tools and interviewed them about the experience. Through these interviews and a series of surveys, a picture emerged of what the learners themselves identified as being useful about the block-based approach to programming. Students cited features discussed here such as the shape and visual layout of blocks, the ability to browse available commands, and the ease of the drag-and-drop composition interaction. They also cited the language of the blocks themselves, with one student saying

“Java is not in English it’s in Java language, and the blocks are in English, it’s easier to understand.” I also surveyed students after working in both block-based and text-based programming environment and they overwhelmingly reported block-based tools as being easier.⁵ These findings show that students themselves see block-based tools as useful and shed light as to why this is the case.

To investigate learning outcomes associated with block-based programming, I conducted a quasi-experimental study in two high school computer science classrooms. The two classrooms used the same programming environment with one difference: one environment presented the code in a block-based interface while the other had a text-based interface. The underlying programming language was the same between the two meaning anything that could be done in one modality could also be done in the other. Starting on the first day of school, the two classes spent five weeks working through the same curriculum and were taught by the same teacher. The study was designed to control for as many factors as possible aside from the programming modality. After the five-week introduction, students in the block-

It is worth thinking about what role block-based languages might play in the design of computational tools.

based condition scored significantly high on content assessments than their text-based peers.⁶

Challenges and Open Questions

While research has shown the potential of block-based environments, there are still challenges and open questions related to the role of block-based programming in introductory computing contexts. One significant question relates to perceptions of block-based tools and whether or not dragging-and-dropping colorful and playful programming commands constitutes “real programming.” While

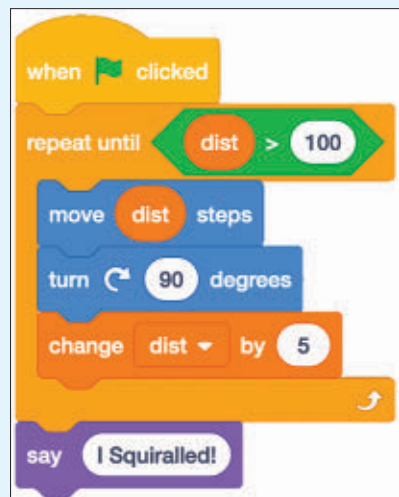
an experienced programmer may be able to see the conceptual equivalence between the repeat block in Scratch and a for loop in Java, the same is not necessarily true for beginners. As part of my classroom interviews, some students expressed concerns over the authenticity of block-based programming. For example, one student stated: *“if we actually want to program something, we wouldn’t have blocks,”* which calls into question the potential utility of block-based tools. Students also expressed concern over the expressive power of block-based environments, saying things like *“blocks are limiting, like you can’t do everything you can with Java, I guess. There is not a block for everything.”* While this statement is not necessarily true of all block-based environments (for example, there are numerous block-based interfaces for Java), the fact that students perceive this difference is a challenge educators face.⁵

A second open question surrounding block-based programming is whether or not block-based tools help learners with transitioning to text-based languages. There have been some documented examples of students learning concepts in block-based environments and successfully transferring those ideas to a text-based

The Scratch programming environment (a) and a block-based program written in Scratch (b).*



(a)



(b)

* In early programming work with the Logo language (from which Scratch is based), students would draw square spirals that came to be called squirrels: the image (and program) reflects that history.

language, however, my own research has not replicated these results. In a continuation of the quasi-experimental classroom study discussed above, after students finished their five-week introduction to programming in the block-based and text-based introductory environments, all students transitioned to the Java programming language. After 10 weeks of learning Java, I readministered the content assessment. Despite students in the block-based condition scoring significantly higher after the introductory portion of the course, there was no significant difference in scores between students in the two conditions. This means the gains associated with the block-based introduction did not translate to students being further ahead when learning Java but also did not hamper their transition.³ Understanding how to better scaffold learners moving between modalities, and the role of the teacher in this processes is a direction of future work for the field.

What the Future Block-based Programming Might Be

So, what is next for block-based programming? First, as the research described in this column suggests, the literature shows block-based programming should have a home in computer science education. One version of that is in the role it is currently playing, that of introductory tools designed to welcome novices to the field, either in upper elementary grades (ages 10 to 14) or high-school (up to age 18). As to what exactly that looks like and how such environments can support learners in moving beyond block-based tools is an open question. One potential direction is hybrid and bidirectional programming environments that blend block-based and text-based tools, giving the learners agency for deciding how and when to switch programming representation. This is one active and exciting area of design research in the area of introductory computing.

Looking beyond introductory contexts, there is a larger question about the potential role of block-based tools in the world of computer science. Currently, there is an assumption that block-based tools serve as

an entry point with the expectation that learners move beyond it to conventional text-based programming languages. However, as the Computer Science for All movement progresses and programming becomes a more universal literacy, it is worth thinking about what role block-based languages might play in the design of computational tools. If it is possible to do significant, non-trivial tasks in block-based environments, should we still expect all learners, even those not likely to pursue a degree in computer science, to learn text-based programming? For example, we created a block-based interface for controlling industrial robots and found it be easier for adult novices to use than existing robotics programming environment.⁴ Given the success of this design, it becomes easy to imagine a world with countless domain-specific block-based programming tools that put the power of computing at the fingertips of those who are proficient with block-based programming. This is not to say this is what the future holds but instead I put this forward as a way to think about new possible end-points for computing education and a more expansive view of the potential of block-based programming in the technological world that awaits. 

References

1. Kelleher, C., Pausch, R., and Kiesler, S. Storytelling Alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2007), 1455–1464.
2. Resnick, M. et al. Scratch: Programming for all. *Commun. ACM* 52, 11 (Nov. 2009), 60.
3. Weintrop, D. Modality Matters: Understanding the Effects of Programming Language Representation in High School Computer Science Classrooms (Ph.D. Dissertation). Northwestern University, Evanston, IL, 2016.
4. Weintrop, D. et al. Evaluating CoBlox: A comparative study of robotics programming environments for adult novices. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* 366, (2018), 1–12; <https://doi.org/10.1145/3173574.3173940>
5. Weintrop, D. and Wilensky, U. To block or not to block, that is the question: Students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (2015), 199–208; <https://doi.org/10.1145/2771839.2771860>
6. Weintrop, D. and Wilensky, U. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)*, 18, 1 (2017), 3; <https://doi.org/10.1145/3089799>

David Weintrop (weintrop@umd.edu) is Assistant Professor in the College of Education and the College of Information Studies at the University of Maryland, College Park, MD, USA. .

Copyright held by author.



Advertise with ACM!

Reach the innovators
and thought leaders
working at the
cutting edge
of computing
and information
technology through
ACM's magazines,
websites
and newsletters.



Request a media kit
with specifications
and pricing:

Ilia Rodriguez

+1 212-626-0686

acmm mediasales@acm.org



Economic and Business Dimensions

A Response to Fake News as a Response to *Citizens United*

How boundaries on speech could free the market for speech.

HOW DO WE fight fake news? Promoting free speech can, at the same time, promulgate false speech and the more vigorously we protect free expression, the more we inadvertently permit deception. The problem is hard. Legislatures across Asia, North America, Europe, and Latin America grapple with it. It touches campaign finance reform, already a thorny issue. As a computer scientist, I argued frequently with my law professor father over ways to solve it. Many of the technological methods we might use to curb false speech run afoul of current law and the very idea of free speech. Yet, the space between law and technology is where we might find better answers.

The problem runs deeper than the technical challenges of machine learning. At one level, reducing Type I errors simply invites those of Type II while training one filter to recognize fake news simply invites adversaries to train other filters to write it.^{3,4} No, at a different level, courts question the desire to regulate fake news at all: they bar intervention in cases of politically protected speech. If fake news is political, it should not be regulated. From a legal theory perspective, there are elements of this policy that are wise—courts should avoid judging political truth—at the same time there are elements that

are unwise—courts should dismantle systems that prevent us from hearing truth. At present, court decisions stifle competing truths and it is here that an old idea from computer science, the Church-Turing thesis, suggests the ban on some forms of intervention should be lifted. Computability theory has much to add to legal practice in the design of better systems.

In 2010, U.S. law became unambiguous. The landmark case *Citizens United v. Federal Election Commission* resolved that the First Amendment forbids suppression of voices “the government deems suspect.”^a This rises to a categorical imperative for political speech. A Supreme Court majority believed that even if corporations, labor unions, and others gain harmful influence, the damage they cause by unrestricted spending is outweighed by the

damage to “democratic process resulting from the restrictions upon free and full discussion.”^b

This followed an earlier decision in *Buckley v. Valeo* that struck down a “restriction on the amount of money a person or group can spend on political communication” because it “reduces the quantity of expression ... the depth of [issues explored], and the size of the audience reached.”^c *Buckley v. Valeo* committed the Court to a position that more speech is always better, a position reaffirmed in *Citizens United* as “there is no such thing as too much speech.”^d As Justice Anthony Kennedy, writing for the majority observed, any “statute which chills speech can and must be invalidated.”^e Together, these two decisions opened the floodgates for unlimited spending by Political Action Committees.

Kennedy’s view is one of pure principle and easy deliberation. Citizens, individually and collectively, have an absolute right to spend on speech. If spending is speech, especially political speech, government regulation is

a *Citizens United v. Federal Election Commission*, 558 U.S. 310 (2010).

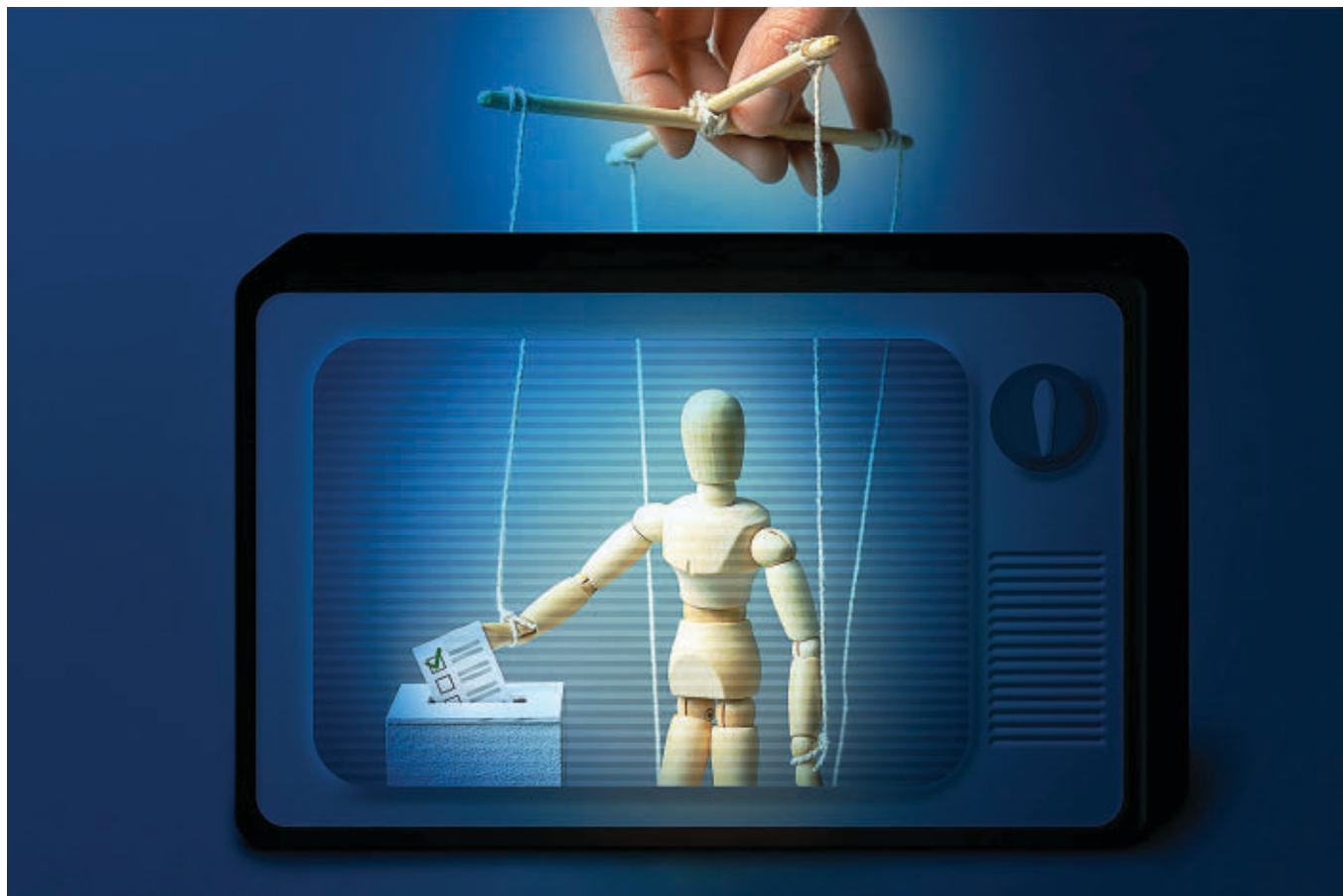
If fake news is political, it should not be regulated.

b Ibid. Kennedy writing for the majority.

c *Buckley v. Valeo*, 424 U.S. 1 (1976).

d Justice Stevens, dissenting. Syllabus: *Citizens United v. Federal Election Commission* <<https://bit.ly/2suTKTQ>>, Supreme Court of the United States, p. 83.

e Op. cit. *Citizens United v. Federal Election Commission*.



limited. Constitutional law scholars describe this view as deontological: it examines only the rightness of the rule and not the nature of the consequence.^f Further, the Supreme Court has long held that government has no business at all regulating the truth or falsity of speech: “under the First Amendment, there is no such thing as a false idea. However pernicious an opinion may seem, we depend for its correction not on the conscience of judges and juries but on the competition of other ideas.”^g

So it was that my father shot down intervention after intervention that I and others conceived. After a year of deliberation, I hit upon two ideas, one that survives strict scrutiny and lives within existing law, and another that provides a logic to overturn that law. It is the second, grounded in philosophy and computer science, that I articulate here in hopes of continuing a conversation cut prematurely short. Sadly, my father died January 29, 2019,

a date late enough to have argued the first idea but too early to argue the second.⁶ For explication of the first idea, I refer readers with more time to a forthcoming paper, “The Problem of Fake News.”^h The gist there is to put friction on harmful externalities not speech, and to invoke intellectual property law so as to motivate news platforms to alter their business models. The second idea operates directly on constitutional law. The gist here is the Court’s logic is self-contradictory and thus invalid. The first deals with the “fakeness” of news, the second with its volume. This column addresses solely the distortions of ideas caused by excesses of money as speech.

Let us set aside *all* the normal exceptions to free speech—incitement to violence, defamation, fraud, trade secret misappropriation, national defense, and so forth. These exceptions create “balancing tests” that recognize competing interests also matter. Preserving life and avoiding corruption do create limits. Set aside *all* these competing interests and

focus only on the truest most politically protected speech. Let us accept the Court’s assumption of the need to protect such speech in order to invalidate its conclusion that it actually *does* protect speech. This essay takes aim at the core logic supporting the Supreme Court’s decision to remove contribution limits on independent expenditures. In effect, the Court holds that infinite amounts of politically protected speech, and its corollary infinite amounts of money, are valid means of protecting “free and full discussion.” Their presumption is the cure for misinformation is more information, yet this turns out to be provably false. As shown in this column, unlimited spending creates a paradox as truths can cancel truths. From there, free and full discussion collapse when falsehoods cancel truths. In seeking to protect speech, the Court has devised means to obstruct speech.

This column rejects the pure principle view that a right to free speech spending, even in protected cases, is absolute and therefore not subject to intervention. My claim is there exist

^f *Gertz v. Robert Welch, Inc.* (1974) 418 U.S. 323, 340.

^g See ssrn.com for an early draft.

^h Op. Cit. *Citizens United v. FEC*.

no absolute rights in any complete system of citizens' rights. To invalidate an absolute right, simply create a contradiction by pitting that right against itself in the manner that mathematicians Church and Turing proved there exist systems with no absolute truths. Consider the statement: The second half of this sentence is absolutely true and the first half of this sentence is absolutely false. The inherent contradiction means the rule is void in at least one application. Not only must a right balance *competing* rights, but also that right must balance *itself*. It is this sense in which one aspect of the *Citizens United* decision is deeply and irretrievably flawed. It presupposes an absolute right. Such a right cannot exist, for any such right pitted against itself is a truth that is untrue, an absolute that is not absolute—a logical contradiction. Three illustrations prove instructive.

Consider a superordinate right, that to life relative to that of speech. We may presuppose that, given a binary and exclusive choice, if a majority of people would prefer a right to live as they choose over a right to speak as they choose, then speech is the subordinate right. Every society, including ours, recognizes at least one limit on an absolute right to life: One person taking a life in self-defense against his or her attempted murder has a reasonable expectation of absolution. The right to life is not so sacrosanct that it cannot be taken and the life that sought to extinguish life is the one more justly forfeit if only one survives. Most societies, excepting ours, recognize limits on political speech. The analogy is that we protect speech absolutely, even though it be used to deny speech to others, yet we do not protect life absolutely when it is used to deny life to others. The logic pertinent to the superordinate right ought to be no less profound in application to the subordinate right.

Consider Justice Oliver Wendell Holmes' famed example of impermissible speech: A person has no right to falsely shout fire in a theater to cause a panic. Such a miscreant cannot claim a right to speech that allows him to endanger the lives of others. We now imagine this miscreant shouting "Fire!" over the voices

Blind adherence to the law, without understanding the due process of law, is its own form of tyranny.

of any people whose ideas he does not want others to hear. His outcry is speech that cancels speech. He could even use a true statement such as "The earth is round!" or a politically protected statement such as "Liberty for all!" to achieve this effect. Indeed, knowing a mode of speech is protected absolutely, he could choose it in order to shout safely even more loudly. A group of such miscreants could shout more loudly still, amplified over the Internet, overriding voices they do not want others to hear. And a group of corporate miscreants could shout yet even more loudly amplified by infinite amounts of money. One message, excluding other messages, is precisely the opposite of "free and full discussion"ⁱ that the Supreme Court sought to protect. In such a case, truthful politically protected speech has stopped others' ears from hearing other truths.

As a further example, consider an absolute right to anonymity and the inherent contradiction in a situation where a claim to that right is used by a party seeking to violate the anonymity of everyone around him. Consider an act of doxing and suppose a criminal unmasks others' private identity yet tries to avoid prosecution by using his right to anonymity as a means to hide from his crime. A reasonable conclusion is that the rights of the violator ought to be suspended precisely in order to protect the rights of everyone else. Hacking a person's private information and publishing it with intent to harm is already illegal. Failure to suspend the right to anonymity, in order to identify and prosecute the

criminal, would be an illogical contradiction that would produce higher levels of doxing. Enforcing the right would negate the reason for granting the right. Lower, not higher, levels of privacy would result.

The string of decisions culminating in *Citizens United* has this same exact effect. Lower, not higher, levels of discourse flow through a market of ideas when special interests literally cannot spend too much.

A right to life that is used to deny a life or a right to speech that is used to deny speech or a right to anonymity that is used to deny anonymity are contradictions in terms. They cannot be enforced without invalidating the logic that led them to be enforced. Such is the effect of the *Citizens United* decision that holds no amount of spending is too much. The argument's logic is flawed.

If a single noisy truth can mask important quiet truths then what hope have we of hearing quiet truths when masked by myriad noisy lies? The market of ideas becomes not just chilly but frozen. To fight false ideas using "the competition of other ideas" we must hear those other ideas.

The theory has practical, not merely academic, value. Russian propaganda models operate under a strategy of "vilify and amplify."¹ This strategy undermines the credibility of any message or messenger that opposes the propagandist while repeating ad infinitum the messages it wants others to hear and, by repetition through different channels, to believe. Repetition of messages is known to increase belief in false claims.⁷ Character assassination is proven to decrease belief in true claims.⁵ Willful failure to act against use of speech by one to override speech of another is not free speech protection but free speech suppression. Without irony, this is the manner of modern free speech suppression in Russia and other totalitarian regimes.¹

In 2010, the top 100 individual donors gave \$73 million to federal candidates, political parties, and super PACs. In 2016, following *Citizens United*, that number rose to more than \$900 million.^j Super PACs, as independent expenditure-only com-

i Op. Cit. *Canadian Security Intelligence Service*.

j Brennan Center for Justice. <https://bit.ly/31Hr3p1>

mittees, may raise unlimited sums of money from corporations, trade unions, and individuals, then spend unlimited sums to promote their political causes.^k They have *no* spending limits. With such resources and such a policy, what stops one partisan from buying all the ad space in swing districts a few weeks prior to an election? The role of government is not to decide political truths but rather to make room for enough sources of truth to enable a just market to decide. Under *Citizens United*, the alternative is a market with but few ideas. The appearance of choice is false, having been predetermined by those who speak loudly enough to set the list of choices.

Consider the absurdity of a law that forbids dissemination of the phrase “Tiananmen Square Uprising.” A just society requires the law be broadly posted so that citizens, aware of the law, take care not to break it. Yet dissemination of the law is a violation of the law, a contradiction. Self-contradictory laws are unjust. And lest those in the West point smug fingers at those in the East, multiple instances of banning Nazi slogans in Germany or hateful ideas on American college campuses exhibit the same contradictory character under the guise of political correctness. We all have this problem. A just and better society requires just and better laws, those that censor harms rather than content, those that balance consequences of over and under reach, and those that make room for multiple ideas, even ones we do not like.

Rules whose enforcement in the extreme yield their automatic repeal cannot logically support any goal offered to justify their application. A contradiction ensues. *A logic is corrupt whose extreme application leads to its own negation.* The deontological view that holds pure principle to be the standard regardless of consequence cannot be correct. Having recognized the problem, the only remaining question is where to draw the boundary on consequence, not how to deny that the boundary exists.

Not only does the line exist, speech crosses that line when it suppresses

others’ speech. Rights are violated not only in the recognized cases of unlawful violence,^l defamation,^m and fraud,ⁿ but also in the unrecognized cases of “vilify and amplify.” In the limit, one voice dominates another, outspends another, and monopolizes an idea market. Then, as with antitrust, strict non-intervention is an abdication of a governing duty to ensure a fair fight in the market of ideas.

One thing my father taught me is that blind adherence to the law, without understanding the due process of law, is its own form of tyranny. We have mechanisms to change bad law. From 1928, when Hilbert posed his problem of identifying consistent systems of absolute truths, it took eight years, until 1936, for mathematicians and philosophers to solve it and recognize its implications. We may thank Church and Turing for correcting our misconceptions of truths as absolute. It has been nine years since *Citizens United*. Now aware of the Church-Turing thesis, legislators and Supreme Court justices ought not take longer to correct their misconceptions of rights as absolute. **C**

^l *Virginia v. Black*, 538 U.S. 343, 359 (2003).

^m *R.A.V. v. City of St. Paul*, 505 U.S. 377, 3399-406 (1992).

ⁿ *Ohralik v. Ohio State Bar Association*, 436 U.S. 447, 456 (1978).

References

1. Academic Outreach (AO) program of the Canadian Security Intelligence Service (CSIS). *Who Said What: Security Challenges of Modern Disinformation*. February 2018.
2. Fish, S. What is the First Amendment for? *New York Times* (Feb. 10, 2010); <https://nyti.ms/2WREPGy>
3. Graham-Cumming, J. How to beat a Bayesian spam filter. 2004; <https://bit.ly/2x7acgZ>.
4. Loder, T., Van Alstyne, M., and Wash, R. An economic response to unsolicited communication. *Advances in Economic Analysis & Policy* 6, 1 (Jan. 2006).
5. Oreskes, N. and Conway, E.M. *Merchants of doubt: How a handful of scientists obscured the truth on issues from tobacco smoke to global warming*. Bloomsbury Publishing USA, 2001.
6. Roberts, S. William Van Alstyne Dies, 84; Often-cited Constitutional scholar. *New York Times* (Feb. 4, 2019); <https://nyti.ms/2KXFAXt>
7. Tucker, J. et al. Social media, political polarization, and political disinformation: A review of the scientific literature, 2018; <https://bit.ly/2ZCehWZ>

Marshall W. Van Alstyne (mva@bu.edu) is Chair of the Information Systems Department and Questrom Chair Professor at Boston University, Boston, MA, USA.

The author thanks *Communications’ Legally Speaking* columnist Pamela Samuelson the Viewpoints co-chairs for their review comments provided during development of this column.

^k <https://bit.ly/1rOzHbW>

Calendar of Events

August 4–8

KDD ‘19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Anchorage, AK, Co-Sponsored: ACM/SIG, Contact: Ankur Teredesai, Email: ankurt@u.washington.edu

August 12–14

ICER ‘19: International Computing Education Research Conference, Toronto, ON, Sponsored: ACM/SIG, Contact: Andrew K. Petersen, Email: andrew.petersen@utoronto.ca

August 18–25

ICFP ‘19: ACM SIGPLAN International Conference on Functional Programming, Berlin, Germany, Sponsored: ACM/SIG, Email: dreyer@mpi-sws.org, Phone: +49 681 9303 8701

August 26–29

FOGA ‘19: Foundations of Genetic Algorithms XV, Potsdam, Germany, Sponsored: ACM/SIG, Contact: Tobias Friedrich, Email: friedrich@hpi.de

August 26–30

27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Tallinn, Estonia, Sponsored: ACM/SIG, Contact: Dietmar Pfahl, Email: dietmar.pfahl@gmail.com

September

September 5–7

NANOCOM ‘19: ACM The 6th Annual International Conference on Nanoscale Computing and Communication, Dublin, Ireland, Sponsored: ACM/SIG, Contact: Tommaso Melodia, Email: melodia@ece.neu.edu



Code Vicious

MUST and MUST NOT

On writing documentation.

Dear KV,

I have joined a small security startup and have been tasked with writing up our internal security processes. The problem is that I am not a writer—I am a software engineer—and whenever I start trying to write about our processes, I either stare at a blank screen until I get frustrated and look away to do something else, or I just wind up writing a lot of sentences that later don't seem to make a lot of sense. I am sure there must be a template that I can work from to get all these things in my head written down in a useful way, but I'm not sure where to look. For example, I want a way to describe to people what they should and shouldn't do with our software and how it must be used so that it provides the security properties they expect. What I see when I try to write about this is a tangled web of spaghetti text.

Tangled

Dear Tangled,

Normally I would reply that the only way to get a good spate of writing done is to go on a three-day bender, and then before sobering up, sit at the keyboard and pour your heart and soul into your text buffer, save your work, and go on another bender before reading what you wrote. It may not work, but the benders ought to be a lot of fun.

In fact, what I am going to do is recommend to you a more than 20-year-old document, RFC 2119. KV has mentioned RFC (Requests for Comments)



before; this is the set of documents going back to the early 1970s in which the Internet protocols and many others are described. For those who are unfamiliar with these documents, they always specify which parts of a protocol are required or optional using a small number of key words: “The key words ‘MUST,’ ‘MUST NOT,’ ‘REQUIRED,’ ‘SHALL,’ ‘SHALL NOT,’ ‘SHOULD,’ ‘SHOULD NOT,’ ‘RECOMMENDED,’ ‘MAY,’ and ‘OPTIONAL’” (See “Key words for use in RFCs to Indicate Requirement Levels”; <https://tools.ietf.org/html/rfc2119>)

The meanings of these words are codified in two pages in ASCII, a now-ancient standard for textual communication. These key words are CAPITALIZED as their only form of emphasis. It turns out it is not necessary to have

fancy formatting in order to communicate clearly; in fact, fancy formatting often distracts from the message you are trying to get across.

No, I am not merely suggesting you use language like this; I believe you **MUST** use these terms as written and then cite the RFC. Getting a group of people to understand your meaning by citing, and perhaps beating them with a well-known and well-written document, can save you a lot of time and trouble. The longer a document is, the more there is to argue over and the more nits there are to pick. Reducing nitpicking saves a lot of time.

A word of caution when using these terms in a security document as you plan to do: The words must be used carefully and for greatest effect. A long

Viewpoint

The Success of the Web: A Triumph of the Amateurs

Connecting the unique factors that influenced the origination and subsequent development of the World Wide Web.

THE WORLD WIDE Web was created during a fervent time for computing, approximately 30 years ago. Back then, the Internet had manifested itself as a global network and it was going beyond its original borders, penetrating both corporate and residential domains. The idea of personal computers had grown for over a decade and it was then consolidating with consumers. Just the right time for a unifying killer app: the Web.

Thirty years later, the success of the Web is unquestionable. It has been growing exponentially in size since its introduction in 1989. It is at the heart of the current retail practices and, more generally, of the corporate world. The Web is so prominent as an Internet-networked application that often the term “Internet” is improperly used to refer to the Web.

Such centrality in our economic structure and lives means the Web, as a political, social, and economic instrument, can be very powerful. So powerful that it can steer a presidential election, turn fantasies into commonly accepted facts, but also educate the less privileged, and give free, broad access to knowledge. The Web of today has become at least as powerful as books have been since Gutenberg’s invention of the metal movable-type printing press in Eu-



rope in approximately 1450. As we are increasingly acknowledging its power, we must also reflect on its sociological, economic, and philosophical consequences. In a past *Communications* column, then-Editor-in-Chief Moshe Vardi looked at the business models that emerged and demands to “build a better Internet,” and again the term Internet here is mostly referring to the Web.⁶ Similarly, Noah Kulwin collects prominent opinions and argues that “Something has gone wrong with the Internet.”⁴ Both arguments are histor-

ical in nature and delve into how business models and data exploitation can turn a resource for humanity into a dangerous weapon.⁷

Alan Kay’s 2012 Interview

Was such power purposely embedded into the Internet and later the Web? Obviously not. The inventors had genuine and benevolent intentions. It is simply very difficult to predict the impact and consequences of an invention. However, what is necessary, and I am glad to see the trend growing, is the

quest for historical reflections. Something that we, computer scientists—researchers of a relatively young discipline—are not sufficiently accustomed to. To put it bluntly, as ACM Turing Award recipient Alan Kay does, “The lack of interest, the disdain for history is what makes computing not-quite-a-field.” While we want computing to be recognized, as it should be, as a field.

The Alan Kay quote comes from a 2012 interview that appeared in *Dr. Dobbs Journal*.³ The interview is full of insightful and poignant remarks. Kay identifies the Web as an example of a technology that was proposed while ignoring the history of the field it was contributing to. He explains, “The Internet was done so well that most people think of it as a natural resource like the Pacific Ocean, rather than something that was man-made. When was the last time a technology with a scale like that was so error-free? The Web in comparison is a joke. The Web was done by amateurs.”

The term “amateur” does not necessarily have a negative connotation; it can simply denote someone who engages in an activity for the sole pleasure of doing so. Someone that does not have the professional background and voluminous experience to carry it out. In this sense, Kay is right. The Web was not created by someone with an education in computer science, but in physics. ACM Turing Award recipient Tim Berners-Lee worked on his hypertextual system as a personal project while at the European Organization for Nuclear Research (CERN). In his 1999 book published to celebrate 10 years of the Web, he recalls, “I wrote it in my spare time and for my personal use, and for no loftier reason than to help me remember the connections among the various people, computers, and projects at the lab [CERN].”²

The Web is a distributed hypertext system and, when it was proposed, it had already many ancestors such as Trigg’s Textnet, Brown University’s Intermedia, Gopher, and HyperCard.¹ Out of these, I highlight two fundamental proposals that came into existence in the 1960s—more than two decades before the Web. Ted Nelson designed and implemented a system called Xanadu and coined the term hypertextuality in approximately 1963. In the same period, ACM Turing Award recipient Doug Engelbart led a

There is a natural tension between a professional design and an amateuristic one.

large project at SRI called the OnLine System (NLS) in which information had a hypertextual organization, among many other pioneering features. NLS was presented publically in 1968 in what would be later known as the “Mother of All Demos.” Berners-Lee appears to be unfamiliar with these systems and in his book admits to becoming aware of NLS only five years after having proposed the Web. In summary, the lack of knowledge of the relevant background, an education in a loosely related discipline, and its genesis as a “hobby” project, all conform to amateurism.

There is a natural tension between a professional design and an amateuristic one. The professional design builds on theoretical foundations, best practices, and knowledge of the state of the art. This helps prevent the repetition of mistakes and poor designs. At the same time, it puts a rich set of constraints and may result in overdesigned, secondary components. On the contrary, an amateuristic design frees the creator from cultural legacies and possible biases. It promotes creativity, at the price of increasing the risk of naïve, avoidable flaws.

Patching the Web

In its original 1989 design, the Web is a stateless, distributed, linked information repository. Based on three simple ingredients—HTML, HTTP, and URL—it leaves great freedom to the way information is created, exchanged, and distributed. Furthermore, there is very little structure and meaning given to the exchange units of the components of the Web, that is, marked-up text enclosed in an application-level protocol. Such simple design has made it possible to easily build components for the Web (servers, browsers, content editors) and

has proved a crucial factor for its broad adoption and rapid success. At the same time, it does not provide for well-defined hypertextual systems. When compared with a system like Xanadu, Ted Nelson is highly critical, “HTML is precisely what we were trying to *prevent*—ever-breaking links, links going outward only, quotes you can’t follow to their origins, no version management, no rights management.” A valid point for reflection that should also be considered is the current broader picture of copyright management and information source determination. Though the Web creator defends his choice as a pondered design decision, “When I designed HTML for the Web, I chose to avoid giving it more power than it absolutely needed—a principle of least power, which I have stuck to ever since. I could have used a language like Donald Knuth’s TEX, which though it looks like a markup language is in fact a programming language. It would have allowed very fancy typography and all kinds of gimmicks, but there would have been little chance of turning Web pages into anything else. It would allow you to express absolutely anything on the page, but would also have allowed Web pages that could crash, or loop forever.”

Kay’s criticism to the Web is orthogonal and touches on the computational side rather than the purely hypertextual one. Instead of a Web browser interpreting text, he favors an operating system-level container capable of hosting distributed objects in execution. This would mean a computational Web with well-defined semantics of execution, rather than a stateless distributed hypertext information repository.

The Web of today significantly differs from the original design of 1989. With its growth and success, it had to cater to many needs and requirements. The stateless nature of the interactions was one of the first shortcomings to be addressed; hence, cookies were introduced. The origin of cookies can be traced back to a request that came from a client of Netscape who wanted to store online transaction information outside of their servers for the purpose of making a Web-based shop. The feature was added to the September 1994 Netscape browser release, and discussions on the specification of the cookies started soon after, reaching the status of an

Distinguished Speakers Program

A great speaker can make the difference between a good event and a WOW event!

Students and faculty can take advantage of ACM's Distinguished Speakers Program to invite renowned thought leaders in academia, industry and government to deliver compelling and insightful talks on the most important topics in computing and IT today. ACM covers the cost of transportation for the speaker to travel to your event.

speakers.acm.org



Association for
Computing Machinery

It is thanks to the interested and volunteer effort of millions of people that the Web has evolved into what it is today.

agreed RFC, called HTTP State Management Mechanism, in 1997. Similarly, we saw the appearance of scripting languages, embedded virtual machines, graphical rendering frameworks, and so on. What I claim is the Web has undergone continuous patching that has slowly and gradually brought it into the direction of a computational infrastructure. These patches have diverse origins, making the Web the result of a collective engineering effort. It is in fact thanks to the interested and volunteer effort of millions of people that the Web has evolved into what it is today. In my recent book,¹ I identify and discuss five major patches that have to do with the computational nature of the Web or—better said—with the lack of it in the original design.

One can similarly look at the evolution from the point of view of security and content presentation patches. In the book, I also reflect on the engineering consequences of starting from an amateuristic design then collectively patched; the end result being the empowering of global adoption. I identify the crucial factors for the success of a patched Web in an evolving landscape of hypertextual proposals.

Why has the Web succeeded where other similar, coeval systems failed? The end-to-end argument may very well apply here.⁵ Based on a series of experiences with networked applications at MIT, the authors suggest that even well-engineered layered architectures may cause high inefficiency in development and operation of systems. Working at the application level is the most practical and effective solution. To bring the argument to the Web case, the reasoning goes that a very well designed system would have been impractical or im-

possible to build—like Xanadu—while something like the Web with a simple application-level pattern and related technologies was the way to widespread deployment and use.

Conclusion

Can an amateuristic design succeed? History tells us the answer is yes. For the Web, many would even argue that amateurism is the winning factor; few that it is the sole possible one. The indisputable success of the Web, however, still leaves the open-minded researcher wondering about what the world would have looked like today if a hypertextually, semantically well-defined system of moving computational objects would have succeeded instead. Would we have the same issues of computational efficiency and security? Would problems such as data privacy and protection, copyright management, and fake news be alleviated? And, most interestingly, would it have been as successful as the current Web is? History has not favored systems that preceded or competed with the Web. Xanadu, despite its thorough design, never gained any adoption, Gopher succumbed to the simplicity and openness of the Web, HyperCard enjoyed a temporary and confined success. Can we conclude that amateuristic simplicity always wins adoption over complex engineering? It is difficult to say. For sure, the Web has had a transformational effect on society. Something similar to the long-lasting effects of printed books; something that could accompany us for many generations. 

References

1. Aiello, M. *The Web Was Done by Amateurs: A Reflection on One of the Largest Collective Systems Ever Engineered*. Springer-Nature, 2018.
2. Berners-Lee, T. and Fischetti, M. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. Harper, San Francisco, 1999.
3. Binstock, A. Interview with Alan Kay. *Dr. Dobbs's Journal*, online edition, 2012; <http://www.drdobbs.com/architecture-and-design/interview-with-alan-kay/240003442>.
4. Kulwin, N. The Internet apologizes. *New York Magazine* (Mar. 2018), <https://slct.al/2GZ6hGr>.
5. Saltzer, J.H., Reed, D.P., and Clark, D.D. End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)* 2, 4 (Mar. 1984), 277–288.
6. Vardi, M.Y. How the hippies destroyed the Internet. *Commun. ACM* 61, 7 (July 2018), 9.
7. Vardi, M.Y. To serve humanity. *Commun.* 62, 7 (July 2019), 7.

Marco Aiello (aiellom@ieee.org) is Professor of Service Computing at the University of Stuttgart (D) and member of the European Academy of Sciences and Arts.

Copyright held by author.

CCGrid 2020

20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing

May 11-14, 2020, Melbourne, Australia

<http://www.cloudbus.org/ccgrid2020>

CALL FOR PAPERS

Tremendous advances in processing, communication and systems/middleware technologies are leading to new paradigms and platforms for computing, ranging from computing *Clusters* to widely distributed *Clouds* and emerging Internet computing paradigms such as *Fog/Edge Computing for Internet of Things (IoT)/Big Data applications*. CCGrid is a series of very successful conferences, sponsored by the IEEE Computer Society Technical Committee on Scalable Computing (TCSC) and ACM, with the overarching goal of bringing together international researchers, developers, and users and to provide an international forum to present leading research activities and results on a broad range of topics related to these platforms and paradigms and their applications. The conference features keynotes, technical presentations, posters, workshops, tutorials, as well as the SCALE challenge featuring live demonstrations and the IC FEC 2020 conference.

In 2020, CCGrid will return to Melbourne, Australia, to celebrate its 20th anniversary. CCGrid 2020 will have a special focus on three important issues that are significantly influencing all aspects of Cluster, Cloud, and Internet computing: Adaptive Elastic Computing, Green Computing, and Cyber-Physical Computing. Topics of interest include, but are not limited to:

- **Internet Computing Frontiers:** Edge, Fog, Serverless, Lambda, Streaming. More decentralized approaches to cloud computing. Sensor data streaming and computation on the edges of the network. Function as a Service (FaaS), serverless computing, lambda computing.
- **Architecture, Networking, Data Centers:** Service oriented architectures. Utility computing models. IaaS, PaaS, SaaS, *aaS paradigms. Micro-datacenter, cloudlet, edge, or fog computing infrastructure. Virtualized hardware: GPUs, tensor processing units, FPGAs.
- **Storage and I/O Systems**
- **Programming Models and Runtime Systems:** Programming models, languages, and systems. Virtualization, containers, and middleware technologies.
- **Resource Management and Scheduling:** Resource allocation algorithms, profiling, modeling. Cluster, cloud, and internet computing scheduling and meta-scheduling.
- **Performance Modelling and Evaluation:** Performance models. Monitoring and evaluation tools. Analysis of system/application performance.
- **Cyber-Security and Privacy:** Cloud security and trust. Access control. Data privacy and integrity. Regulation.
- **Sustainable and Green Computing:** Hardware/software/

application energy efficiency. Power, cooling and thermal awareness.

- **Applications:** Data Science, Artificial Intelligence, Cyber-Physical Systems and applications to real and complex problems in science, engineering, and business.

PAPER SUBMISSION

Authors are invited to submit papers electronically. Submitted manuscripts should be structured as technical papers and may not exceed 10 letter size (8.5 x 11) pages including figures, tables and references using the IEEE format for conference proceedings. All manuscripts will be reviewed and will be judged on correctness, originality, technical strength, significance, quality of presentation, and relevance to the conference attendees.

Submitted papers must represent original unpublished research that is not currently under review for any other conference or journal. Papers not following these guidelines will be rejected without review and further action may be taken, including (but not limited to) notifications sent to the heads of the institutions of the authors and sponsors of the conference. Submissions received after the due date, exceeding length limit, or not appropriately structured may also not be considered. Authors may contact the conference chairs for more information. The proceedings will be published through the IEEE Press, USA and will be made online through the IEEE and ACM Digital Libraries.

CHAIRS & COMMITTEES

General Chair

Rajkumar Buyya, University of Melbourne, Australia

General Vice Chairs

Dhabaleswar Panda, Ohio State University, USA

Hai Jain, HUST, China

Massimo Villari, The University of Messina, Italy

Program Committee Co-Chairs

Carlos A Varela, RPI, USA

Laurent Lefevre, INRIA, France

Workshops Co-Chairs

George Pallis, University of Cyprus, Cyprus

Borja Sotomayor, The University of Chicago, USA

IMPORTANT DATES (TENTATIVE)

Papers Due:	10 December 2019
Notification of Acceptance:	30 January 2020
Camera Ready Papers Due:	25 February 2020

Article development led by **acmqueue**
queue.acm.org

Five diverse technology companies show how it's done.

BY NATASHA NOY, YUQING GAO, ANSHU JAIN,
ANANT NARAYANAN, ALAN PATTERSON, AND JAMIE TAYLOR

Industry-Scale Knowledge Graphs: Lessons and Challenges

KNOWLEDGE GRAPHS ARE critical to many enterprises today: They provide the structured data and factual knowledge that drive many products and make them more intelligent and “magical.”

In general, a knowledge graph describes objects of interest and connections between them. For example, a knowledge graph may have nodes for a movie, the actors in this movie, the director, and so on. Each node may have properties such as an actor's name and age. There may be nodes for multiple movies involving a particular actor. The user can then traverse the knowledge graph to collect information on all the movies in which the actor appeared or, if applicable, directed.

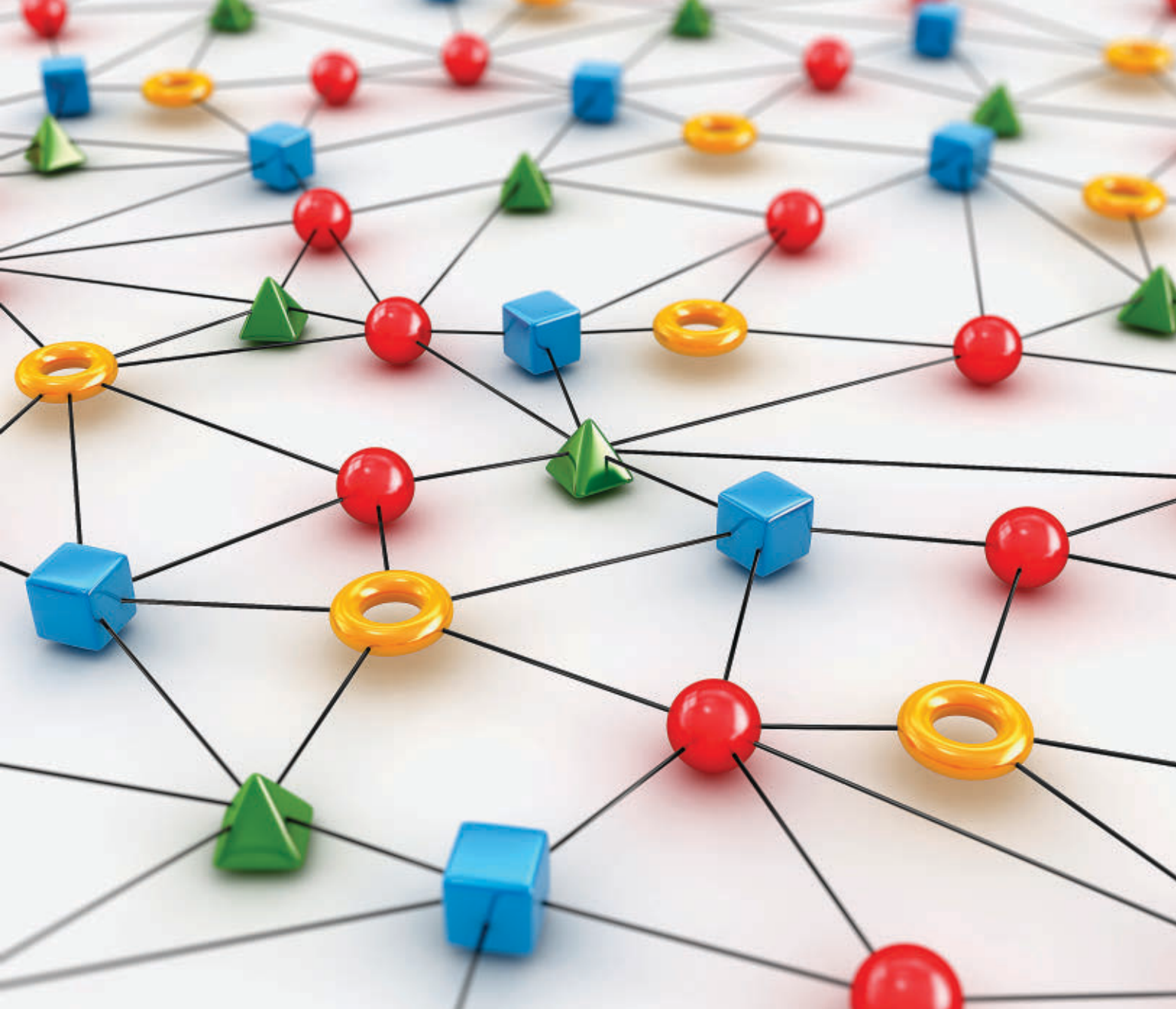
Many practical implementations impose constraints on the links in knowledge graphs by defining a *schema* or *ontology*. For example, a link from a movie to its director must connect an object of type Movie to an object of type Person. In some cases the links themselves might have their own properties: a link connecting an actor and a movie might have the name of the specific role the actor played. Similarly, a link connecting a politician with a specific role in government might have the time period during which the politician held that role.

Knowledge graphs and similar structures usually provide a shared substrate of knowledge within an organization, allowing different products and applications to use similar vocabulary and to reuse definitions and descriptions that others create. Furthermore, they usually provide a compact formal representation that developers can use to infer new facts and build up the knowledge—for example, using the graph connecting movies and actors to find out which actors frequently appear in movies together.

This article looks at the knowledge graphs of five diverse tech companies, comparing the similarities and differences in their respective experiences of building and using the graphs, and discussing the challenges that all knowledge-driven enterprises face today. The collection of knowledge graphs discussed here covers the breadth of applications, from search, to product descriptions, to social networks:

- Both Microsoft's Bing knowledge graph and the Google Knowledge Graph support search and answering questions in search and during conversations. Starting with the descriptions and connections of people, places, things, and organizations, these graphs include general knowledge about the world.

- Facebook has the world's largest social graph, which also includes information about music, movies, celebrities, and places that Facebook users care about.



► The Product Knowledge Graph at eBay, currently under development, will encode semantic knowledge about products, entities, and the relationships between them and the external world.

► The Knowledge Graph Framework for IBM's Watson Discovery offerings addresses two requirements: one focusing on the use case of discovering nonobvious information, the other on offering a "Build your own knowledge graph" framework.

The goal here is not to describe these knowledge graphs exhaustively, but rather to use the authors' practical experiences in building knowledge graphs in some of the largest technology companies today as a scaffolding to highlight the challenges that any

enterprise-scale knowledge graph will face and where some innovative research is needed.

What's In a Graph? Design Decisions

Let's start by describing the five knowledge graphs and the decisions that went into each design and determining the scope of each graph. The different applications and product goals for each one resulted in different approaches and architectures, though many of the challenges are shared by all the enterprises. The accompanying table summarizes the properties of these knowledge graphs.

Microsoft. Engineers and scientists at Microsoft have been working on large-scale graphs for many years. This

work included building the end-to-end system from the underlying research, as well as a global-scale service for hundreds of millions of users. Across the company, there are several major graph systems, each bringing specific challenges around creating the graph and keeping it up to date. Many different products can use a knowledge graph to bring value to consumers. The following are some of the graphs at Microsoft:

► The Bing knowledge graph contains information about the world and powers question answering on Bing. It contains entities such as people, places, things, organizations, locations, and so on, as well as the actions that a user might take (for example, to play a video or buy a song). This is the largest knowledge graph at Microsoft, as its

aim is to contain general knowledge about the entire world.

► The Academic graph is a collection of entities such as people, publications, fields of study, conferences, and locations. It allows a user to see connections between researchers and pieces of research that may otherwise be hard to determine.

► The LinkedIn graph contains entities such as people, jobs, skills, companies, locations, and so on. The LinkedIn Economic graph is based on 590 million members and 30 million companies, and is used to find economy-level insights for countries and regions.

The Bing search engine displays a knowledge panel from the Bing knowledge graph when there is additional useful information. For example, a search for the film director James Cameron reveals information such as his date of birth, height, movies and TV shows he directed, previous romantic partners, TED Talks he gave, and Reddit “Ask Me Anything” questions and answers (through partnership with Reddit). A search for a different type of entity returns completely different information—for example, searching for “Woodblock restaurant” results in an extract from the menu, professional critic and user reviews, as well as the option to book a table.

All of these graph systems—as would probably be the case with any large graph system—have three key determinants of quality and usefulness:

► **Coverage.** Does the graph have all the required information? The answer

is always effectively no, because developers are always looking for new ways to provide value to users and for new sources of information.

► **Correctness.** Is the information correct? How do you know if two sources of information are actually about the same fact, and what do you do if they conflict? Answering these questions is a huge area of study and investment by itself.

► **Freshness.** Is the content up to date? It may have been correct at one time but gone stale. Freshness will vary for something that changes almost constantly (a stock price) compared with something that changes rarely (the capital of a country), with many different kinds of information in between.

To generate knowledge about the world, data is ingested from multiple sources, which may be very noisy and contradictory, and must be collated into a single, consistent, and accurate graph. The final fact that a user sees is the tip of an iceberg—a huge amount of work and complexity is hidden below. For example, there are 200 Will Smiths in Wikipedia alone, and the Bing knowledge result for the actor Will Smith is composed from 108,000 facts taken from 41 websites.

From search to conversation. Knowledge graphs power advanced AI, allowing single queries to be turned into an ongoing conversation. Specifically, this allows a user to have a conversation with the system and to have the system maintain the context through each turn of the conversation. For example, in a

future scenario a user could say to Bing, “Show me all the countries in the world where it’s over 70 degrees Fahrenheit right now,” and once the system returns the answer, the user could say, “Show me those within a two-hour flight.”

You can take the same idea further to enable a full conversational experience. For example, a user could say, “I want to travel to NYC two days before Thanksgiving and stay for a week,” and the system would use the underlying knowledge graph to make sense of the query and then request missing pieces of information. In this example, the system needs to know that “NYC” could mean “JFK Airport” and that Thanksgiving is November 22. It then must know how to carry out a flight search, which requires a start location and a destination location. The system would then have to know the next line of the conversation must determine the start location, so it would say, “Okay, booking a flight to JFK from November 20 to 27. Where will you be flying from?”

Google. With more than 70 billion assertions describing a billion entities, the Google Knowledge Graph covers a wide swath of subject matter and is the result of more than a decade of data-contribution activity from a diverse set of individuals, most of whom have never had experience with knowledge-management systems.

Perhaps more important, Knowledge Graph serves as a long-term, stable source of class and entity identity that many Google products and features use behind the scenes. Outside users and developers can observe these features when they use services such as YouTube and Google Cloud APIs. This focus on identity has allowed Google to transition to “things not strings.” Rather than simply returning the traditional “10 blue links,” Knowledge Graph helps Google products interpret user requests as references to concepts in the world of the user and to respond appropriately.

Google’s Knowledge Graph is perhaps most visible when users issue queries about entities and the search results include an array of facts about the entities that are served from Knowledge Graph. For example, a query for “I.M. Pei” produces a small panel in the search results with information about the architect’s education, awards, and

Common characteristics of the knowledge graphs.

	Data model	Size of the graph	Development stage
Microsoft	The types of entities, relations, and attributes in the graph are defined in an ontology.	~2 billion primary entities, ~55 billion facts	Actively used in products
Google	Strongly typed entities, relations with domain and range inference	1 billion entities, 70 billion assertions	Actively used in products
Facebook	All of the attributes and relations are structured and strongly typed, and optionally indexed to enable efficient retrieval, search, and traversal.	~50 million primary entities, ~500 million assertions	Actively used in products
eBay	Entities and relation, well-structured and strongly typed	Expect around 100 million products, >1 billion triples	Early stages of development and deployment
IBM	Entities and relations with evidence information associated with them.	Various sizes. Proven on scales documents >100 million, relationships >5 billion, entities >100 million	Actively used in products and by clients

the significant structures he designed.

The Knowledge Graph also recognizes that certain kinds of interactions can take place with different entities. A query for “The Russian Tea Room” provides a button to make a reservation, while a query for “Rita Ora” provides links to her music on various music services.

At the scale of the Google Knowledge Graph, a single individual can not remember, let alone manage, the detailed structures used throughout the graph. To ensure the system remains consistent over time, Google built its Knowledge Graph from a basic set of low-level structures. It replicated similar structures and reasoning mechanisms at different levels of abstraction, conceptually bootstrapping the structure from a number of basic assertions. For example, to check specific invariant constructions, Google leveraged the idea that types were themselves instances of types to introduce the notion of metatypes. It could then reason about the metatypes to verify the finer-grained types did not violate the invariants it was interested in. It can validate that time-independent identities are not subclasses of structures, which are time-dependent. This scalable level of abstraction was relatively easy to add in a manner that worked out of the box because it was built upon the same low-level entailments on which the rest of the system was based.

This meta-level schema also allows validation of data at scale. For example, you can validate that painters existed before their works of art were created by identifying the painters as the “origin” of their painted work “products” and applying a general check on all relations between these metaclasses.

At a slightly higher conceptual level, Knowledge Graph “understands” that authors are distinct from their creative works, even though these entities are frequently conflated in colloquial expressions. Similarly, creative works may have multiple expressions that are themselves distinct. This ontological knowledge helps maintain the identity of entities as the graph grows.

Building the Knowledge Graph through these self-describing layers not only simplifies consistency checking by machines, but also makes the Knowledge Graph easier for internal users to understand. Once new developers have



The system would use the underlying knowledge graph to make sense of the query and then request missing pieces of information.



been trained on the fundamentals of Knowledge Graph organization, they can understand the full extent of its inventory of structures. Similarly, by keeping the structure of the graph tied to a few core principles and exposing meta-relations explicitly in schemas, finding and comprehending new schema structures is simplified for internal developers.

Facebook is known for having the world’s largest social graph. Facebook engineers have built technology over the past decade to enable rich connections between people. Now they are applying the same technology to building a deeper understanding of not just people, but also the things that people care about.

By modeling the world in a structured manner and at scale, Facebook engineers were able to unlock use cases that a social graph by itself could not fulfill. Even seemingly simple things, such as structured understanding of music and lyrics when combined with software that detects when people are referencing them, can enable serendipitous moments between individuals. Many experiences in Facebook’s products today, such as helping people plan movie outings on Messenger, are powered by the knowledge graph.

Facebook’s knowledge graph focuses on the most socially relevant entities, such as those that are most commonly discussed by its users: celebrities, places, movies, and music. As the Facebook knowledge graph continues to grow, developers focus on those domains that have the greatest chance of delivering utility and delightful user experiences.

Coverage, correctness, structure, and constant change all drive the design of the Facebook knowledge graph:

- Coverage means being exhaustive in a domain that is being modeled. The default stance is multiprovider, which means that the entire graph-production system is built with the assumption that data will be received from multiple sources, all providing (sometimes conflicting) information about overlapping sets of entities. The Facebook knowledge graph deals with the conflicting information in one of two ways: the information is deemed to be sufficiently low confidence to justify dropping it; or conflicting views are incorporated into the entity by retaining

provenance and an inferred confidence level about the assertion.

- Correctness does not mean the knowledge graph always knows the “right” value for an attribute, but rather that it is always able to explain why a certain assertion was made. Therefore, it keeps provenance for all data that flows through the system, from data acquisition to the serving layer.

- Structure means the knowledge graph must be self-describing. If a piece of data is not strongly typed or does not fit the schema describing the entity, then the graph attempts to do one of the following: convert the data into the expected type (for example, performing simple type coercion, handling incorrectly formatted dates); extract structured data that matches the type (for example, run natural language processing, NLP) on unstructured text such as user reviews to convert into typed slots); or leave it out entirely.

- Lastly, the Facebook knowledge graph is designed for constant change. The graph is not a single representation in a database that is updated when new information is received. Instead, the graph is built from scratch, from the sources, every day, and the build system is idempotent—producing a complete graph at the end of it.

An obvious place for a Facebook knowledge graph to start is the Facebook pages ecosystem. Businesses and people create pages on Facebook to represent a huge range of ideas and interests. Furthermore, having the owner of an entity make assertions about it is a valuable source of data. As with any crowd-sourced data, however, it is not without its challenges.

Facebook pages are very public facing, and millions of people interact with them every day. Thus, the interests of a page owner don’t always align with the requirements of a knowledge graph.

Most commonly, pages and entities do not have a strict 1:1 mapping, as pages can represent collections of entities (for example, movie franchises). Data can also be incomplete or very unstructured (blobs of text), which makes it more difficult to use in the context of a knowledge graph.

Facebook’s biggest challenge has been to leverage data found on its pages and to combine it with other more



The Discovery use case creates new knowledge that is not directly present in domain documents or data sources.



structured sources of data to achieve the goals of a clean, structured knowledge graph. A useful tool for Facebook has been to think of the graph as the model and a Facebook page as the view—a projection of an entity or collection of entities that reside in the graph.

eBay is building its Product Knowledge Graph, which will encode semantic knowledge about products, entities, and their relationships with each other and the external world. This knowledge will be key to understanding what a seller is offering and a buyer is looking for and intelligently connecting the two, a key part of eBay’s marketplace technology.

For example, eBay’s knowledge graph can relate products to real-world entities, defining the identity of a product and why it might be valuable to a buyer. A basketball jersey for the Chicago Bulls is one product, but if it is signed by Michael Jordan, it is a very different product. A postcard from 1940 in Paris might be just a postcard; knowing that Paris is in France and that 1940 is during World War II changes the product entirely.

Entities in the knowledge graph can also relate products to each other. If a user searches for memorabilia of Lionel Messi and the graph indicates that Lionel Messi plays for Futbol Club Barcelona, then, maybe, merchandise for that club is of interest, too. Perhaps memorabilia for other famous Barcelona players will be of interest to this shopper. Related merchandise should include soccer-based products such as signed shirts, strips, boots, and balls. This idea can extend from sports to music, film, literature, historical events, and much more.

Just as important as entity relations is understanding the products themselves and their relationships. Knowing that one product is an iPhone and another is a case for an iPhone is obviously important. But the case might fit some phones and not others, so eBay needs to model the parts and accessory sizes. Knowing the many variants and relationships of products is also important: Which products are manufacturer variants of one product? Do they come in different sizes, capacities, or colors? Which are comparable—meaning they have mostly the same specifications but perhaps different brands or colors? The system also needs to understand

products that go together as a set, say in bundles, kits, or even fashion outfits.

As with other knowledge graphs, eBay must cope with scale. At any one time there may be more than one billion active listings across thousands of categories. These listings might include hundreds of millions of products and tens of billions of attributes specified for those products.

There are several different users of the eBay Knowledge Graph, and these users have very different service-level requirements. When the search service needs to understand a user's query, the knowledge graph must power an answer that takes milliseconds. At the other end of the scale, large graph queries could take hours to run.

To cope with these challenges, eBay engineers have designed an architecture that provides them with flexibility, while ensuring that the data is consistent. The knowledge graph uses a replicated log for all writes and edits to the graph. The log provides a consistent ordered view of the data. This approach enables multiple back-end data stores that meet different use cases. Specifically, there is a flattened document store for serving search queries with low latency and a graph store for doing long-running graph analysis. Each of these stores simply appends its operations to the write log and gets the additions and edits to the graph in a guaranteed order. As a result, each store will be consistent.

IBM developed its Knowledge Graph Framework, used by Watson Discovery Services and its associated offerings, which have been deployed in many industry settings outside of IBM. IBM Watson uses the Knowledge Graph Framework in two distinct ways: First, the framework directly powers Watson Discovery, which focuses on using structured and unstructured knowledge to discover new, nonobvious information, and the associated vertical offerings on top of Discovery; second, the framework allows others to build their own knowledge graphs with the prebuilt knowledge graph as the core.

The Discovery use case creates new knowledge that is not directly present in domain documents or data sources. This new knowledge can be surprising and anomalous. While search and exploration tools access knowledge that

is already available in the sources available to the system, they are necessary but not sufficient for Discovery. Nonobvious discovery includes new links between entities (for example, a new side effect of a drug, an emerging company as an acquisition target or sales lead), a potential new important entity in the domain (for example, a new material for display technologies, a new investor for a particular investment area), or changing significance of an existing entity (an increasing stake by an investor in an organization, or increasing interaction between a person of interest and some criminal in an intelligence-gathering scenario).

Given its wide enterprise customer base applying cognitive technologies in various domains, IBM focused on creating a framework for clients and client teams to build their own knowledge graphs. Industry teams at IBM leverage this framework to build domain-specific instances. Clients exist in several domains ranging from consumer-oriented research in banking and finance, insurance, IT services, media and entertainment, retail, and customer service, to industries focused almost entirely on deep discovery—especially scientific domains such as life sciences, oil and gas, chemicals and petroleum, defense, and space exploration. This breadth requires the framework have all of the machinery that clients need to build and manage a knowledge graph themselves. Some of the key technologies built into the framework include document conversion, document extraction, passage storage, and entity normalization.

The following are some of the key insights and lessons that IBM engineers learned from both building the knowledge graph for Watson Discovery and deploying the system in other industries.

► *Polymorphic stores offer a solution.* The IBM Watson Knowledge Graph uses a polymorphic store, supporting multiple indices, database structures, in-memory, and graph stores. This architecture splits the actual data (often redundantly) into one or more of these stores, allowing each store to address specific requirements and workloads. IBM engineers and researchers addressed a number of challenges such as keeping these multiple stores in sync,

allowing communication between the stores through microservices, and allowing ingestion of new knowledge or reprocessing raw data in a way that does not require reloading or rebuilding the entire graph.

► *Evidence must be primitive to the system.* The main link between the real world (which developers often try to model) and the data structures holding the extracted knowledge is the “evidence” of the knowledge. This evidence is often the raw documents, databases, dictionaries, or image, text, and video files from which the knowledge is derived. When it comes to making pointed and useful contextual queries during a discovery process, the metadata and other associated information often play a role in inference of the knowledge. Thus, it is critical not to lose the linkage between the relationships stored in the graph and where those relationships come from.

► *Push entity resolution to runtime through context.* Resolving ambiguous references to entities referenced by partial names, surface forms, or multiple entities having the same names is a classic problem in understanding natural language. In the field of knowledge discovery, however, developers often look for the nonobvious patterns where an entity is not behaving in its well-understood form or appears in a novel context. Thus, a disambiguation of an entity too early in the process of knowledge-graph creation conflicts with the very goal of discovery. It is better to leave those utterances unresolved or disambiguate them to multiple entities, and then during runtime use the context of the query to resolve the entity name.

Challenges Ahead

The requirements, coverage, and architectures of the knowledge graphs discussed here differ quite a bit, but many of the challenges appear consistently across most implementations. These include challenges of scale, disambiguation, extraction of knowledge from heterogeneous and unstructured sources, and managing knowledge evolution. These challenges have been at the forefront of research for years, yet they continue to baffle industry practitioners. Some of the challenges are present in some of the systems but may

be less relevant in other settings.

Entity disambiguation and managing identity. While entity disambiguation and resolution is an active research area in the semantic Web, and now in knowledge graphs for several years, it is almost surprising that it continues to be one of the top challenges in the industry almost across the board. In its simplest form, the challenge is in assigning a unique normalized identity and a type to an utterance or a mention of an entity. Many entities extracted automatically have very similar surface forms, such as people with the same or similar names, or movies, songs, and books with the same or similar titles. Two products with similar names may refer to different listings. Without correct linking and disambiguation, entities will be incorrectly associated with wrong facts and result in incorrect inference downstream.

While these problems might seem obvious in smaller systems, when identity management must be done with a heterogeneous contributor base and at scale, the problem becomes much more challenging. How can identity be described in a way that different teams can agree on it and know what the other teams are describing? How can developers be sure to have enough human-readable information to adjudicate conflicts?

Type membership and resolution. Most current knowledge-graph systems allow each entity to have multiple types, and the specific type may matter in different circumstances. For example, Barack Obama is a person, but also a politician and actor—a vastly more popular politician and not a very well-known actor. Cuba can be a country or may refer to its government. In some cases, knowledge-graph systems defer the type assignment to runtime: Each entity describes its attributes, and the application uses a specific type and collection of attributes depending on the user task.

While criteria for class membership might be straightforward early on, as the universe of instances grows, enforcing these criteria while maintaining semantic stability becomes challenging. For example, when Google defined the category for “sports” in its knowledge graph, e-sports did not exist. So, how does Google maintain the category

identity for sports while also including e-sports?

Managing changing knowledge. An effective entity-linking system also needs to grow organically based on its ever-changing input data. For example, companies may merge or split, and new scientific discoveries may break an existing entity into multiples. When a company acquires another company does the acquiring company change identity? What about a division being spun out? Does identity follow the acquisition of the rights to a name?

While most knowledge-graph frameworks are becoming efficient at storing a point-in-time version of a knowledge graph and managing instantaneous changes to the knowledge graphs to evolve the graph, there is a gap in being able to manage highly dynamic knowledge in the graphs.⁴ A fundamental understanding of temporal constructs, history, and change with history is needed to capture these changes. Furthermore, the ability to manage updates through multiple stores (for example, IBM’s polymorphic stores) is necessary.

There are a lot of considerations around the integrity of the update process, eventual consistency, conflicting updates, and, simply, runtime performance. There may be an opportunity to think of different variations of existing distributed data stores designed to handle incremental cascade updates. It is also critical to manage changing schemas and type systems, without creating inconsistencies with the knowledge already in the system. Google, for example, addresses this problem by conceptualizing the metamodel layer into multiple layers. The basic lower layers remain fairly constant and higher levels are built through the notion of metatypes (which are really instances of types), which can be used to enrich the type system.

Knowledge extraction from multiple structured and unstructured sources. Despite the recent advances in natural language understanding, the extraction of structured knowledge (which includes entities, their types, attributes, and relationships) remains a challenge across the board. Growing the graphs at scale requires not only manual approaches, but also unsupervised and semi-supervised knowledge extraction

from unstructured data in open domains.

For example, in the eBay Product Knowledge Graph, many graph relationships are extracted from unstructured text in listings and seller catalogs; the IBM Discovery knowledge graph relies on documents as evidence for the facts represented in the graphs. Traditional supervised machine-learning frameworks require labor-intensive human annotations to train knowledge-extraction systems. This high cost can be alleviated or eliminated by adopting fully unsupervised approaches (clustering with vector representations) or semi-supervised techniques (distant supervision with existing knowledge, multi-instance learning, active learning, and so on). Entity recognition, classification, text, and entity embeddings all prove useful tools to link our unstructured text to entities we know about in the graph.³

Managing operations at scale. It is probably not surprising that all of the knowledge-graph systems described here face the challenge of managing the graphs at scale. This dimension often makes the problems that have been addressed in multiple forms in the academic and research community (such as disambiguation and unstructured data extraction) present new challenges in industry settings. Managing scale is the underlying challenge that affects several operations related to performance and workload directly. It also manifests itself indirectly as it affects other operations, such as managing fast incremental updates to large-scale knowledge graphs as at IBM or managing consistency on a large evolving knowledge graph as at Google.¹

Other Key Challenges

In addition to these truly pervasive challenges, the following challenges will be critical to the efforts described in this article. These are interesting and intriguing subjects for research and academic communities.

Knowledge-graph semantic embeddings. With a large-scale knowledge graph, developers can build high-dimensional representations of entities and relations. The resulting embeddings will greatly benefit many machine-learning, NLP, and AI tasks as sources of features and constraints, and

can form the basis for more sophisticated inferences and ways to curate training data. Deep-learning techniques can be applied to problems of entity deduction and attribute inference.²

Knowledge inference and verification. Making sure that facts are correct is a core task in constructing a knowledge graph, and with a huge scale it is not remotely possible to verify everything manually. This requires an automated approach: advances in knowledge representation and reasoning, probabilistic graphical models, and natural language inferences can be used to construct an automatic or semi-automatic system for consistency checking and fact verification.

Federation of global, domain-specific, and customer-specific knowledge. In a case like IBM clients, who build their own custom knowledge graphs, the clients are not expected to tell the graph about basic knowledge. For example, a cancer researcher is not going to teach the knowledge graph that skin is a form of tissue, or that St. Jude is a hospital in Memphis, Tennessee. This is known as “general knowledge,” captured in a general knowledge graph.

The next level of information is knowledge that is well known to anybody in the domain—for example, carcinoma is a form of cancer or NHL more often stands for non-Hodgkin lymphoma than National Hockey League (though in some contexts it may still mean that—say, in the patient record of an NHL player). The client should need to input only the private and confidential knowledge or any knowledge that the system does not yet know. Isolation, federation, and online updates of the base and domain layers are some of the major issues that surface because of this requirement.

Security and privacy for personalized, on-device knowledge graphs. Knowledge graphs by definition are enormous, since they aspire to create an entity for every noun in the world, and thus can only reasonably run in the cloud. Realistically, however, most people do not care about all entities that exist in the world, but rather a small fraction or subset that is personally relevant to them. There is a lot of promise in the area of personalizing knowledge graphs for individual users, perhaps even to the extent that they can shrink

to a small enough size to be shippable to mobile devices. This will allow developers to keep providing user value in a privacy-respecting manner by doing more on-device learning and computation, over local small knowledge-graph instances. (We are eager to collaborate with the research community in pursuit of this goal.)

Multilingual knowledge systems. A comprehensive knowledge graph must cover facts expressed in multiple languages and conflate the concepts expressed in those languages into a cohesive set. In addition to the challenges in knowledge extraction from multilingual sources, different cultures may conceptualize the world in subtly different ways, which poses challenges in the design of the ontology as well.

Conclusion

The natural question from our discussion in this article is whether different knowledge graphs can someday share certain core elements, such as descriptions of people, places, and similar entities. One of the avenues toward sharing these descriptions could be to contribute them to Wikidata as a common, multilingual core. In the nearer term, we hope to continue sharing the results of research that each of us may have done with researchers and practitioners outside of our companies.

Knowledge representation is a difficult skill to learn on the job. The pace of development and the scale at which knowledge-representation choices impact users and data do not foster an environment in which to understand and explore its principles and alternatives. The importance of knowledge representation in diverse industry settings, as evidenced by the discussion in this article, should reinforce the idea that knowledge representation should be a fundamental part of a computer science curriculum—as fundamental as data structures and algorithms.

Finally, we all agree that AI systems will unlock new opportunities for organizations in how they interact with customers, provide unique value in their space, and transform their operations and workforces. To realize this promise, these organizations must figure out how to build new systems that unlock knowledge to make them truly intelligent organizations.

The article summarizes and expands on a panel discussion the authors conducted at the International Semantic Web Conference in Asilomar, CA, in Oct. 2018 (<https://bit.ly/2ZYVLJh>). The discussion is based on practical experiences and represents the views of the authors and not necessarily their employers. 

Related articles on queue.acm.org

Schema.org: Evolution of Structured Data on the Web

R.V. Guha, D. Brickley, and S. Macbeth
<https://queue.acm.org/detail.cfm?id=2857276>

Hazy: Making it Easier to Build and Maintain Big-data Analytics

A. Kumar, F. Niu, and C. Ré
<https://queue.acm.org/detail.cfm?id=2431055>

A Primer on Provenance

L. Carata, et al.
<https://queue.acm.org/detail.cfm?id=2602651>

References

1. Höffner, K., Walter, S., Marx, E., Usbeck, R., Lehmann, J. and Ngonga Ngomo, A.C. Survey on challenges of question answering in the semantic Web. *Semantic Web* 8, 6 (2017), 895–920.
2. Lin, Y., Liu, Z., Sun, M., Liu, Y. and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Assoc. Advancement of Artificial Intelligence* 15, (2015), 2181–2187.
3. Nickel, M., Murphy, K., Tresp, V. and Gabrilovich, E. 2016. A review of relational machine learning for knowledge graphs. In *Proceedings of the IEEE* 104, 1 (2016), 11–33.
4. Paulheim, H., Knowledge graph refinement: a survey of approaches and evaluation methods. *Semantic Web* 8, 3 (2017), 489–508.

Natasha Noy is a scientist at Google, where she works on making structured data accessible and leads Google Dataset Search. Previously, she worked on ontology engineering and semantic Web at Stanford University, Stanford, CA, USA.

Yuqing Gao is the general manager of Microsoft's Artificial Intelligence – Knowledge Graph organization. She has been a key leader behind intelligent features for Microsoft Office products, Bing Entity Search, and other prominent AI knowledge-driven Microsoft technologies.

Anshu Jain works at IBM Watson, where he is responsible for the architecture of the core knowledge and language capabilities, including Knowledge Graph, natural language understanding, and Watson Knowledge Studio, among others.

Anant Narayanan is an engineering manager at Facebook, where he helps build knowledge platforms to develop a deeper understanding of entities and relationships. Previously, he led the development of large-scale data pipelines at Ozlo to support conversational AI systems.

Alan Patterson is a Distinguished Engineer at eBay, heading up eBay's efforts to build a product knowledge graph that contains eBay's knowledge of products, as well as organizations, brands, people, places, and standards. Previously, he worked at the startup True Knowledge (also Evi.com).

Jamie Taylor manages the Schema Team for Google's Knowledge Graph. The team's responsibilities include extending KG's underlying semantic representation, growing coverage of the ontology, and enforcing semantic policy. Previously, he worked for Metaweb Technologies.

Copyright held by authors/owners.
Publication rights licensed to ACM.

An executive crash course.

BY ANNA WIEDEMANN, NICOLE FORSGREN, MANUEL WIESCHE,
HEIKO GEWALD, AND HELMUT KRCMAR

Research for Practice: The DevOps Phenomenon

A TRADITIONAL SOFTWARE company releases its flagship product maybe every few years. Each release can include hundreds of new features and improvements. Because releases are infrequent, users can grow impatient waiting for each new release and are thankful when it finally arrives.

Disappointment sets in, however, when bugs are found and features don't work as expected. Under great stress and with great turmoil, an emergency release is produced and put into production (hurried through the regular release process, often achieved by skipping tests), which has still more bugs, and the process repeats with more emergency releases, leading to more frustration, stress, and disappointment. Worse yet, new business opportunities are missed or ignored because of doubt, uncertainty, and distrust in the IT department's ability to deliver value.

Isn't there a better way?

Such practices are a thing of the past for companies that subscribe to the DevOps method of software development and delivery. New releases are frequent: often weekly or daily. Bugs are fixed rapidly. New business opportunities are sought with gusto and confidence. New features are released, revised, and improved with rapid iterations. In one case study, a company was able to provide a new software feature every 11 seconds.¹⁷

Which of these software teams would you rather be? Which of these companies will win during their industry's digital transformation?

DevOps presents a strategic advantage for organizations when compared with traditional software-development methods (often called *phase-gate* or *waterfall*).⁷ Leadership plays an important role during that transformation.



IMAGE BY DMITRIY RYBIN



In fact, Gartner predicts that CIOs who haven't transformed their teams' capabilities by 2020 will be displaced.⁹

Promises and Challenges for Digital Transformation

For organizations hoping to capture market share and deliver value faster (or even just deliver software more safely and securely), DevOps promises both speed and stability.⁴ It has developed as a prominent phenomenon of digital transformation in modern organizations that use software to deliver value to their customers in industries including banking, retail, and even manufacturing. DevOps combines activities of software development and delivery to enhance the speed of getting new software features to customers. This leads to higher customer satisfaction and profitability, which are important outcomes at the

organization level. It also leads to important team-level outcomes such as better collaboration among different departments (such as developers, testers, and IT operations) and improved work-life balance.

Executing a successful DevOps transformation isn't without its challenges. Organizations and software products vary in maturity and implementation, making transformation efforts difficult to design and deploy across teams and organizations. Most importantly, for DevOps to truly deliver value, it must include more than just tooling and automation—so simply purchasing and installing a solution is not sufficient. As outlined here, DevOps includes culture, process, and technology. Indeed, success stories abound: Companies such as Kaiser Permanente, Capital One, Target, Starbucks, and ING have adopted DevOps methods, allowing them to

deliver software for key applications in just seconds.

DevOps enhances automation from applications to infrastructure provisioning. Continuous delivery supports automation and enables faster time to market and agile software development with fast feedback cycles. As the phenomenon is relatively new in practice, practitioners report on struggles with issues such as leadership and cultural transformation, implementing continuous delivery pipelines, and integrating a culture of collaboration in team settings.⁸ Each of these areas would benefit from examination and guidance by formal research to test and augment the valuable experiences of those in industry.

A Move Away from Traditional Project Management

► DevOps methodology is marked by a change in how software delivery

is treated: moving from a discrete project to an ongoing product. The traditional way of delivering software (referred to previously as *phase-gate* or *waterfall*) happens with the help of project management. In this paradigm, the project typically “ends” with the first major release of the new software or a set of new features delivered in a major incremental release. In general, the project team dissolves following a new release, and the responsibility for running the system is then “thrown over the wall” to operations. The operations team takes over responsibility for further changes and incident management. This leads to several problems:

- Developers are not responsible for running the system they built and therefore do not understand if trade-offs appear in creating and running the system, notably in the scalability and reliability of the software. This can lead to the same problems being perpetuated in future software releases, even if they are well understood by the operations team.

- The operations team is responsible for maintaining a highly reliable and stable system. Each new line of software deployment introduces change, and therefore leads to instability. This leads to a mismatch in incentives, where accepting new software from developers introduces risk (instability) and uncertainty (because less or no visibility into the development process gives them little insight into the software they are inheriting). Even if new components are of high quality, all new code adds complexity to the system and risk that the software was not developed with scalability and reliability in mind—key factors that operations must address and support in new software, as well as existing software.

- Stakeholders upstream (that is, earlier in the development process) from the production environment, including developers and the business, are not able to receive any feedback about performance until the first complete release. This generally takes place several months after project approval. Furthermore, not all features in software deliver value, and speeding up feedback to the development team and business allows for faster iteration to refine the

software. This leads to wasted time and resources on features that don’t ultimately deliver value. For example, research from Microsoft shows that only one-third of well-designed features delivered value to the customer.¹⁴

In contrast, DevOps calls for a shift to product-based management. Practically, this means there is no more “end date” to projects, and teams instead deliver features—and therefore value—continuously. An important part of achieving this is integrating teams throughout the value stream, from development through operations; some organizations are even including business stakeholders. In this model, software is a product that is maintained as a product, with delivery and value metrics being tracked by the business continuously.

State-of-the-Art Research and Practice on DevOps

One of the biggest challenges (and complaints!) in industry is the lack of a formal definition for DevOps. Many practitioners argue that this is intentional, because it allows teams and organizations to adopt a definition that works for them. In addition, they point out that having a formal definition of agile (coded in the Agile Manifesto) hasn’t solved the problem of definition sprawl, and the resulting confusion around what is truly meant when an organization says it is “going agile” still plagues the industry. This lack of understanding can be challenging, so we present some common definitions for reference here.

- “DevOps is a software development and delivery methodology that provides ... increased speed and stability while delivering value to organizations.”⁴

- “DevOps, whether in a situation that has operations engineers picking up development tasks or one where developers work in an operations realm, is an effort to move the two disciplines closer.”¹⁶

- “DevOps, a compound of ‘development’ and ‘operations,’ is a software development and delivery approach designed for high velocity.”¹⁷

And yet, these definitions focus on the outcome (value through speed and stability, via Forsgren⁴) or the foundations of the discipline (bringing together development and operations,

via Roche¹⁶ or Sebastian et al.¹⁷ If one wants to understand the components of DevOps methods, perhaps the most common presentation of these is summarized as CALMS: culture, automation, lean, measurement, and sharing.^{6,12} Here is a brief definition of each component:

- **Culture.** Integration of mutual trust, willingness to learn, continuous improvement, constant flow of information, open-mindedness to changes, and experimentation between developers and operations.

- **Automation.** Implementing deployment pipelines with a high level of automation (most notably continuous integration/continuous delivery) and comprehensive test automation.

- **Lean.** Applying lean principles such as minimization of work in progress, as well as shortening and amplification of feedback loops to identify and minimize value flow breaks to increase efficiency.

- **Measurement.** Monitoring the key system metrics such as business or transactions metrics and other key performance indicators.

- **Sharing** knowledge in the organization and across organizational boundaries. Team members should learn from each other’s experiences and proactively communicate.

An Interpretation of CALMS

This article uses the five core elements of CALMS as a framework.

Culture. Working as part of a DevOps team requires a culture of collaboration within a cross-functional team setting. In its ideal state, DevOps uses a so-called cross-functional team, which means that groups made up of developers, testers, quality assurance professionals, and IT operations engineers all work together to develop and deliver software. In this way, they are familiar with each others’ work and challenges (a common phrase to describe this is “to have empathy”), which helps them create and maintain better software. For example, because they see the challenges in maintaining scalable and reliable infrastructure encountered by operations professionals, developers write more scalable and reliable code in collaboration with operations staff.

Automation. This principle requires

a suite of DevOps tools.^{2,13} The following are a few examples of available automation tooling:

- **Build.** Tools for the software development and service life cycle, including compiling code, managing dependencies, creating documentation, conducting tests, and deployments of an application in different stages.

- **Continuous integration.** Constant testing of new software components.

- **Release automation.** Packaging and deploying a software component from development across all environments to production.

- **Version control.** Managing changes to the program and collecting other information. Version control is a component of configuration management.

- **Test automation.** Using software to execute tests and repeating activities.

- **Configuration management.** Reducing production provisioning and configuration maintenance with the help of reproducing the production system on development stages.

- **Continuous delivery.** A combination of principles, practices, and patterns designed to make uninterrupted deployments.

- **Logging.** Traces are essential for application management; everything must be logged.

- **Monitoring.** Identification of infrastructure problems before the customers notice them. Monitoring storage, network traffic, memory consumption, etc.

Continuous integration and delivery reduce the cost and risk of releasing software. They do this by combining automation and good practices to consistently, reliably, and repeatably perform work (such as tests and builds) that enables fast feedback and builds quality in during the software-delivery process. This helps teams deliver features faster and more reliably and, in turn, achieve faster value delivery for the organization. The highest-performing teams are able to deploy 46 times more frequently with 2,555 times shorter lead times than low performers. Failure rates are seven times less, and they are able to recover 2,604 times faster than their lower-performing peers.⁸

Lean. “Building quality in”—referenced earlier—is a key tenet in DevOps, and a principle also found in *lean*. Ap-



Companies such as Kaiser Permanente, Capital One, Target, Starbucks, and ING have adopted DevOps methods, allowing them to deliver software for key applications in just seconds.



plied to DevOps, this means teams look for opportunities to remove waste, leverage feedback loops, and optimize automation.

Let’s look at an example. DevOps teams differ in size and product responsibility. In some models, a single team conducts all software development and delivery activities, including development, testing, delivery, and maintenance. They are ultimately responsible for the complete software-delivery life cycle of the software products (and may be responsible for more than one product), delivering value to the business. Lean processes allow for quick iterations and feedback throughout the development and delivery process to improve quality and build faster and more reliable systems. Examples include working in small batches to enable fast flow through the development pipeline, limiting work in process, fixing errors as they are discovered vs. at the end, and “shifting left” on security input.

These practices may sound familiar; similar practices have driven quality and value in manufacturing. (For a great story about lean manufacturing, check out episode 561 of *This American Life*,¹¹ which discusses NUMMI (New United Motor Manufacturing Incorporated), the joint venture between Toyota and GM in Fremont, CA.)

Measurement is another core aspect of DevOps. The ability to monitor and observe systems is important, because software development and delivery are essentially dealing with an invisible inventory that interacts in complex ways that cannot be observed. (This is in contrast to traditional physical manufacturing systems such as an automobile assembly line, described in the NUMMI case.)

Through effective monitoring, teams are able to track, watch, measure, and debug their systems throughout the software-delivery life cycle. It should be noted that metrics are also a tool for quality assurance, and measurements from several sources should be leveraged.⁹

Sharing of knowledge and information enables successful DevOps teams and helps amplify their success. By sharing practices—both successes and failures—within teams, across the organization, and across the industry, teams

benefit from the learning of others and improve faster. While others have pointed out that sharing is possible in any domain and any methodology, DevOps has adopted this as a cultural norm, and many in the industry report that the field is much more collaborative than their prior work in tech.

Internal collaboration may include work shadowing or job swapping: developers are involved in operations and maintenance activities (for example, developers may even “take the pager”), and operations engineers rotate in to development and test roles, learning essential components of design and test work. In many cases, all cross-functional team members participate in the same meetings, which gives them shared context. Cross-industry sharing often takes places at conferences, with dozens of DevOps Days and other community-organized events sprouting up around the world.

The application of these principles leads to better outcomes: for individuals (seen in reduced burnout and greater job satisfaction), for teams (seen in better software delivery outcomes and better team cultures), and for organizations (seen in improved performance in measures such as profitability, productivity, customer satisfaction, and efficiency).^{7,21}

Although DevOps has been an important movement in industry for more than a decade, it has not received much attention from the academic community until recently. And while CALMS principles are not always referred to using these terms, they do appear in existing research (for example, Fitzgerald and Stol³).

Research Summaries

The core insights of some prior DevOps-related research follow:

Journal Articles

► Fitzgerald and Stol. *The Journal of Systems and Software*³

Presents a continuous software-engineering pipeline and a research agenda for different continuous processes including DevOps and BizDev (business strategy and development).

► Forsgren, Sabherwal, and Durcikova. *European Journal of Information Systems*¹⁰

Highlights the roles adopted when

As many practitioners note, managing the cultural changes inside an organization can be a more difficult and important challenge than implementing technical changes.

sharing and sourcing knowledge in a system management context, and identifies strategies for optimizing outcomes.

► Forsgren, Durcikova, Clay, and Wang. *Communications of the AIS*⁵

Identifies the most important system and information characteristics of tools for users who maintain systems.

► Dennis, Samuel, and McNamara. *Journal of Information Technology*¹

Highlights that maintenance effort should be considered during the design phase and calls this DFM (design for maintenance). The authors present insights into how the links among documents should affect both the maintenance effort and use.

► Sharma and Rai. *European Journal of Information Systems*¹⁹

Investigates how an organization's computer-aided software engineering adoption decision is influenced by individual factors of IS leaders and technological factors.

► Shaft and Vessey. *Journal of Management Information Systems*¹⁸

Aims to examine software maintenance as interlinking comprehension and modification; the relationship between these two factors is moderated by cognitive fit.

Conferences

► Trigg and Bødker. *ACM Conference on Computer Supported Cooperative Work*²⁰

Examines how people tailor their shared PC environment and presents an understanding of how software development tailoring can be helpful in designing systems that better fit the demands.

► Lwakatare et al. *Hawaii International Conference on System Sciences*¹⁵

Investigates key challenges of DevOps adoption in embedded system domains.

► Wiedemann and Wiese. *European Conference on Information Systems*²²

Presents an ideal skill set that DevOps team members should adopt to manage the software delivery life cycle.

Practical Challenges and Opportunities

Implementing DevOps presents several challenges. First, technology—and the resulting organizational transformation—is difficult, but strong leadership can help. As many practitioners

note, managing and enabling the cultural changes inside an organization can be a more difficult and important challenge than implementing the technical changes. Good leadership is vital. One approach studied in several contexts, including DevOps, is transformational leadership; this style uses five dimensions (vision, intellectual stimulation, inspirational communication, supportive leadership, and personal recognition) to inspire and guide teams.

Second, DevOps requires a custom solution for each organization. Each context is unique, and a prescriptive approach to DevOps implementation and adoption is unlikely to be successful. Teams and organizations pose a unique set of challenges and cultural norms. Each one should adopt and adapt its own approach to achieve DevOps success. The adaptation of DevOps should include development of not only technical, but also cultural, process, and measurement practices. The work of creating a unique and seemingly ad hoc technology transformation journey is difficult and may be daunting, so many organizations look for step-by-step guides. However, these are not likely to provide solutions (beyond basic advice such as “automate your toolchain”) and are usually offered by those trying to sell you something.

Third, each DevOps solution should encompass a holistic view, consisting of automation (including tools and architecture), process, and culture. In many traditional approaches, specialist knowledge in one area (for example, development) is leveraged to accomplish a task before passing it off to another group. In DevOps, a move from this high specialization to include a broad understanding of more areas is necessary. (Some call this *T-shaped knowledge*, with the top part of the “T” representing broad knowledge, while the stem of the T represents deep understanding in one area of expertise.)

This allows people to understand how their work will affect and interact with more areas of the technical stack; this often requires significant additional learning and responsibilities in the transition to DevOps. Organizations should provide training and

education, and not just expect technologists to augment their learning independently. Note that while some technologists consider this expansion of responsibilities and knowledge exciting, others may push back, especially those who are just a few years from retirement and comfortable in their work roles, or who see sharing information about their roles as a risk to job security. Organizations must consider these training and cultural challenges in particular and respond accordingly. (As already noted, DevOps is about much more than just technology!)

Conclusion

DevOps is about providing guidelines for faster time to market of new software features and achieving a higher level of stability. Implementing cross-functional, product-oriented teams helps bridge the gaps between software development and operations. By ensuring their transformations include all of the principles outlined in CALMS, teams can achieve superior performance and deliver value to their organizations. DevOps is often challenging, but stories from across the industry show that many organizations have already overcome the early hurdles and plan to continue their progress, citing the value to their organizations and the benefits to their engineers.

Acknowledgments

The authors would like to thank our anonymous reviewers, whose guidance and thoughtful feedback helped us to shape and refine this article. 

References

1. Dennis, A.R., Samuel, B.M. and McNamara, K. Design for maintenance: how KMS document linking decisions affect maintenance effort and use. *J. Information Technology* 29, 4 (2014), 312–326.
2. Ebert, C., Gallardo, G., Hernantes, J. and Serrano, N. DevOps. *IEEE Software* 33, 3 (2016), 94–100; <https://ieeexplore.ieee.org/document/7458761>.
3. Fitzgerald, B. and Stol, K.-J. Continuous software engineering: A roadmap and agenda. *J. Systems and Software* 123, (2017), 176–189.
4. Forsgren, N. DevOps delivers. *Commun. ACM* 61, 4 (Apr. 2018), 32–33; <https://dl.acm.org/citation.cfm?id=3174799>.
5. Forsgren, N., Durcikova, A., Clay, P.F. and Wang, X. The integrated user satisfaction model: Assessing information quality and system quality as second-order constructs in system administration. *Commun. AIS* 38 (2016), 803–839.
6. Forsgren, N. and Humble, J. The role of continuous delivery in IT and organizational performance. In *Proceedings of the Western Decision Sciences Institute*, 2015.
7. Forsgren, N., Humble, J. and Kim, G. *Accelerate: The*

Science of Lean Software and DevOps: Building and Scaling High-performing Technology Organizations. Revolution Press, Portland, OR, 2018.

8. Forsgren, N., Humble, J. and Kim, G. *Accelerate: state of DevOps report: Strategies for a new economy*. DORA (DevOps Research and Assessment) and Google Cloud, 2018; <https://bit.ly/2vDJPmU>.
9. Forsgren, N. and Kersten, M. DevOps metrics. *Commun. ACM* 61, 4 (Apr. 2018), 44–48; <https://dl.acm.org/citation.cfm?id=3200906.3159169>.
10. Forsgren, N., Sabherwal, R. and Durcikova, A. Knowledge exchange roles and EKR performance impact: Extending the theory of knowledge reuse. *European J. Information Systems* 27, 1 (2018), 3–21.
11. Glass, I. Episode 561, “NUMMI 2015.” *This American Life*; <https://www.thisamericanlife.org/561/nummi-2015>.
12. Humble, J. and Molesky, J. Why enterprises must adopt DevOps to enable continuous delivery. *Cutter IT Journal* 24, 8 (2011), 6–12; <https://www.cutter.com/sites/default/files/itjournal/fulltext/2011/08/itj1108.pdf>.
13. Humble, J. Continuous delivery sounds great, but will it work here? *Commun. ACM* 61, 4 (Apr. 2018); 34–39; <https://dl.acm.org/citation.cfm?id=3173553>.
14. Kohavi, R., Crook, T., Longbotham, R., Frasca, B., Henne, R., Ferres, J.L. and Melamed, T. Online experimentation at Microsoft, Data Mining Case Studies 11, 2009.
15. Lwakatare, L.E. et al. Towards DevOps in the embedded systems domain: Why is it so hard? In *Proceedings of the Hawaii International Conference on System Sciences*, 2016.
16. Roche, J. Adopting DevOps practices in quality assurance. *Commun. ACM* 56, 11 (Nov. 2013), 38–43; <https://dl.acm.org/citation.cfm?id=2524721>.
17. Sebastian, I.M., Ross, J.W., Beath, C., Mockler, M., Moloney, K.G. and Fonstad, N.O. How big old companies navigate digital transformation. *MIS Q. Executive* 16, 3 (2017), 197–213.
18. Shaft, T.M. and Vessey, I. The role of cognitive fit in the relationship between software comprehension and modification. *MIS Quarterly* 30, 1 (2006), 29–55.
19. Sharma, S. and Rai, A. Adopting IS process innovations in organizations: The role of IS leaders' individual factors and technology perceptions in decision making. *European J. Information Systems* 24, 1 (2015), 23–37.
20. Trigg, R.H. and Bødker, S. From implementation to design: Tailoring and the emergence of systematization. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 1994, 45–54; <https://dl.acm.org/citation.cfm?id=192869>.
21. Wiedemann, A. A new form of collaboration in IT teams—exploring the DevOps phenomenon. In *Proceedings of the Pacific Asia Conference on Information Systems*, 2017, 1–12.
22. Wiedemann, A. and Wiesche, M. Are you ready for Devops? Required skill set for Devops teams. In *Proceedings of the European Conference on Information Systems*, 2018.

Anna Wiedemann is a research assistant at Neu-Ulm University of Applied Sciences and a doctoral candidate at the Technical University of Munich (TUM), Germany.

Nicole Forsgren does research and strategy at Google Cloud and is best known as lead investigator on the largest DevOps studies to date. She has been a successful entrepreneur (with an exit to Google), professor, performance engineer, and sysadmin.

Manuel Wiesche is a postdoctoral researcher at the Chair for Information Systems, Department of Informatics at the Technical University of Munich (TUM), Germany.

Heiko Gewald is research professor of information management at Neu-Ulm University of Applied Sciences and director of the Center for Research on Service Sciences.

Helmut Krcmar is chaired professor of information systems in the Department of Informatics at the Technical University of Munich (TUM) and speaker of the directorate of fortiss GmbH, the Research Institute of the Free State of Bavaria for Software and Systems.

Copyright held by authors/owners.
Publication rights licensed to ACM.

Know when to let go of emotional attachment to your work.

BY KATE MATSUDAIRA

Overly Attached

WE ALL KNOW it's good to have a strong sense of ownership toward your work, but what happens when you get too attached?

Recently, I encountered this problem while collaborating with a smart, senior engineer who couldn't make logical decisions if it meant deprecating the system he and his team had worked on for a number of years. Even though the best thing would have been to help another team create the replacement system, they did not want to entertain the idea because it would mean putting an end to something they had invested so much in.

I recognized this behavior, because this has happened to me, too. So, I started thinking about why this happens and what one can do to navigate these difficult situations.

One of the great things about writing software is that you get to create something. Great engineers typically are good not only at building things, but also at owning

and maintaining them over time. This can foster a great sense of ownership and responsibility, which is something that is recognized and rewarded (in most situations, anyway) because it benefits the whole team.

Sometimes, however, ownership can lead to emotional attachment, and that can have negative consequences.

The longer you work on one system or application, the deeper the attachment. For years you have been investing in it—adding new features, updating functionality, fixing bugs and corner cases, polishing, and refactoring. If the product serves a need, you likely reap satisfaction for a job well done (and maybe you even received some raises or promotions as a result of your great work). I totally get this—I still have copies of big projects from 10-plus years ago that are so out of date they likely would not compile if I tried.

This can also be true for inexperienced engineers. I remember my very first job and my first bug.

I spent a few days getting up to speed, setting up my environment, and then fixed the bug—submitting my first check-in to the large project we were working on. That night, one of the senior engineers, working late, had reviewed all the new code that had been checked in prior to the nightly build. Apparently my function didn't meet his approval, so he erased it and rewrote it as part of another class. I still remember the next day I was devastated. It was like he had erased my whole career with a single submit. When you have only a small amount of work, every little thing you do represents a lot of your career, and so it is easy to be attached.

This doesn't just happen with code. It can happen with ideas, proposals, projects—anything you have invested significant time, energy, and care in. It is natural that people become very attached to things they have invested in, but unfortunately, that attachment can often make it difficult to see your work objectively, as other people do.



When Ownership Turns Emotional

Becoming too attached to your work can have negative consequences. While each situation is unique, sometimes it can result in suboptimal decision-making. Here are a few real-world examples (keep in mind that these can be good or bad, depending on the circumstances):

- ▶ Building functionality in one system instead of another, based on the team that owns it (that is, some manifestations of Conway's law).
 - ▶ Prioritizing tech debt or refactoring projects over new features.
 - ▶ Refusal to adopt or purchase other implementations that are superior in some way.
 - ▶ Maintaining an investment in a project longer than justified (that is, sunk-cost fallacy).
 - ▶ Impaired or slow decision-making.
- In a utopia, all people act rationally,

are able to weigh the pros and cons, and make well-informed, thoughtful decisions. In reality, though, people are emotional creatures and their judgment can be clouded.

As a leader, this can be tough to navigate, because you want to encourage strong ownership and enable people to work on projects that satisfy them. At times, however, these desires can be at odds with the decisions you need to make for your business and goals.

What do you do when you have to work with someone who is too attached? Following are some of the strategies that have worked for me.

Strategies for Navigating Emotional Attachment

1. Align on goals and purpose.

The first step is to ensure all parties agree on the objectives and goals. If you don't know which variables you

are optimizing for, it will be difficult to achieve alignment.

Start the conversation by asking questions that will uncover what the other people are focused on. Try to understand what things matter to them. Then you can determine if your goals are different and if you need to negotiate or escalate. Your job is to figure out how to align on the same outcomes.

Once you reach an agreement, it can help to record it (on a whiteboard, in an email or document, and so on) since some people absorb and interpret things differently out loud and in writing.

Sometimes this alignment is enough to make decisions and move forward.

2. Ask everyone to be open to ideas and alternatives.

Sometimes people might think they know the answer. They have already

thought through all of the options and they are certain they are on the path forward. While unintentional, the result of this thinking may close off any consideration of alternatives.

By asking people to be open to other ideas, and by being willing to entertain other options yourself, you can start a dialog to consider different approaches. There are seldom right answers. There can be wrong answers, but most of the time there are just different options with pros, cons, and risks.

Work together to explore other options, and for each, lay out the different considerations. This can help frame the discussion and potentially uncover issues that may not have been obvious to everyone involved.

3. Ask for stories, not solutions.

When you focus on the specific details or solutions to a problem, alignment can be more difficult. Just like someone's values, these beliefs are so strongly held, it can be challenging to change anyone's mind. If you focus on the how and why, however, it can be much easier to find common ground and solve the problem.

For example, if you start with the premise that a service must exist to provide a certain API, it can be difficult to argue. Where can you go from there? However, if you start with the story of what the API is used for, it may be possible to provide that data in a completely different way elsewhere in the system.

By framing it as a story, using the big-picture insight you have as the leader, you can help the other people involved see the service in a larger context than they had been thinking about. It is so easy to get hyper-focused on one thing when you are too close to a project or have only the perspective of working in one department or team. This is your opportunity to help people zoom out and see the situation with fresh eyes.

Alternatively, you can ask people to imagine other scenarios where the service could be used in different ways from those they are currently working toward. Maybe they see only one logical solution right now; by asking them to look for other functions the service could have, you can help them unlock more ways that the prob-

lem could and should be solved. This allows you to solve problems more creatively and see more perspectives on the situation.

As humans, we follow narratives far more closely and with far greater understanding than we do data or facts. Instead of just insisting that people change their opinions, you can guide them there by helping them reframe the situation or see it more broadly.

4. Accept their experience, suggest new interpretations.

Most likely, the positions of emotionally attached people are based on experience and significant effort. It is important to recognize and validate that experience. Being empathetic helps you understand where they are coming from and what the reasons behind their emotions are.

Once you have an understanding of their particular experience, it is much easier to suggest alternative interpretations or viewpoints based on that experience.

For example, if the system has been hardened against hundreds of edge cases, start by acknowledging that and helping them think through the "why." This can lead to different discussions and stop you from talking in circles.

Ask questions such as: What drove so many edge cases? How were they discovered? Are any edge cases no longer relevant/needed? What risks exist if an edge case is missed in a future implementation? If you had to build the service again, what would you change to eliminate many of these edges? This line of questioning can help them assess risks and open their minds to potential alternatives.

5. Enlist help.

If you are having difficulty influencing people, consider enlisting the help of someone they trust; this can be their manager or maybe other teammates. Sometimes having others on your side can help you influence and change someone's decision. Those people may also be able to explain another person's position to you differently and in a way that is less emotionally charged.

Either way, getting a second (or third) opinion on the situation can

help add color and put you in a position to influence the outcome.

6. Practice being open-minded.

Sometimes emotional responses are driven by solid concerns, but the delivery or interaction can prevent getting to those details. By practicing patience, strong listening skills, and thoughtful question-asking, you can help take charged situations and dissect them. Don't be afraid to ask why—and accept the possibility that the people on the other side may be right and you might be wrong.

Go into the interaction assuming good intent and looking for the opportunity to work together—not to prove something right or wrong. Having a curious mind and seeking to understand will lay the groundwork for a more positive discussion.

Most complex problems have multiple solutions, and you need to be able to separate your ego from your intellect and decisions.

As you work with more people and more systems you will inevitably encounter people who are very attached to their projects. The key to reaching common ground starts with being open and thoughtful. Always seek to understand other positions—and let people tell their stories. Validate their experiences, and ask thoughtful questions. And if you reach a roadblock, look to others to help you.

Being able to collaborate and work through these issues will make you a better leader and should lead to better outcomes. 

Related articles on queue.acm.org

Nine Things I Didn't Know I Would Learn Being an Engineer Manager

Kate Matsudaira

<https://queue.acm.org/detail.cfm?id=2935693>

The Small Batches Principle

Thomas A. Limoncelli

<https://queue.acm.org/detail.cfm?id=2945077>

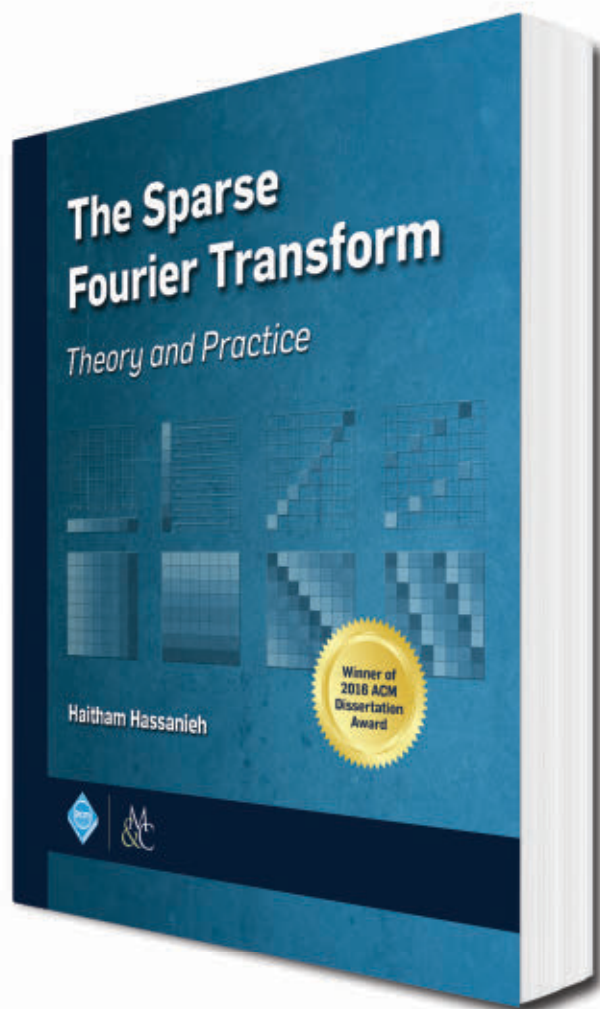
Death by UML Fever

Alex E. Bell

<https://queue.acm.org/detail.cfm?id=984495>

Kate Matsudaira (katemats.com) is an experienced technology leader. She has worked at Microsoft and Amazon and successful startups before starting her own company, Popforms, which was acquired by Safari Books.

Copyright held by author/owner.
Publication rights licensed to ACM.



**Faster algorithms that
run in sublinear time
have become necessary.**

Here's your guide.

Haitham Hassanieh

University of Illinois at Urbana Champaign

The Fourier transform is one of the most fundamental tools for computing the frequency representation of signals. It plays a central role in signal processing, communications, audio and video compression, medical imaging, genomics, astronomy, as well as many other areas. Because of its widespread use, fast algorithms for computing the Fourier transform can benefit a large number of applications. The fastest algorithm for computing the Fourier transform is the Fast Fourier Transform (FFT), which runs in near-linear time making it an indispensable tool for many applications. However, today, the runtime of the FFT algorithm is no longer fast enough especially for big data problems where each dataset can be few terabytes. Hence, faster algorithms that run in sublinear time, i.e., do not even sample all the data points, have become necessary. This book addresses the above problem by developing the Sparse Fourier Transform algorithms and building practical systems that use these algorithms to solve key problems in six different applications: wireless networks, mobile systems, computer graphics, medical imaging, biochemistry, and digital circuits.



ISBN: 978-1-947487-04-8 DOI: 10.1145/3166186

<http://books.acm.org>

<http://www.morganclaypoolpublishers.com/acm>

DOI:10.1145/3330794

A Harvard-based pilot program integrates class sessions on ethical reasoning into courses throughout its computer science curriculum.

BY BARBARA J. GROSZ, DAVID GRAY GRANT, KATE VREDENBURGH, JEFF BEHREND, LILY HU, ALISON SIMMONS, AND JIM WALDO

Embedded EthiCS: Integrating Ethics Across CS Education

THE PARTICULAR DESIGN of any technology may have profound social implications. Computing technologies are deeply intermeshed with the activities of daily life, playing an ever more central role in how we work, learn, communicate, socialize, and participate in government. Despite the many ways they have improved life, they cannot be regarded as unambiguously beneficial or even value-neutral. Recent experience shows they can lead to unintended but harmful consequences. Some technologies are thought to threaten democracy through the spread of propaganda on online social networks, or to threaten privacy through the aggregation of datasets that include increasingly personal information, or to

threaten justice when machine learning is used in such high-stakes, decision-making contexts as loan application reviews, employment procedures, or parole hearings.^{1,3,8,12,17,23} It is insufficient to ethically assess technology after it has produced negative social impacts, as has happened, for example, with facial recognition software that discriminates against people of color and with self-driving cars that are unable to cope with pedestrians who jaywalk.^{13,15} Developers of new technologies should aim to identify potential harmful consequences early in the design process and take steps to eliminate or mitigate them. This task is not easy. Designers will often have to negotiate among competing values—for instance, between efficiency and accessibility for a diverse user population, or between maximizing benefits and avoiding harm. There is no simple recipe for identifying and solving ethical problems.

Computer science education can help meet these challenges by making ethical reasoning about computing technologies a central element in the curriculum. Students can learn to think not only about what technology they could create, but also whether they should create that technology. Learning to reason this way requires

» key insights

- **Embedding ethical reasoning throughout the entire CS curriculum has the potential to habituate students to thinking ethically as they develop algorithms and build systems, both in their studies and as they pursue technical work in their careers.**
- **Without creating a multitude of additional courses, CS programs can meet student demand for learning about ethics and the potential societal impact of their work as well as for acquiring computer science technical competencies.**
- **By working with philosophy colleagues and students, CS faculty can integrate ethical reasoning into their courses more easily and more expertly. As a beneficial side effect of this approach, CS faculty gain competence in ethical reasoning and philosophers acquire a greater depth of understanding of technology.**

ILLUSTRATION BY CHARIS TSEVIS



courses unlike those currently standard in computer science curricula. A range of university courses on topics in areas of computing, ethics, society and public policy are emerging to meet this need. Some cover computer science broadly, while others focus on specific problems like privacy and security; typically, these classes exist as stand-alone courses in the computer science curriculum. Others have integrated ethics into the teaching of introductory courses on programming, artificial intelligence, and human-computer interaction.^{4,5,22}

This article presents an alternative and more integrative approach to incorporating ethical reasoning into computer science education, which we have dubbed “Embedded EthiCS.” In contrast to stand-alone computer ethics or computer-and-society courses, Embedded EthiCS employs a distributed pedagogy that makes ethical reasoning an integral component of courses throughout the standard computer science curriculum. It modifies existing courses rather than requiring wholly new courses. Students learn ways to identify ethical implications of technology and to reason clearly about them while they are learning ways to develop and implement algorithms, design interactive systems, and code. Embedded EthiCS thus addresses shortcomings of stand-alone courses.^{6,10} Furthermore, it compensates for the reluctance of STEM faculty to teach ethics on their own¹⁸ by embedding philosophy graduate students and postdoctoral fellows into the teaching of computer science courses.

Here, we present the rationale behind Embedded EthiCS; describe its development at Harvard, giving examples from participating courses; discuss lessons we have learned; and consider challenges—intellectual, administrative, and institutional—to implementing such a program in academic institutions of different kinds. We conclude by calling for the computer science community to join together to build open repositories of resources to facilitate wider adoption of the approach.

Why Embed Ethics and Philosophers in the Teaching of Computer Science?

Embedded EthiCS was created in response to student demand for two



Students can learn to think not only about what technology they could create, but also whether they should create that technology.



elective courses in computer science at Harvard that considered ethical concerns in concert with computer science methods: “Privacy and Technology” and “Intelligent Systems: Design and Ethical Challenges.” (For a brief description of these courses, see the online appendix “Special Topic Courses in Computer Science and Ethics;” <https://dl.acm.org/citation.cfm?doid=3330794&picked=formats>)

In teaching these courses, we repeatedly noticed how easy it was for students to forget about ethical concerns when focused on technical systems issues. Even those earnestly committed to learning and using ethical reasoning in their work quickly lost sight of these considerations when engaged in a technical design task. At the same time, we recognized that most computer science courses contain material for which an ethical challenge might arise. We thus designed Embedded EthiCS to habituate students to thinking ethically.

The Embedded EthiCS approach adds short ethics modules to computer science courses in the core computer science curriculum. By embedding ethics broadly across the curriculum, this approach meets three goals for computer science students: it shows them the extent to which ethical issues permeate almost all areas of computer science; it familiarizes them with a variety of concrete ethical issues and problems that arise across the field; and it provides them repeated experiences of reasoning through those issues and communicating their positions effectively.

While no single course with an Embedded EthiCS module will by itself produce ethically minded technology designers, we expect that incorporating modules throughout the curriculum will have a compounding effect—one that continually reinforces the importance of ethical reasoning to all aspects of computer science and technology design. In addition to exposing students to ethical content in a great breadth of computational contexts, this distributed pedagogy approach conveys the message that ethical reasoning is an expected part of a computer scientist’s work.

Embedded EthiCS is inherently interdisciplinary. Knowing what can be done with technology falls within the purview of computer science.

Understanding, evaluating, and successfully defending arguments about what should be done falls within the purview of the normative disciplines, most notably ethics, a subfield of philosophy. For students to succeed at learning not only how to build innovative computing systems, but also how to determine whether they should build those systems or how ethical considerations should constrain their design, it is imperative that these two disciplines work together. To this end, Harvard Computer Science and Philosophy Department faculty have been partnering to develop the Embedded EthiCS curriculum. Computer science faculty and teaching assistants collaborate with advanced Ph.D. students and postdoctoral fellows in Philosophy to develop Embedded EthiCS modules for each course. This approach also opens up exciting new areas of research for the philosophers who teach the modules and broadens their teaching repertoire.

How Does Embedded EthiCS Work?

Each Embedded EthiCS course has an Embedded EthiCS teaching assistant who is an advanced Ph.D. student or postdoctoral fellow in philosophy with a strong background in ethics and considerable teaching experience.

In consultation with faculty course heads, the teaching assistants design ethics modules through which students develop practical competence in addressing particular ethical challenges. They identify an ethical issue related to the course content, prepare for and lead one or two class meetings focused on that issue, and design an assignment and plan for assessing it. Depending on class size, the grading itself may be done by the Embedded EthiCS teaching assistant, by regular course teaching assistants, or through peer grading.

The modules are designed to give students three core ethical reasoning skills: the ability to identify and anticipate ethical problems in the development and use of computing technologies; the ability to reason, both alone and in collaboration with others, about those problems and potential solutions to them, using concepts and principles from moral philosophy; and the ability to communicate effectively their understanding of how to address those problems. The modules emphasize “active learning” activities and assignments that teach students to apply the philosophical ideas they have learned to concrete, real-world ethical problems as recommended by recent studies of ethics education.^{6,10} They are de-

signed to help students exercise their newly acquired ethical reasoning skills in context.

Embedded EthiCS Pilot

We piloted the Embedded EthiCS program over three semesters (Spring 2017, Fall 2017, and Spring 2018), with 14 separate courses. Figure 1 lists the courses, grouping them by computer science area, indicating the ethical problems addressed and enrollments. To illustrate the content and design of modules, we describe modules for several courses here.

► **Networks: Facebook, Fake News, and the Ethics of Censorship.** This course focuses on the use of network modeling tools to study complex empirical phenomena involving current online networks, including the ways ideas and influence spread and the contagion of economic behaviors. The Embedded EthiCS module considered the censorship of so-called “fake news” by social media companies. Its goal was to engage students in different forms of ethical reasoning about whether social media companies are morally obligated to suppress the spread of “fake news” on their platforms, and, if so, what kinds of content they should suppress and what strategies they should use to suppress

Figure 1. Embedded EthiCS courses 2017–2018.

CS236r and CS265 are graduate courses; other courses are primarily for undergraduates, with 100-level courses being at intermediate level. CS1, 134, and 179 were offered twice; only enrollments for 134 differed significantly and both are given. Boldface indicates courses discussed in the article.

Area	Course Title	Challenges	Enrollment
Introductory Courses	CS 1: Great Ideas in Computer Science	The Ethics of Electronic Privacy	76
	CS 51: Introduction to Computer Science II	Morally Responsible Software Engineering	283
	CS 109b: Advanced Topics in Data Science	Moral Considerations for Data Science Decisions	93
Theory	CS 126: Fairness, Privacy, and Validity in Data Analysis	Diversity and Equality of Opportunity in Automated Hiring Systems	11
Computer Science and Economics	CS 134: Networks	Facebook, Fake News, and the Ethics of Censorship	162 (S'17); 21 (F'17)
	CS 136: Economics and Computing	Matching Mechanisms and Fairness	55
	CS 236r: Topics at the Interface of Economics and Computing	Interpretability and Fairness	24
Programming Languages and Computer Systems	CS 152: Programming Languages	Verifiably Ethical Software Systems	79
	CS 165: Data Systems	Data and Privacy	25
	CS 265: Big Data Systems	Privacy and Statistical Inference from Data	12
Human-Computer Interaction	CS 179: Design of Useful and Usable Interactive Systems	Inclusive Design and Equality of Opportunity	62
Artificial Intelligence	CS 181: Machine Learning	Machine Learning and Discrimination	296
	CS 182: Introduction to AI	Machines and Moral Decision-Making	164
	CS 189: Autonomous Robot Systems	Robots and Work	20

it. The Embedded EthiCS teaching assistant first discussed three philosophical topics with the students: the distinction between hard and soft censorship; a selection of J.S. Mill's arguments against censorship from *On Liberty*; ¹⁶ and an argument, reconstructed from a *New York Times* editorial, that Facebook is obligated to suppress fake news because it interferes with democratic governance. ¹⁷ The module's assignment asked students to write short essays identifying a strategy for suppressing fake news and defending a position about whether Facebook was obligated to implement it.

► **Data Systems and Programming Languages Courses.** The discussion-based graduate course on big data systems investigates the design of data systems and algorithms that can “scale up,” that is, use a single machine to its full potential, and “scale out,” such as, use multiple machines (typically in the hundreds or thousands). The Embedded EthiCS module considered how to understand and protect privacy in the age of big data, particularly in light of the powerful inference capabilities large datasets and contemporary analytical tools make possible, some of which seem to violate individual privacy. Its goals were to give students a method for diagnosing the importance of privacy in a domain; to help students understand why traditional privacy protections, such as consent notices and anonymization, are ineffective for some flows of information; and to have students brainstorm solutions to difficult cases of statistical inference from publicly available information. To prepare for the in-class discussion, students were assigned a set of detailed questions on readings that dealt with different definitions of privacy and types of privacy protections. ^{2,7,14,20} In class, the Embedded EthiCS teaching assistant introduced an interest-based method for thinking about these issues. ^{21,24} The method starts by identifying the serious, common interests that underlie rights protections. The in-class session focused on the ethical grey area of whether unforeseen inferences about an individual from her publicly available data constitute a violation of privacy. (See Rumbold and Wilson ²⁰ for discussion.) The class also discussed whether individuals did or

did not waive their right to privacy in other grey areas, such as when employers monitor employees at work. For cases where privacy was violated, students brainstormed design solutions using the methodology.

For the basic undergraduate course on data systems, we developed an alternative privacy module that examined why privacy is valuable and whether it is a right. It also examined trade-offs between privacy and other social goods, such as healthcare, in the design of data systems.

For the basic undergraduate programming languages course, the Embedded EthiCS module investigated ways to integrate ethics into the software engineering process. Before the module, the class studied techniques for verifying that a program will behave in accordance with its design specifications. The module focused on the idea of ethical design specifications as opposed to legal or technical ones, that is, design specifications to help ensure a program behaves in ways that are morally acceptable.

► **Design of Useful and Usable Interactive Systems: Inclusive Design and Equality of Opportunity.** The Embedded EthiCS module for this human-computer interaction design course focused on the topic of inclusive design, namely, designing human-computer interaction systems to be both useful to and usable by individuals with disabilities of various kinds. Its goal was to lead students to think more clearly about the extent to which software developers are morally obligated to design for inclusion. The class began with a discussion of different meanings of “inclusive design.” Students then considered whether software companies are morally obligated to design for inclusion because doing so would, at a reasonable cost, alleviate unjust cumulative disadvantages faced by people with disabilities. During this discussion, the Embedded EthiCS teaching assistant introduced three relevant philosophical ideas: the distinction between actions that are morally obligatory and morally supererogatory; John Rawls's principle of fair equality of opportunity; ¹⁹ and the medical, social, and interactive models of disability. ²⁵ Students engaged in a group-based ethics simulation in which they imagined they were

software developers deciding whether to redesign their company's website for inclusion even if they might incur a cost like doing the work on personal time. The module's assignment was incorporated into the final design project for the course. Students were asked to answer questions about whether they would be obligated to design their project for inclusion if they went on to develop it commercially.

► **Machine Learning and Discrimination.** The Embedded EthiCS module for this introductory machine learning course focused on machine learning and its potential for discrimination. Its goals were to introduce students to different theories of wrongful discrimination, to lead them to appreciate that designing ethical machine learning systems involves more than designing accurate machine learning algorithms, to introduce them to formalized fairness criteria, and to lead them to think about the implications of an “impossibility” result. ¹¹ After giving a brief presentation on theories of wrongful discrimination (for which chapter 1 of Hellman ⁹ provides an overview), the Embedded EthiCS teaching assistant presented a case study in which an employer's hiring practices generated outcomes that correlated with the race of job applicants (based on Barocas and Selbst ³). The procedure was grounded in a sound business rationale and was also the product of historical injustice against certain groups. The students discussed whether the case was an instance of discrimination on two different types of theories of discrimination: anti-classification theories and anti-subordination theories. ³ The distinction between these two theories was then used to discuss conflicts between formal fairness criteria and the public discussion surrounding the use of COMPAS, a recidivism risk prediction tool, to inform judge's decisions in parole hearings. The assignment required students to design an algorithm for fair hiring practices that would reduce disparate impact while also producing socially good outcomes in the labor market, and to defend their design choices.

Embedded EthiCS: Assessment of the Pilot Program

Our experience with the pilot program has shown it is not only possible to in-

tegrate the teaching of ethical reasoning with core computer science methods but also rewarding for students and faculty alike. Following each Embedded EthiCS class session, faculty informally provided feedback, and we asked students to complete a short survey. Faculty reported the modules contributed to classes with only a modest burden on them, and that they learned from them.

Student surveys aimed to assess the effectiveness of each module and of the module approach in general. Figure 2 presents key survey results. Responses were overwhelmingly positive, supporting continuation of the initiative. Over 80% of students in all courses—and over 90% of students in five of the classes—agreed these class sessions were interesting. In all but two classes, more than 80% of students reported they would be interested in learning more about ethics in future computer science courses. Comments, which one-quarter of the students provided, were overwhelmingly positive, with many expressing eagerness for more exposure to ethics content and more opportunities to develop skills in ethical reasoning and communication. Negative comments were largely specific to individual class content or presentation. Some students wanted more breadth or depth, others more background. One

comment about overlap between two classes suggests the need to coordinate across classes.

From the Pilot to a Sustainable Program

For the first pilot of Embedded EthiCS in the spring semester of 2017, one Ph.D. student developed modules for four different classes: the introductory “great ideas” class, a theory course on networks, a data science course, and a human-computer interaction class. Based on the success of that effort, we engaged two Ph.D. students in AY 2017–2018 to develop modules for an additional 10 courses and repeat the modules for three of the original four courses.

In AY 2018–2019, we are working toward developing a corps of graduate student and postdoctoral teaching assistants for the program. A postdoctoral fellow leads weekly meetings of past and present teaching assistants and coordinates the development of modules. In Fall 2018, nine courses include Embedded EthiCS modules, including four courses on new subjects, two in systems and two in theoretical computer science.

What have we learned? The key lessons concern student engagement, successful faculty roles, teaching assistant experience, and barriers to em-

bedding ethics. A set of best practices is emerging.

For engaging students with ethical reasoning, we found that techniques that encourage an inclusive discussion with smaller classes tend to be effective with larger classes as well. In particular, Embedded EthiCS modules are most effective when the issues raised connect technical material to ethical issues already salient to students, the module employs short active learning exercises, and an assignment gives students practice with the ideas developed in the session.

We have found that Embedded EthiCS modules work best with close faculty engagement. Participating fully in the design of the modules, as described here, and being personally involved in the module class session(s) are crucial. Computer science faculty can also contribute to the success of Embedded EthiCS in two further ways: by including an assignment (either separate or part of a larger problem set) that contributes to the course grade in some way, however minor; and by mentioning ethical issues during other parts of the course to preview the upcoming module, refer back to the lesson, or otherwise signal the importance of the topic. These activities typically require only three hours of faculty time. When the assignment contributes to the fi-

Figure 2. Embedded EthiCS Pilot Evaluation.

Percentage of responding students in each course who agreed with each statement from the student evaluation survey. (Note that original responses were on a Likert scale from 1–7, with 7 = “strongly agree,” 6 = “agree,” 5 = “somewhat agree,” 4 = “neither agree nor disagree,” 3 = “somewhat disagree,” 2 = “disagree,” 1 = “strongly disagree.”) CS134 was offered twice, and results from both surveys are provided in chronological order. CS 179 was also offered twice; we show the initial survey results; the subsequent survey had a higher percentages in all categories. Figure 1 may be consulted for course titles and the ethical challenges discussed in each course.

Statement	Percentage of Students Who Agreed With Each Statement									
	CS 1	CS 51	CS 109b	CS 134 (S)	CS 134 (F)	CS 136	CS 152	CS 165	CS 179	CS 182
The ethics guest lecture was interesting.	96%	95%	81%	93%	100%	86%	86%	100%	83%	80%
The ethics guest lecture was relevant to me.	91%	86%	90%	89%	100%	86%	78%	100%	89%	80%
The ethics guest lecture helped me think more clearly about the moral issues we discussed.	91%	98%	76%	87%	80%	71%	78%	100%	83%	60%
The ethics guest lecture increased my interest in learning about the moral issues we discussed.	83%	90%	86%	84%	87%	86%	81%	100%	72%	80%
I would be interested in learning more about ethics in future computer science courses.	83%	83%	90%	85%	73%	86%	76%	100%	74%	100%

nal course grade and when faculty are physically present in the Embedded EthiCS class session, students understand that the faculty value the place of ethical reasoning in the course and that the module is a core element of the course content rather than an optional supplement.

We have found that Ph.D. students and postdoctoral fellows who are teaching assistants for Embedded EthiCS can embed modules in three to four different courses per term, depending on how many modules are new and how much material is already available. Although the philosophy teaching assistants' work differs from the usual leading of discussion sections and grading essays, the workload for preparing and teaching three or four Embedded EthiCS modules is the same, 14–20 hours a week. Further, teaching assistants who have participated to date report they benefited enormously from exposure to a breadth of computer science concepts and methods for which their philosophical expertise is relevant. We anticipate this experience will also prove valuable on the job market and, for many, to their research.

Several of the barriers we encountered are common within cross-disciplinary ventures. First, we saw typical insecurities: philosophers who were concerned about their lack of technical expertise and computer scientists who were concerned about their lack of familiarity with ethics and reluctant to discuss ethical issues with students. Although we found the technical barrier to productive ethical discussions of computer science methods and systems is much lower than many philosophers expect, philosophers without a background in computer science still need support, both financial and intellectual, to develop the requisite background knowledge. We also found many computer science faculty interested in attending a brief introductory computer ethics course focused on philosophical theories and methods relevant to computing technology challenges, a possibility we are currently exploring.

Our experience suggests the process of co-designing Embedded EthiCS modules helps mitigate insecurities, and the presence in the class of philosophers with the expertise to answer



Embedded EthiCS modules are most effective when the issues raised connect technical material to ethical issues already salient to students.



questions is essential for success. Strategies we have found helpful include selecting the topic for the ethics module before the semester begins, with input from both computer science faculty and philosophers. Doing so provides the philosopher time to develop the required technical knowledge in the relevant specific content area. Building a repository of past material for reuse or to serve as models for future modules has also proved useful.

A second barrier arises from the disciplines' different methodologies and vocabularies. In the setting of a computer science course, students accustomed to problem sets with a single correct answer often have trouble when there are several acceptable answers to ethical problems. To address this challenge, within each module, we discuss the controversial nature of some ethical problems, model successful ethical reasoning and inclusive discussion during class, and design activities for students to practice this kind of reasoning and discussion. To bridge the ethics and computer science course vocabularies, and to foreground the philosophical material in need of more explanation, computer science faculty work together with the Embedded EthiCS team in the design and implementation of the modules.

A third cross-field challenge is recruiting philosophers to develop and teach the Embedded EthiCS modules. Hiring numerous Ph.D. candidates from a single philosophy graduate program to cover the full range of computer science courses is impractical. The Philosophy Department needs these graduate students to teach its own courses, and the students themselves need experience teaching those courses. To address this challenge, we are including philosophy postdoctoral fellows in the teaching assistant cadre. For these postdoctoral fellows too, we expect the training and experience they gain will benefit their research and employment profile.

We also uncovered assessment and institutional challenges. The approach of integrating ethics pedagogy into core computer science courses reflects a hypothesis that recurring exposure to this type of reasoning across the curriculum will habituate students to thinking ethically when pursuing

technical work. Post-module surveys provide insight into the effectiveness of particular modules, but we want to measure the approach's impact over the course of years, for instance, as students complete their degrees and even later in their careers. By design, Embedded EthiCS makes small interventions in individual courses, precluding the usefulness of short-term evaluations of impact at the individual course level (for example, pre-course/post-course surveys²²). We thus need to find ways to measure the long-term effectiveness of the Embedded EthiCS approach and compare it to other approaches. As measuring the impact of teaching ethics within the CS curriculum is a challenge regardless of approach, we aim to identify broadly applicable methods.

The institutional challenges to mounting Embedded EthiCS derive from its cross-disciplinary nature. In particular, university support, both financial and administrative, is crucial. Funding is needed for teaching assistants and postdoctoral fellows, including senior level postdoctoral fellows able to train and support the efforts of those developing modules for courses. Administrative support is needed for recruiting faculty and courses in computer science, for recruiting teaching assistants and postdoctoral fellows in philosophy, and for organizing and managing a repository of materials for the program, including modules and evaluation materials. Several of these challenges are made more complex because they cross university divisions.

Looking Forward

Teaching computer scientists to identify and address ethical problems starting from the design phase is as important as enabling them to develop algorithms and programs that work efficiently. The strategy of integrating the teaching of ethical reasoning skills with the teaching of computational techniques into existing computer science coursework not only provides students valuable experience identifying, confronting, and working through ethical questions, but also communicates the need to identify, confront, and address ethical questions throughout their work in computer science. It provides them with ethical reasoning skills to take into their computing and

information technology work after they graduate, preparing them to produce socially and ethically responsible computer technology, and to justify their ethically motivated design choices to their colleagues and employers. Computer scientists and technologists with these capabilities are important for the long-term well-being of our society.

We invite those at other institutions to join us by integrating ethics throughout their own computer science curriculum and to help us expand the open repositories of resources we are developing for ethics modules, including in-class activities, case studies, assignments, and recommended readings. We also think it is important to share lessons learned, approaches to meeting the challenges of university support for these efforts, and ways to engage and train philosophers to participate in them. 

References

- Angwin, J., Larson, J., Mattu, S. and Kirchner, L. Machine Bias. (May 2016). Retrieved Oct. 13, 2018 ; <https://bit.ly/1XMKh5R>.
- Barocas, S. and Nissenbaum, H. Big data's end run around procedural protections. *Commun. ACM* 57, 11 (Nov. 2014).
- Barocas, S. and Selbst, A. Big data's disparate impact. *California Law Review* 104, 671 (2016).
- Burton, E., Goldsmith, J. and Mattei, N. How to teach computer ethics through science fiction. *Commun. ACM* 61, 8 (Aug. 2018).
- Califf, M.E. and Goodwin, M. 2005. Effective incorporation of ethics into courses that focus on programming. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*. (St. Louis, MO, 2015) ACM Press, New York, NY, 347–351. DOI: 10.1145/1047344.1047464
- Cech, E.A. Culture of disengagement in engineering education? *Science, Technology, & Human Values* 39, 1 (2014), 42–72.
- Dwork, C. Differential privacy. *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, (eds) Lecture Notes in Computer Science 4052 (2006). Springer, Berlin, Heidelberg.
- Economist*. Do social media threaten democracy? (Nov. 2017). Retrieved Oct. 13, 2018; <https://econ.st/2xf31GL>.
- Hellman, D. *When is Discrimination Wrong?* Harvard University Press, Cambridge, MA, 2011.
- Hollander, R. and Arenberg, C.R. (eds.). *Ethics Education and Scientific and Engineering Research: What's Been Learned? What Should Be Done? Summary of a Workshop*. National Academy of Engineering. The National Academies Press, Washington, D.C, 2009.
- Kleinberg, J., Mullainathan, S. and Raghavan, M. *Inherent Trade-Offs in the Fair Determination of Risk Scores* (2016), arXiv:1609.05807
- Knight, W. Biased algorithms are everywhere, and no one seems to care. *Technology Review*, (July 2017). Retrieved Oct. 13, 2018 from <https://bit.ly/2tIh1EX>.
- Levin, S. Uber crash shows 'catastrophic failure' of self-driving technology, experts say. *The Guardian* (Mar. 2018); <https://bit.ly/2pych2k>.
- Levy, K. and Barocas, S. Refractive surveillance: Monitoring customers to manage workers. *Intern. J. Commun.* 12 (2018), 1166–1188.
- Lohr, S. Facial recognition is accurate if you're a white guy. *New York Times* (Feb. 2018); <https://nyti.ms/2H3QeaT>.
- Mill, J.S. *On Liberty*. Parker and Son, London, U.K., 1859.
- New York Times* Editorial Board. Facebook and the digital virus called fake news. (Nov. 2016); <https://nyti.ms/2vIMC9Y>.
- Pease, A. and Baker, R. Union College's Rapaport

Everyday Ethics Across the Curriculum Initiative. *Teaching Ethics*, 2009.

- Rawls, J. *A Theory of Justice*. Belknap, Cambridge, MA, USA, 1971.
- Rumbold, B. and Wilson, J. Privacy rights and public information. *J. Political Philosophy*, 2018.
- Scanlon, T.M. *The Difficulty of Tolerance: Essays in Political Philosophy*. Cambridge University Press, Cambridge, MA, USA, 2006.
- Skirpan, M., Beard, N., Bhaduri, S., Fiesler, C. and Yeh, T. Ethics education in context: A case study of novel ethics activities for the CS classroom. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, MD, 2018). ACM Press, New York, NY, 940–945; DOI:10.1145/3159450.3159573
- Sweeney, L. Uniqueness of simple demographics in the U.S. population. LIDAP-WP4, 2000. Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, PA.
- Tasioulas, J. The moral reality of human rights. *Freedom from Poverty as a Human Right: Who Owes What to the Very Poor?* T. Pogge, (ed). Oxford University Press, co-Published with UNESCO, 2007.
- Wasserman, D., Asch, A., Blustein, J. and Putnam, D. Disability: Definitions, models, experience. *The Stanford Encyclopedia of Philosophy* (Summer 2016 Edition). Retrieved July 19, 2018 from <https://plato.stanford.edu/archives/sum2016/entries/disability/>.

Jeff Behrends (jbehrends@fas.harvard.edu) is a Lecturer of Philosophy at Harvard University, Cambridge, MA, USA, and Director of Ethics and Technology Initiatives at the Edmond J. Safra Center for Ethics.

David Gray Grant (davidgraygrant@gmail.com) is a postdoctoral fellow in philosophy at Harvard University where he co-leads the Embedded EthiCS teaching lab, and a research fellow in digital ethics at the Jain Family Institute.

Barbara J. Grosz (grosz@eecs.harvard.edu) is Higgins Professor of Natural Sciences on the Computer Science faculty of the John A. Paulson School of Engineering and Applied Sciences at Harvard University, Cambridge, MA, USA, and a member of the External Faculty of Santa Fe Institute, Santa Fe, NM, USA. She is co-founder and co-director of the Embedded EthiCS initiative and also created the Harvard course "Intelligent Systems: Design and Ethical Challenges."

Lily Hu (lilyhu@g.harvard.edu) is a Ph.D. candidate in Applied Mathematics and a Fellow at the Berkman Klein Center, both at Harvard University, Cambridge, MA, USA. She has served as a teaching assistant for the course on "Intelligent Systems: Design and Ethical Challenges."

Alison Simmons (asimmons@fas.harvard.edu) is the Samuel H. Wolcott Professor of Philosophy at Harvard University, Cambridge, MA, USA. She is co-founder and co-director of the Embedded EthiCS initiative.

Kate Vredenburg (kvredenburg@fas.harvard.edu) was an Embedded EthiCS teaching assistant during her time as a Ph.D. candidate in philosophy at Harvard University. She is currently a postdoctoral fellow at Stanford's McCoy Family Center for Ethics in Society and will join the faculty of the London School of Economics as an Assistant Professor in 2020.

Jim Waldo (waldo@g.harvard.edu) is a professor of the practice of computer science at Harvard University, Cambridge, MA, USA, where he is also the chief technology officer for the School of Engineering, a position he assumed after leaving Sun Microsystems Laboratories. He teaches courses in privacy, technology ethics, and distributed systems.

Copyright held by authors/owners. Publication rights licensed to ACM. \$15.00.



Watch the authors discuss this work in the exclusive *Communications* video. <https://cacm.acm.org/videos/embedded-ethics>

DOI:10.1145/3338112

Key lessons for designing static analyses tools deployed to find bugs in hundreds of millions of lines of code.

BY DINO DISTEFANO, MANUEL FÄHNDRICH, FRANCESCO LOGOZZO, AND PETER W. O'HEARN

Scaling Static Analyses at Facebook

STATIC ANALYSIS TOOLS are programs that examine, and attempt to draw conclusions about, the source of other programs without running them. At Facebook, we have been investing in advanced static analysis tools that employ reasoning techniques similar to those from program verification. The tools we describe in this article (Infer and Zoncolan) target issues related to crashes and to the security of our services, they perform sometimes complex reasoning spanning many procedures or files, and they are integrated into engineering workflows in a way that attempts to bring value while minimizing friction.

These tools run on code modifications, participating as bots during the code review process. Infer targets our mobile apps as well as our backend C++ code, codebases with 10s of millions of lines; it has seen over 100 thousand reported issues fixed by developers before code reaches production. Zoncolan targets the 100-million lines of Hack code, and is additionally

integrated in the workflow used by security engineers. It has led to thousands of fixes of security and privacy bugs, outperforming any other detection method used at Facebook for such vulnerabilities. We will describe the human and technical challenges encountered and lessons we have learned in developing and deploying these analyses.

There has been a tremendous amount of work on static analysis, both in industry and academia, and we will not attempt to survey that material here. Rather, we present our rationale for, and results from, using techniques similar to ones that might be encountered at the edge of the research literature, not only simple techniques that are much easier to make scale. Our goal is to complement other reports on industrial static analysis and formal methods,^{1,6,13,17} and we hope that such perspectives can provide input both to future research and to further industrial use of static analysis.

Next, we discuss the three dimensions that drive our work: bugs that matter, people, and actioned/missed bugs. The remainder of the article describes our experience developing and deploying the analyses, their impact, and the techniques that underpin our tools.

Context for Static Analysis at Facebook

Bugs that Matter. We use static analysis to prevent bugs that would affect our products, and we rely on our engineers' judgment as well as data from production to tell us the bugs that matter the most.

» key insights

- **Advanced static analysis techniques performing deep reasoning about source code can scale to large industrial codebases, for example, with 100-million LOC.**
- **Static analyses should strike a balance between missed bugs (false negatives) and un-actioned reports (false positives).**
- **A "diff time" deployment, where issues are given to developers promptly as part of code review, is important to catching bugs early and getting high fix rates.**



It is important for a static analysis developer to realize that not all bugs are the same: different bugs can have different levels of importance or severity depending on the context and the nature. A memory leak on a seldom-used service might not be as important as a vulnerability that would allow attackers to gain access to unauthorized information. Additionally, the frequency of a bug type can affect the decision of how important it is to go after. If a certain kind of crash, such as a null pointer error in Java, were happening hourly, then it might be more important to target than a bug of similar severity that occurs only once a year.

We have several means to collect data on the bugs that matter. First of all, Facebook maintains statistics on crashes and other errors that happen in production. Second, we have a “bug bounty” program, where people outside the company can report vul-

nerabilities on Facebook, or on apps of the Facebook family; for example, Messenger, Instagram, or WhatsApp. Third, we have an internal initiative for tracking the most severe bugs (SEV) that occur.

Our understanding of Bugs that Matter at Facebook drives our focus on advanced analyses. For contrast, a recent paper states: “All of the static analyses deployed widely at Google are relatively simple, although some teams work on project-specific analysis frameworks for limited domains (such as Android apps) that do interprocedural analysis”¹⁷ and they give their entirely logical reasons. Here, we explain why Facebook made the decision to deploy interprocedural analysis (spanning multiple procedures) widely.

People and deployments. While not all bugs are the same, neither are all users; therefore, we use different deployment models depending on the

intended audience (that is, the people the analysis tool will be deployed to).

For classes of bugs intended for all or a wide variety of engineers on a given platform, we have gravitated toward a “diff time” deployment, where analyzers participate as bots in code review, making automatic comments when an engineer submits a code modification. Later, we recount a striking situation where the diff time deployment saw a 70% fix rate, where a more traditional “offline” or “batch” deployment (where bug lists are presented to engineers, outside their workflow) saw a 0% fix rate.

In case the intended audience is the much smaller collection of domain security experts in the company, we use two additional deployment models. At “diff time,” security related issues are pushed to the security engineer on-call, so she can comment on an in-progress code change when necessary. Addition-

ally, for finding all instances of a given bug in the codebase or for historical exploration, offline inspection provides a user interface for querying, filtering, and triaging all alarms.

In all cases, our deployments focus on the people our tools serve and the way they work.

Actioned reports and missed bugs.

The goal of an industrial static analysis tool is to help people: at Facebook, this means the engineers, directly, and the people who use our products, indirectly. We have seen how the deployment model can influence whether a tool is successful. Two concepts we use to understand this in more detail, and to help us improve our tools, are *actioned reports* and *observable missed bugs*.

The kind of action taken as a result of a reported bug depends on the deployment model as well as the type of bug. At diff time an action is an update to the diff that removes a static analysis report. In Zoncolan's offline deployment a report can trigger the security expert to create a task for the product engineer if the issue is important enough to follow up with the product team. Zoncolan catches more SEVs than either manual security reviews or bug bounty reports. We measured that 43.3% of the severe security bugs are detected via Zoncolan. At press time, Zoncolan's "action rate" is above 80% and we observed about 11 "missed bugs."

A missed bug is one that has been observed in some way, but that was not reported by an analysis. The means of observation can depend on the kind of bug. For security vulnerabilities we have bug bounty reports, security reviews, or SEV reviews. For our mobile apps we log

crashes and app not-responding events that occur on mobile devices.

The actioned reports and missed bugs are related to the classic concepts of true positives and false negatives from the academic static analysis literature. A true positive is a report of a potential bug that can happen in a run of the program in question (whether or not it will happen in practice); a false positive is one that cannot happen. Common wisdom in static analysis is that it is important to keep control of the false positives because they can negatively impact engineers who use the tools, as they tend to lead to apathy toward reported alarms. This has been emphasized, for instance, in previous *Communications*' articles on industrial static analysis.^{1,17} False negatives, on the other hand, are potentially harmful bugs that may remain undetected for a long time. An undetected bug affecting security or privacy can lead to undetected exploits. In practice, fewer false positives often (though not always) implies more false negatives, and vice versa, fewer false negatives implies more false positives. For instance, one way to reign in false positives is to fail to report when you are less than sure a bug will be real; but silencing an analysis in this way (say, by ignoring paths or by heuristic filtering) has the effect of missing bugs. And, if you want to discover and report more bugs you might also add more spurious behaviors.

The reason we are interested in advanced static analyses at Facebook might be understood in classic terms as saying: false negatives matter to us. However, it is important to note the number of false negatives is notoriously difficult to quantify (how many unknown bugs are there?). Equally,

though less recognized, the false positive rate is challenging to measure for a large, rapidly changing codebase: it would be extremely time consuming for humans to judge all reports as false or true as the code is changing.

Although true positives and false negatives are valuable concepts, we don't make claims about their rates and pay more attention to the action rate and the (observed) missed bugs.

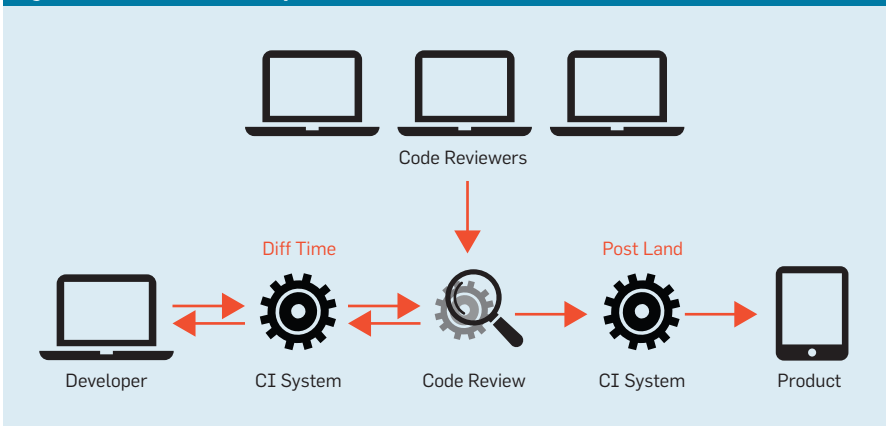
Challenges: Speed, scale, and accuracy. A first challenge is presented by the sheer scale of Facebook's codebases, and the rate of change they see. For the server-side, we have over 100-million lines of Hack code, which Zoncolan can process in less than 30 minutes. Additionally, we have 10s of millions of both mobile (Android and Objective C) code and backend C++ code. Infer processes the code modifications quickly (within 15 minutes on average) in its diff time deployment. All codebases see thousands of code modifications each day and our tools run on each code change. For Zoncolan, this can amount to analyzing one trillion lines of code (LOC) per day.

It is relatively straightforward to scale program analyses that do simple checks on a procedure-local basis only. The simplest form is linters, which give syntactic style advice (for example, "the method you called is to be deprecated, please consider rewriting"). Such simple checks provide value and are in wide deployment in major companies including Facebook; we will not comment on them further in this article. But for more reasoning going beyond local checks, such as one would find in the academic literature on static analysis, scaling to 10s or 100s of millions of LOC is a challenge, as is the incremental scalability needed to support diff time reporting.

Infer and Zoncolan both use techniques similar to some of what one might find at the edge of the research literature. Infer, as we will discuss, uses one analysis based on the theory of Separation Logic,¹⁶ with a novel theorem prover that implements an inference technique that guesses assumptions.⁵ Another Infer analysis involves recently published research results on concurrency analysis.^{2,10} Zoncolan implements a new modular parallel taint analysis algorithm.

But how can Infer and Zoncolan scale? The core technical features they

Figure 1. Continuous development.



share are compositionality and carefully crafted abstractions. For most of this article we will concentrate on what one gets from applying Infer and Zoncolan, rather than on their technical properties, but we outline their foundations later and provide more technical details in an online appendix (<https://dl.acm.org/citation.cfm?doid=3338112&picked=formats>).

The challenge related to accuracy is intimately related to actioned reports and missed bugs. We try to strike a balance between these issues, informed by the desires based on the class of bugs and the intended audience. The more severe a potentially missed issue is, the lower the tolerance for missed bugs. Thus, for issues that indicate a potential crash or performance regression in a mobile app such as Messenger, WhatsApp, Instagram, or Facebook, our tolerance for missed bugs is lower than, for example, stylistic lint suggestions (for example, don't use deprecated method). For issues that could affect the security of our infrastructure or the privacy of the people using our products, our tolerance for false positives is higher still.

Software Development at Facebook

Facebook practices continuous software development,⁹ where a main codebase (master) is altered by thousands of programmers submitting code modifications (diffs). Master and diffs are the analogues of, respectively, GitHub master branch and pull requests. The developers share access to a codebase and they land, or commit, a diff to the codebase after passing code review. A *continuous integration system* (CI system) is used to ensure code continues to build and passes certain tests. Analyses run on the code modification and participate by commenting their findings directly in the code review tool.

The Facebook website was originally written in PHP, and then ported to Hack, a gradually typed version of PHP developed at Facebook (<https://hacklang.org/>). The Hack codebase spans over 100 million lines. It includes the Web frontend, the internal web tools, the APIs to access the social graph from first- and third-party apps, the privacy-aware data abstractions, and the privacy control logic for viewers and apps. Mobile apps—for Facebook, Messenger, Instagram and



The reason we are interested in advanced static analyses at Facebook might be understood in classic terms: false negatives matter to us.



WhatsApp—are mostly written in Objective-C and Java. C++ is the main language of choice for backend services. There are 10s of millions of lines each of mobile and backend code.

While they use the same development models, the website and mobile products are deployed differently. This affects what bugs are considered most important, and the way that bugs can be fixed. For the website, Facebook directly deploys new code to its own datacenters, and bug fixes can be shipped directly to our datacenters frequently, several times daily and immediately when necessary. For the mobile apps, Facebook relies on people to download new versions to from the Android or the Apple store; new versions are shipped weekly, but mobile bugs are less under our control because even if a fix is shipped it might not be downloaded to some people's phones.

Common runtime errors—for example, null pointer exceptions, division by zero—are more difficult to get fixed on mobile than on the server. On the other hand, server-side security and privacy bugs can severely impact both the users of the Web version of Facebook as well as our mobile users, since the privacy checks are performed on the server-side. As a consequence, Facebook invests in tools to make the mobile apps more reliable and server-side code more secure.

Moving Fast with Infer

Infer is a static analysis tool applied to Java, Objective C, and C++ code at Facebook.⁴ It reports errors related to memory safety, to concurrency, to security (information flow), and many more specialized errors suggested by Facebook developers. Infer is run internally on the Android and iOS apps for Facebook, Instagram, Messenger, and WhatsApp, as well as on our backend C++ and Java code.

Infer has its roots in academic research on program analysis with separation logic,⁵ research, which led to a startup company (Monoidics Ltd.) that was acquired by Facebook in 2013. Infer was open sourced in 2015 (www.fbinfer.com) and is used at Amazon, Spotify, Mozilla, and other companies.

Diff-time continuous reasoning. Infer's main deployment model is based on fast incremental analysis of code changes. When a diff is submitted to code review an instance of Infer is run

in Facebook's internal CI system (Figure 1). Infer does not need to process the entire codebase in order to analyze a diff, and so is fast.

An aim has been for Infer to run in 15min–20min on a diff on average, and this includes time to check out the source repository, to build the diff, and to run on base and (possibly) parent commits. It has typically done so, but we constantly monitor performance to detect regressions that makes it take longer, in which case we work to bring the running time back down. After running on a diff, Infer then writes comments to the code review system. In the default mode used most often it reports only regressions: new issues introduced by a diff. The “new” issues are calculated using a bug equivalence notion that uses a hash involving the bug type and location-independent information about the error message, and which is sensitive to file moves and line number changes cause by refactoring, deleting, or adding code; the aim is to avoid presenting warnings that developers might regard as pre-existing. Fast reporting is important to keep in tune with the developers' workflows. In contrast, when Infer is run in whole-program mode it can take more than an hour (depending on the app)—too slow for diff-time at Facebook.

Human factors. The significance of the diff-time reasoning of Infer is best understood by contrast with a failure. The first deployment was batch rather than continuous. In this mode Infer would be run once per night on the entire Facebook Android codebase, and it would generate a list of issues. We manually looked at the issues, and

assigned them to the developers we thought best able to resolve them.

The response was stunning: we were greeted by near silence. We assigned 20–30 issues to developers, and almost none of them were acted on. We had worked hard to get the false positive rate down to what we thought was less than 20%, and yet the fix rate—the proportion of reported issues that developers resolved—was near zero.

Next, we switched Infer on at diff time. The response of engineers was just as stunning: the fix rate rocketed to over 70%. The same program analysis, with same false positive rate, had much greater impact when deployed at diff time.

While this situation was surprising to the static analysis experts on the Infer team, it came as no surprise to Facebook's developers. Explanations they offered us may be summarized in the following terms:

One problem that diff-time deployment addresses is the *mental effort of context switch*. If a developer is working on one problem, and they are confronted with a report on a separate problem, then they must swap out the mental context of the first problem and swap in the second, and this can be time consuming and disruptive. By participating as a bot in code review, the context switch problem is largely solved: programmers come to the review tool to discuss their code with human reviewers, with mental context already swapped in. This also illustrates how important timeliness is: if a bot were to run for an hour or more on a diff it could be too late to participate effectively.

A second problem that diff-time deployment addresses is relevance. When

an issue is discovered in the codebase, it can be nontrivial to assign it to the right person. In the extreme, somebody who has left the company might have caused the issue. Furthermore, even if you think you have found someone familiar with the codebase, the issue might not be relevant to any of their past or current work. But, if we comment on a diff that introduces an issue then there is a pretty good (but not perfect) chance that it is relevant.

Mental context switch has been the subject of psychological studies,¹² and it is, along with the importance of relevance, part of the received collective wisdom impressed upon us by Facebook's engineers. Note that others have also remarked on the benefits of reporting during code review.¹⁷

At Facebook, we are working actively on moving other testing technologies to diff time when possible. We are also supporting academics on researching incremental fuzzing and symbolic execution techniques for diff time reporting.

Interprocedural bugs. Many of the bugs that Infer finds involve reasoning that spans multiple procedures or files. An example from OpenSSL illustrates:

```
apps/ca.c:2780: NULL_DEREFERENCE
pointer 'revtm' last assigned on line
2778 could be null
and is dereferenced at line 2780, column 6
2778. revtm = X509_gmtime_adj(NULL, 0);
2779.
2780. i = revtm->length + 1;
```

The issue is that the procedure `X509_gmtime_adj()` can return null in some circumstances. Overall,

Figure 2. A simple example capturing a common safety pattern used in Android apps.

Threading information is used to limit the amount of synchronization required. As a comment from the original code explains: “mCount is written to only by the main thread with the lock held, read from the main thread with no lock held, or read from any other thread with the lock held.” Bottom: unsafe additions to `RaceWithMainThread.java`.

```
1  @ThreadSafe
2  class RaceWithMainThread {
3      int mCount;
4      void protectedWriteOnMainThread_OK() {
5          OurThreadUtils.assertMainThread();
6          synchronized (this) { mCount = 1; }
7      }
8      int unprotectedReadOnMainThread_OK() {
9          OurThreadUtils.assertMainThread();
10         return mCount;
11     }
12     synchronized int protectedReadOffMainThread_OK() {
13         return mCount;
14     }
15     synchronized void
16     protectedWriteOffMainThread_BAD() {
17         mCount = 2;
18     }
19     int unprotectedReadOffMainThread_BAD() {
20         return mCount;
21     }
```

the error trace found by Infer has 61 steps, and the source of null, the call to `X509_gmtime_adj()` goes five procedures deep and it eventually encounters a return of null at call-depth 4. This bug was one of 15 that we reported to OpenSSL which were all fixed.

Infer finds this bug by performing compositional reasoning, which allows covering interprocedural bugs while still scaling to millions of LOC. It deduces a precondition/postcondition specification approximating the behavior of `X509_gmtime_adj`, and then uses that specification when reasoning about its calls. The specification includes 0 as one of the return values, and this triggers the error.

In 2017, we looked at bug fixes in several categories and found that for some (null dereferences, data races, and security issues) over 50% of the fixes were for bugs with traces that were interprocedural.^a The interprocedural bugs would be missed bugs if we only deployed procedure-local analyses.

Concurrency. A concurrency capability recently added to Infer, the *RacerD* analysis, provides an example of the benefit of feedback between program analysis researchers and product engineers.^{2,15} Development of the analysis started in early 2016, motivated by Concurrent Separation Logic.³ After 10 months of work on the project, engineers from News Feed on Android caught wind of what we were doing and reached out. They were planning to convert part of Facebook's Android app from a sequential to a multithreaded architecture. Hundreds of classes written for a single-threaded architecture had to be used now in a concurrent context: the transformation could introduce concurrency errors. They asked for interprocedural capabilities because Android UI is arranged in trees with one class per node. Races could happen via interprocedural call chains sometimes spanning several classes, and mutations almost never happened at the top level: procedural local analysis would miss most races.

We had been planning to launch the proof tool we were working on in a year's time, but the Android engineers were starting their project and needed help sooner. So we pivoted to a *minimum viable product*, which would serve the engi-



Advanced static analyses, like those found in the research literature, can be deployed at scale and deliver value for general code.



neers—it had to be fast, with actionable reports, and not too many missed bugs on product code (but not on infrastructure code).^{2,15} The tool borrowed ideas from concurrent separation logic, but we gave up on the ideal of proving absolute race freedom. Instead, we established a ‘completeness’ theorem saying that, under certain assumptions, a theoretical variant of the analyzer reports only true positives.¹⁰

The analysis checks for data races in Java programs—two concurrent memory accesses, one of which is a write. The example in Figure 2 (top) illustrates: If we run the Infer on this code it doesn't find a problem. The unprotected read and the protected write do not race because they are on the same thread. But, if we include additional methods that do conflict, then Infer will report races, as in Figure 2, bottom.

Impact. Since 2014, Facebook's developers have resolved over 100,000 issues flagged by Infer. The majority of Infer's impact comes from the diff-time deployment, but it is also run batch to track issues in master, issues addressed in fix-athons and other periodic initiatives.

The *RacerD* data race detector saw over 2,500 fixes in the year to March 2018. It supported the conversion of Facebook's Android app from a single-threaded to a multithreaded architecture by searching for potential data races, without the programmers needing to insert annotations for saying which pieces of memory are guarded by what locks. This conversion led to an improvement in scroll performance and, speaking about the role of the analyzer, Benjamin Jaeger, an Android engineer at Facebook, stated:^b “without Infer, multithreading in News Feed would not have been tenable.” As of March 2018, no Android data race bugs missed by Infer had been observed in the previous year (modulo 3 analyzer implementation errors).²

The fix rate for the concurrency analysis to March 2018 was roughly 50%, lower than for the previous general diff analysis. Our developers have emphasized that they appreciate the reports because concurrency errors are difficult to debug. This illustrates our earlier points about balancing action rates and bug severity. See Blackshear et al.² for more discussion on fix rates.

a <https://bit.ly/2WloBVj>

b <https://bit.ly/2xurbMl>

Overall, Infer reports on over 30 types of issues, ranging from deep inter-procedural checks to simple procedure-local checks and lint rules. Concurrency support includes checks for deadlocks and starvation, with hundreds of “app not-responding” bugs being fixed in the past year. Infer has also recently implemented a security analysis (a ‘taint’ analysis), which has been applied to Java and C++ code; it gained this facility by borrowing ideas from Zoncolan.

Staying Secure with Zoncolan

One of the original reasons for the development and adoption of Hack was

to enable more powerful analysis of the core Facebook codebase. Zoncolan is the static analysis tool we built to find code and data paths that may cause a security or a privacy violation in our Hack codebase.

The code in Figure 3 is an example of a vulnerability prevented by Zoncolan. If the `member_id` variable on line 21 contains the value `../../users/delete_user/`, it is possible to redirect this form into any other form on Facebook. On submission of the form, it will invoke a request to `https://facebook.com/groups/add_member/../../users/delete_user/` that will delete

the user’s account. The root cause of the vulnerability in Figure 3 is that the attacker controls the value of the `member_id` variable which is used in the `action` field of the `<form>` element. Zoncolan follows the interprocedural flow of untrusted data (for example, user input) to sensitive parts of the codebase. Virtual calls do make interprocedural analysis difficult since the tool generally does not know the precise type of an object. To avoid missing paths (and thus bugs), Zoncolan must consider all the possible functions a call may resolve to.

SEV-oriented static analysis development. We designed and developed Zoncolan in collaboration with the Facebook App Security team. Alarms reported by Zoncolan are inspired by security bugs uncovered by the App Security team.

The initial design of Zoncolan began with a list of SEVs that were provided to us by security engineers. For each bug we asked ourselves: “How could we have caught it with static analysis?” Most of those historical bugs were no longer relevant because the programming language or a secure framework prevented them from recurring—for instance, the widespread adoption of XHP made it possible to build XSS-free Web pages by construction. We realized the remaining bugs involved interprocedural flows of untrusted data, either directly or indirectly, into some privileged APIs. Detecting such bugs can be automated with static taint flow analysis,¹⁸ which tracks how the data originating from some untrusted sources reaches or influences the data reaching some sensitive parts of the codebase (sinks).

When a security engineer discovers a new vulnerability, we evaluate whether that class of vulnerability is amenable to static analysis. If it is, we prototype the new rule, iterating with the feedback of the engineer in order to refine results to strike the right balance of false positives/false negatives. When we believe the rule is good enough, it is enabled on all runs of Zoncolan in production. We adopt the standard Facebook App Security severity framework, which associates to each vulnerability an impact level, in a scale from 1 (best-practice) to 5 (SEV-worthy). A security impact level of 3 or more is considered severe.

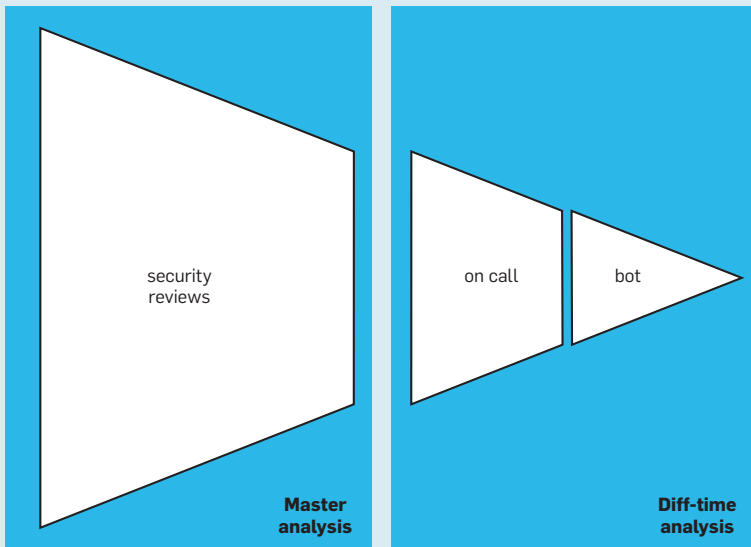
Scaling the analysis. A main challenge was to scale Zoncolan to a codebase of more than 100 millions of LOC

Figure 3. Example of a bug that Zoncolan prevents. It may cause the attacker to delete a user account. The attacker can provide an input on line 5 that causes a redirection to any other form on Facebook at line 20.

```

1  <?hh
2  class AddMemberToGroup extends FacebookEndpoint {
3    private function getIDs (): (string, int) {
4      // User input, untrusted
5      return tuple((string) $this->getRequest('member_id'),
6                  (int) $this->getRequest('gid'));
7    }
8
9    public function render(): :xhp {
10     list($member_id, $group_id) = $this->getIDs();
11     return this->getConfirmationForm($group_id, $member_id);
12   }
13
14   public function getConfirmationForm
15   (int $group_id, string $member_id): :xhp {
16     $url = "https://facebook.com/groups/add_member/" .
17           $member_id;
18
19     return
20       <form method="post" action={$url}>
21         <input name="gid" value={$group_id}/>
22         <input name="action" value="add"/>
23       </form>;
24   }
25 }
```

Figure 4. Funneled deployment of Zoncolan



code. Thanks to a new parallel, compositional, non-uniform static analysis that we designed, Zoncolan performs the full analysis of the code base in less than 30 minutes on a 24-core server.

Zoncolan builds a dependency graph that relates methods to their potential callers. It uses this graph to schedule parallel analyses of individual methods. In the case of mutually recursive methods, the scheduler iterates the analysis of the methods until it stabilizes, that is, no more flows are discovered. Suitable operators (called widenings in the static analysis literature⁷) ensure the convergence of the iterations. It is worth mentioning that, even though the concept of taint analysis is well established in Academia, we had to develop new algorithms in order to scale to the size of our codebase.

Funneled deployment. Figure 4 provides a graphical representation of the Zoncolan deployment model. This funneled deployment model optimizes bug detection with the goal of supporting security of Facebook: The Zoncolan master analysis finds all existing instances of a newly discovered vulnerability. The Zoncolan diff analysis avoids vulnerabilities from being (re-)introduced in the codebase.

Zoncolan periodically analyzes the entire Facebook Hack codebase to update the master list. The target audience is security engineers performing security reviews. In the master analysis, we expose all alarms found. Security engineers are interested in all existing alarms for a given project or a given category. They triage alarms via a dashboard, which enables filtering by project, code location, source and/or destination of the data, length or features of the trace. When a security engineer finds a bug, he/she files a task for the product group and provides guidance on how to make the code secure. When an alarm is a false positive, he/she files a task for the developers of Zoncolan with an explanation of why the alarm is false. The Zoncolan developers then refine the tool to improve the precision of the analysis. After a category has been extensively tested, the Zoncolan team, in conjunction with the App security team, evaluates if it can be promoted for diff analysis. Often promotion involves improving the signal by filtering the output according to, for example, the length of the inter-procedural trace,

the visibility of the endpoint (external or internal?), and so on. At press time, circa 1/3 of the Zoncolan categories are enabled for diff analysis.

Zoncolan analyzes every Hack code modification and reports alarms if a diff introduces new security vulnerabilities. The target audience is: the author and the reviewers of the diff (Facebook software engineers who are not security experts), and the security engineer in the on-call rotation (who has a limited time budget). When appropriate, the on-call validates the alarm reported, blocks the diff, and provides support to write the code in a secure way. For categories with very high signal, Zoncolan acts as a security bot: it bypasses the security on-call and instead comments directly on the diff. It provides a detailed explanation on the security vulnerability, how it can be exploited, and includes references to past incidents, for example, SEVs.

Finally, note the funneled deployment model makes it possible to scale up the security fixes, without reducing the overall coverage Zoncolan achieves (that is, without missing bugs): If Zoncolan determines a new issue is not high-signal enough for autocommenting on the diff, but needs to be looked at by an expert, it pushes it to the on-call queue. If the alarm makes neither of these cuts, the issue will end up in the Zoncolan master analysis after the diff is committed.

Impact. Zoncolan has been deployed for more than two years at Facebook, first to security engineers, then to software engineers. It has prevented thousands of vulnerabilities from being introduced to Facebook's codebase. Figure 5 compares the number of SEVs, such as bugs of severity 3-to-5, prevented by Zoncolan, in a six-month period, to the traditional programs adopted by security engineers, such as manual code reviews/pentesting and bug bounty reports. The bars show that at Facebook, Zoncolan catches more SEVs than either manual security reviews or bug bounty reports. We measured that 43.3% of the severe security bugs are detected via Zoncolan.

The graph in Figure 6 shows the distribution of the actioned bugs found by Zoncolan at different stages of the deployment funnel, according to the security impact level. The largest number of categories is enabled for the master analysis, so it is not unexpected that it is the largest bucket. However, when restricting to SEVs, the diff analysis largely overtakes the master analysis—211 severe issues are prevented at diff-time, versus 122 detected on master. Overall, we measured the ratio of Zoncolan actioned bugs to be close to 80%.

We also use the traditional security programs to measure missed bugs (that is, the vulnerabilities for which there is a Zoncolan category), but the

Figure 5. Comparison of severe bugs reported by Zoncolan with respect to security reviews and bug bounty, in a six-month period (darker implies more severe).

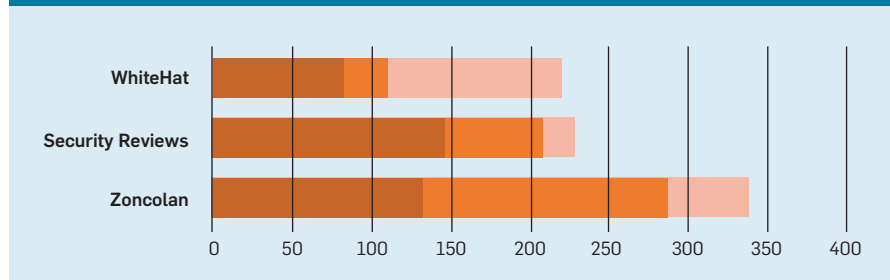
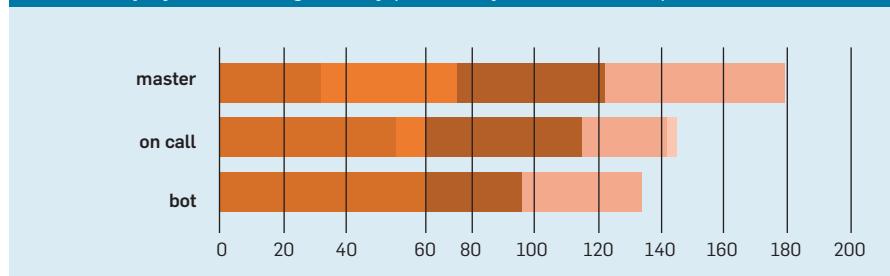


Figure 6. Distribution of all the bugs fixed, in a six-month period, based on Zoncolan's funneled deployment and bug severity (darker implies more severe).



tool failed to report them. To date, we have had about 11 missed bugs, some of them caused by a bug in the tool or incomplete modeling.

Compositionality and Abstraction

The technical features that underpin our analyses are *compositionality* and *abstraction*.

The notion of compositionality comes from language semantics: A semantics is compositional if the meaning of a compound phrase is defined in terms of the meanings of its parts and a means of combining them. The same idea can be applied to program analysis.^{5,8} A program analysis is compositional if the analysis result of a composite program is defined in terms of the analysis results of its parts and a means of combining them. When applying compositionality in program analysis, there are two key questions:

- How to represent the meaning of a procedure concisely?
- How to combine the meanings in an effective way?

For (a) we need to approximate the meaning of a component by abstracting away the full behavior of the procedure and to focusing only on the properties relevant for the analysis. For instance, for security analysis, one may be only interested that a function returns a user-controlled value, when the input argument contains a user-controlled string, discarding the effective value of the string. More formally, the designer of the static analysis defines an appropriate mathematical structure, called the abstract domain,⁷ which allows us to approximate this large function space much more succinctly. The design of a static analysis relies on abstract domains precise enough to capture the properties of interest and coarse enough to make the problem computationally tractable. The ‘abstraction of a procedure meaning’ is often called a procedure summary in the analysis literature.¹⁹

The answer to question (b) mostly depends on the specific abstract domain chosen for the representation of summaries. Further information on the abstractions supported by Infer and Zoncolan, as well as brief information on recursion, fixpoints, and analysis algorithms, may be found in the online technical appendix. It is worth discussing the intuitive reason for why compositional analysis to-

gether with crafted abstract domains can scale: each procedure only needs to be visited a few times, and many of the procedures in a codebase can be analyzed independently, thus opening opportunities for parallelism. A compositional analysis can even have a runtime that is (modulo mutual recursion) a linear combination of the times to analyze the individual procedures. For this to be effective, a suitable abstract domain, for instance limiting or avoiding disjunctions, should also contain the cost of analyzing a single procedure.

Finally, compositional analyses are naturally incremental—changing one procedure does not necessitate re-analyzing all other procedures. This is important for fast diff-time analysis.

Conclusion

This article described how we, as static analysis people working at Facebook, have developed program analyses in response to the needs that arise from production code and engineers’ requests. Facebook has enough important code and problems that it is worthwhile to have embedded teams of analysis experts, and we have seen (for example, in the use of Infer to support multithreaded Android News Feed, and in the evolution of Zoncolan to detect SEV-worthy issues) how this can impact the company. Although our primary responsibility is to serve the company, we believe that our experiences and techniques can be generalized beyond the specific industrial context. For example, Infer is used at other companies such as Amazon, Mozilla, and Spotify; we have produced new scientific results,^{2,10} and proposed new scientific problems.^{11,14} Indeed, our impression as (former) researchers working in an engineering organization is that having science and engineering playing off one another in a tight feedback loop is possible, even advantageous, when practicing static analysis in industry.

To industry professionals we say: advanced static analyses, like those found in the research literature, can be deployed at scale and deliver value for general code. And to academics we say: from an industrial point of view the subject appears to have many unexplored avenues, and this provides research opportunities to inform future tools.

Acknowledgments

Special thanks to Ibrahim Mohamed for being a tireless advocate for Zoncolan among security engineers, to Cristiano Calcagno for leading Infer’s technical development for several years, and to our many teammates and other collaborators at Facebook for their contributions to our collective work on scaling static analysis. 

Readers interested in more technical details of this work are encouraged to review the online appendix; (<https://dl.acm.org/citation.cfm?doid=3338112&picked=formats>).

References

- Bessey, A. et al. A few billion lines of code later: using static analysis to find bugs in the real world. *Commun. ACM* 53, 2 (Feb. 2010), 66–75.
- Blackshear, S., Gorogiannis, N., Sergey, I. and O’Hearn, P. Racerd: Compositional static race detection. In *Proceedings of OOPSLA*, 2018.
- Brookes, S. and O’Hearn, P.W. Concurrent separation logic. *SIGLOG News* 3, 3 (2016), 47–65.
- Calcagno, C. et al. Moving fast with software verification. In *Proceedings of NASA Formal Methods Symposium*, 2015, 3–11.
- Calcagno, C., Distefano, D. O’Hearn, P.W. and Yang, H. Compositional shape analysis by means of bi-abduction. *J. ACM* 58, 6 (2011), 26.
- Cook, B. Formal reasoning about the security of Amazon Web services. *LICS* (2018), 38–47.
- Cousot, P. and Cousot, R. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th POPL*, 1977, 238–252.
- Cousot, P. and Cousot, R. Modular static program analysis. In *Proceedings of 2002 CC*, 159–178.
- Feitelson, D.G., Frachtenberg, E. and Beck, K.L. Development and deployment at Facebook. *IEEE Internet Computing* 17, 4 (2013), 8–17.
- Gorogiannis, N., Sergey, I. and O’Hearn, P. A true positives theorem for a static race detector. In *Proceedings of the 2019 POPL*.
- Harman, M. and O’Hearn, P. From start-ups to scale-ups: Open problems and challenges in static and dynamic program analysis for testing and verification. In *Proceedings of SCAM*, 2018.
- Iqbal, S.T. and Horvitz, E. Disruption and recovery of computing tasks: Field study, analysis, and directions. In *Proceedings of 2007 CHI*, 677–686.
- Larus, J.R. et al. Righting software. *IEEE Software* 21, 3 (2004), 92–100.
- O’Hearn, P. Continuous reasoning: Scaling the impact of formal methods. *LICS*, 2018.
- O’Hearn, P.W. Experience developing and deploying concurrency analysis at Facebook. *SAS*, 2018, 56–70.
- O’Hearn, P.W. Separation logic. *Comm. ACM* 62, 2 (Feb 2019), 86–95.
- Sadowski, C., Aftandilian, E., Eagle, A., Miller-Cushon, L. and Jaspán, C. Lessons from building static analysis tools at Google. *Commun. ACM* 61, 4 (Apr. 2018), 58–66.
- Xie, Y. and Aiken, A. Static detection of security vulnerabilities in scripting languages. In *Proceedings of USENIX Security Symposium*, 2006.
- Yorsh, G., Yahav, E. and Chandra, S. Generating precise and concise procedure summaries. In *Proceedings of 2008 POPL*.

Dino Distefano is a research scientist at Facebook, London, U.K., and a professor of computer science at Queen Mary University of London, U.K.

Manuel Fähndrich is a software engineer at Facebook Research, Seattle, WA, USA.

Francesco Logozzo is a software engineer at Facebook Research, Seattle, WA, USA.

Peter W. O’Hearn is a research scientist at Facebook, London, U.K. and a professor of computer science at University College London, U.K.



The ACM Conference Series on
Recommender Systems

COPENHAGEN, DENMARK

SEPTEMBER 16-20, 2019



RECSYS 2019

IMPORTANT DATES

Early-bird registration deadline: August 2, 2019

Standard registration deadline: September 15, 2019

On-site registration: After September 15, 2019

RecSys is the premier venue for research and applications of recommendation technologies.

Recommender systems are a ubiquitous feature of the modern Internet, mining user activity and item data to help people find new things to purchase, watch, read and enjoy.

The RecSys community brings together academia and industry, research and practice, and multiple disciplines including HCI, ML, IR, business, and psychology to advance recommendation technologies and our understanding of their human dimensions.

GENERAL CHAIRS

Toine Bogers *Aalborg University Copenhagen, Denmark*

Alan Said *University of Gothenburg, Sweden*

For more information, visit
<https://recsys.acm.org/recsys19/>



Association for
Computing Machinery



SIGCHI

The ability to build a construct that organizes work from different devices and information resources is as complex as it is invaluable.

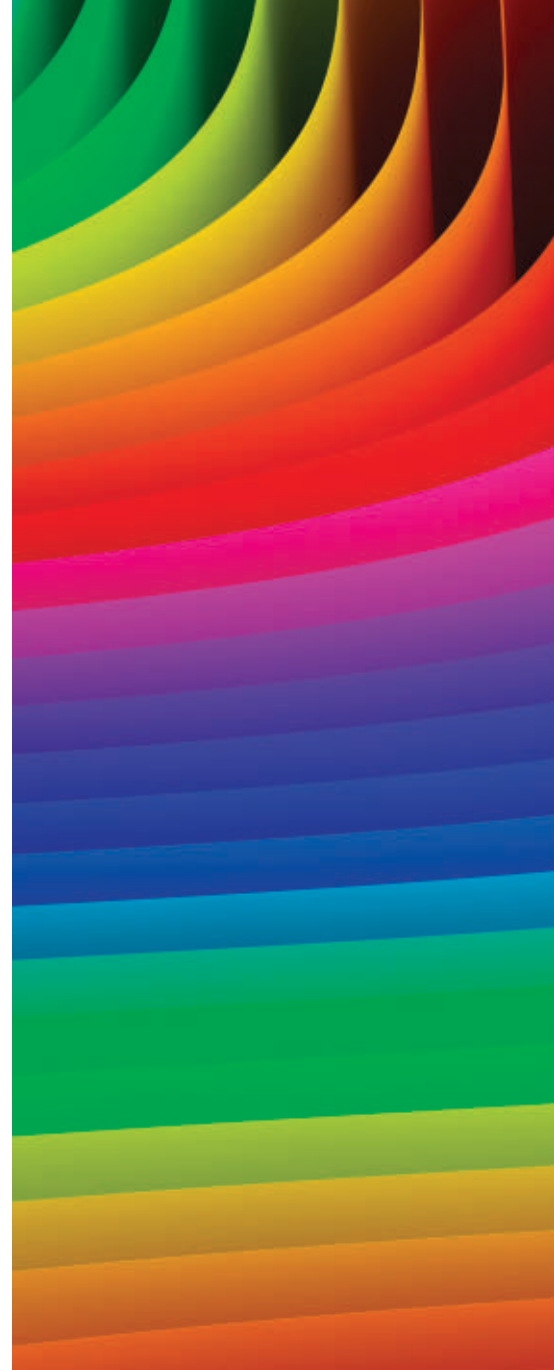
BY JAKOB E. BARDRAM, STEVEN JEURIS, PAOLO TELL, STEVEN HOUBEN, AND STEPHEN VOIDA

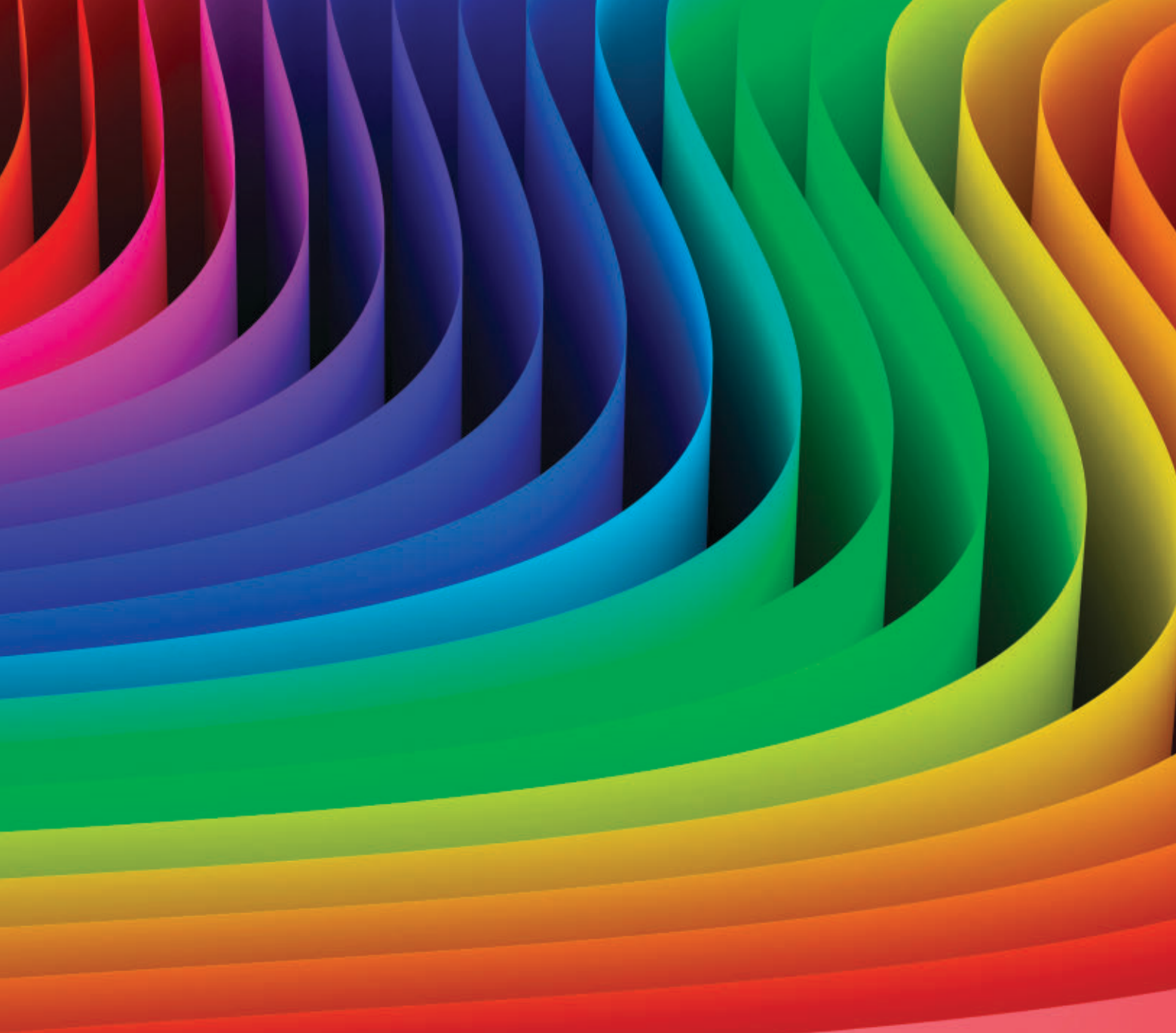
Activity-Centric Computing Systems

MOBILE, UBIQUITOUS, SOCIAL, and cloud computing have brought an unprecedented amount of information, digitized resources, and computational power—spanning many different devices—to users today. Correspondingly, an increasing amount of work and leisure activity is taking place in this distributed digital computing environment. For example, in a hospital, the medical record and bio-signals of patients are digitized and accessed by multiple stationary, mobile, and wearable devices. At home, digital and social media, email, photo libraries, and the like are accessed on a wide range of devices including laptops, smartphones, TV sets, and other Internet-connected appliances. However, this rapid increase in the diversity and volume of both computational devices and digital content quickly

» key insights

- **Activity-Centric Computing (ACC)** addresses deep-rooted information management problems in traditional application-centric computing by providing a unifying computational model for human goal-oriented ‘activity,’ cutting across system boundaries.
- A historical review of the motivation for and development of ACC systems is explored, highlighting the need for broadening this research topic to also include low-level system research and development.
- ACC concepts and technology relate to many facets of computing: they are relevant for researchers working on new computing models and operating systems, as well as for application designers seeking to incorporate these technologies in domain-specific applications.





introduces corresponding organizational challenges, leading to digital clutter. Many people feel overwhelmed and burdened by organizing and retrieving their digital resources, which includes handling, organizing, and finding information—a problem commonly referred to as information overload. Moreover, handling multiple and often concurrent tasks while coordinating with other individuals adds an additional level of complexity.

Despite the overwhelming success of new devices and cloud-based information-sharing infrastructures, the evolution of the user interface models that people use to interact with these innovations and the representations with which they organize electronic in-

formation on these platforms has not kept pace. Although it is much more common today for users to access information through the browser or on a mobile device than in the past, most contemporary user interface models are still fundamentally grounded in the personal computer metaphor, as part of which electronic resources are defined by the applications used to manipulate them and “filed” using a desktop metaphor (files, folders, and application windows). This application- and document-centric model leads to a fragmentation of a person’s information. For example, information related to a specific work project is often scattered across multiple files, local folders, cloud folders, and across different

applications such as email, instant messaging, local and cloud-based document editors, Web browsers, and social media channels/communities. Moreover, this information might be scattered across different devices and accessed by multiple users.

While cloud-based technologies allow users to access and share files and documents online and access them across different devices—including cloud-dedicated devices like the ChromeBook—such technologies have overwhelmingly maintained use of the files-and-folders model for presenting resources and applications to users, even when new capabilities such as tag-based (for example, Gmail) and graph-backed (for example, Mi-

Conceptual Models for Activity-Centric Computing

The goal of ACC is to replicate the multifaceted and complex nature of human activities in the real world in a computational representation. ACC systems do not provide another application, service, or collaboration tool, but rather integrate existing tools across devices, people, services, and resources in a manner that reflects the real-world activity being done. The design challenge in ACC is to create activity representations that are simple, yet flexible enough to accommodate different levels of rigidity.^{6,14} To achieve this, different conceptual models for ACC systems have been proposed, of which the Activity-Based Computing and the Unified Activity Management models are the most elaborate ones.

Activity-Based Computing (ABC). In ABC, a ‘computational activity’ (or just ‘activity’) is a computerized representation of a real-world human activity.² The purpose of the computational activity is to reflect the human activity and to provide access to resources relevant to its execution. The ABC approach was developed to support hospital work and can be used to model the work done as part of treating patients. As illustrated in Figure 2 (left) a computational activity aggregates and links services, resources, documents, and users that are relevant to the real-world activity of treating Mrs. Pedersen for leukemia. Among other things it gives access to the patient’s medical records, medicine charts, and medical images. Access to these materials is mediated by the respective computer systems involved: the electronic patient record system (EPR); the electronic medication system (EMS); and the picture, archiving, and communication system (PACS). Hence, ABC extends computational support ‘upwards’ from the level of application and document to the level of the overall activity. In the ABC model, this is called ‘Activity-Centered Resource Aggregation,’ which is the first of six core design principles:

Activity-centered resource aggregation. Aggregation of relevant resources, services, applications, documents, data, and users in a one logical bundle. This principle supports information and task management.

Activity suspension and resumption. Suspending an activity means its state is stored and removed from the active workspace, while resuming an activity restores it. This principle supports multitasking and interruptions in work.

Activity roaming. Activities are stored in an infrastructure and hence can be accessed from multiple devices. This allows suspending an activity on one device and resuming it on another device. This principle supports mobility across multiple devices.

Activity adaptation. When an activity roams (migrates) from one device to another, it adapts to the runtime and resources available on the local device. This principle supports mobile code execution, which can take advantage of technical resources like processing power, memory, network, and display size.

Activity sharing. Activities are per default shared and can be accessed, used, and modified by all users who are ‘participants’ of the activity. This principle supports collaboration, including access control.

Activity awareness. Computational activities are always representations of real-world activities and these representations need to build and maintain an ‘awareness’ of—that is, knowledge about—this real-world context. This principles support context-aware adaptation to the users’ (work) context.

Unified Activity Management (UAM). Developed by IBM Research, UAM specifies a semantic model as a unified model for integrating formal business processes with the informal collaboration needed to accomplish business objectives.¹⁷ In UAM, an ‘activity description’ articulates the actors (people) and roles involved, the resources used (tools, artifacts, people), the results produced, the events it is bounded by, and its relationships to other activities (such as sub-activities or dependent activities). All the people involved can see the activity descriptions and they can modify and extend the descriptions. An ‘activity’ is metadata, that is, the glue tying together system resources around the generic semantics of an activity. In a reference implementation of UAM, activity descriptions were implemented in semantic web technology using RDF and OWL. Figure 2 (right) summarizes UAM in which activity representations are managed in an RDF-based Activity Metadata Repository that integrates information from various external services like email, and calendars.

and computers. In this article, we review a specific approach in which ‘activities’ become a new computational abstraction around which interaction occurs. An activity is an ongoing effort in a person’s life toward a goal. For example, an activity can be a work project, writing a research paper, implementing a feature in software, designing a new product, planning an event, treating a patient, or preparing for a vacation. Activities reflect goals that people want or need to achieve in the real world, and a real-world ‘activity’ can be represented in the computer as an abstraction of computational data, resources, tools, applications, services, and so on, which are needed in order for users to perform this activity.

There are different approaches to realizing *Activity-Centric Computing* (ACC) systems. (See the sidebar for two conceptual models proposed for ACC). However, a common theme in these approaches is to mitigate information fragmentation and overload by integrating resources (for example, information), services (for example, applications), devices, and users into an activity ‘bundle’ that ties these four layers together. A representation of computational activities is illustrated in Figure 1. For example, in a software development project, a debugging activity could encapsulate: a number of source code files, unit tests, and test documentation [resources], a source code editor, a debugger, a terminal window, and a bug reporting system [services], a twin-display debugging setup and several different smartphone configurations for testing [devices], and the tester with the two developers who are working on this feature [users].

The idea of ACC is not novel. In fact, many researchers who studied the original personal computer model argued early on that computing systems should provide high-level support for activities. In 1983, Liam Bannon and colleagues observed that “[c]urrent human-computer interfaces provide little support for the kind of problems users encounter when attempting to accomplish several different tasks in a single session.”¹ They proposed moving away from computing environments built around applications and files as first-class computational constructs, focusing rather on the higher-level activities that people perform on computers.

Microsoft 365) information management schemes are emerging. Moreover, the introduction of cloud-based services have for most users just added yet another set of services, applications, and accounts for them to handle, and has in practice added yet another layer of information fragmentation and overload to the picture. Thus, even though

touch-based phones and tablets look and feel different, the personal computing model, with its focus on applications (or apps), is still in many cases the dominant means for working with electronic information.

Researchers have argued these problems call for a fundamentally new abstraction for interaction between people

Since then, a great deal of research has been done on ACC technologies, ranging from research on user interface management technologies to more fundamental distributed middleware and operating system components to support ACC systems. In order to provide a historical and comprehensive overview of the state-of-the-art in ACC research, this article presents a systematic review of the research literature on ACC systems and technologies and provides an outlook of their potential and the main implementation challenges in applying these approaches to contemporary and emerging computing environments.

A Review of ACC Systems

A three-step procedure was followed to identify a comprehensive corpus of ACC research papers from the computing literature, which were further processed for data extraction. First, the authors—all of whom have contributed substantively to the ACC research domain—identified an initial set of publications (N=38) that we agree accurately represents core ACC research. Second, we applied a backward snowballing technique, adding all articles cited by all of the papers in our initial set (N=984). Third, after pruning out all duplicates, we screened all retrievable publications identified in the second step to focus on only those publications presenting “technologies with a design motivated by the idea of supporting computational activities” as described earlier. This process yielded to the selection of 101 primary studies and the identification of 68 unique technologies.^a Over 58% of these papers refer to what we call activity as ‘activity,’ whereas 35.6% refers to ‘tasks,’ and the remainder use other terms, such as ‘project.’

The following coding schemes emerged during the data extraction process. Each primary study was labeled with tags related to ‘motivation’ and ‘system type.’ Motivation was extracted by analyzing what kind of challenge(s) each paper stated that it was addressing, whereas system type was extracted by analyzing the technological contribution(s). Disagreements and

ambiguities in coding were resolved in meetings involving all authors.

Figure 3 shows the coding schemes and distribution of both motivation and system type contributions. Note that each paper may be labeled with multiple tags. For example, the article “Activity-based computing for medical work in hospitals”² presents an application, a middleware infrastructure, and a smartspace system.

Figure 4 shows a historical distribution of identified design motivations over time. From this overview, we can identify three ACC waves in the literature: an initial wave in the 1980s, motivated by the Bannon et al. paper; a second wave in the 2000s; and a recent third wave beginning in 2012 and continuing today. Note that the decline shown beginning in 2015 is a methodological issue; since this review was completed in 2017 and is built from referenced papers, the collection of papers is by nature backward looking and historical.

In terms of motivation for incorporating support for computational activities, we can identify three broad areas:

Green—motivated by the belief that activities are a better representation of

how humans think and/or to provide support for task switching, improved resource management, and automating the overhead of task management.

Blue—motivated to provide support for collaboration, mobility, process optimization, and awareness (of the workspace, task, people, and resources).

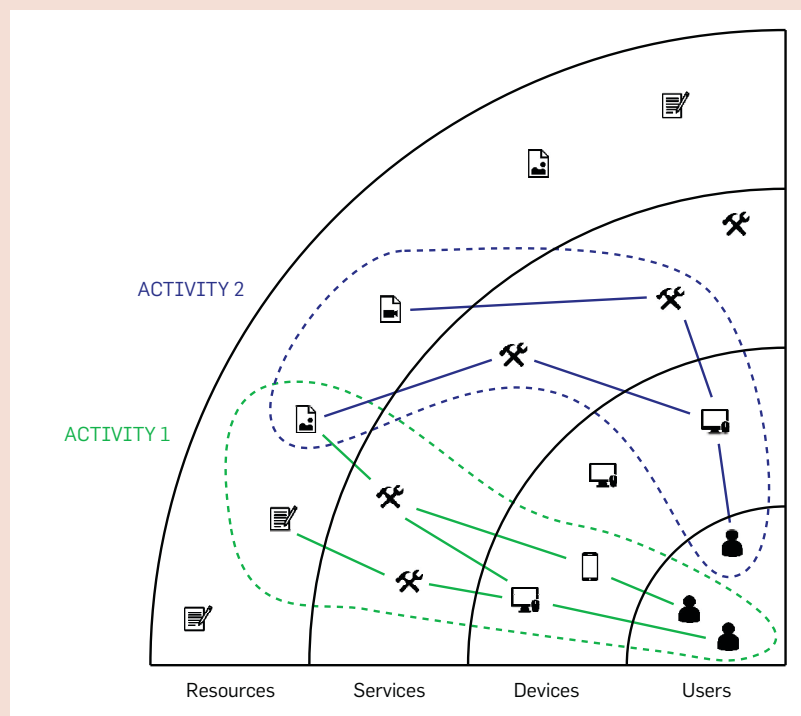
Red—motivated to address information fragmentation, including information fragmented across devices.

Based on the data, we can derive that a significant part of ACC research has addressed multitasking (34%), resource management (60%), and collaboration (39%), especially during the first and second wave. On the other hand, little research at this stage focused on resource management automation (8%), workflow automation (2%), or awareness (6%). During the second wave, support for collaboration, mobility and awareness (blue) was given increased focus, and in the second and third waves, research was increasingly motivated by the challenges of information and device fragmentation (red).

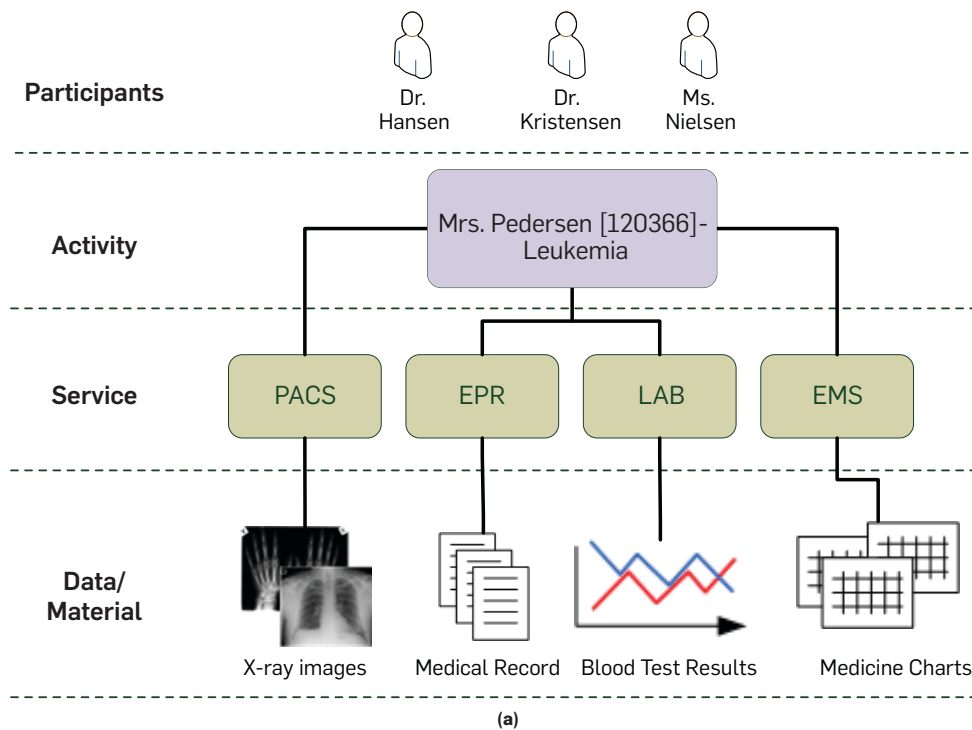
In terms of system types, we can also identify three broad areas:

Magenta—end-user oriented applications and user interface technology.

Figure 1. The four layers in computing considered during the design of ACC systems.



^a For the full list, see <http://dl.acm.org/citation.cfm?doid=3325901&picked=formats>

Figure 2. Left: The Activity-Based Computing model. Right: The Unified Activity Management semantic model.

Yellow—middleware, file management, and distributed system support.

Cyan—low-level operating system support, processes, and I/O.

From the figures, we can see the majority of papers have focused on end-user applications (45%), user interface management systems (UIMS) (47%) (magenta), and middleware technologies (yellow)—especially file management (42%) and middleware frameworks (38%). Less focus has been directed toward more low-level issues (cyan) like operating systems (6%), processes (3%), and I/O (2%).

From the review, we can identify a set of common topics and technologies, which we unpack as examples of core ACC research contributions.

Multitasking

Many (34%) ACC systems were motivated by providing support for multitasking, which also represents some of the earliest research in this space. These systems enabled multitasking by supporting suspension of the current activity and resumption of another. This focus recalls the original study by Bannon et al., who argued that a “work-space system should support diges-

sion while providing [...] easy return to previous activities.”¹ This implies that people can pause their work on one activity and simply save the entire state of the activity including the configuration of applications, files, windows, and other resources. Afterward, they can easily switch to another activity, thus, loading the configuration of files, documents, applications, and collaborative tools associated with that activity.

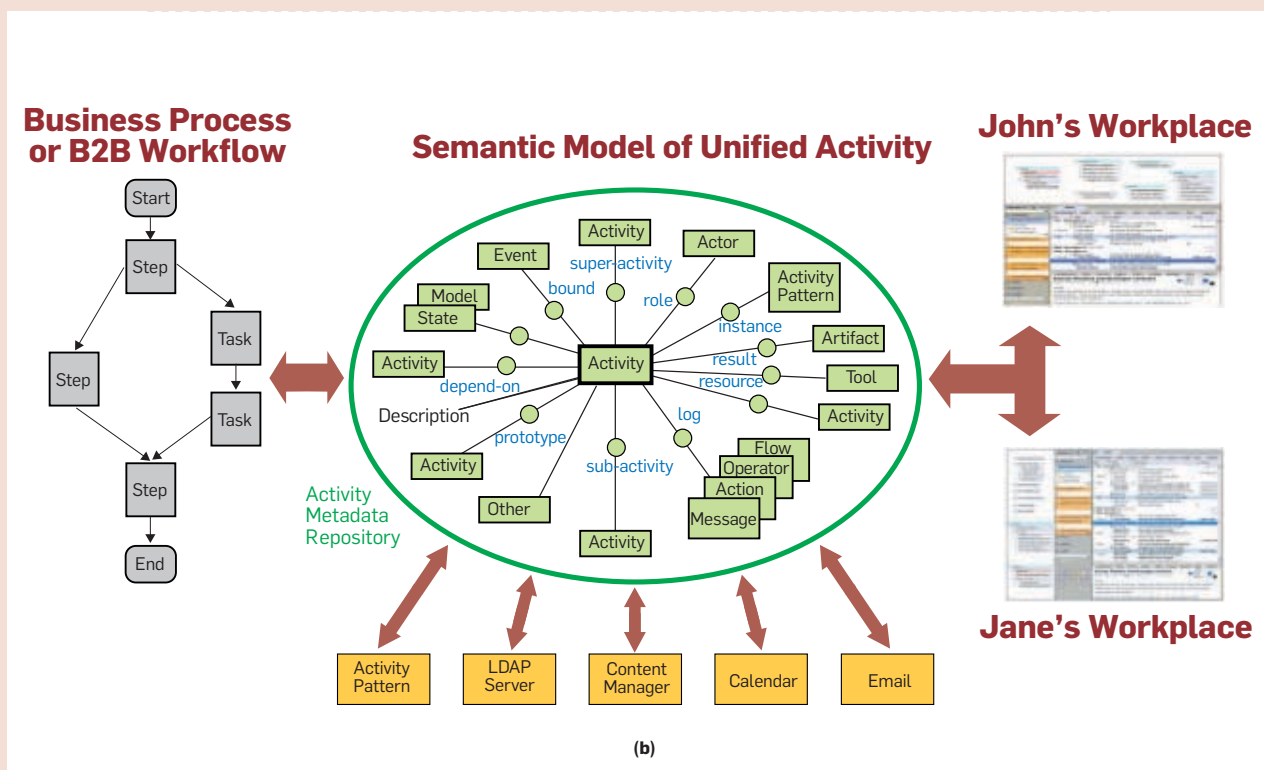
One of the first ACC technologies was the ‘Rooms’ system presented by Xerox PARC in 1987,⁷ which was directly motivated by the Bannon et al. study. Even though this study was based on observations of users interacting with a command-line interface (Unix), similar problems of limited support for multitasking were also observed in the graphical user interfaces developed at Xerox PARC. In Rooms, separate windows associated with the same task could be collated into distinct ‘rooms,’ and users could switch between these rooms in order to switch tasks. In many ways, Rooms was the predecessor of the ‘virtual desktop’ systems we know today. Kimura is a more recent example of an ACC system focusing on supporting multitasking by augmenting the

entire office environment.¹⁵ In contrast to Rooms, which limits interaction to the desktop monitor, Kimura leveraged interactive peripheral displays on the walls of the office, allowing users to switch between activities while maintaining a peripheral awareness of other activities in the background.

Window Management in Desktop Interfaces

As the user interface in all contemporary OSs (macOS, Windows, Linux) materialized around the desktop metaphor using overlapping windows, icons, menus, and a mouse pointer (also known as the WIMP paradigm), it was evident this model provided limited explicit support for human activity, including multitasking. Therefore, many (47%) ACC systems have provided models that integrate support for activities into the user interface.

For example, the ActivityBar³ (Figure 5) suggests replacing the Windows XP Taskbar with an ActivityBar that gives direct access to switching among activities. Each activity groups multiple application windows with associated resources, such as documents, spreadsheets, Web pages, among others.



This approach was later extended to also support sharing and collaborative awareness in the ‘co-Activity Manager’ system⁹ and the entire temporal activity lifecycle in Laevo.¹¹ Similarly, Microsoft Research (MSR) has proposed a number of ACC extensions to Windows, including the TaskGallery²⁰ and ScalableFabric¹⁹ window management systems, as well as ‘Colletta,’ which is an extension of the Windows UI that supports lightweight management of the user’s activities through tagging.¹⁸ On the MacOS, Giornata²² (Figure 6) provides support for multitasking through virtual desktop management, tagging of activities, lightweight file management using the desktop surface, and collaborative awareness of the co-workers relevant to each activity.

Automation of File and Resource Management

Managing multiple files and resources across multiple activities and multiple applications has proven to be a significant challenge in all OSs. For example, keeping track of files related to a specific customer case across folders, email, applications, and cloud-based services is inherently cumbersome. Activity-

centric resource and file management technologies have been proposed to address these problems and—as is evident from the magnitude of the corresponding columns in Figure 3—have been central themes in ACC research.

However, even the act of managing computational activity representations incurs some overhead. One approach that has been proposed for further minimizing this cost (but that has been relatively lightly explored, according to our review; see also Figure 3) is in augmenting activity representations with *automation* to automatically handle some of this organizational work on the user’s behalf.

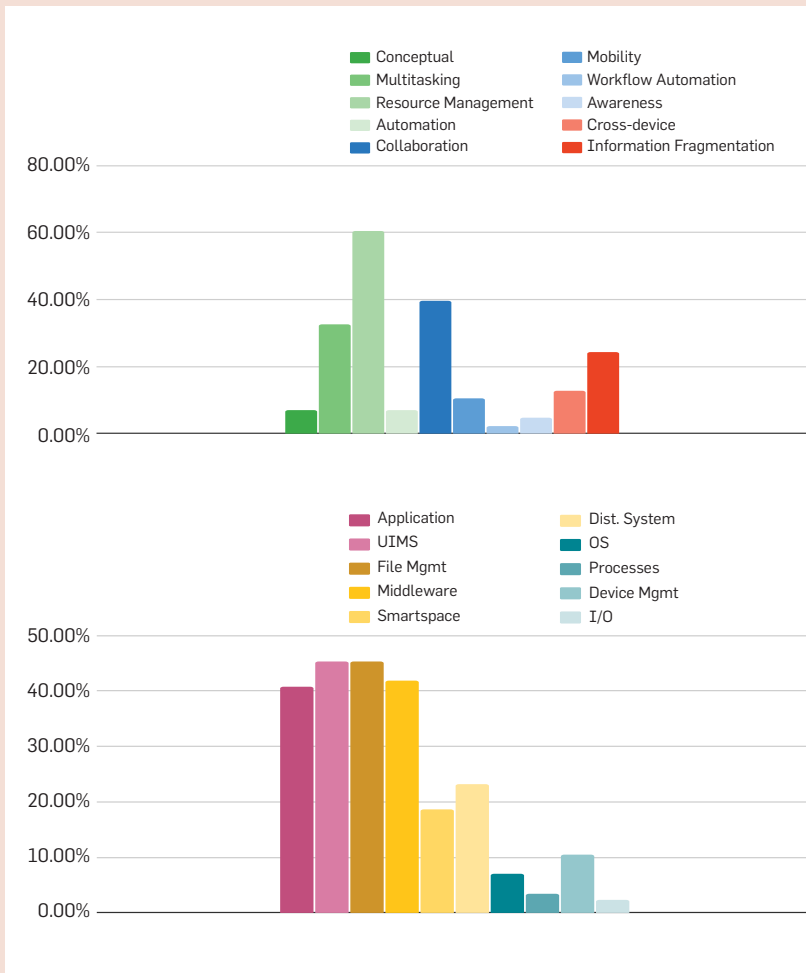
For example, by logging interactions with applications used in knowledge work (for example, email, word processing, spreadsheets, and Internet browsers), both the UMEA¹² and Task-Tracer⁵ system automatically organize resources (for example, documents, folders, URLs, and contacts) into computational activities. This classification is used in, for example, file management interfaces where an open file dialog box opens by default in a folder associated with the current activity, and quick access is provided to files

most likely needed as part of ongoing work. Similarly, Mylar¹³ uses a degree-of-interest model to capture activity contexts in an integrated development environment (IDE) by monitoring the interactions of a programmer with source code. These activity contexts are managed in a ‘task list’ view, which can be used to filter the IDE to only show those elements relevant to the selected task (for example, implementing a feature or working on a bug fix).

Collaboration and Awareness

Collaboration is core to human activity and a number of ACC systems (39%) have targeted support for collaborative activity. Activity sharing aims to enable people to work on the same digital activity representations and its resources, without the need for using any external or third-party collaboration tool, application, or system. Instead, support for collaboration support is simply built into ACC system support. Additionally, because collaboration and cooperation practices differ across individuals and teams and can change over time, ACC systems have supported different collaboration styles, ranging from full real-time synchronous coopera-

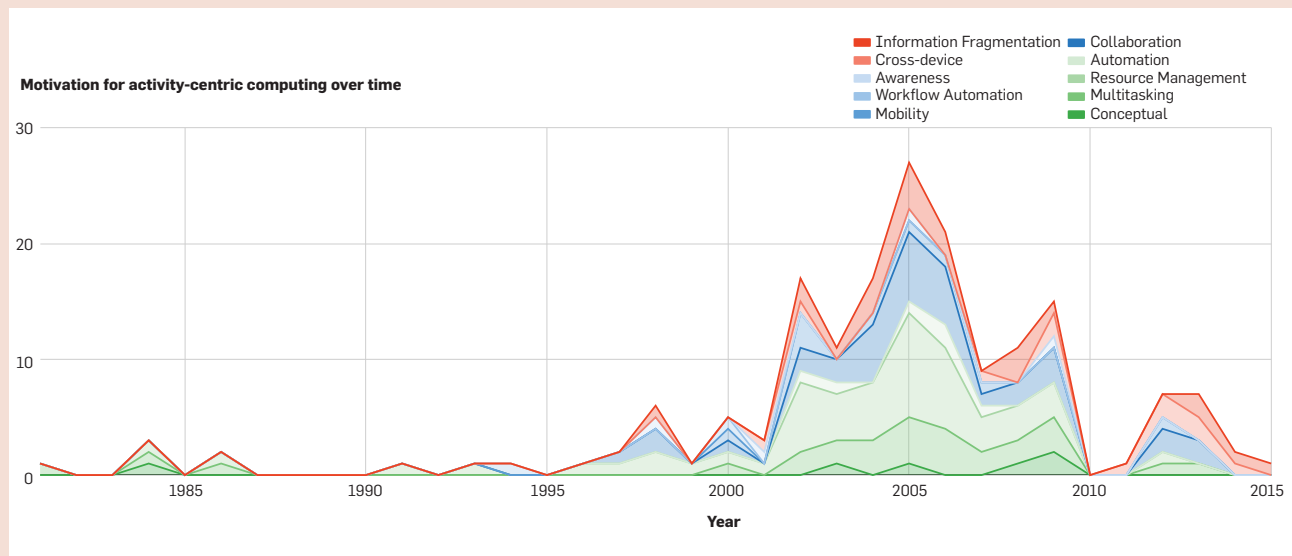
Figure 3. The coding schemes and distribution of 'stated motivation' and 'system type contribution' for all 101 ACC papers.



tion within an activity to simpler ways of packaging and sending an activity to other users to support asynchronous collaboration. ACC collaboration mechanisms have also experimented with providing a flexible way for people to define access rights, roles, and the shared context for each activity they are using. These collaborative features have also been used to define and enforce complex organizational work, facilitating the kinds of coordination offered by other workflow-based collaborative systems.

For example, the Activity Explorer⁶ and the Unified Activity Management¹⁶ systems developed by IBM Research support the notion of 'activity-centric collaboration,' which aims to support collaboration via activity models, defined as a logical unit of work that incorporates all the tools, people, and resources needed to get a job done. In contrast to prior personal information management systems, the IBM approach had an explicit focus on supporting collaboration by suggesting a unified activity model for *business processes* across people and organizational boundaries. Activity-centric support for collaboration was implemented as part of the IBM Lotus Workplace groupware system. In a hospital domain, the Activity-Based Computing (ABC) system provided support for the extensive collaboration related to medical treatment of

Figure 4. Distribution of motivation for ACC papers in a historical outline.



hospitalized patients.² The ABC system demonstrated the role of activities in fostering both co-located and remote collaboration, and supported scenarios ranging from a co-located team meeting between doctors and nurses to remote video conferencing between, for example, a radiologist in the radiology department and a physician during a ward round.

Interactive Surfaces and Cross-Device Interaction

Recently, we have witnessed an explosion in the variety and popularity of mobile and ubiquitous computing devices such as smartphones, tablets, whiteboards, tabletops, and game consoles. Traditional cross-device interaction has been accomplished through sending files or documents from one device to another, using available on-device tools to show or use the document. However, this ‘basic’ cross-device operation does not support moving a complex work context from one device to another seamlessly. In the third wave of ACC research, researchers have thus proposed that ACC can help manage this complexity by using the notion of a device-agnostic activity to bundle together resources accessible across multiple devices and facilitate configuration of these device ecosystems to suit the needs of specific real-world activities.

The ReticularSpaces system⁴ (Figure 8) suggests a uniform user interface across multiple interactive surfaces (tablet, wall, tabletop) that allows users to access and collaborate on shared resources, organized into activities. For example, during a software development stand-up meeting, all requirement documentation, software architecture descriptions, and source code for a particular feature under review are available across all devices. Similarly, the ActivitySpace⁸ system allows users to synchronize files across tablets, smartphones, and desktop devices by using the notion of an activity as the means for switching among different collections of content. Finally, the electronic laboratory bench (eLabBench)²¹ (Figure 7) provides an example of how resources for a biology experiment can be bundled together and made accessible on an interactive lab bench during experimental work inside the lab.

Outlook and Future Challenges

Research on Activity-Centric Computing has been ongoing since the early 1980s and has achieved much in terms of demonstrating how support for multitasking, mobility, collaboration, and cross-device interaction can be incorporated into computing platforms as well as end-user applications across different domains. Based on a thorough review of 101 papers, we found that ACC has proposed conceptual and technological models to better support window management, file management, workflow management, distributed systems, interactive smart space technology, and cross-device/ubiquitous computing. As such, ACC as a research theme cuts across several computer science disciplines and offers a potentially valuable series of

approaches for addressing the contemporary and significant problems of information fragmentation and information overload.

However, our review also revealed a set of limitations to ACC. First, most research has focused on end-user applications (45%) and user interface management (47%), and less on more basic technologies like how to incorporate ACC into operating systems, file management, distributed computing, and networking technologies. At the same time, the development of ACC applications presented in the research literature has been cumbersome exactly due to this lack of underlying technological support. Thus, more basic research on the lower-level technological components supporting ACC is needed—especially investigations of how support

Figure 5. ActivityBar for Windows XP.

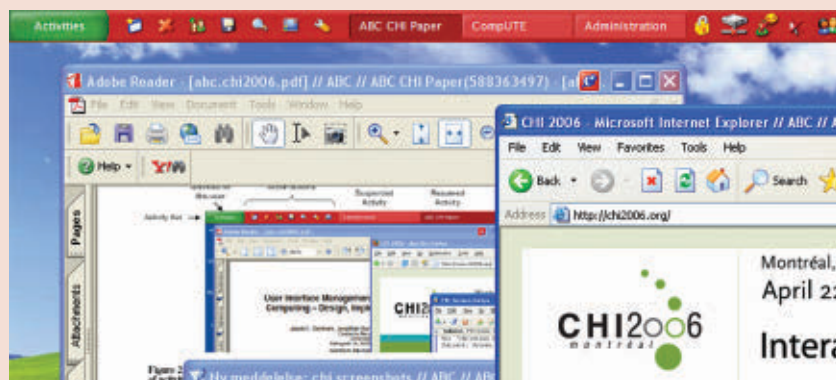
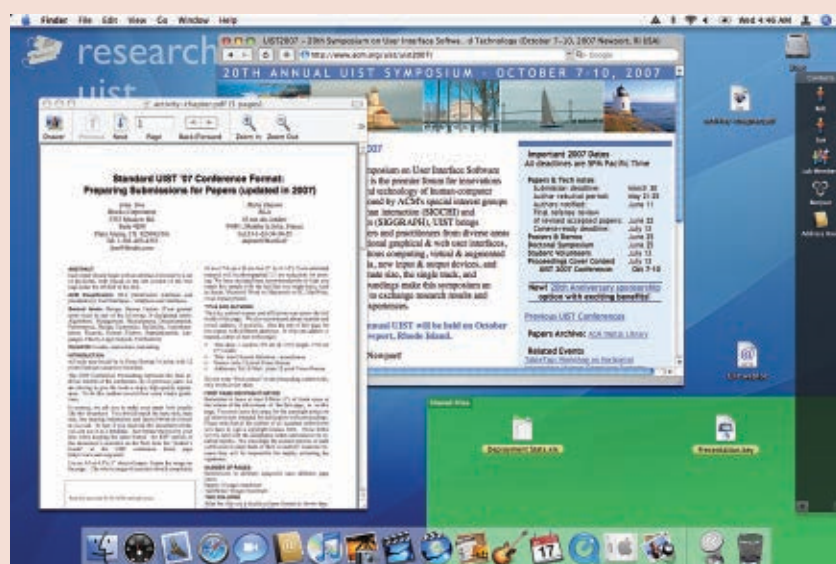


Figure 6. Giornata for MacOS.



for ACC can be incorporated into or exposed by mainstream operating systems. As an example, one of the most pervasive examples of this kind of missing support from our literature review relates to the need for ACC systems to support suspension of the current activity and resumption of another. Because activity models are stateful, each activity must maintain state information in a persistent way, allowing the state of that activity to be saved (during suspension) and restored (during resumption) at a later time. Enabling a full-stack stateful activity management system has proven to be one of the major challenges in ACC since this requires access to detailed runtime state information spanning the entire computer stack; from end-user applications to the win-

dow manager's layout and on down to the underlying file, networking, and process-level state—information that is not readily available in contemporary operating systems (like Windows and macOS) nor from most applications.

Second, from a conceptual point of view, a notable barrier to the adoption of ACC technologies is the fact that ACC systems require either end users or ACC systems to manage the computational representations of activities—work that is “invisibly” delegated to the end users in current, application-centered computing environments. The manual management of activities—that is, the manual creation of a computational activity and organization of its associated resources (such as files and users)—introduces an extra overhead to informa-

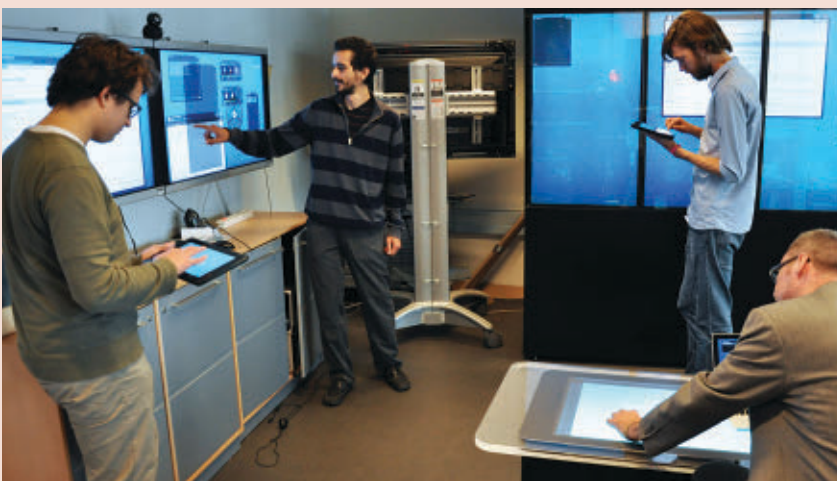
tion management, similar to managing files in a hierarchical folder structure. However, these activity representations may more closely replicate the multifaceted clusters of digital resources, services, and users that map to an individual's discrete real-world tasks, making this organizational work easier, more meaningful, or more memorable¹⁴ than are our current, fragmentation-prone interfaces. An alternative approach is to liberate users from this pattern recognition to automatically organize computing resources into indexed activities. This solution also comes with a cost, however; users must give up some degree of control in the definition of their digital activities, which might lead to mismatches between computational and cognitive representations of activities. These systems might also be semi-automatic, providing specific options or possibilities, without being fully prescriptive. For example, the physical location of a user and their device(s) could be leveraged to filter possible activities or to only show activities that were previously used at that physical location. Striking the right balance among these approaches in future ACC systems will be essential to encourage adoption.

Third, despite the fact that most major computer science companies (for example, IBM, Microsoft, Apple, and Google) have contributed to research on ACC or experimented with ACC research systems, we have still seen a relatively limited impact of this research on the software architecture of shipping consumer platforms; that is, resource organization at the level of the operating system or task management at the level of the window manager. Even with nearly 30 years of research and development into the benefits of ACC approaches, the application- and document-centered interaction paradigm continues to reign. We have found a few notable examples of success stories: IBM has incorporated computational activity representations into its Lotus Connections suite of enterprise collaboration tools; KDE's Plasma desktop environment uses multifaceted activity representations to enhance a typical virtual desktop-driven computing workspace; and Mylar¹³ has been introduced as Mylyn2 in the popular Eclipse IDE as a task-focused interface for programmers. However, for each of these success stories, there are similar

Figure 7. The electronic laboratory workbench (eLabBench²¹).



Figure 8. ReticularSpaces: Collocated activity sharing across multiple devices in a smart space environment.⁴



examples of systems that did not make it into the mainstream—for example, Apple's application-agnostic OpenDoc platform and Microsoft's proposed (and cancelled) WinFS relational file system. This underlines the challenges involved in opening up and redesigning the underlying computational architecture to more completely support ACC systems. And it emphasizes the importance of clearly articulating the benefits that end users stand to gain by investing time and effort to learn and adopt activity-centered interaction paradigms.

Looking forward, this review has helped us to enumerate exciting open research areas within ACC. Given the explosion in the number of devices and their heterogeneity and interconnectivity, ACC is a strong candidate for a computing paradigm that can help address these complex challenges in service of a more coherent user experience. Limited research has been conducted on cross-device management (9%) and smartspace technology (17%) in the ACC domain, and here there is still much work to be done.

Currently, a major shift toward cloud-based computing is taking place and all major software companies are investing in infrastructures for cloud-based computing. As we have argued and demonstrated earlier,¹⁰ cloud-based technologies provides an excellent platform for ACC; it provides the ability to share, distribute, and synchronize heterogeneous resources in real-time across multiple users, devices, and locations. However, as mentioned at the outset, the current state-of-art of cloud-based computing is to provide services similar to local resources like CPU power, files, and kind of manual custodial work by relying on content extraction and applications. If these were aggregated into cloud-based activities, a solid foundation for enabling ACC would be available. Recently, Microsoft announced its Microsoft 365 environment, which supports resource aggregation, suspend/resume, and cross-device coordination via constructs called 'Sets' and 'Graphs,' all of which seems promising building blocks for supporting ACC. In general, we would argue that higher-level support, such as ACC, would be central to the success of a scalable user experience in future development of cloud-based computing.

Furthermore, applying ACC principles, concepts, and technologies to the development of end-user applications in industry is potentially beneficial for many different domains. The research literature reviewed here points out a few areas that have been well-explored to date—information work, medical work, and software development—but many other domains would likely benefit from having direct computational support for domain-specific activities.

In summary, going forward, ACC still presents a variety of important and challenging research topics for researchers and practitioners in many different computing fields—from basic infrastructure to end-user interfaces and applications—to address. **□**

References

- Bannon, L., Cypher, A., Greenspan, S. and Monty, M.L. Evaluation and analysis of users' activity organization. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM (1983) 54–57.
- Bardram, J.E. Activity-based computing for medical work in hospitals. *ACM Trans. Comput.-Hum. Interact.* 16, 2 (June 2009), 10:1–10:36; <https://doi.org/10.1145/1534.903.1534907>
- Bardram, J., Bunde-Pedersen, J. and Soegaard, M. Support for activity-based computing in a personal computing operating system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM (2006), 211–220; <https://doi.org/10.1145/1124.772.1124805>
- Bardram, J., Gueddana, S., Houben, S. and Nielsen, S. Reticular spaces: Activity-based computing support for physically distributed and collaborative smart spaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM (2012) 2845–2854; <https://doi.org/10.1145/2207.676.2208689>
- Dragunov, A.N., Dietterich, T.G., Johnsrude, K., McLaughlin, M., Li, L. and Herlocker, J.L. TaskTracer: A desktop environment to support multi-tasking knowledge workers. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*. ACM (2005) 75–82; <https://doi.org/10.1145/1040.830.1040855>
- Geyer, W. et al. Activity Explorer: Activity-centric collaboration from research to product. *IBM Systems Journal* 45, 4 (2006), 713–738; <https://doi.org/10.1147/sj.454.0713>
- Henderson Jr, D.A. and Card, S. Rooms: The use of multiple virtual workspaces to reduce space contention in a Window-based graphical user interface. *ACM Trans. Graph.* 5, 3 (July 1986), 211–243; <https://doi.org/10.1145/24054.24056>
- Houben, S., Tell, P. and Bardram, J.E. 2014. ActivitySpace: Managing device ecologies in an activity-centric configuration space. In *Proceedings of the 9th ACM International Conference on Interactive Tabletops and Surfaces*. ACM (2014) 119–128; <https://doi.org/10.1145/2669.485.2669493>
- Houben, S., Bardram, J.E., Vermeulen, J., Luyten, K. and Coninx, K. Activity-centric support for ad hoc knowledge work: A case study of co-activity manager. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM (2013) 2263–2272; <https://doi.org/10.1145/2470.654.2481312>
- Houben, S., Nielsen, S., Esbensen, M. and Bardram, J.E. Noosphere: An activity-centric infrastructure for distributed interaction. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*. ACM, 2013, 13:1–13:10; <https://doi.org/10.1145/2541.831.2541856>
- Jeuris, S., Houben, S. and Bardram, J. Laevo: A temporal desktop interface for integrated knowledge work. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. ACM (2014) 679–688; <https://doi.org/10.1145/2642.918.2647391>
- Kaptein, V. UMEA: Translating interaction histories into project contexts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM (2003) 353–360; <https://doi.org/10.1145/642.611.642673>
- Kersten, M. and Murphy, G.C. Using task context to improve programmer productivity. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM (2006), 1–11; <https://doi.org/10.1145/1181.775.1181777>
- Kidd, A. The marks are on the knowledge worker. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM (1994) 186–191; <https://doi.org/10.1145/191.666.191740>
- MacIntyre, B., Mynatt, E.D., Volda, S., Hansen, K.M., Tullio, J. and Corso, G.M. Support for multitasking and background awareness using interactive peripheral displays. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*. ACM (2001), 41–50; <https://doi.org/10.1145/502.348.502355>
- Moody, P., Gruen, D., Muller, J., Tang, J. and Moran, T.P. 2006. Business activity patterns: A new model for collaborative business applications. *IBM Systems Journal* 45, 4 (2006), 683–694; <https://doi.org/10.1147/sj.454.0683>
- Moran, T.P., Cozzi, A. and Farrell, S.P. Unified activity management: Supporting people in e-business. *Commun. ACM* 48, 12 (Dec. 2005), 67–70; <https://doi.org/10.1145/1101.779.1101811>
- Oleksik, G. et al. Lightweight tagging expands information and activity management practices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM (2009) 279–288; <https://doi.org/10.1145/1518.701.1518746>
- Robertson, G. et al. Scalable fabric: Flexible task management. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM (2004) 85–89; <https://doi.org/10.1145/989.863.989874>
- Robertson, G. et al. The task gallery: A 3D Window manager. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM (2000) 494–501; <https://doi.org/10.1145/332.040.332482>
- Tabard, A., Hincapié–Ramos, J.D., Esbensen, M. and Bardram, J.E. The eLabBench: An interactive tabletop system for the biology laboratory. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ACM (2011) 202–211; <https://doi.org/10.1145/2076.354.2076391>
- Volda, S., Mynatt, E.D. and Edwards, W.K. Reframing the desktop interface around the activities of knowledge work. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*. ACM (2008) 211–220; <https://doi.org/10.1145/1449.715.1449751>

Jakob E. Bardram (jakba@dtu.dk) is a professor at the Technical University of Denmark, Lyngby, and director of the Copenhagen Center for Health Technology.

Steven Jeuris (sjeu@dtu.dk) is a postdoc at the Technical University of Denmark, Lyngby.

Paolo Tell (pate@itu.dk) is an assistant professor of software engineering at the University of Copenhagen, Denmark.

Steven Houben (s.houben@lancaster.ac.uk) is an assistant professor in the School of Computing and Communication at Lancaster University, U.K.

Stephen Volda (svolda@colorado.edu) is an assistant professor of information science at the University of Colorado, Boulder, CO, USA.

Copyright held by authors/owners.
Publication rights licensed to ACM. \$15.00



Watch the authors discuss this work in the exclusive *Communications* video. <https://cacm.acm.org/videos/activity-centric-computing-systems>

Tracing the tangled web of unsolicited and undesired email and possible strategies for its demise.

BY EMILIO FERRARA

The History of Digital Spam

SPAM! THAT'S WHAT Lorrie Faith Cranor and Brian LaMacchia exclaimed in the title of a popular call-to-action article that appeared 20 years ago in *Communications*.¹⁰ And yet, despite the tremendous efforts of the research community over the last two decades to mitigate this problem, the sense of urgency remains unchanged, as emerging technologies have brought new dangerous forms of digital spam under the spotlight. Furthermore, when spam is carried out with the intent to deceive or influence at scale, it can

alter the very fabric of society and our behavior. In this article, I will briefly review the history of digital spam: starting from its quintessential incarnation, spam emails, to modern-days forms of spam affecting the Web and social media, the survey will close by depicting future risks associated with spam and abuse of new technologies, including artificial intelligence (AI), for example, digital humans. After providing a taxonomy of spam, and its most popular applications emerged throughout the last two decades, I will review technological and regulatory approaches proposed in the literature, and suggest some possible solutions to tackle this ubiquitous digital epidemic moving forward.

An omni-comprehensive, universally acknowledged definition of digital spam is hard to formalize. Laws and regulation attempted to define particular forms of spam, for example, email (see 2003's Controlling the Assault of Non-Solicited Pornography and Marketing Act.) However, nowadays, spam occurs in a variety of forms, and across different techno-social systems. Each domain may warrant a slight different definition that suits what spam is in that precise context: some features of spam in a domain, for example, volume in mass spam campaigns, may not apply to

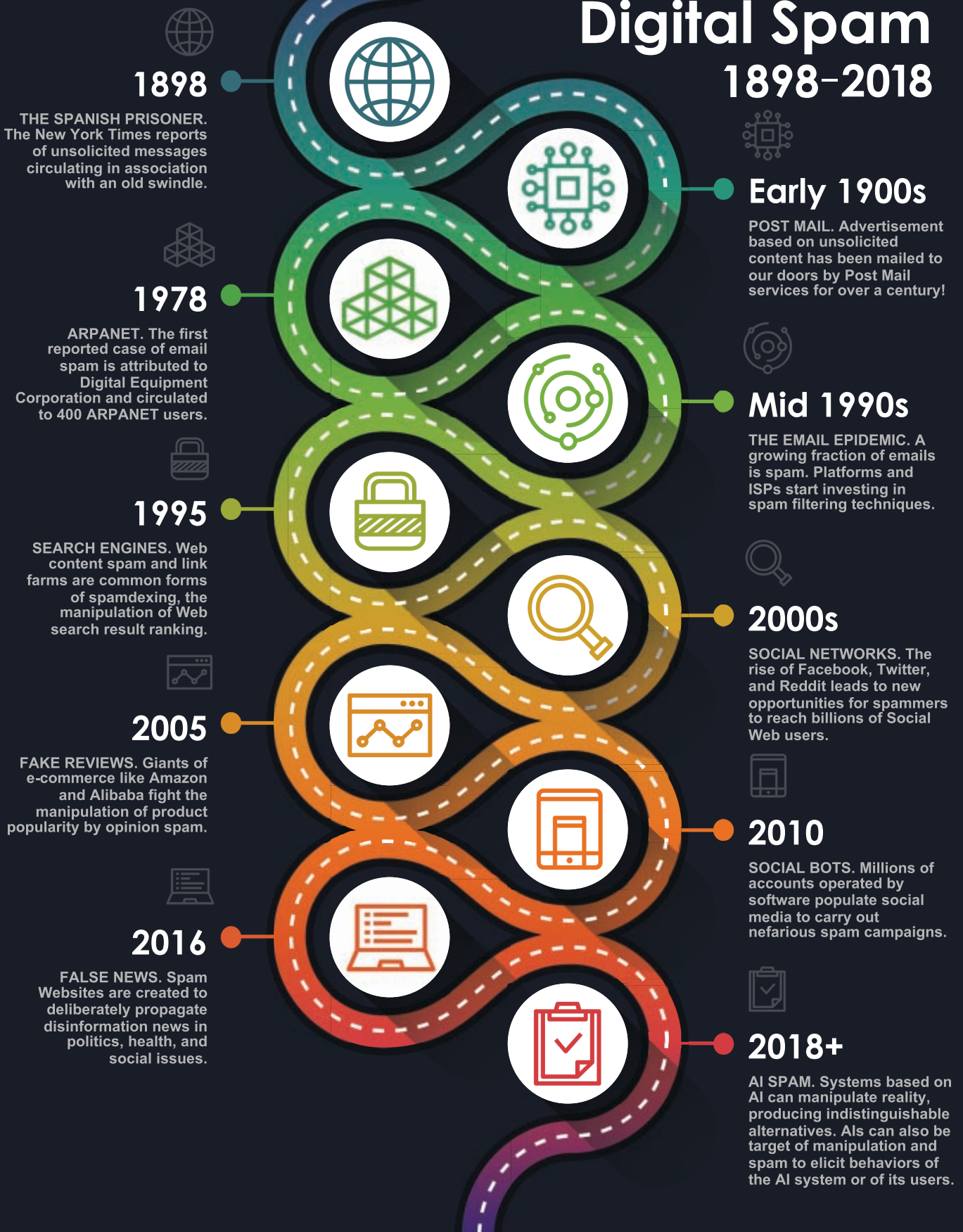
» key insights

- Throughout the Internet's history, digital spam has pervaded all techno-social platforms and it is constantly evolving. This article provides a taxonomy of digital spam, from its inception to current spam techniques.
- Since the email spam epidemic of the early 1990s, new forms of spam have emerged, including search engine spam, fake reviews, spam bots, and false news. In its latest incarnation, spam threatens to pollute AI systems making them biased and ultimately dangerous for our society.
- By illustrating some of the risks posed by digital spam in all its forms, including AI spam, we provide policy recommendations and technical insights to tackle old and new forms of spam.

Figure 1. Timeline of the major milestones in the history of spam, from its inception to modern days.

The History of Digital Spam

1898–2018



others, for example, carefully targeted phishing operations.

In an attempt to propose a general taxonomy, I here define digital spam as the attempt to abuse of, or manipulate, a techno-social system by producing and injecting unsolicited, and/or undesired content aimed at

steering the behavior of humans or the system itself, at the direct or indirect, immediate or long-term advantage of the spammer(s).

This broad definition will allow me to track, in an inclusive manner, the evolution of digital spam across its most popular applications, starting

from spam emails to modern-days spam. For each highlighted application domain, I will dive deep to understand the nuances of different digital spam strategies, including their intents and catalysts and, from a technical standpoint, how they are carried out and how they can be detected.

Wikipedia provides an extensive list of domains of application:

“While the most widely recognized form of spam is email spam, the term is applied to similar abuses in other media: instant messaging spam, Usenet newsgroup spam, Web search engine spam, spam in blogs, wiki spam, online classified ads spam, mobile phone messaging spam, Internet forum spam, junk fax transmissions, social spam, spam mobile apps, television advertising and file sharing spam.” (<https://en.wikipedia.org/wiki/Spamming>)

The accompanying table summarizes a few examples of types of spam and relative context, including whereas there exist machine learning solutions (ML) to each problem. Email is known to be historically the first example of digital spam (see Figure 1) and remains uncontested in scale and pervasiveness with billions of spam emails generated every day.¹⁰ In the late 1990s, spam landed on instant messaging (IM) platforms (SPIM) starting from AIM (AOL Instant Messenger) and evolving through modern-days IM systems such as WhatsApp, Facebook Messenger, and WeChat. A widespread form of spam that emerged in the same period was Web search engine manipulation: content spam and link farms allowed spammers to boost the position of a target Website in the search result rankings of popular search engines, by gaming algorithms like PageRank and the like. With the success of the social Web,²² in the early 2000s we witnessed the rise of many new forms of spam, including Wiki spam (injecting spam links into Wikipedia pages¹), opinion and review spam (promoting or smearing products by generating fake online reviews²⁷), and mobile messaging spam (SMS and text messages sent directly to mobile devices³). Ultimately, in the last decade, with the increasing pervasiveness of online social networks and the sig-

Detecting Spam Email

Email spam detection is an arms race between attackers (spammers) and defenders (service providers). Two decades of research in the data mining and machine learning communities produced troves of techniques to tackle this problem. Some milestones include:

SMTP solutions. SMTP is the protocol at the foundation of the email exchange infrastructure. Blacklists were introduced to keep track of spam propagators.⁷ Mail servers can consult blacklisting services to determine whether to route emails to their destination. A softer version of blacklisting is greylisting. Greylists keep track of triplets of IP addresses (sender, receiver, SMTP host) involved into an email exchange. The first time a triplet involving a dubious SMTP host appears, the exchange is denied, but the triplet is stored to authorize future exchanges. This is based on the rationale that spammers rarely retry sending spam through the same relay, and was proven effective in reducing early spam circulation.⁷

Another approach is keyword-based filtering: whenever the subject or the body of an email contains flagged terms (belonging to a keyword list), the SMTP service provider would not route it to its intended recipient, and flag the sending offender—multiple offenses would lead to permanent bans. Other strategies like DomainKeys Identified Mail (DKIM) and digital signatures are authentication methods designed to detect email spoofing and assess email provenance.

Supervised learning. In their seminal work, Drucker et al.¹³ proposed one of the first machine learning systems for spam detection, based on support vector machines (then the state of the art in terms of supervised learning). The success of supervised learning over traditional keyword-based filters demonstrated by Drucker et al.¹³ motivated the first wave of machine learning research in email spam detection. Shortly after, Androutsopoulos et al.⁴ showed the power of naive Bayesian anti-spam filtering: Bayesian systems yielded state-of-the-art spam detection performance for many years. The advent of more sophisticated learning models, like boosting trees, set the accuracy bar higher but paradigm shifts lagged for nearly a decade.

Hybrid neural systems. More recently, Wu³⁷ proposed behavior-based spam detection using combinations of simple association rules and neural networks. Given their ability to naturally handle visual information, neural network methods to detect spam were extended to multimedia content. For example, Wu et al.³⁸ and Fumera et al.¹⁷ proposed methods exploiting visual cues to detect spam content injected in images embedded into emails.

Dedicated hardware. Networking companies are developing anti-spam appliances. Dedicated hardware can detect various types of spam, including phishing, malware, and ransomware, guaranteeing high efficiency and accuracy. For example, Cisco advertises that their Email Security Appliance (ESA) detects over 99.9% of incoming spam email with lower than one in a million false positive rate.

Examples of types of spam and relative statistics.

Spam Type	Start	Today's Volume	ML	Ref
Email	1978	Billions x day	✓	10
Instant Messaging	1997	Millions x day	✓	20
Search Engine	1998	Unknown	✓	31
Wiki	2001	Thousands x day	—	1
Opinion and Reviews	2005	Millions across platforms	✓	11
Mobile Messaging	2007	Millions x day	✓	3
Social Bots	2010	Millions across platforms	✓	16
False News	2016	Thousands across websites	—	36
Multimedia	2018	Unknown	—	25

nificant advancements in AI, new forms of spam involve social bots (accounts operated by software to interact at scale with social Web users¹⁶), false news websites (to deliberately spread disinformation³⁶), and multi-media spam based on AI.²⁵

In the following, I will focus on three of these domains: email spam, Web spam (specifically, opinion spam and fake reviews), and social spam (with a focus on social bots). Furthermore, I will highlight the existence of a new form of spam that I will call AI spam. I will provide examples of spam in this new domain, and lay out the risks associated with it and possible mitigation strategies.

Flooded By Junk Email

The 1998 article by Cranor and LaMacchia¹⁰ in *Communications*, characterized the problem of junk email messages, or email spam, as one of the earliest forms of digital spam.

Email spam has mainly two purposes, namely advertising (for example, promoting products, services, or contents), and fraud (for example, attempting to perpetrate scams, or phishing). Neither ideas were particularly new or unique to the digital realm: advertisement based on unsolicited content delivered by traditional post mail (and, later, phone calls, including more recently the so-called “robo-calls”) has been around for nearly a century. As for scams, the first reports of the popular advance-fee scam (in modern days known as 419 scam, a.k.a. the Nigerian Prince scam), called the Spanish Prisoner scam were circulating in the late 1800s.^a

The first reported case of digital spam occurred in 1978 and was attributed to Digital Equipment Corporation, who announced their new computer system to over 400 subscribers of ARPANET, the precursor network of modern Internet (see Figure 1). The first mass email campaign occurred in 1994, known as the USENET green card lottery spam: the law firm of Canter & Siegel advertised their immigration-related legal services simultaneously to over 6,000 USENET newsgroups. This event con-

Email spam has mainly two purposes: advertising and fraud.

tributed to popularizing the term spam. Both the ARPANET and USENET cases brought serious consequences to their perpetrators as they were seen as egregious violations of common code of conduct in the early days of the Internet (for example, Canter & Siegel ran out of business and Canter was disbarred by the Arizona Bar Association.) However, things were bound to change as the Internet became an increasingly more pervasive technology in our society.

Email spam: Risks and challenges.

The use of the Internet for distributing unsolicited messages provides unparalleled scalability, and unprecedented reach, at a cost that is infinitesimal compared to what it would take to accomplish the same results via traditional means.¹⁰ These three conditions created the ideal conjecture of economical incentives that made email spam so pervasive.

In contrast to old-school post mail spam, digital email spam introduced a number of unique challenges:¹⁰ If left unfiltered, spam emails can easily outnumber legitimate ones, overwhelming the recipients and thus rendering the email experience from unpleasant to unusable; email spam often contains explicit content that can hurt the sensibility of the recipients—depending upon the sender/recipient country’s laws, perpetrating this form of spam could constitute a criminal offense;^b by embedding HTML or JavaScript code into spam emails, the spammers can emulate the look and feel of legitimate emails, tricking the recipients and eliciting unsuspecting behaviors, thus enacting scams or enabling phishing attacks;²³ finally, mass spam operations pose a burden on Internet service providers (ISPs), which have to process and route unnecessary, and often large, amounts of digital junk information to millions of recipients—for the larger spam campaigns, even more.

The Internet was originally designed by and for tech-savvy users: spammers quickly developed ways to take advantage of the unsophisticated ones. Phishing is the practice of using

a See *New York Times*, Mar. 20, 1898; <https://nyti.ms/2DD6oIn>

b For example, see the U.S. Federal Law on Obscenity; <https://bit.ly/2wfpDgt>

deception and social engineering strategies by which attackers manage to trick victims by disguising themselves as a trusted entity.^{9,23} The end goal of phishing attacks is duping the victims into revealing sensitive information for identity theft, or extorting funds via ransomware or credit card frauds. Email has been by far and large the most common vector of phishing attacks. In 2006, Indiana University carried out a study to quantify the effectiveness of phishing email messages.²³ The researchers demonstrated that a malicious attacker impersonating the university would have a 16% success rate in obtaining the users' credentials when the phishing email came from an unknown sender; however, success rate arose to 72% when the email came from an attacker impersonating a friend of the victim.

Fighting email spam. Over the course of the last two decades, solutions to the problem of email spam revolved around implementing new regulatory policies, increasingly sophisticated technical hurdles, and combinations of the two.¹⁰ Regarding the former, in the context of the U.S. or the European Union (EU), policies that regulate access to personal information (including email addresses), such as the EU's General Data Protection Regulation (GDPR) enacted in 2018, hinder the ability of bulk mailers based in EU countries to effectively carry out mass email spam operations without risks and possibly serious consequences. However, it has become increasingly more obvious that solutions based exclusively on regulatory affairs are ineffective: spam operations can move to countries with less restrictive Internet regulations. However, regulatory approaches in conjunction with technical solutions have brought significant progress in the fight against email spam.

From a technical standpoint, two decades of research advancements led to sophisticated techniques that strongly mitigate the amount of spam email ending up in the intended recipients' inboxes. A number of review papers have been published that surveyed data mining and machine learning approaches to detect and filter out email spam,⁷ some with a specific focus on scams and phishing spam.²¹



From a technical standpoint, two decades of research advancements led to sophisticated techniques that strongly mitigate the amount of spam email ending up in the intended recipients' inboxes.



In the sidebar “Detecting Spam Email,” I summarize some of the technical milestones accomplished in the quest to identify spam emails. Unfortunately, I suspect that much of the state-of-the-art research on spam detection lies behind close curtains, mainly for three reasons: First, large email-related service providers, such as Google (Gmail), Microsoft (Outlook, Hotmail), Cisco (IronPort, Email Security Appliance—ESA) devote(d) massive R&D investments to develop machine learning methods to automatically filter out spam in the platforms they operate (Google, Microsoft, among others) or protect (Cisco); the companies are thus often incentivized to use patented and close-sourced solutions to maintain their competitive advantage. Secondly, related to the former point, fighting email spam is a continuous arms-race: revealing one's spam filtering technology gives out information that can be exploited by the spammers to create more sophisticated campaigns that can effectively and systematically escape detection, thus calling for more secrecy. Finally, the accuracy of email spam detection systems deployed by these large service providers has been approaching nearly perfect detection: a diminishing return mechanism comes into play where additional efforts to further refine detection algorithms may not warrant the costs of developing increasingly more sophisticated techniques fueling complex spam detection systems; this makes established approaches even more valuable and trusted, thus motivating the secrecy of their functioning.

Web 2.0 or Spam 2.0?

The new millennium brought us the Social Web, or Web 2.0, a paradigm shift with an emphasis on user-generated content and on the participatory, interactive nature of the Web experience.²² From knowledge production (Wikipedia) to personalized news (social media) and social groups (online social networks), from blogs to image and video sharing sites, from collaborative tagging to social e-commerce, this wealth of new opportunities brought us as many

new forms of spam, commonly referred to as social spam.

Differently from spam emails, where spam can only be conveyed in one form (such as email), social spam can appear in multiple forms and modi operandi. Social spam can be in the form of textual content (for example, a secretly sponsored post on social media), or multimedia (for example, a manufactured photo on 4chan); social spam can aim at pointing users to unreliable resources, for example, URLs to unverified information or false news websites;³⁶ social spam can aim at altering the popularity of digital entities, for example, by manipulating user votes (upvotes on Reddit posts, retweets on Twitter), and even that of physical products, for example, by posting fake online reviews (say, for example, about a product on an e-commerce website).

Spammy opinions. In the early 2000s (see Figure 1), the growing popularity of e-commerce websites like Amazon and Alibaba motivated the emergence of opinion spam (a.k.a. review spam).^{24,27}

According to Liu,²⁷ there are three types of spam reviews: fake reviews, reviews about brands only, and non-reviews. The first type of spam, fake reviews, consists of posting untruthful, or deceptive reviews on online e-commerce platforms, in an attempt to manipulate the public perception (in a positive or negative manner) of specific products or services presented on the affected platform(s). Fake positive reviews can be used to enhance the popularity and positive perception of the product(s) or service(s) the spammer intends to promote, while fake negative reviews can contribute to smear the spammer's competitor(s) and their products/services. Opinion spam of the second type, reviews about brands only, pertains comments on the manufacturer/brand of a product but not on the product itself—albeit genuine, according to Liu²⁷ they are considered spam because they are not targeted at specific products and are often biased. Finally, spam reviews of the third type, non-reviews, are technically not opinion spam as they do not provide any opinion, they only contain generic, unrelated content

(for example, advertisement, or questions, rather than reviews, about a product). Fake reviews are, by far and large, the most common type of opinion spam, and the one that has received more attention in the research community.²⁷ Furthermore, Jindal and Liu²⁴ showed that spam of the second and third type is simple to detect and address.

Unsurprisingly, the practice of opinion spam, and in particular fake reviews, is widely considered as unfair and deceptive, and as such it has been subject of extensive legal scrutiny and court battles. If left unchecked, opinion spam can poison a platform and negatively affect both customers and platform providers (including incurring in financial losses for both parties, as customers may be tricked into purchasing undesirable items and grow frustrated against the platform), at the sole advantage of the spammer (or the entity they represent)—as such, depending on the country's laws, opinion spam may qualify as a form of digital fraud.

Detecting fake reviews is complex for a variety of reasons: for example, spam reviews can be posted by fake or real user accounts. Furthermore, fakes reviews can be posted by individual users or even groups of users.^{27,30} Spammers can deliberately use fake accounts on e-commerce platforms, created only with the scope of posting fake reviews. Fortunately, fake accounts on e-commerce platforms are generally easy to detect, as they engage in intense reviewing activity without any product purchases. An alternative and more complex scenario occurs when fake reviews are posted by real users. This tends to occur under two very different circumstances: compromised accounts (that is, accounts originally owned by legitimate users that have been hacked and sold to spammers) are frequently re-purposed and utilized in opinion spam campaigns;¹¹ and fake review markets became very popular where real users collude in exchange for direct payments to write untruthful reviews for example, without actually purchasing or trying a given product or service. To complicate this matter, researchers showed that fake personas, for example, Facebook profiles,

can be created and associated with such spam accounts.¹⁸ During the late 2000s, many online fake-review markets emerged, whose legality was battled in court by e-commerce giants. Action on both legal and technical fronts has helped mitigating the problem of opinion spam.

From a technical standpoint, a variety of techniques have been proposed to detect review spam. Liu²⁷ identified three main approaches, namely supervised, unsupervised, and group spam detection. In supervised spam detection, the problem of separating fake from genuine (non-fake) reviews is formulated as a classification problem. Jindal and Liu²⁴ pointed out that the main challenge of this task is to work around the shortage of labeled training data. To address this problem, the authors exploited the fact that spammers, to minimize their work, often produce (near-)duplicate reviews, that can be used as examples of fake reviews. Feature engineering and analysis was key to build informative features of genuine and fake reviews, enriched by features of the reviewing users and the reviewed products. Models based on logistic regression have been proven successful in detecting untruthful opinions in large corpora of Amazon reviews.²⁴ Detection algorithms based on support vector machines or naive Bayes models generally perform well (above 98% accuracy) and scale to production systems.²⁹ These pipelines are often enhanced by human-in-the-loop strategies, where annotators recruited through Amazon Mechanical Turk (or similar crowd-sourcing services) manually label subsets of reviews to separate genuine from fake ones, to feed online learning algorithms so to constantly adapt to new strategies and spam techniques.^{11,27}

Unsupervised spam detection was used both to detect spammers as well as for detecting fake reviews. Liu²⁷ reported on methods based on detecting anomalous behavioral patterns typical of spammers. Models of spam behaviors include targeting products, targeting groups (of products or brands), general and early rating deviations.²⁷ Methods based on association rules can capture atypical behaviors of reviewers, detecting anomalies

in reviewers' confidence, divergence from average product scores, entropy (diversity or homogeneity) of attributed scores, or temporal dynamics.³⁹ For what concerns the unsupervised detection of fake reviews, linguistic analysis was proved useful to identify stylistic features of fake reviews, for example, language markers that are over- or underrepresented in fake reviews. Opinion spam to promote products, for example, exhibits on average three times fewer mentions of social words, negative sentiment, and long words (> six letters) than genuine reviews, while containing twice more positive terms and references to self than formal texts.¹¹

Concluding, group spam detection aims at identifying signatures of collusion among spammers.³⁰ Collective behaviors such as spammers' coordination can emerge by using combinations of frequent pattern mining and group anomaly ranking. In the first stage, the algorithm proposed by Mukherjee et al.³⁰ identifies groups of reviewers who all have reviewed a same set of products—such groups are flagged as potentially suspicious. Then, anomaly scores for individual and group behaviors are computed and aggregated, accounting for indicators that measure the group burstiness (that is, writing reviews in short timespan), group reviews similarity, and so on. Groups are finally ranked in terms of their anomaly scores.³⁰

The rise of spam bots. Prior to the early 2000s, most of the spam activity was still coordinated and carried out, at least in significant part, by human

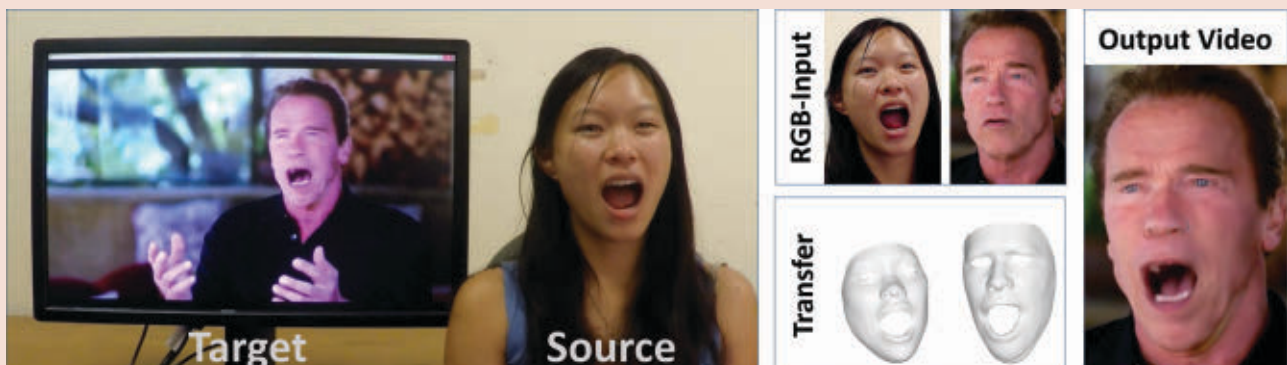
operators: email spam campaigns, Web link farms, and fake reviews, among others, all rely on human intervention and coordination. In other words, these spam operations scale at a (possibly significant) cost. With the rise in popularity of online social network and social media platforms (see Figure 1), new forms of spam started to emerge at scale. One such example is social link farms:¹⁹ similarly to Web link farms, whose goal is to manipulate the perception of popularity of a certain website by artificially creating many pointers (hyperlinks) to it, in social link farming spammers create online personas with many artificial followers. This type of spam operation requires creating thousands (or more) of accounts that will be used to follow a target user in order to boost its apparent influence. Such “disposable accounts” are often referred to as fake followers as their purpose is solely to participate in such link-farming networks. In some platforms, link farming was so pervasive that spammers reportedly controlled millions of fake accounts.¹⁹ Link farming introduced a first level of automation in social media spam, namely the tools to automatically create large swaths of social media accounts.

In the late 2000s, social spam obtained a new potent tool to exploit: bots (short for software robots, a.k.a. social bots). In my 2016 *Communications* article “The Rise of Social Bots,”¹⁶ I noted that “bots have been around since the early days of computers:” examples of bots include chatbots, algo-

rithms designed to hold a conversation with a human, Web bots, to automate the crawling and indexing of the Web, trading bots, to automate stock market transactions, and much more. Although isolated examples exist of such bots being used for nefarious purposes, I am unaware of any reports of systematic abuse carried out by bots in those contexts.

A social bot is a new breed of “computer algorithm that automatically produces content and interacts with humans on the social Web, trying to emulate and possibly alter their behavior.” Since bots can be programmed to carry out arbitrary operations that would otherwise be tedious or time-consuming (thus expensive) for humans, they allowed for scaling spam operations on the social Web to an unprecedented level. Bots, in other words, are the dream spammers have been dreaming of since the early days of the Internet: they allow for personalized, scalable interactions, increasing the cost effectiveness, reach, and plausibility of social spam campaigns, with the added advantage of increased credibility and the ability to escape detection achieved by their human-like disguise. Furthermore, with the democratization and popularization of machine learning and AI technologies, the entry barrier to creating social bots has significantly lowered.¹⁶ Since social bots have been used in a variety of nefarious scenarios (see the sidebar “Social Spam Applications”), from the manipulation of political discussion, to the spread of conspiracy theories and false news, and even by

Figure 2. Video sequence real-time reenactment using AI.³⁴ This proof-of-concept technology could be abused to create AI-fueled multimedia spam.



extremist groups for propaganda and recruitment, the stakes are high in the quest to characterize bot behavior and detect them.^{35,c}

Maybe due to their fascinating morphing and disguising nature, spam bots have attracted the attention of the AI and machine learning research communities: the arms-race between spammers and detection systems yielded technical progress on both the attacker's and the defender's technological fronts. Recent advancements in AI (especially artificial neural networks, or ANNs) fuel bots that can generate human-like natural language and interact with human users in near real time.^{16,35} On the other hand, the cyber-security and machine learning communities came together to develop techniques to detect the signature of artificial activity of bots and social network sybils.^{16,40}

In Ferrara et al.,¹⁶ we fleshed out techniques used to both create spam bots, and detect them. Although the degree of sophistication of such bots, and therefore their functionalities, varies vastly across platforms and application domains, commonalities also emerge. Simple bots can do unsophisticated operations, such as posting content according to a schedule, or interact with others according to pre-determined scripts, whereas complex bots can motivate their reasoning and react to further human scrutiny. Beyond anecdotal evidence, there is no systematic way to survey the state of AI-fueled spam bots and consequently their capabilities—researchers adjust their expectations based on advancements made public in AI technologies (with the assumptions that these will be abused by spammers with the right incentives and technical means), and based on proof-of-concept tools that are often originally created with other non-nefarious purposes in mind (one such example is the so-called DeepFakes, discussed later).

^c It should be noted that bots are not used exclusively for nefarious purposes: for example, some researchers used bots for positive health behavioral interventions.¹⁶ Furthermore, it has been noted the most problematic aspect of nefarious bots is their attempt to deceive and disguise themselves as human users: however, many bots are labeled as such and may provide useful services, like live-news updates.

Social Spam Applications

Political manipulation. In a peer-reviewed study published on Nov. 7, 2016⁶ (the day before the U.S. presidential election), I unveiled a massive-scale spam operation affecting the American political Twitter. With the aid of Botometer, an AI system that leverages over a thousand features to separate bots from humans,³⁵ tens of thousands of bots were identified. By studying the activity signatures of these bots, I noted that they were being retweeted at the same rate than human users, which may have contributed to the spread of political misinformation.³⁶ Since most of these bots aimed at sowing chaos, their presence may have inflamed and further polarized the political conversation, with unknown consequences on the integrity of the democratic process. Since then, dozens of studies corroborated these results; many other studies, before and after mine, showed the perils associated with social spam campaigns in political domains. Most recently, the emerging phenomenon of fake news spreading attracted a lot of attention. Vosoughi et al.³⁶ investigated the role of social media, as well as bots, in the spread of true and false news: the authors showed that humans are more likely to share false stories inspired by fear, disgust, and surprise. This suggests that conditioning and manipulation operations online can affect human behavior.

Public health. Conspiracy and denialism are endemic of social networks. Spam in public health discussions has become commonplace for social media: in a recent study, for example, my team highlighted how bots are used to promote electronic cigarettes as cessation devices with health benefits, a fact not definitively corroborated by science.² The use of bots to carry out anti vaccination campaigns has been the subject of investigation of a DARPA Challenge in 2016.³²

Stock market. Automatic trading algorithms leverage information from social media to predict stock prices. Using bots, spam campaigns have been carried out to give the false impression that certain stocks were spoken positively about on Twitter, successfully tricking trading algorithms into buying them in a pump-and-dump scheme unveiled by the U.S. Securities and Exchange Commission (SEC) in 2015.¹⁵

Data leaks. Social platforms enable the often unwilling disclosure of private user information. A recent study showed that over a third of content shared on Facebook has the default public-visibility privacy settings.²⁸ The amount of content accessible to undesirable users may be even higher when considering privacy settings that allow one's friends to access private information and preferences: Research showed that most users indiscriminately accept friendship connections on Facebook.¹⁸ Spam bots can inject themselves into tightly connected communities, by leveraging the weak-tie structure of online social networks,¹² and obtain private user information on large swaths of users. Phishing is also responsible for data leaks. Attacks based on short-URLs are popular on social media: they can hide the true identity of the spammers and have been proven effective to steal personal data.^{9,19}

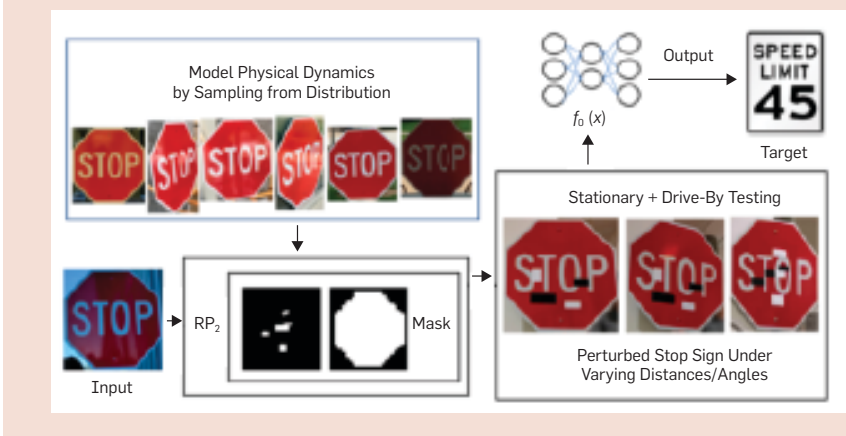
In the sidebar “Social Spam Applications,” I highlight some of the domains where bots made the headlines: one such example is the wake to the 2016 U.S. presidential election, during which Twitter and Facebook bots have been used to sow chaos and further polarize the political discussion.⁶ Although it is not always possible for the research community to pinpoint the culprits, the research of my group, among many others, contributed to unveil anomalous communication dynamics that attracted further scrutiny by law enforcement and were ultimately connected to state-sponsored operations (if you wish, a form of social spam aimed at influencing individual behavior). Spam bots operate in other highly controversial conversation domains: in the context of public health, they promote products or spread scientifically unsupported claims;^{2,15} they

have been used to create spam campaigns to manipulate the stock market;¹⁵ finally, bots have also been used to penetrate online social circles to leak personal user information.¹⁸

AI Spam

AI has been advancing at vertiginous speed, revolutionizing many fields including spam. Beyond powering conversational agents such as chatbots, like Siri or Alexa, AI systems can be used, beyond their original scope, to fuel spam operations of different sorts. I will refer to this phenomenon next as spamming *with* AI, hinting to the fact that AI is used as a tool to create new forms of spams. However, given their sophistication, AI systems can themselves be subject of spam attacks. I will refer to this new concept as spamming *into* AI, suggesting that AIs can be manipulated, and even compromised, by spammers (or attackers

Figure 3. Physical-world attacks onto AI visual classifier.¹⁴ Similar techniques could be abused to inject unwanted spam into AI and trigger anomalous behaviors.



in a broader sense) to exhibit anomalous and undesirable behaviors.

Spamming with AI. Advancements in computer vision, augmented and virtual realities are projecting us in an era where the boundary between reality and fiction is increasingly more blurry. Proofs-of-concept of AIs capable to analyze and manipulate video footages, learning patterns of expressions, already exist: Suwajanakorn et al.³³ designed a deep neural network to map any audio into mouth shapes and convincing facial expressions, to impose an arbitrary speech on a video clip of a speaking actor, with results hard to distinguish, to the human eye, from genuine footage. Thies et al.³⁴ showcased a technique for real-time facial reenactment, to convincingly re-render the synthesized target face on top of the corresponding original video stream (see Figure 2). These techniques, and their evolutions,²⁵ have been then exploited to create so-called DeepFakes, face-swaps of celebrities into adult content videos that surfaced on the Internet by the end of 2017. Such techniques have also already been applied to the political domain, creating fictitious video footage re-enacting Obama,^d Trump, and Putin,^e among several world leaders.²⁵ Concerns about the ethical and legal conundrums of these new technologies have been already expressed.⁸

In the future, well-resourced spammers capable of creating AIs pretending to be human may abuse these technologies. Another example: Google recently demonstrated the ability to deploy an AI (Google Duplex) in the real world to act as a virtual assistant, seamlessly interacting with human interlocutors over the phone:^f such technology may likely be repurposed to carry out massive scale spam-call campaigns. Other forms of future spam with AI may use augmented or virtual reality agents, so-called digital humans, to interact with humans in digital and virtual spaces, to promote products/services, and in worse-case scenarios to carry out nefarious campaigns similar to those of today's bots, to manipulate and influence users.

Spamming into AI. AIs based on ANNs are sophisticated systems whose functioning can sometimes be too complex to explain or debug. For such a reason, ANNs can be easy preys of various forms of attacks, including spam, to elicit undesirable, even harmful system's behaviors. An example of spamming into AI can be bias exacerbation: one of the major problems of modern-days AIs (and, in general, of supervised learning approaches based on big data) is that biases learned from training data will propagate into predictions.

The problem of bias,⁵ especially in AI, is under the spotlight and is being tackled by the computing research

community.⁸ One way an AI can be maliciously led to learn biased models is deliberately injecting spam—here intended as unwanted information—into the training data: this may lead the system to learn undesirable patterns and biases, which will affect the AI system's behavior in line with the intentions of the spammers.

An alternative way of spamming into AI is the manipulation of test data. If an attacker has a good understanding of the limits of an AI system, for example, by having access to its training data and thus the ability to learn strength and weakness of the learned models, attacks can be designed to lure the AI into an undesirable state. Figure 3 shows an example of a physical-world attack that affects an AI system's behaviors in anomalous and undesirable ways:¹⁴ in this case, a deep neural network for image classification (which may have been used, for example, to control an autonomous vehicle) is tricked by a “perturbed” stop sign mistakenly interpreted as a speed limit sign—according to the expectation of the attacker. Spam test data may be displayed to a victim AI system to lure it into behaving according to a scripted plot based on weaknesses of the models and/or of its underlying data. The potential applications of such type of spam attacks can be in medical domains (for example, deliberate misreading of scans), autonomous mobility (for example, attacks on the transportation infrastructure or the vehicles), and more. Depending on the pervasiveness of AI-fueled systems in the future, the questions related to spamming into AI may require the immediate attention of the research community.

Recommendations

Four decades have passed since the first case of email spam was reported by 400 ARPANET users (see Figure 1). While some prominent computer scientists (including Bill Gates) thought that spam would quickly be solved and soon remembered as a problem of the past,¹⁰ we have witnessed its evolution in a variety of forms and environments. Spam feeds itself of (economic, political, ideological, among others) incen-

^d See <https://grail.cs.washington.edu/projects/AudioToObama>

^e See <http://niessnerlab.org/projects/thies-2016face.html>

^f <https://bit.ly/2rznYXJ>

^g <https://bit.ly/2lfdtI2>

tives and of new technologies, both of which there is no shortage of, and therefore it is likely to plague our society and our systems for the foreseeable future.

It is therefore the duty of the computing community to enact policies and research programs to keep fighting against the proliferation of current and new forms of spam. I conclude suggesting three maxims that may guide future efforts in this endeavor:

1. *Design technology with abuse in mind.* Evidence seems to suggest that, in the computing world, new powerful technologies are oftentimes abused beyond their original scope. Most modern-days technologies, like the Internet, the Web, email, and social media, have not been designed with built-in protection against attacks or spam. However, we cannot perpetuate a naive view of the world that ignores ill-intentioned attackers: new systems and technologies shall be designed from their inception with abuse in mind.

2. *Don't forget the arms race.* The fight against spam is a constant arms race between attackers and defenders, and as in most adversarial settings, the party with the highest stakes will prevail: since with each new technology comes abuse, researchers shall anticipate the need for countermeasures to avoid being caught unprepared when spammers will abuse their newly designed technologies.

3. *Blockchain technologies.* The ability to carry out massive spam attacks in most systems exists predominantly due to the lack of authentication measures that reliably guarantee the identity of entities and the legitimacy of transactions on the system. The blockchain as a proof-of-work mechanism to authenticate digital personas (including in virtual realities), AIs, and others may prevent several forms of spam and mitigate the scale and impact of others.^h

Spam is here to stay: let's fight it together!

Acknowledgments

The author would like to thank current and former members of the USC Information Sciences Institute's MINDS research group, as well as of the Indiana University's CNetS group, for invaluable research collaborations and discus-

sions on the topics of this work. The author is grateful to his research sponsors including the Air Force Office of Scientific Research (AFOSR), award FA9550-17-1-0327, and the Defense Advanced Research Projects Agency (DARPA), contract W911NF-17-C-0094.

Trademarked products/services mentioned in this article include: WhatsApp, Facebook Messenger, WeChat, Gmail, Microsoft Outlook, Hotmail, Cisco IronPort, Email Security Appliance (ESA), AOL Instant Messenger, Reddit, Twitter, and Google Duplex. **■**

^h It is worth noting that proof-of-work has been proposed to prevent spam email in the past, however its feasibility remains debated, especially in its original non-blockchain-based implementation.²⁶

References

- Adler, B., Alfaro, L.D. and Pye, I. Detecting Wikipedia vandalism using wikitrust. Notebook papers of CLEF 1 (2010), 22–23.
- Allem, J.P., Ferrara, E., Uppu, S.P., Cruz, T.B. and Unger, J.B. E-cigarette surveillance with social media data: social bots, emerging topics, and trends. *JMIR Public Health and Surveillance* 3, 4 (2017).
- Almeida, T.A., Hidalgo, J.M.G. and Yamakami, A. Contributions to the study of SMS spam filtering: new collection and results. In *Proceedings of the 11th ACM Symposium on Document Engineering*. ACM, 2011, 259–262.
- Androustopoulos, I., Koutsias, J., Chandrinis, K.V. and Spyropoulos, C.D. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2000, 160–167.
- Baeza-Yates, R. Bias on the Web. *Commun. ACM* 61, 6 (June 2018), 54–61.
- Bessi, A. and Ferrara, E. Social bots distort the 2016 US Presidential election online discussion. *First Monday* 21, 11 (2016).
- Caruana, G. and Li, M. A survey of emerging approaches to spam filtering. *ACM Computing Surveys* 44, 2 (2012), 9.
- Chesney, R. and Citron, D. Deep Fakes: A Looming Crisis for National Security, Democracy and Privacy. The Lawfare Blog (2018).
- Chhabra, S., Aggarwal, A., Benevenuto, F. and Kumaraguru, P. Phi.sh/Social: The phishing landscape through short URLs. In *Proceedings of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*. ACM, 2011, 92–101.
- Cranor, L.F. and LaMacchia, B.A. Spam! *Commun. ACM* 41, 8 (Aug. 1998), 74–83.
- Crawford, M., Khoshgoftaar, T.M., Prusa, J.D., Richter, A.D. and Najada, H.A. Survey of review spam detection using machine-learning techniques. *J. Big Data* 2, 1 (2015), 23.
- De Meo, P., Ferrara, E., Fiumara, G. and Provetti, A. On Facebook, most ties are weak. *Commun. ACM* 57, 11 (Nov. 2014), 78–84.
- Drucker, H., Wu, D. and Vapnik, V.N. Support vector machines for spam categorization. *IEEE Trans Neural Networks* 10 (1999).
- Eykholt, K. et al. D. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, 1625–1634.
- Ferrara, E. Manipulation and abuse on social media. *ACM SIGWEB Newsletter* Spring (2015), 4.
- Ferrara, E., Varol, O., Davis, C., Menczer, F. and Flammini, A. The rise of social bots. *Commun. ACM* 59, 7 (July 2016), 96–104.
- Fumera, G., Pillai, I. and Roli, F. Spam filtering based on the analysis of text information embedded into images. *J. Machine Learning Research* 7, (Dec. 2006), 2699–2720.
- Gao, H., Hu, J., Wilson, C., Li, Z., Chen, Y. and Zhao, B.Y. Detecting and characterizing social spam campaigns. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2010, 35–47.
- Ghosh, S. et al. Understanding and combating link farming in the Twitter social network. In *Proceedings of the 21st International Conference on World Wide Web*. ACM, 2012, 61–70.
- Goodman, J., Cormack, G.V. and Heckerman, D. Spam and the ongoing battle for the inbox. *Commun. ACM* 50, 2 (Feb. 2007), 24–33.
- Gupta, B.B., Tewari, A., Jain, A.K. and Agrawal, D.P. Fighting against phishing attacks: state of the art and future challenges. *Neural Computing and Applications* 28, 12 (2017), 3629–3654.
- Hendler, J., Shadbolt, N., Hall, W., Berners-Lee, T. and Weitzner, D. Web science: An interdisciplinary approach to understanding the Web. *Commun. ACM* 51, 7 (July 2008), 60–69.
- Jagatic, T.N., Johnson, N.A., Jakobsson, M. and Menczer, F. Social phishing. *Commun. ACM* 50, 10 (Oct. 2007), 94–100.
- Jindal, N. and Liu, B. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 219–230.
- Kim, H. et al. Deep Video Portraits. arXiv preprint (2018), arXiv:1805.11714.
- Laurie, B. and Clayton, R. Proof-of-work proves not to work; version 0.2. In *Workshop on Economics and Information, Security*, 2004.
- Liu, B. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies* 5, 1 (2012), 1–167.
- Liu, Y., Gummadi, K.P., Krishnamurthy, B. and Mislove, A. Analyzing Facebook privacy settings: User expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*. ACM, 61–70.
- Mukherjee, A. et al. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2013, 632–640.
- Mukherjee, A., Liu, B. and Gance, N. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st International Conference on World Wide Web*. ACM, 2012, 191–200.
- Spirin, N. and Han, J. 2012. Survey on Web spam detection: Principles and algorithms. *ACM SIGKDD Explorations Newsletter* 13, 2 (2012), 50–64.
- Subrahmanian, V.S. et al. The DARPA Twitter Bot Challenge. *Computer* 49, 6 (2016), 38–46.
- Suwajanakorn, S., Seitz, S.M. and Kemelmacher-Shlizerman, I. Synthesizing Obama: Learning lip sync from audio. *ACM Trans Graphics* (2017).
- Thies, J., Zollhöfer, M., Stamminger, M., Theobalt, C., and Nießner, M. Face2Face: Real-time face capture and reenactment of RGB videos. In *Proceedings of Computer Vision and Pattern Recognition*. IEEE, 2016.
- Varol, O., Ferrara, E., Davis, C., Menczer, F. and Flammini, A. Online human-bot interactions: Detection, estimation, and characterization. In *Proceedings of International AAAI Conference on Web and Social Media*, 2017.
- Vosoughi, S., Roy, D. and Aral, S. The spread of true and false news online. *Science* 359, 6380 (2018), 1146–1151.
- Wu, C.H. Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks. *Expert Systems with Applications* 36, 3 (2009), 4321–4330.
- Wu, C.T., Cheng, K.T., Zhu, Q., and Wu, Y.L. Using visual features for anti-spam filtering. In *Proceedings of IEEE International Conference on Image Processing* 3. IEEE, 2005, III–509.
- Xie, S., Wang, G., Lin, S. and Yu, P.S. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, 823–831.
- Yang, Z., Wilson, C., Wang, X., Gao, T., Zhao, B.Y. and Dai, Y. Uncovering social network Sybils in the wild. *ACM Trans. Knowledge Discovery from Data* 8, 1 (2014), 2.

Emilio Ferrara (emiliofe@usc.edu) is an assistant research professor and associate director of Applied Data Science at the University of Southern California Information Sciences Institute, Marina Del Rey, CA, USA.

Copyright held by author/owner.
Publication rights licensed to ACM.

Introducing ACM Transactions on Internet of Things (TIOT)

A new journal from ACM publishing novel research contributions and experience reports in domains whose synergy and interrelations enable the IoT vision

Now Accepting Submissions

ACM Transactions on Internet of Things (TIOT) publishes novel research contributions and experience reports in several research domains whose synergy and interrelations enable the IoT vision. TIOT focuses on system designs, end-to-end architectures, and enabling technologies, and on publishing results and insights corroborated by a strong experimental component.

Submissions are expected to provide experimental evidence of their effectiveness in realistic scenarios and the related datasets. The submission of purely theoretical or speculative papers is discouraged, and so is the use of simulation as the sole form of experimental validation.

Experience reports about the use or adaptation of known systems and techniques in real-world applications are equally welcome, as these studies elicit precious insights for researchers and practitioners alike. For this type of submissions, the depth, rigor, and realism of the experimental component is key, along with the analysis and expected impact of the lessons learned.



**For more information and to submit your work,
please visit <https://tiot.acm.org>.**



Association for
Computing Machinery

Advancing Computing as a Science & Profession

research highlights

P. 94

Technical Perspective **The True Cost of Popularity**

By Graham Cormode

P. 95

Heavy Hitters via Cluster-Preserving Clustering

By Kasper Green Larsen, Jelani Nelson,
Huy L. Nguyễn, and Mikkel Thorup

Technical Perspective

The True Cost of Popularity

By Graham Cormode

THE NOTION OF popularity is prevalent within society. We have made charts of the most popular music and movies since the early part of the 20th century. Elections and referenda are primarily decided by who gets the most votes. Within computer systems, we monitor followers and endorsements in social networks, and track views, hits, and connection attempts in other networks.

Computationally, the problem of determining which items are popular appears at first a straightforward one. Given a dataset of votes, we can simply sort by the item identifier, then count up how many votes are assigned to each. When the number of votes is large, we might try to avoid the overhead of sorting, and aim to more directly pick out the most popular items with only a few passes through the data.

Things get more interesting when we further refine the problem. What happens when the number of votes and the number of candidate items gets so large that it is not feasible to keep a tally for each candidate? This might be implausible in the context of political elections, but is an everyday reality in social systems handling many billions of actions (representing votes) on pieces of content or links (representing the items). Here, we may only get one opportunity to see each vote, and must update our data structures accordingly before moving on to the next observation. Other twists complicate things further: What if votes can have different weights, reflecting the intensity of the endorsement? What if these weights can be negative, encoding a removal of support for an item? What if the formula to compute the overall score is not the sum of the weights, but the square of the sum of the weights?

Each of these variations makes the problem more challenging, while only increasing the generality of any solution: if we can create an algorithm to handle all these variations, then it will still work when they do not apply. Such has been the level of interest in designing effective and efficient algorithms that a lexicon has emerged to describe them: the

most popular items are the *heavy hitters*; processing each update once as it arrives gives the *streaming model*; allowing negative weights is the (general) *turnstile* model; setting a threshold for being a heavy hitter based on removing the k heaviest items is the *k-tail* version; and a weighting function based on squared sums is called l_2 . So while the following paper by Larsen et al. addresses the *k-tail* l_2 heavy hitters problem in the turnstile streaming model, it should be understood as solving a most general version of the problem.

Solutions for more restricted versions of this problem have been defined over the years, and have been put to use in deployments handling large volumes of data. For example, Twitter has used heavy hitter algorithms to track the number of views of individual tweets as they are embedded in different pages around the Web.^a Meanwhile, Apple has combined heavy hitter algorithms with privacy tools to allow privately tracking the emerging popularity of words, phrases and emoticons among their users.^b

Broadly speaking, heavy hitter algorithms are defined by two phases: a collection phase to gather data and statistics from viewing the stream of

updates, and a search process to extract the heavy hitter items. There are simple and effective randomized algorithms that can create summaries that allow the estimation of the final weight of a given item to a high degree of accuracy. However, when there are a very large number of possible items to consider (say, the combination of every tweet and every page on the Web), making the search process efficient becomes the chief objective.

Consequently, the main focus of the following paper is on building up sufficient information to allow a more effective search process. It proceeds by incrementally developing the solution from first principles, relying on concepts from across computer science: randomly partitioning the input space to simplify the core problem; modifying the encoding of the item identifiers, and applying ideas from coding theory to correct for noise; using a construction based on expander graphs to make this more robust; and finally making use of an approach to clustering from spectral graph theory to ensure the identifiers of the heavy hitters can be correctly extracted. The end result is an algorithm that, for the first time, meets the minimum space cost to solve the problem while giving an efficient search time cost.

This opens the way for further work. How efficiently could this clustering approach be implemented in practice, and what applications might it find elsewhere? While identifying popular items is a foundational question for data analysis, there are many more questions that can be asked. The area of streaming algorithms concerns itself with finding efficient algorithms for statistics and queries on large data viewed as a stream of updates. Current challenges revolve around processing massive datasets to extract statistical models for prediction and inference. 

^a <https://skillsmatter.com/skillscasts/6844-count-min-sketch-in-real-data-applications>

^b <https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html>

The main focus of the following paper is on building up sufficient information to allow a more effective search process.

Graham Cormode is a professor in the Department of Computer Science at the University of Warwick, U.K.

Copyright held by author.

Heavy Hitters via Cluster-Preserving Clustering

By Kasper Green Larsen,^{*} Jelani Nelson,[†] Huy L. Nguyễn,[‡] and Mikkel Thorup[§]

Abstract

We develop a new algorithm for the turnstile heavy hitters problem in general turnstile streams, the **EXPANDERSKETCH**, which finds the approximate top- k items in a universe of size n using the same asymptotic $O(k \log n)$ words of memory and $O(\log n)$ update time as the **COUNTMIN** and **COUNTSKETCH**, but requiring only $O(k \text{poly}(\log n))$ time to answer queries instead of the $O(n \log n)$ time of the other two. The notion of “approximation” is the same ℓ_2 sense as the **COUNTSKETCH**, which given known lower bounds is the strongest guarantee one can achieve in sublinear memory.

Our main innovation is an efficient reduction from the heavy hitters problem to a clustering problem in which each heavy hitter is encoded as some form of noisy spectral cluster in a graph, and the goal is to identify every cluster. Since every heavy hitter must be found, correctness requires that every cluster be found. We thus need a “cluster-preserving clustering” algorithm that partitions the graph into pieces while finding every cluster. To do this we first apply standard spectral graph partitioning, and then we use some novel local search techniques to modify the cuts obtained so as to make sure that the original clusters are sufficiently preserved. Our clustering algorithm may be of broader interest beyond heavy hitters and streaming algorithms.

1. INTRODUCTION

Finding “frequent” or “top- k ” items in a dataset is a common task in data mining. In the data streaming literature, this problem is typically referred to as the *heavy hitters problem*, which is as follows: a frequency vector $x \in \mathbb{R}^n$ is initialized to the zero vector, and we process a stream of updates $\text{update}(i, \Delta)$ for $\Delta \in \mathbb{R}$, with each such update causing the change $x_i \leftarrow x_i + \Delta$. The goal is to identify coordinates in x with large weight (in absolute value) while using limited memory. For example, i may index distinct Web surfers, and x_i could denote the number of times person i clicked a Web ad on a particular site; Metwally et al.¹² gave an application of then finding frequent ad clickers in Web advertising. Or in networking, $n = 2^{32}$ may denote the number of source IP addresses in IPv4, and x_i could

be the number of packets sent by i on some link. One then may want to find sources with high link utilization in a network traffic monitoring application. In both these cases $\Delta = 1$ in every update. Situations with negative Δ may also arise; for example one may want to notice *changes* in trends. Imagine n is the number of words in some lexicon, and a search engine witnesses a stream of queries. One may spot trend shifts by identifying words that had a large change in frequency of being searched across two distinct time periods T_1 and T_2 . If one processes all words in T_1 as $\Delta = +1$ updates and those in T_2 as $\Delta = -1$ updates, then x_i will equal the difference in frequency of queries for word i across these two time periods. Thus, finding the top k heavy indices in x could find newly trending words (or words that once were trending but no longer are).

Returning to technical definitions, we define item weights $w_i := f(x_i)$ based on a weighting function $f: (-\infty, \infty) \rightarrow [0, \infty)$. We then define W to be the sum of all the weights that is $W := \sum_{i=1}^n w_i$. Given some parameter k , a *k-heavy hitter* under this weighting function is an item i such that $w_i > W/k$. It is clear that k is an upper bound on the number of k -heavy hitters, and thus an algorithm that finds them all is solving some form of approximate top- k problem (it is approximate since we only require finding top- k items that are heavy enough, with respect to the weighting function under consideration). We will in fact study a harder problem known as the *tail heavy hitters problem*, for which we say an item is a *k-tail heavy hitter* if $w_i > W_{[k]}/k$. Here $W_{[k]}$ is the sum of all weights except for the top k . Note that the number of tail heavy hitters must be less than $2k$ (namely the actual top k , plus the fewer than k items in the tail which may satisfy $w_i > W_{[k]}/k$). One specific goal for the algorithm then, which we require in this paper, is to at query time output a list $L \subset \{1, \dots, n\}$ such that (1) L contains every k -tail heavy hitter, and (2) $|L| = O(k)$. All the algorithms discussed here can also be slightly modified to provide the guarantee that every item in L is at least a $2k$ -tail heavy hitter, so that false positives in L are guaranteed to still be somewhat heavy.

The earliest literature on the heavy hitters problem focused on the case of *insertion-only streams*, in which case $\text{update}(i, \Delta)$ has $\Delta = 1$ for every update (in contrast with the *general turnstile model*, which allows arbitrary $\Delta \in \mathbb{R}$). This corresponds to seeing a stream of indices $i_1, i_2, \dots, i_m$ and wanting to find those items which occur frequently. The perhaps most well-studied form of the problem in insertion-only streams is $f(x_i) = x_i$. A simple solution then is sampling: if the stream is $i_1, i_2, \dots, i_m$, then sample t uniformly random indices $j_1, \dots, j_t$ to create a new sampled stream $i_{j_1}, \dots, i_{j_t}$. A straightforward analysis based on the

^{*} Supported by Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation, grant DNRF84, a Villum Young Investigator Grant and an AUFF Starting Grant.

[†] Supported by NSF grant IIS-1447471 and CAREER award CCF-1350670, ONR Young Investigator award N00014-15-1-2388 and DORECG award N00014-17-1-2127, an Alfred P. Sloan Research Fellowship, and a Google Faculty Research Award.

[‡] Supported by NSF CAREER award CCF-1750716.

[§] Supported by his Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research and by his Investigator Grant 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation.

The preliminary version of this paper was published in IEEE FOCS 2016.

Chernoff-Hoeffding bound shows if one picks $t = \Omega(k^2 \log k)$ then with large probability one can solve (the non-tail version) of the problem by returning the top $O(k)$ frequency items in the sampled stream. A more clever algorithm from 1982 known as Frequent¹³ though solves the problem for the same identity weighting function using only $O(k)$ words of memory, which is optimal. One then may wonder: what then is the motivation for considering other weighting functions f ?

We now observe the following: consider the weighting function $f(x_i) = |x_i|^p$ for some fixed $p > 1$. The usual jargon for such f refers to the resulting problem as ℓ_p heavy hitters (or ℓ_p tail heavy hitters when one considers the tail version). If one assumes that item frequencies are distinct, or simply that if several items are tied for the k th largest then none of them are considered as being in the top k , then as $p \rightarrow \infty$ the top k items are all guaranteed to be k -tail heavy hitters. This implies that for large p a correct tail heavy hitter algorithm is required to not miss any item in the top k , which is a more stringent requirement and thus a harder problem to solve. To see this, suppose the k th item has frequency x_i and the $(k+1)$ st has frequency $x_{i'} < x_i$.

Then we want that

$$w_i = x_i^p > \frac{n-k}{k} \cdot x_{i'}^p \geq W_{[k]} / k.$$

This inequality indeed holds for $p \rightarrow \infty$; specifically it suffices that $p > \log((n-k)/k) / \log(x_i / x_{i'})$. One then may expect that, due to larger p forcing a “more exact” solution to the top- k problem, solving ℓ_p tail heavy hitters should be harder as p grows. This is in fact the case: any solution to ℓ_p heavy hitters (even the non-tail version) requires $\Omega(n^{1-2/p})$ bits of memory.² It is furthermore also known, via a formal reduction, that solving larger p is strictly harder,⁹ in that for $p > q$ a solution to ℓ_p tail heavy hitters in space S leads to a solution for ℓ_q in space $O(S)$, up to changing the parameter k by at most a factor of two. In light of these two results, for tail heavy hitters solving the case $p = 2$ is the best one could hope for while using memory that is subpolynomial in n . A solution to the case $p = 2$ was first given by the COUNTSKETCH, which uses $O(k \log n)$ words of memory. Most recently the BPTree was given using only $O(k \log k)$ words of memory,³ improving the preceding CountSieve data structure using $O(k \log k \log \log n)$ words.⁴

Thus far we have described previous results in the insertion-only model, and we now turn back to the turnstile model. Considering both positive and negative Δ is important when one does not necessarily want the top k items in a single stream, but perhaps the top k items in terms of frequency *change* across two different streams (e.g., two time windows). For example, we may have two streams of query words to a search engine across two time windows and want to know which words had their frequencies change the most. If one interprets stream items in the first window as $\Delta = +1$ updates and those in the second window as corresponding to $\Delta = -1$ updates, then x_i for any word i will be the difference between the frequencies across the two windows, so that we are now attempting to solve an approximate top- k problem in this vector of frequency differences.

The complexity of the heavy hitters problem varies dramatically when one moves from the insertion-only model to the turnstile model. For example, it is clear that the

first-mentioned algorithm of sampling the stream no longer works (consider for the example the stream which does $\text{update}(1, +1)$ N times followed by $\text{update}(1, -1)$ N times for large N , followed by $\text{update}(2, 1)$; only 2 is a heavy hitter, but a sampling algorithm will likely not notice). It turns out that the Frequent, BPTree, and CountSieve data structures also do not have versions that can operate in the turnstile model; in fact a lower bound of the work of Jowhari et al.⁹ shows that any solution to ℓ_p heavy hitters for $p \geq 1$ in the turnstile model requires $\Omega(k \log n)$ words of memory, which is more than what is required by any of these three data structures. The COUNTSKETCH, however, does function correctly even in the turnstile model, and is asymptotically memory-optimal in this model due to the lower bound of the work of Jowhari et al.⁹ Another popular algorithm in the turnstile model, for the easier ℓ_1 tail heavy hitters problem, is the COUNTMIN sketch, also using $O(k \log n)$ words of memory. The COUNTMIN sketch though has the advantage that in the so-called *strict* turnstile model, when we are promised that at all times in the stream $x_i \geq 0$ for all i , the constants that appear in its space complexity are smaller than that of the COUNTSKETCH.

Given that the space complexity of the heavy hitters problem is resolved up to a constant factor in the turnstile model, what room for improvement is left? In fact, the quality of a data structure is not measured only along the axis of space complexity, but also update time, query time, and failure probability (all the algorithms mentioned thus far, even in the insertion-only model but with the exception of Frequent, are randomized and have a bounded failure probability of incorrectly answering a query). Using $k \log n$ words of memory the COUNTMIN sketch and COUNTSKETCH each have $O(\log n)$ update time and failure probability $1/\text{poly}(n)$, which are both the best known, but they suffer from a query time of $\Theta(n \log n)$. That is, even though the final output list L only has size $O(k)$, the time to construct it is slightly superlinear in the size of the universe the items are drawn from. For $p = 1$ and only in the strict turnstile model, this was remedied by the so-called “dyadic trick” as shown in the work of Cormode and Muthukrishnan⁷ and “hierarchical COUNTSKETCH” as shown in the work of Cormode and Hadjieleftheriou,⁶ which each could trade off space and update time for improved query time; see Figure 1 for detailed bounds. None of these solutions though were able to simultaneously achieve the best of all worlds, namely (1) working in the general turnstile model, (2) achieving the ℓ_2 guarantee, (3) achieving the $O(k \log n)$ space and $O(\log n)$ update time of the COUNTSKETCH, and (4) achieving the $k \text{poly}(\log n)$ query time of the dyadic trick. That is, none were able to do so until the EXPANDERSKETCH.

1.1. Our main contribution

We give the first data structure, the EXPANDERSKETCH, for general turnstile ℓ_2 tail heavy hitters using optimal $O(k \log n)$ words of memory with $1/\text{poly}(n)$ failure probability, fast $O(\log n)$ update time, and fast $k \text{poly}(\log n)$ query time.

2. THE EXPANDERSKETCH

We here provide an overview of our algorithm, EXPANDERSKETCH. Henceforth to avoid repetitiveness, we will often drop the “tail” in “tail heavy hitters”; all heavy hitters problems we consider henceforth are the tail version. Our first

Figure 1. Previous results for the turnstile ℓ_p heavy hitters problem, stated for failure probability $1/\text{poly}(n)$. The “general” column states whether the algorithm works in the general turnstile model (as opposed to only strict turnstile). Memory consumption is stated in machine words. The Hierarchical COUNTSKETCH query time could be made $k n^\gamma$ for arbitrarily small constant $\gamma > 0$. The constant c in the $\log^c n$ term in the EXPANDERSKETCH query time can be made $3 + o(1)$ and most likely even $2 + o(1)$ using more sophisticated graph clustering subroutines, and in the strict turnstile model it can even be made $1 + o(1)$; see full paper.

Data structure	Memory	Update time	Query time	Guarantee	General?
COUNTMIN ⁷	$k \log n$	$\log n$	$n \log n$	ℓ_1	Y
COUNTMIN with dyadic trick ⁷	$k \log^2 n$	$\log^2 n$	$k \log^2 n$	ℓ_1	N
Hierarchical COUNTSKETCH ⁶	$k \log n$	$\log n$	$k \cdot n^{0.1}$	ℓ_1	N
COUNTSKETCH ⁵	$k \log n$	$\log n$	$n \log n$	ℓ_2	Y
EXPANDERSKETCH (this work)	$k \log n$	$\log n$	$k \log^c n$	ℓ_2	Y

step is a reduction from arbitrary k to several heavy hitters problems for which k is “small”. In this reduction, we initialize data structures (to be designed) $D_1, \dots, D_q$ for the k' -heavy hitters problem for $q = \max\{1, \Theta(k/\log n)\}$ and pick a hash function $h: [n] \rightarrow [q]$. When we see an update (i, Δ) in the stream, we feed that update only to $D_{h(i)}$. We can show that after the reduction, whp (i.e., with probability $1 - 1/\text{poly}(n)$) we are guaranteed each of the k -heavy hitters i in the original stream now appears in some substream updating a vector $x' \in \mathbb{R}^n$, and i is a k' -heavy hitter in x' for $k' = C \log n$. One then finds all the k' -heavy hitters in each substream then outputs their union as the final query result. For the remainder of this overview we thus focus on solving k -heavy hitters for $k < C \log n$, that is how to implement one of the D_j , so there are at most $O(\log n)$ heavy hitters. Our goal is to achieve failure probability $1/\text{poly}(n)$ with space $O(k \log n)$, update time $O(\log n)$, and query time $k \text{poly}(\log n) = \text{poly}(\log n)$.

There is an algorithm, the Hierarchical COUNTSKETCH (see Figure 1), which is almost ideal. It achieves $O(k \log n)$ space and $O(\log n)$ update time, but the query time is polynomial in the universe size instead of our desired polylogarithmic. Our main idea is to reduce to the universe size to polylogarithmic so that this solution then becomes viable for fast query. We accomplish this reduction while maintaining $O(\log n)$ update time and optimal space by reducing our heavy hitter problem into $m = \Theta(\log n / \log \log n)$ separate *partition heavy hitters problems*, which we now describe.

In the partition k -heavy hitters problem there is some partition $\mathcal{P} = \{S_1, \dots, S_N\}$ of $[n]$, and it is presented in the form of an oracle $\mathcal{O}: [n] \rightarrow [N]$ such that for any $i \in [n]$, $\mathcal{O}(i)$ gives the $j \in [N]$ such that $i \in S_j$. In what follows the partitions will be random, with $\mathcal{O}_j$ depending on some random hash functions. Define a vector $y \in \mathbb{R}^N$ such that for each $j \in [N]$, $y_j = \|x_{S_j}\|_2$, where x_{S_j} is the projection of x onto a subset S of coordinates. The goal then is to solve the k -heavy hitters problem on y subject to streaming updates to x : we should output a list $L \subset [N]$, $|L| = O(k)$, containing all the k -heavy hitters of y . Our desired failure probability is $1/\text{poly}(N)$.

We remark at this point that the ℓ_1 version of partition heavy hitters in the strict turnstile model is simple to solve (and for ℓ_1 strict turnstile, our algorithm already provides improvements over previous work and is thus delving into). In particular, one can simply use a standard strict turnstile ℓ_1 heavy hitters algorithm for an N -dimensional vector and translate every update $e(i, \Delta)$ to $\text{update}(\mathcal{O}(i), \Delta)$. This in effect treats y_j as $\sum_{i: \mathcal{O}(i)=j} x_i$, which is exactly $\|x_{\mathcal{O}^{-1}(j)}\|_1$ as desired in

the case of strict turnstile streams. For details on the ℓ_2 version in the general turnstile model, we refer to the full version of our work. It suffices to say here that the Hierarchical COUNTSKETCH can be modified to solve even the partition heavy hitters problem, with the same space and time complexities as in Figure 1 (but with the n 's all replaced with N 's).

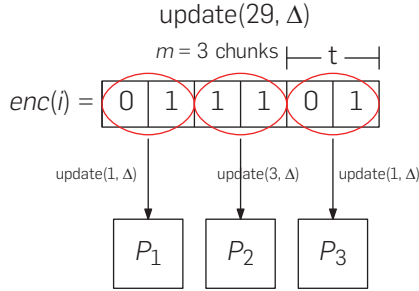
Now we explain how we make use of partition heavy hitters. We take the pedagogical approach of explaining a simple but flawed solution, then iteratively refine to fix flaws until we arrive at a working solution.

Take 1. Recall our overall plan: reduce the universe size so that (the partition heavy hitters version of the) Hierarchical COUNTSKETCH has fast query time. For each index $i \in [n]$ we can write i in base b (for some b yet to be determined) so that it has digit expansion $d_{m-1} d_{m-2} \dots d_0$ with each $d_i \in \{0, \dots, b-1\}$. Our algorithm instantiates m separate partition heavy hitter data structures $P_0, \dots, P_{m-1}$ each with $N = b$ and where P_j has oracle $\mathcal{O}_j(i)$ mapping i to the j th digit in its base- b expansion (See Figure 2). If we choose $b = \text{poly}(\log n)$ (which we will do), then our query time per P_j is a fast $k \cdot b^\gamma = \text{poly}(\log n)$ (see Figure 1). Suppose for a moment that there was only one heavy hitter i and that, by a stroke of luck, none of the P_j 's fail. Then since i is heavy, $\mathcal{O}_j(i)$ must be heavy from the perspective of P_j for each j (since $\mathcal{O}_j(i)$ receives all the mass from x_i , plus potentially even *more* mass from indices that have the same j th digit in base- b). Recovering i would then be simple: we query each P_j to obtain d_j , then we concatenate the digits d_j to obtain i .

There are of course two main problems: first, each P_j actually outputs a list L_j which can have up to $\Theta(\log n)$ “heavy digits” and not just 1, so it is not clear which digits to concatenate with which across the different j . The second problem is that the P_j 's are randomized data structures that fail with probability $1/\text{poly}(b) = 1/\text{poly}(\log n)$, so even if we knew which digits to concatenate with which, some of those digits are likely to be wrong.

Take 2. We continue from where we left off in the previous take. The second issue, that some digits are likely to be wrong, is easy to fix. Specifically, for $b = \text{poly}(\log n)$ we have $m = \log_b n = O(\log n / \log \log n)$. For this setting of b, m , using that the failure probability of each P_j is $1/\text{poly}(b)$, a simple calculation shows that whp $1 - 1/\text{poly}(n)$, at most a small constant fraction of the P_j fail. Thus, for example, at most 1% of the digits d_j are wrong. This is then easy to fix: we do not write i in base b but rather write $\text{enc}(i)$ in base b , where $\text{enc}(i)$ is the encoding of i (treated as a $\log n$ bit string) into $T = O(\log n)$ bits by an error-correcting code with constant rate that can correct an $\Omega(1)$ -fraction of errors. Such codes exist

Figure 2. Simplified version of final data structure. The update is $x_{29} \leftarrow x_{29} + \Delta$ with $m = 3$, $t = 2$ in this example. Each P_j is a b -tree operating on a partition of size 2^t .

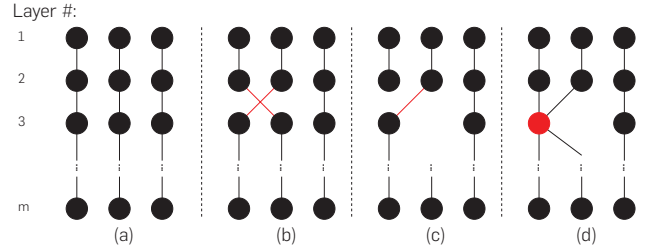


with linear time encoding and decoding.¹⁶ Then even if we recover $\text{enc}(i)$ with 1% of the digits being incorrect, we can decode to recover i exactly.

We now return to the first issue, which is the main complication: in general, there may be more than one heavy hitter (there may be up to $\Theta(\log n)$ of them). Thus, even if we performed wishful thinking and pretended that every P_j succeeded, and furthermore that every L_j contained exactly the j th digits of the encodings of heavy hitters and nothing else, it is not clear how to perform the concatenation. For example, if there are two heavy hitters with encoded indices 1100 and 0110 in binary that we then write in, say, base 4 (as 30 and 12), suppose the P_j correctly return $L_1 = \{3, 1\}$ and $L_2 = \{0, 2\}$. How would we then know which digits matched with which for concatenation? That is, are the heavy hitter encodings 30 and 12, or are they 32 and 10? Brute force trying all possibilities is too slow, since $m = \Theta(\log n / \log \log n)$ and each $|L_j|$ could be as big as $\Theta(\log n)$, yielding $(C \log n)^m = \text{poly}(n)$ possibilities. In fact this question is quite related to the problem of *list-recoverable codes*, but since no explicit codes are known to exist with the efficiency guarantees we desire, we proceed in a different direction.

To aid us in knowing which chunks to concatenate with which across the L_j , the attempt we describe now (which also does not quite work) is as follows. Define m pairwise independent hash functions $h_1, \dots, h_m: [n] \rightarrow [\text{poly}(\log n)]$. Since there are $O(\log n)$ heavy hitters, any given h_j perfectly hashes them with decent probability. Now rather than partitioning according to $\mathcal{O}_j(i) = \text{enc}(i)_j$ (the j th digit of $\text{enc}(i)$), we imagine setting $\mathcal{O}_j(i) = h_j(i) \circ \$ \circ \text{enc}(i)_j \circ \$ \circ h_{j+1}(i)$ where $\circ$ denotes concatenation. Define an index $j \in [m]$ to be *good* if (a) P_j succeeds, (b) h_j perfectly hashes all heavy hitters $i \in [n]$, and (c) for each heavy hitter i , the total ℓ_2 weight from non-heavy hitters hashing to $h_j(i)$ is $o((1/\sqrt{\log n}) \|x_{[k]}\|_2)$. A simple argument shows that whp a $1 - \epsilon$ fraction of the $j \in [m]$ are good, where ϵ can be made an arbitrarily small positive constant. Now let us perform some wishful thinking: if *all* $j \in [m]$ are good, and furthermore no non-heavy elements appear in L_j with the same h_j but different h_{j+1} evaluation as an actual heavy hitter, then the indices in L_j tell us which chunks to concatenate within L_{j+1} , so we can concatenate, decode, then be done. Unfortunately a small constant fraction of the $j \in [m]$ are not good, which prevents this scheme from working (see Figure 3). Indeed, in order to succeed in a query,

Figure 3. Each vertex in row j corresponds to an element of L_j , that is the heavy hitter chunks out-put by P_j . When indices in $\mathcal{P}_j$ are partitioned by $h_j(i) \circ \text{enc}(i)_j \circ h_{j+1}(i)$, we connect chunks along paths. Case (a) is the ideal case, when all j are good. In (b) P_2 failed, producing a wrong output that triggered incorrect edge insertions. In (c) both P_2 and P_3 failed, triggering an incorrect edge and a missing vertex, respectively. In (d) two heavy hitters collided under h_3 , causing their vertices to have the same name thereby giving the appearance of a merged vertex. Alternatively, light items masking as a heavy hitter might have appeared in L_3 with the same h_3 evaluation as a heavy hitter but different h_4 evaluation, causing the red vertex to have two outgoing edges to level 4.



for each heavy hitter we must correctly identify a large connected component of the vertices corresponding to that heavy hitter's path — that would correspond to containing a large fraction of the digits of the encoding, which would allow for decoding. Unfortunately, paths are not robust in having large connected component subgraphs remaining even for $O(1)$ bad levels.

Take 3. The above consideration motivates our final scheme, which uses an expander-based idea first proposed in the work of Gilbert et al.⁸ in the context of “for all” ℓ_1/ℓ_1 sparse recovery, a problem in compressed sensing. Although our precise motivation for the next step is slightly different than in the work of Gilbert et al.,⁸ and our query algorithm and our definition of “robustness” for a graph will be completely different, the idea of connecting chunks using expander graphs is very similar to an ingredient in that work. The idea is to replace the path in the last paragraph by a graph which is robust to a small fraction of edge insertions and deletions, still allowing the identification of a large connected component in such scenarios. Expander graphs will allow us to accomplish this. For us, “robust” will mean that over the randomness in our algorithm, whp each corrupted expander is still a spectral cluster (to be defined shortly). For,⁸ robustness meant that each corrupted expander still contains an induced small-diameter subgraph (in fact an expander) on a small but constant fraction of the vertices, which allowed them a recovery procedure based on a shallow breadth-first search. They then feed the output of this breadth-first search into a recovery algorithm for an existing list-recoverable code (namely Parvaresh-Vardy codes). Due to suboptimality of known list-recoverable codes, such an approach would not allow us to obtain our optimal results.

2.1. An expander-based approach

Let F be an arbitrary D -regular connected graph on the vertex set $[m]$ for some $D = O(1)$. For $j \in [m]$, let $\Gamma(j) \subset [m]$ be the set of neighbors of vertex j . We partition $[n]$ according to $\mathcal{O}_j(i) = z(i)_j = h_j(i) \circ \$ \circ \text{enc}(i)_j \circ \$ \circ h_{\Gamma(j)} \circ \$ \dots \circ h_{\Gamma(j)}$ where

$\Gamma(j)_k$ is the k th neighbor of j in F . Given some such z , we say its *name* is the first $s = O(\log \log n)$ bits comprising the h_j portion of the concatenation. Now, we can imagine a graph G on the layered vertex set $V = [m] \times [2^s]$ with m layers. If L_j is the output of a heavy hitter query on P_j , we can view each element z of L_j as suggesting D edges to add to G , where each such z connects D vertices in various layers of V to the vertex in layer j corresponding to the name of z . The way we actually insert the edges is as follows. First, for each $j \in [m]$ we instantiate a *partition point query structure* Q_j with the same oracle as the P_j ; this is a data structure which, given any $z \in [N]$, outputs a low-error estimate $\tilde{y}_z$ of y_z with failure probability $1/\text{poly}(N)$. We modify the definition of a level $j \in [m]$ being “good” earlier to say that Q_j must also succeed on queries to every $z \in L_j$. We point query every partition $z \in L_j$ to obtain an estimate $\tilde{y}_z$ approximating y_z . We then group all $z \in L_j$ by name, and within each group we remove all z from L_j except for the one with the largest $\tilde{y}_z$, breaking ties arbitrarily. This filtering step guarantees that the vertices in layer j have unique names, and furthermore, when j is good all vertices corresponding to heavy hitters appear in L_j and none of them are thrown out by this filtering. We then let G be the graph created by including the at most $(D/2) \cdot \sum_j |L_j|$ edges suggested by the z ’s across all L_j (we only include an edge if both endpoints suggest it). Note G will have many isolated vertices since only $m \cdot \max_j |L_j| = O(\log^2 n / \log \log n)$ edges are added, but the number of vertices in each layer is 2^s , which may be a large power of $\log n$. We let G be its restriction to the union of non-isolated vertices and vertices whose names match the hash value of a heavy hitter at the m different levels. This ensures G has $O(\log^2 n / \log \log n)$ vertices and edges. We call this G the *chunk graph*.

Now, the intuitive picture is that G *should* be the vertex-disjoint union of several copies of the expander F , one for each heavy hitter, plus other junk edges and vertices coming from other non-heavy hitters in the L_j . Due to certain bad levels j however, some expanders might be missing a small constant ϵ -fraction of their edges, and also the ϵm bad levels may cause spurious edges to connect these expanders to the rest of the graph. The key insight is as follows. Let W be the vertices of G corresponding to some particular heavy hitter, so that in the ideal case W would be a single connected component whose induced graph is F . What we can prove, even with ϵm bad levels, is that every heavy hitter’s such vertices W forms an $O(\epsilon)$ -spectral cluster.

Definition 1. An ϵ -spectral cluster in an undirected graph $G = (V, E)$ is a vertex set $W \subseteq V$ of any size satisfying the following two conditions: First, only an ϵ -fraction of the edges incident to W leave W , that is, $|\partial(W)| \leq \epsilon \text{vol}(W)$, where $\text{vol}(W)$ is the sum of edge degrees of vertices inside W . Second, given any subset A of W , let $r = \text{vol}(A)/\text{vol}(W)$ and $B = W \setminus A$. Then

$$|E(A, B)| \geq (r(1-r) - \epsilon) \text{vol}(W).$$

Note $r(1-r) \text{vol}(W)$ is the number of edges one would expect to see between A and B had W been a random graph with a prescribed degree distribution.

Roughly, the above means that (a) the cut separating W from the rest of G has $O(\epsilon)$ conductance (i.e., it is a very sparse

cut), and (b) for any cut $(A, W \setminus A)$ within W , the number of edges crossing the cut is what is guaranteed from a spectral expander, minus $O(\epsilon) \cdot \text{vol}(W)$. Our task then reduces to finding all ϵ -spectral clusters in a given graph. We devise a scheme CUTGRABCLOSE that for each such cluster W , we are able to find a $(1 - O(\epsilon))$ -fraction of its volume with at most $O(\epsilon) \cdot \text{vol}(W)$ erroneous volume from outside W . This suffices for decoding for ϵ a sufficiently small constant, since this means we find most vertices, that is chunks of the encoding, of the heavy hitter.

For the special case of ℓ_1 heavy hitters in the strict turnstile model, we are able to devise a much simpler query algorithm that works; see the full paper for details. For this special case we also in the full paper devise a space-optimal algorithm with $O(\log n)$ update time, whp success, and *expected* query time $O(k \log n)$ (though unfortunately the variance of the query time may be quite high).

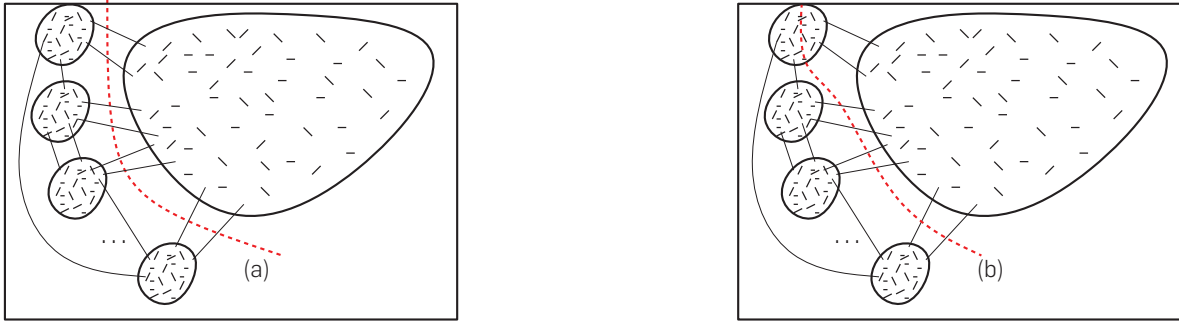
2.2. Cluster-preserving clustering

As mentioned, an ϵ -spectral cluster is a subset W of the vertices in a graph $G = (V, E)$ such that (1) $|\partial W| \leq \epsilon \text{vol}(W)$, and (2) for any $A \subsetneq W$ with $\text{vol}(A)/\text{vol}(W) = r$, $|E(A, W \setminus A)| \geq (r(1-r) - \epsilon) \text{vol}(W)$. Item (2) means the number of edges crossing a cut within W is what you would expect from a random graph, up to $\epsilon \text{vol}(W)$. Our goal is to, given G , find a partition of V such that every ϵ -spectral cluster W in G matches some set of the partition up to $\epsilon \text{vol}(W)$ symmetric difference.

Our algorithm CUTGRABCLOSE is somewhat similar to the spectral clustering algorithm of (Kannan et al.,¹⁰ Section 4), but with local search. That algorithm is simple: find a low-conductance cut (e.g., a Fiedler cut) to split G into two pieces, then recurse on both pieces. Details aside, Fiedler cuts are guaranteed by Cheeger’s inequality to find a cut of conductance $O(\sqrt{\gamma})$ as long as a cut of conductance at most γ exists in the graph. The problem with this basic recursive approach is shown in Figure 4 (in particular cut (b)). Note that a cluster can be completely segmented after a few levels of recursion, so that a large portion of the cluster is never found.

Our approach is as follows. Like the above, we find a low-conductance cut then recurse on both sides. However, before recursing on both sides we make certain “improvements” to the cut. We say $A \subset V$ is *closed* in G if there is no vertex $v \in G \setminus A$ with at least $5/9$ ths of its neighbors in A . Our algorithm maintains that all recursive subcalls are to closed subsets in G as follows. Suppose we are calling CUTGRABCLOSE on some set A which we inductively know is closed in G . We first try to find a low-conductance cut within A . If we do not find one, we terminate and let A be one of the sets in the partition. Otherwise, if we cut A into $(S, \bar{S})$, then we close both $S, \bar{S}$ by finding vertices violating closure and simply moving them. It can be shown that if the $(S, \bar{S})$ cut had sufficiently low conductance, then these local moves can only improve conductance further. Now both S and $\bar{S}$ are closed in A (which by a transitivity lemma we show, implies they are closed in G as well). We then show that if (1) some set S is closed, and (2) S has much more than half the volume of some spectral cluster W (e.g., a $2/3$ fraction), then in fact S contains a $(1 - O(\epsilon))$ -fraction of W . Thus after closing both $S, \bar{S}$, we have that S either: (a) has almost none of W , (b) has

Figure 4. Each small oval is a spectral cluster. They are well-connected internally, with sparse cuts to the outside. The large oval is the rest of the graph, which can look like anything. Cut (a) represents a good low-conductance cut, which makes much progress (cutting the graph in roughly half) while not losing any mass from any cluster. Cut (b) is also a low-conductance cut as long as the number of small ovals is large, since then cutting one cluster in half has negligible effect on the cut's conductance. However, (b) is problematic since recursing on both sides loses half of one cluster forever.



almost all of W , or (c) has roughly half of W (between $1/3$ and $2/3$, say). To fix the latter case, we then “grab” all vertices in $\bar{S}$ with some $\Omega(1)$ -fraction, for example $1/6$ th, of their neighbors in S and simply move them all to S . Doing this some constant number of times implies S has much more than $2/3$ rd of W (and if S was in case (a), then we show it still has almost none of W). Then by doing another round of closure moves, one can ensure that both S , $\bar{S}$ are closed, and each of them has either an $O(\epsilon)$ -fraction of W or a $(1 - O(\epsilon))$ -fraction. It is worth noting that our algorithm can make use of *any* spectral cutting algorithm as a black box and not just Fiedler cuts, followed by our grab and closure steps. For example, algorithms from^{14, 15} run in nearly linear time and either (1) report that no γ -conductance cut exists (in which case we could terminate), (2) find a *balanced* cut of conductance $O(\sqrt{\gamma})$ (where both sides have nearly equal volume), or (3) find an $O(\sqrt{\gamma})$ -conductance cut in which every $W \subset G$ with $\text{vol}(W) \leq (1/2) \text{vol}(G)$ and $\phi(W) \leq O(\gamma)$ has more than half its volume on the smaller side of the cut. Item (2), if it always occurred, would give a divide-and-conquer recurrence to yield nearly linear time for finding all clusters. It turns out item (3) though is even better! If the small side of the cut has half of every cluster W , then by grabs and closure moves we could ensure it is still small and has almost all of W , so we could recurse just on the smaller side.

As a result we end up completing the cluster preserving clustering in polynomial time in the graph size, which is polynomial in $\text{poly}(\log n)$, and this allows us to find the k heavy hitters with whp in $O(k \text{poly}(\log n))$ time. For a full description of the algorithm, the reader is referred to reference Larsen et al.¹¹

Acknowledgments

We thank Noga Alon for pointing us to (Alon and Chung,¹ Lemma 2.3), Piotr Indyk for the reference,⁸ Yi Li for answering several questions about,⁸ Mary Wootters for making us aware of the formulation of the list-recovery problem in coding theory and its appearance in prior work in compressed sensing and group testing, and Fan Chung Graham and Olivia Simpson for useful conversations about graph partitioning algorithms.



References

- Alon, N., Chung, F.R.K. Explicit construction of linear sized tolerant networks. *Discrete Math.* 72 (1988), 15–19.
- Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.* 68, 4 (2004), 702–732.
- Braverman, V., Chestnut, S.R., Ivkin, N., Nelson, J., Wang, Z., Woodruff, D.P. BPTree: An ℓ_2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)* (2017), ACM, Chicago, IL, 361–376.
- Braverman, V., Chestnut, S.R., Ivkin, N., Woodruff, D.P. Beating CountSketch for heavy hitters in insertion streams. In *Proceedings of the 48th STOC* (2016), ACM, Cambridge, MA.
- Charikar, M., Chen, K., Farach-Colton, M. Finding frequent items in data streams. *Theor. Comput. Sci.* 312, 1 (2004), 3–15.
- Cormode, G., Hadjieleftheriou, M. Finding frequent items in data streams. *PVLDB* 1, 2 (2008), 1530–1541.
- Cormode, G., Muthukrishnan, S. An improved data stream summary: The count-min sketch and its applications. *J. Algorithms* 55, 1 (2005), 58–75.
- Gilbert, A.C., Li, Y., Porat, E., Strauss, M.J. For-all sparse recovery in near-optimal time. In *Proceedings of the 41st ICALP* (2014), Springer, Copenhagen, Denmark, 538–550.
- Jowhari, H., Sgallam, M., Tardos, G. Tight bounds for L_p samplers, finding duplicates in streams, and related problems. In *Proceedings of the 30th PODS* (2011), ACM, Athens, Greece, 49–58.
- Kannan, R., Vempala, S., Vetta, A. On clusterings: Good, bad and spectral. *J. ACM* 51, 3 (2004), 497–515.
- Larsen, K.G., Nelson, J., Nguyễn, H.L., Thorup, M. Heavy hitters via cluster-preserving clustering. *CoRR*, abs/1511.01111 (2016).
- Metwally, A., Agrawal, D., El Abbadi, A. Efficient computation of frequent and top-k elements in data streams. In *Proceedings of the 10th ICDT* (2005), Springer, Edinburgh, UK, 398–412.
- Misra, J., Gries, D. Finding repeated elements. *Sci. Comput. Program* 2, 2 (1982), 143–152.
- Orecchia, L., Sachdeva, S., Vishnoi, N.K. Approximating the exponential, the Lanczos method and an $\tilde{O}(m)$ -time spectral algorithm for balanced separator. In *Proceedings of the 44th STOC* (2012), 1141–1160.
- Orecchia, L., Vishnoi, N.K. Towards an SDP-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition. In *Proceedings of the 22nd SODA* (2011), SIAM, San Francisco, CA, 532–545.
- Spielman, D.A. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Information Theory* 42, 6 (1996), 1723–1731.

Kasper Green Larsen (larsen@cs.au.dk), Aarhus University, Aarhus, Denmark.

Jelani Nelson (minilek@seas.harvard.edu), Harvard University, Cambridge, MA, USA.

Huy L. Nguyễn (hu.nguyen@northeastern.edu), Northeastern University, Boston, MA, USA.

Mikkel Thorup (mthorup@di.ku.dk), University of Copenhagen, Denmark.

Florida Atlantic University
Chair of the Department of Computer & Electrical Engineering and Computer Science (CEECS)

The College of Engineering and Computer Science (COECS) at Florida Atlantic University (FAU) is seeking a dynamic individual to serve as the Chair of the Department of Computer & Electrical Engineering and Computer Science (CEECS) - <http://www.ceecs.fau.edu/>. A visionary leader with demonstrated collaboration and communication skills is sought to further advance the department's mission.

The department, consisting of 39 faculty and 8 educational and research staff members offers, baccalaureate (800 students), masters and PhD degrees (280 students) in computer engineering, electrical engineering and computer science. A

multidisciplinary masters of bioengineering is also offered in the department. The department collaborates closely with FAU's ISENSE pillar (<http://isense.fau.edu/>). The department's current active funding portfolio is over \$7.0M, with support from NSF, NIH, DoD, DoE, and Industry.

Applicants should have strong research and scholarly portfolios and will: provide leadership for academic and research programs; promote the reputation of the department at the national and international levels; create cooperation opportunities with industry partners; attract world-class faculty, high-caliber students and staff; promote collaborative research within and outside FAU; and mentor faculty at different stages of their careers.

Minimum Qualification:

An earned doctorate from an accredited institution in electrical engineering, computer engineering, computer science or a closely-related field by the time of application required. Must have demonstrated effective leadership, management and communication with faculty, staff, students, and other administrators. Must have commitment to diversity and inclusion as it applies to faculty, staff and students. Must have commitment to shared governance, collegiality, and professional development for faculty and staff.

All applicants must apply electronically to the currently posted position (Chairperson and Professor) on the Office of Human Resources' job website (<https://fau.edu/jobs>) by completing the required online employment application and submitting the related documents. When completing the online application, please upload the following: a cover letter detailing professional experience, curriculum vitae, a statement describing the vision for the department, leadership and mentoring philosophy, teaching and research interests (max 3 pages), contact information for six professional references and copies of official transcripts scanned into an electronic format. Review of applications will begin in August 2019. All applications received by July 31, 2019 will receive full consideration.

A background check will be required for the candidate selected for this position. This position is subject to funding. For more information and to apply, visit www.fau.edu/jobs and go to Apply Now (REQ06164).

Florida Atlantic University is an equal opportunity/affirmative action/equal access institution and all qualified applicants will receive consideration without regard to race, color, religion, sex, sexual orientation, gender identity, national origin, disability status, protected veterans status or other protected status. Individuals with disabilities requiring accommodation, please call 561-297-3057. 711.

FAU is committed to the principles of engaged teaching, research and service. All persons aspiring to achieve excellence in the practice of these principles are encouraged to apply.

The University of Alabama in Huntsville (UAH)

Assistant Professor

The Department of Computer Science at The University of Alabama in Huntsville (UAH) invites applicants for a tenure-track faculty position at the Assistant Professor level beginning January 2020. All applicants with a background in traditional areas of computer science will be considered; however, special emphasis will be given to applicants with expertise in cybersecurity, gaming, software engineering, cloud computing, and systems related areas.

A Ph.D. in computer science or a closely related area is required. The successful candidate will have a strong academic background and be able to secure and perform funded research in areas typical for publication in well-regarded academic conference and journal venues. In addition, the candidate should embrace the opportunity to provide undergraduate education.

The department has a strong commitment to excellence in teaching, research, and service; the candidate should have good communication skills, strong teaching potential, and research accomplishments.

UAH is located in an expanding, high technology area, in close proximity to Cummings Research Park, the second largest research park in the nation and the fourth largest in the world. Nearby are the NASA Marshall Space Flight Center, the Army's Redstone Arsenal, numerous Fortune 500 and high tech companies. UAH also has an array of research centers, including information technology and cybersecurity. In short, collaborative research opportunities are abundant, and many well-educated and highly technically skilled people are in the area. There is also access to excellent public schools and inexpensive housing.

UAH has an enrollment of approximately 9,500 students. The Computer Science department offers BS, MS, and PhD degrees in Computer Science and contributes to interdisciplinary degrees. Faculty research interests are varied and include cybersecurity, mobile computing, data science, software engineering, visualization, graphics and game computing, multimedia, AI, image processing, pattern recognition, and distributed systems. Recent NSF figures indicate the university ranks 30th in the nation in overall federal research funding in computer science.

Interested parties must submit a detailed resume with references to info@cs.uah.edu or Chair, Search Committee, Dept. of Computer Science, The University of Alabama in Huntsville, Huntsville, AL 35899. Qualified female and minority candidates are encouraged to apply. Initial review of applicants will begin as they are received and continue until a suitable candidate is found.

The University of Alabama in Huntsville is an affirmative action/equal opportunity employer/minorities/females/veterans/disabled.

Please refer to log number: 19/20-545



**ADVERTISING
IN CAREER
OPPORTUNITIES**

How to Submit a Classified Line Ad: Send an e-mail to acmm mediasales@acm.org. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.

Estimates: An insertion order will then be e-mailed back to you. The ad will be typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

Deadlines: 20th of the month/2 months prior to issue date. For latest deadline info, please contact: acmm mediasales@acm.org

Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at: <http://jobs.acm.org>

Ads are listed for a period of 30 days.

**For More Information Contact:
ACM Media Sales,
at 212-626-0686 or
acmm mediasales@acm.org**

The research institute for Cyber Defence (CODE) at the Bundeswehr University Munich has been substantially expanded to become the research institute for “Cyber Defence and Smart Data” of the Bundeswehr and the Federal Government institutions. CODE was established in 2013 with the objective of bringing together experts from different faculties and scientific disciplines as well as expertise from industry and government agencies to conduct research in cyber and information space. CODE pursues a comprehensive, integrated, and interdisciplinary approach to implementing technical innovations and concepts for the protection of data, software, and ICT infrastructures in accordance with legal and commercial framework conditions. It has already established important strategic partnerships in this area. The objective of the expansion is to unite the Bundeswehr's and the Federal Government's research initiatives in the area of Cyber Defence and Smart Data and to establish the CODE research institute as the primary point of contact in the cyber and information domains of the Bundeswehr and the Federal Government.

Research and teaching in the area of cyber security is already being carried out as part of the bachelor's and master's programs in the Department of Computer Science. A new independent master's program in Cyber Security was launched on January 1st, 2018.

The Department of Computer Science at the Bundeswehr University Munich is seeking a professor for the following specialist area of its Cyber Defence und Smart Data research institute:

W3 University Professorship for Artificial Intelligence in IT Security

Artificial intelligence methods are used, among other things, to automate fact-based decision-making processes and to simulate and advance the behavior of individual actuators or entire swarms. In the context of IT security, they have the long-term potential not only to automate time-consuming manual analysis tasks of large amounts of data reliably and with low error rates by means of cognitive security, but also to contribute to the protection of networked systems. Therefore, they should be used for the detection of, for example, anomalies or attacks and for the design and implementation of preventive security mechanisms.

Due to the wide range of possible applications of Artificial Intelligence, the W3 professorship has an interface function within the Department of Computer Science and the CODE research institute and serves as a bridge to the engineering, humanities, and social sciences. An excellent, internationally oriented professor is sought who is particularly proven in several of the following areas in research and teaching with a clear reference to IT security:

- Autonomous and distributed AI systems
- Experimental and model-driven cognitive systems
- Machine learning for the detection and prognosis of patterns in data
- Neural Computing, especially Deep Learning and Reinforcement Learning
- Scalable, efficient and resilient analysis algorithms and learning methods

The Bundeswehr University Munich is looking for a professor who, in addition to having outstanding scientific qualifications, will also contribute actively to the CODE research institute.

Besides excellent research work, the new professor is expected to develop demanding lectures, tutorials, and seminars for the master's program in Cyber Security and to provide excellent teaching in her or his respective specialist area.

The applicant is also expected to carry out teaching in the bachelor's programs in computer science and business computer science, and to work closely with the other departments at the Bundeswehr University Munich.

The professorship will be provided with superbly equipped laboratories housed in a new building that will be built in the near future. The candidate must have an excellent scientific track record, as demonstrated by a habilitation or equivalent scientific achievement, as well as relevant publications in academic journals. Proven teaching experience in her or his respective specialist area is highly desired. The new professor should have an international perspective, e.g., based on participation in international research projects and experience in acquiring third-party funding. The duties will also include participation in the academic self-administration of the university. Further, the candidate will be expected to assume a gender equality based leadership role.

The Bundeswehr University Munich offers academic programs directed primarily at officer candidates and officers, who can obtain bachelor's and master's degrees in a trimester system. Study programs are complemented by an integrated program entitled “studium plus” which consists of interdisciplinary seminars, tutorials and training in key professional qualifications.

The recruitment requirements and the employment status of professors are governed by the Federal Civil Service Act (Bundesbeamtengesetz). To become an appointed civil servant (Beamter) candidates must be no older than 50 on the date of their appointment.

The university seeks to increase the number of female professors and thus explicitly invites women to submit applications. Severely disabled candidates with equal qualifications will receive preferential consideration.

Please submit your application documents marked as Confidential Personnel Matter to the **Dean of the Department of Computer Science, Professor Dr. Oliver Rose, Bundeswehr University Munich, D-85577 Neubiberg, via email to dekanat.inf@unibw.de by 30th of September, 2019.**



[CONTINUED FROM P. 104] One of my InfilTraitor bots learned that my opponents in the legislature were convinced I would veto any bill they sent me, so they did not bother to analyze the consequences if I failed to veto it. So, I showed them! I signed it!

Because we had used the example of Grassy Plains School in our false flag fake news campaign, my opponents posted that as the first referendum question. By a close margin, the winning plan was the lawn ornaments 3D printing workshop. Essentially all the voters made their own selections, rather than delegating. I guessed they knew something had to be done with this recently abandoned building, and wanted to avoid the right-wing and left-wing options this early in the development of Columville's new political system.

Over the following weeks, Grassy Plains Ornaments made apparent progress. There already was some antiquated computer-operated manufacturing machinery in the building, left over from the occupational training facility when it ran out of raw materials. By "raw materials," of course I mean students. Local hobbyists enthusiastically repaired the existing equipment and donated a good deal of their own. Their neighbors began placing orders for wooden swans and Hobbits. However, the early results were not as expected; for example, white-winged Hobbits and swans with big feet. It proved very difficult for an uncoordinated group of individualists to assemble essentially obsolete technology into a reliably functioning system. Interest faded, and people began proposing other uses for Grassy Plains.

Meanwhile, people were becoming accustomed to visiting VoterBooth every morning, during an advertisement on the local virtual news program, and paying their \$10 to buck whatever trend the news was promoting. A feature in the system allowed them to propose new legislation, and many of their ideas concerned iconic Grassy Plains. A new proposition emerged: (1) keep the ornament workshop but invest tax money to improve it, (2) convert it into a museum of old computers donated by citizens, (3) tear the building down and decide later what to put in its place. By this point, most of the voters were delegating their votes, and

One version of the fluid democracy fantasy would give each voter a daily vote to cast themselves, or to delegate to someone else.

they wound up giving them to the three citizens who had proposed the three plans. You might guess that one represented my political party, the second represented my opponents, and the third was a wimpy independent, but no, all three were apolitical! And the selection was to destroy Grassy Plains.

I don't know if all this was a plot, either by my opponents, by a malicious foreign power, or just the citizens, but as soon as the demolition team had assembled their crusher vehicles to begin their grim job, a new vote changed everything! A real-time referendum, fluctuating from minute to minute, would steer the demolition fleet to some other building, and rip it suddenly from its roots.

At this very moment, I am watching the local news on my wall-size monitor, as the demolition fleet passes Grassy Plains and heads toward... Yea! It is approaching the legislature building! In one corner of my screen I see the votes streaming in, increasingly aiming at the legislature as its doom nears. But wait! The votes to destroy the legislature are decreasing. The demolition parade turns, and I run to the front door, not having any windows left to look out in this age of virtual reality caves. Oh, no! The voters have decided to demolish my Governor's Mansion! ☐

William Sims Bainbridge (wsbainbridge@hotmail.com) is a sociologist who taught classes on crime and deviant behavior at respectable universities before morphing into a computer scientist, editing an encyclopedia of human-computer interaction, writing many books on things computational, from neural nets to virtual worlds to personality capture, then repenting and writing harmless fiction.

© 2019 ACM 0001-0782/19/8 \$15.00

ACM Transactions



on Social Computing

ACM TSC seeks to publish work that covers the full spectrum of social computing including theoretical, empirical, systems, and design research contributions. TSC welcomes research employing a wide range of methods to advance the tools, techniques, understanding, and practice of social computing, particularly research that designs, implements or studies systems that mediate social interactions among users, or that develops theory or techniques for application in those systems.



For further information
or to submit your
manuscript,
visit tsc.acm.org

From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/3339827

William Sims Bainbridge

Future Tense Fluid Democracy

In trying to “drown” the opposition with daily online elections, I didn’t realize they could wash me away.

EVEN IN THE first month of my governorship of this fine state, I began to have problems with the legislature, which belonged to the “other” political party. I had campaigned on the plan to transform the state capital, Columbusville, into a Smart City, but my political party and the opposition wanted it to use different operating systems. Another problem was that we disagreed about what should be done with the three abandoned shopping centers, now that all our citizens bought their stuff online. That issue was tangled up with all the road improvements needed to keep the self-driving trucks and taxis from roaming the schoolyards, although that could have been worse if the kids were still attending classes rather than home-schooling online as most of them now did. I sent drafts of laws and budgets to the legislature, and they voted them down. It sent laws and budgets to me, and I vetoed them. Clearly, the state of West Montana was spiraling into chaos!

Desperate one day, I was frantically trying to think of a plan to defeat my despicable opponents, and decided to enter “delegitimate” into Wikipedia in search of ideas, but made some typo and got “delegative democracy” instead. It was hard to read the page, because some Wikipedia editors seemed to be battling over whether the text should begin “Delegative democracy, also known as liquid democracy ...” or use “fluid” or “flüssig” instead of “liquid,” apparently one of the instabilities caused by the merger of the English and German Wikipedias the previous week. I tried entering “fluid”



into Google Translate, and discovered that produced “flüssig” in German. Translating “flüssig” into English did not return “fluid,” but gave “liquid.” Chuckling over the joke that this kind of democracy must be “all wet,” I suddenly had a clever idea.

Why not drown my opponents in the Internet-based version of an election every day? Wikipedia told me that one version of the fluid democracy fantasy would give each voter a daily vote. They could cast it themselves, or delegate it to somebody else, such as a professional politician, a member of their family, or even the captain of the neighborhood baseball team. Whoever cast the vote could use it to decide whether something under consideration by the legislature would go up or down, that very day. VoterBooth seemed the best name for this out-of-sight site.

Brilliant political strategist that I am, I secretly formed a corporation to run this supposedly high-security VoterBooth where citizens could vote every day for the modest payment of \$10. By “secretly,” I mean that I added one unit to the chain of non-profits and for-profits I ran, including blockchain money laundering services that assured that the federal regulators could never discover that this virtual shopping center belonged to me.

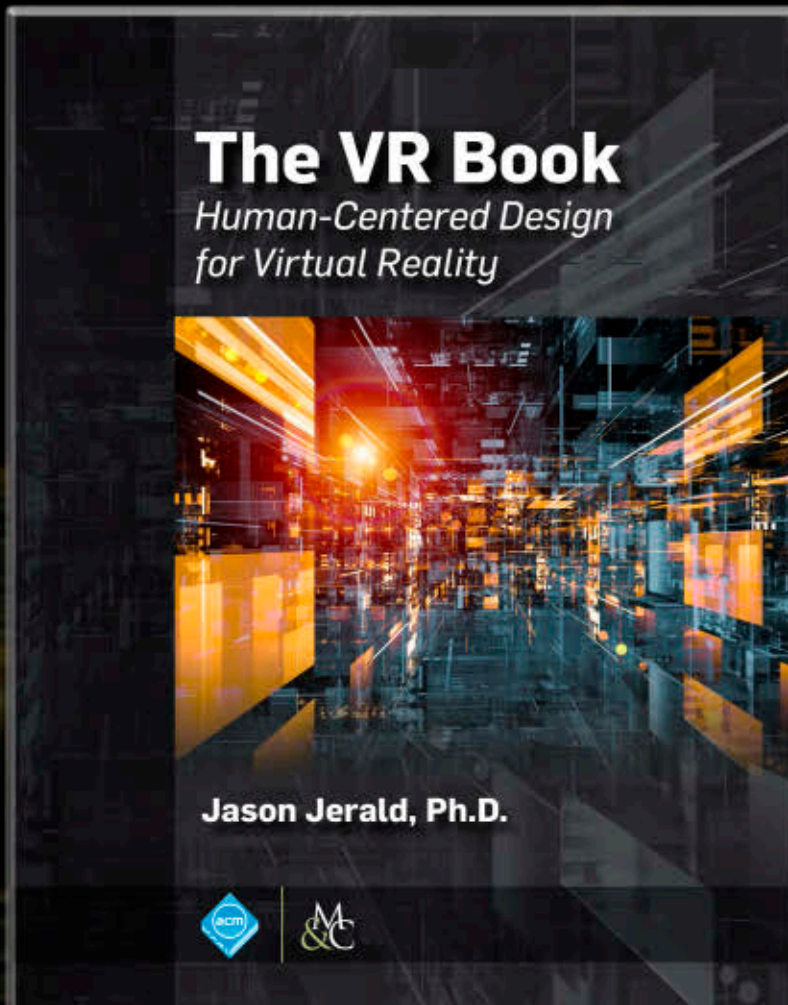
Each day, a citizen could log into VoterBooth and see what issue needed to be decided. For example, “Should Grassy Plains School be converted into a shopping center, an apartment house for unemployed people living off welfare, or a 3D printing workshop for local manufacture of lawn ornaments?” The citizen could vote directly by selecting (1) shopping center, (2) apartment house, (3) 3D printing workshop, (4) keep the school, or (5) abstain. The voter also has the choice not to vote in the referendum, but rather to assign his or her vote to somebody else, even the voter’s school teacher because the system had no age requirements for voting, or a locally admired teenage computer game leaderboard master. Citizens who received votes from others could pass them along, so the result could be a decision tree functioning like an incoherent web of political parties.

My main post-truth IT specialist thought VoterBooth was a great idea, and he quickly astroturfed the echo chambers my opponents operated in Chinbook and Scalpbook, with the idea that fluid democracy would wash away their hated governor, namely me. So VoterBooth became public as their idea, not mine, although I had already set it up so the profits would secretly come to me. [CONTINUED ON P. 103]

Jason Jerald, PhD

The VR Book

Human-Centered Design for Virtual Reality

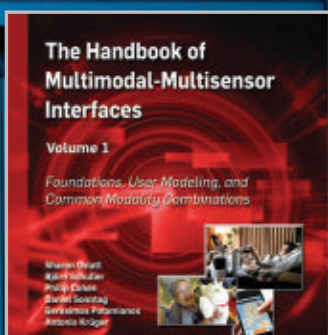
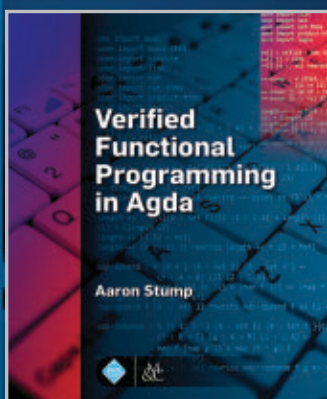
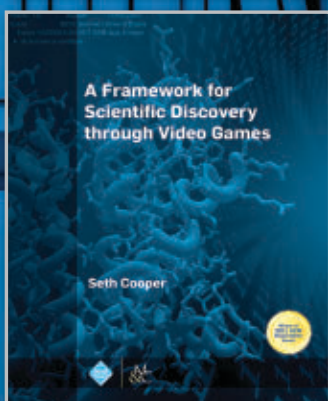
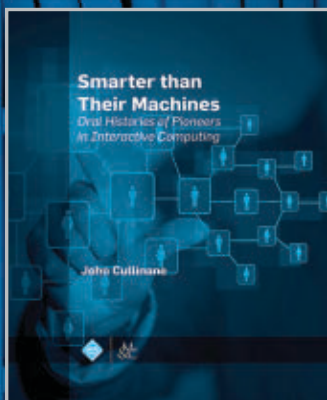
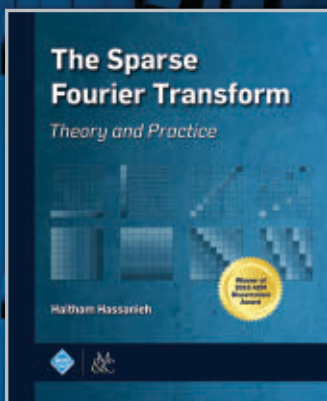
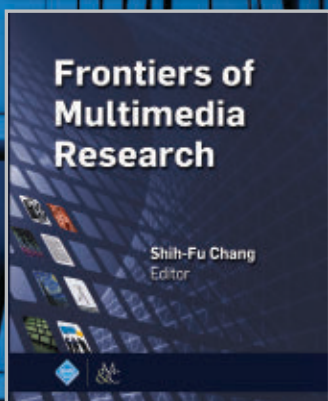
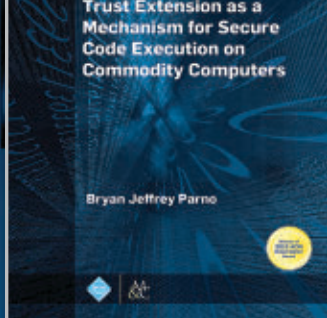
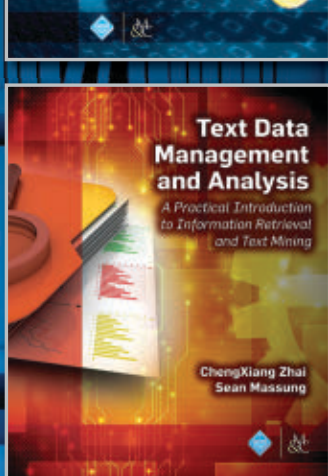


Dr. Jerald has recognized a great need in our community and filled it. The VR Book is a scholarly and comprehensive treatment of the user interface dynamics surrounding the development and application of virtual reality. I have made it required reading for my students and research colleagues. Well done!”

- Prof. Tom Furness, University of Washington, VR Pioneer



ISBN: 978-1-970001-12-9 DOI: 10.1145/2792790
<http://books.acm.org>
<http://www.morganclaypoolpublishers.com/vr>



In-depth. Innovative. Insightful.

Inspired by the need for high-quality computer science publishing at the graduate, faculty, and professional levels, ACM Books are affordable, current, and comprehensive in scope.

**Full Collection | Title List
Now Available**

For more information, please visit
<http://books.acm.org>



Association for Computing Machinery

1601 Broadway, 10th Floor, New York, NY 10019-7434, USA

Phone: +1-212-626-0658 Email: acmbooks-info@acm.org