Mapping the Democratization of AI on GitHub
A First Approach

Marcus Burkhardt

Over the course of the past 80 years the digital computer has radically changed the world we inhabit and the ways in which we relate to it and to each other. Likewise, computing has radically changed as well. At first, practical computing machines[1] carried individual names, but in the early 1950s proper names were quickly replaced by series designators which are emblematic for the era of mainframe computers and time-sharing systems. The 1970s and 1980s gave rise to micro, home and personal computers as well as graphical user interfaces that became prevalent in the 1990s. With the rise of the World Wide Web (WWW) during this decade networked computing and networking changed the face of computing again. Regardless of the burst of the dot-com bubble the Web flourished throughout the early 2000s by being reframed as Web 2.0 and social web. During this period computing devices became increasingly mobile and desktop computers were superseded by notebooks, smartphones and tablets, software gradually morphed into services and apps that rely on cloud infrastructures for distributed processing and storage.

In June 2017 Sundar Pichai, CEO of Google Inc., declared yet another paradigm shift in the history of computing. Innovation should neither be driven by approaching problems as first and foremost digital nor mobile, but instead by taking an AI first approach that is fueled by recent advances in the field of machine learning: "We believe smartphones should be smarter; they should learn from you and they should adapt to you. Technologies such as on-device machine learning can learn your usage patterns and automatically anticipate your next action saving you time" (Pichai 2018). This statement reflects a central promise of machine learning applications, namely the ability to adapt to unforeseen futures without prior programming of a particular event: visual recognition of specific objects or persons that the program did neither "see" nor was trained on before, self-driving cars that can deal with new situations safely or chatbots that conduct conversa-

1 For the concept of *practical computing machines* see Turing (1992).

tions with humans in an engaging manner. Conversely, the more such technologies are built into the fabric of everyday life the more concerns are raised about their potential risks, e.g. biases and inequalities inherent in training data sets. As a result, machine learning models often produce (social) structures instead of adapting to them. Drawing on debates in critical algorithm studies this paper asks how machine learning and artificial intelligence as fields of technological development and innovation are in themselves structured. By providing an initial mapping of the *coding cultures* of machine learning and artificial intelligence on GitHub the paper argues for the importance to attend more closely to the hitherto largely neglected infrastructural layers of code libraries and programming frameworks for the development of critical perspectives on the social and cultural implications of machine learning technologies to come.

Democratization of AI

In recent years the interest in artificial intelligence (AI) in general and in machine learning (ML) in particular skyrocketed once again. This ongoing development is to some extend driven by leading technology companies such as Google and its parent company Alphabet, Amazon Web Services (AWS), Facebook, IBM, Microsoft etc. It rests upon the massive accumulation of data by these companies on the one hand and the establishment of large-scale cloud infrastructures as well as infrastructural services on the other hand. However, these companies do not simply contribute to the rapid technological developments in AI and ML, they also take part in shaping the imaginaries of smart, intelligent and autonomous technologies as cornerstones of technological progress and enablers of social progress as well as economic prosperity for the years to come.

Central to this is the recent push towards the democratization of artificial intelligence. Google (IANS 2017), IBM (Moore 2018), Apple (Simonite 2017) and Microsoft (n.d.) alike mobilize the notion of democratic AI to promote the shift towards ML driven technological innovation. In this context democratization can be understood "as the action/development of making something accessible to everyone, to the 'common masses'" (Schmarzo 2017). For Microsoft this entails allowing "every person and every organization" (n.d.) to partake in the anticipated benefits of AI whose effects will supposedly be as far-reaching as that of the printing press:

> With the advent of the printing press in the 1400s we have an explosion of information—the first democratizing event around access that made it possible for humans everywhere to start learning. Access to information has only spread from there. […] The question is, how can we use all we have in terms of computational

> power to solve this fundamental constraint? To make better sense of the world? That's the essence of what AI is. It's not about having AI that beats humans in games, it's about helping everyone achieve more — humans and machines working together to make the world a better place. (Ibid.)

In view of the long history of exaggerated expectations of artificial intelligence, a certain skepticism regarding such a claim of a revolutionary caesura is warranted. What is more important, nevertheless, is the question how AI is actually made accessible, i.e. how democratization is enacted. Once again, the case of Microsoft can serve as a paradigmatic example. The company pursues a four-fold strategy: (1) utilize AI to develop new modes of interaction with ambient computing technologies; (2) build intelligence into every application; (3) allow developers to make use of these "intelligent capabilities"; (4) make computing infrastructure available as a service (ibid.).

The promise of democratization is directed towards both technology developers and their users. For developers this democratization entails the possibility to make use of AI in their own products and to partake in shaping the future of AI by having open or paid access to resources and services such as software libraries, pre-trained machine learning models, frameworks, platforms and infrastructures. Users on the other hand are enlisted in the democratization of AI as beneficiaries of technologies that are "infused" (ibid.) with artificial intelligence and machine learning. In such technologies, intelligence is typically enacted as a wide range of limited scope capabilities and features: software applications that play chess or the game of go, cameras that recognize faces and take pictures when people are smiling, speakers that are capable to recognize, interpret and execute voice-based commands, cars that drive autonomously, chatbots that engage in entertaining, helpful or informative conversations with human beings etc.

Practices of Machine Learning

The capacities of ML technologies are designed as well as staged to astonish its users. Among many researchers in the fields of science and technology studies and media studies Kate Crawford and Ryan Calo have argued for the "need to assess the impact of technologies on their social, cultural and political settings" (2016: 311). It is, thus, important to gain a critical understanding of machine learning technologies in general and the current drive towards democratic AI in particular. Such a critical understanding is all the more significant since modern day deep neural networks as well as other ML approaches supposedly evade human comprehension in principle.

Despite the secrecy imposed on algorithmic systems by corporations and the intrinsic opacity of machine learning systems (cp. Burrell 2016), there is much to know about ML technologies as Adrian Mackenzie eloquently argued:

> Machine learning is hardly obscure or arcane knowledge today. These techniques are heavily documented in textbooks […], in how-to books […], and numerous video and website tutorials, lectures and demonstrations […]. We can more or less read about and indeed play about with implementations in software […]. (2015: 431p.)

Drawing on such diverse resources Mackenzie himself became a critical student, practitioner and investigator of multiple situated, hybrid machine learning practices. In *Machine Learners* Mackenzie presents a hands-on inquiry of "machine learning as a form of knowledge production and a strategy of power" (ibid.: 9). Following Foucault's notion of archaeology Mackenzie unfolds an archaeology of six operational formations that are central to ML:

> vectorization, optimization, probabilization, pattern recognition, regularization, and propagation. These generic operations intersect in a diagram of machine learning spanning hardware and software architectures, organizations of data and datasets, practices of designing and testing models, intersections between scientific and engineering disciplines, and professional and popular pedagogies. (Ibid.: 18)

The approach to inquire machine learning not from afar, but by "learning to machine learn" (ibid.: 18) resonates well with Wendy Chuns claim "that software can only be understood *in media res*" (Chun 2008: 323). In the middle of things that constitute ML today Mackenzie engages not only with textbooks, tutorials, mathematical formulae, algorithms, and data sets, but also with numerous software libraries. While code libraries and programming frameworks are often referenced as crucial infrastructural elements of today's software culture they are hardly studied closely in critical research (cp. Berry 2011; Marino 2014). *Machine Learners*, too, acknowledges the importance of code libraries frequently, but does not research them in detail. In passing, however, Mackenzie offers some valuable insights into how code libraries and programming frameworks shape machine learning practices by crystallizing "a repertoire of standard operations, patterns, and functions for reshaping data and constructing models that classify and predict events and associations among things, people, processes, and so on" (Mackenzie 2017: 23). Their architecture "classifies and orders" (ibid.: 77) machine learning as a field of interrelated practices as much as a domain of knowledge production. They constitute the "accumulating sediment of coding and related data practices […] in which machine learners take root" (ibid.: 23). For Mackenzie the

implementation and widespread use of code libraries are, thus, articulations of contemporary *coding cultures*.

Code libraries provide resources that developers can draw from and build upon. By offering predefined functions and functionalities they relieve developers from building software from the ground up. At the same time code libraries impose their functional logics and practical affordances on developers. In this regard code libraries can be considered as media of "co-operative action" consisting of accumulative resources that can be *laminated* into operational software (Goodwin 2018: 129). However, code libraries are sites of cooperation as well. They are created, contributed to, maintained, updated and deprecated in the "recursive publics" (Kelty 2008: 28) of code-sharing platforms such as GitHub.[2]

As the "Facebook of coding" (Wulf 2017) GitHub today hosts more than 96 million repositories (GitHub 2019), most of them contain codes or coding related resources. For today's coding culture GitHub serves as a center of gravity for hosting open and closed source software projects as well as for finding, contributing and debating code resources including libraries and frameworks (Mackenzie 2018). This applies in particular to current developments in ML. Yet, how exactly does ML as a practice and field take shape on GitHub? How do algorithmic techniques for ML, data sets, machine learned models and other resources circulate on GitHub? Which actors are involved in shaping this space of cooperation, exchange and negotiation and which strategies of power are deployed? Or to put it differently: How is the democratization of AI enacted on the infrastructural level of code and coding sharing practices?

Mapping the Democratization of AI in Code

For mapping the numerous heterogenous articulations and manifestations of machine learning and artificial intelligence on GitHub researchers can make use of the application programming interface (API) provided by the platform which allows for the retrieval of repository metadata that match certain search criteria, like specific keywords contained in the repositories description or topics assigned to the repository by its creator.[3] However, the GitHub search API poses certain restrictions: it allows only for a limited number of keywords per query and returns only a maximum of 1.000 results for each query. While this might be sufficient to explore the most visible projects on GitHub for a specific keyword in regard to

2 For a discussion of the relevance of version control systems and GitHub for contemporary coding cultures see Burkhardt (2019) and Mackenzie (2017, 2018).

3 A similar mapping of cultures of coding and sharing has been undertaken by Kollanyi (2016) in respect to Bots.

the number of forks or stars a repository received, this limitation disregards the variety of keywords associated with ML and AI as well as the long tail of software development and code sharing practices in the field of ML. In order to map the enactments and materializations of machine learning more comprehensively a search tactic must be deployed that charts the space of ML and AI related repositories iteratively by identifying and working through an extensive list of search terms and by restricting the search parameters (programming languages, numbers of forks, received stars and size) step by step:

→ search terms
→ search terms + programming language
→ search terms + programming language + fork count
→ search terms + programming language + fork count + star count
→ search terms + programming language + fork count + star count + size

This search tactic provides a snapshot view of a dynamically changing environment. The results remain incomplete, but are more comprehensive than the default limit of 1.000 results per query.[4] What is left out and remains invisible in principle are all the private repositories hosted on GitHub.

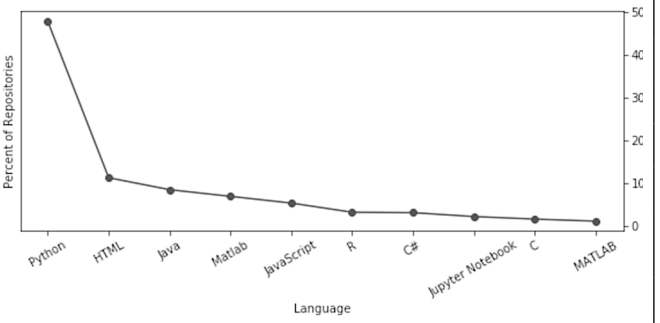
Assembling a dataset is all but the first step in mapping the enactments of machine learning in contemporary coding culture. The dataset contains 211.802 unique repositories.[5] Among those 41.818 contain artificial intelligence[6] as a keyword and 103.344 machine learning[7]. Remarkably only 3.064 repositories reference both conceptual fields although in public discourse the current summer of AI is largely attributed to developments in the area of machine learning.

4 Certain gaps in the result sets can be pinned down precisely: The result limit is exceeded for programming languages like Python that have neither been forked nor received a star and are relatively small (in the case of Python it is 0, 1, 2, 3 …, 11 KB). However, in other cases gaps emerge as inconsistencies between the supposed number of results returned by the GitHub API and the actual number of results retrieved.

5 Search terms: ai, dl, nn, ml, artificial-intelligence, artificialintelligence, artificial intelligence, machine learning, machinelearning, machine-learning, machine intelligence, deep learning, deep-learning, deeplearning, neural nets, neuralnets, neural-nets, neural net, neuralnet, neural-net, neural networks, neuralnetworks, neural-networks, neural network, neuralnetwork, neural-network, neural, bigdl, caffe, caffe2, cntk, coreml, deeplearning4j, keras, lasagne, mlib, mlpack, moa, mocha.jl, mxnet, neon, Paddle Paddle, pylearn, pylearn2, pytorch, scikit-learn, shogun, singa, tensorflow, tflearn, theano.

6 Variations considered: artificial intelligence, artificial-intelligence, artificialintelligence, ai.

7 Variations considered: machine learning, machine-learning, machinelearning, ml.



Language	%
Python	47.74
HTML	11.25
Java	8.46
Matlab	6.90
JavaScript	5.33
R	3.19
C#	3.09
Jupyter Notebook	2.18
C	1.59
MATLAB	1.10
Total	90,83

Fig. 1: Distribution of top 10 programming languages used in the retrieve repositories.

Machine learning and artificial intelligence materializes in a range of programming languages. *Python* (together with Python-based *Juypter Notebooks*) is by far the most often used programming language for ML. Remarkably this is followed by eleven percent HTML repositories. As will be discussed later this is because not all repositories on GitHub contain code as a resource. Many contain informational and educational resources such as tutorials, book manuscripts, research papers, course materials or collections of links.

The popularity or relevance of repositories can be inferred from the number of forks as well as stars they received. In the context of GitHub in particular and the Git version control system in general forks are copies of a repository. Forking constitutes a central practice in Git-based collaboration: "Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea" (GitHub a). Stars on the other hand are platform specific indicators of some kind of interest of users in a repository: "You can star repositories and topics to keep track of projects you find interesting and discover related content in your news feed" (GitHub b).

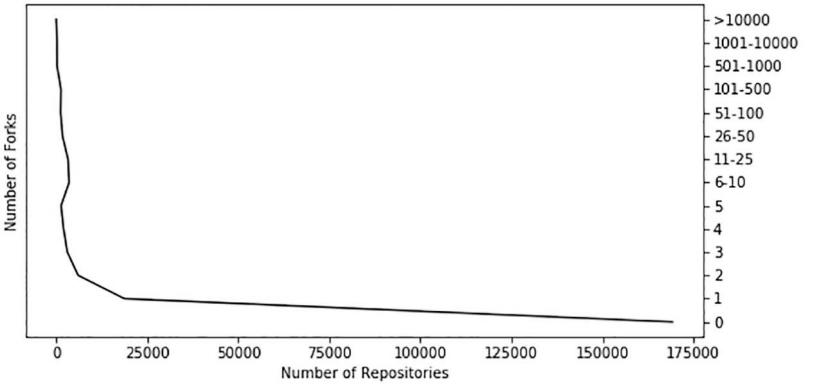


Fig. 2: Distribution of forks by repository

The distribution of stars and forks across repositories adheres to the power law. Only few repositories gain high visibility, while the largest number of repositories has no forks and received little to no stars. The long tail of machine learning is a space of testing out, experimenting with and learning by doing. This space, however, is also populated with course assignments, original research in progress, personal collections of resources on machine learning, and new, emerging, failed or abandoned code libraries. Among the more than 200.000 unique repositories that were retrieved for this article less than 0.8 percent have more than 100 or more forks and only 1.8 percent received 100 or more stars.

	Total number	≥100 forks	≥100 stars
Artificial intelligence repositories	41.818	284	593
Machine learning repositories	103.344	799	1.677
Deep learning repositories	33.749	857	1.842
Neural network repositories	46.681	558	1.365
All repositories	211.802	1.751	3.951

High level observations on the use of programming languages or the distribution of popularity offer some initial insights into how artificial intelligence and machine learning is articulated on GitHub. A more detailed analysis of the 200 most often forked and starred repositories, however, reveals the heterogeneity and diversity of resources developed, published, collaborated, maintained, debated, updated and downloaded. Among those top 200 repositories are 42 that can be categorized as code libraries or programming frameworks. However, more than

fifty percent contain informational and educational resources. About 15 percent of the repositories provide reference implementations for algorithms, machine learned models for specific application domains or software applications based on machine learning. And only few provide infrastructural resources, programming languages, experiments or datasets for ML.

Type	#
Informational/Educational resource	115
Library/Framework	42
Algorithms/Models/Applications	31
Infrastructure/Optimization	8
Experiments	2
Languages	1
Datasets	1

The informational and educational resources address a range of distinct audiences like machine learning beginners, learners of a framework, machine learning professionals as well as researchers. As a result, some repositories provide resources on the fundamentals of machine learning, materials for online courses, tutorials for different code libraries or introductory book publications. Others are framed as *collections*, *comprehensive lists* or *curated lists* on machine learning in general, state of the art research or more specific topics like infrastructures. In part these resources are created, assembled and provided by individual developers, researchers or authors. Yet, many informational and educational repositories are created and maintained by corporations to educate users *about* and recruite them *for* their products. The repository *amazon-sagemaker-examples*[8] released by *Amazon Web Services Labs* for example contains basic information and training materials on using the companies *SageMaker* platform for the training, optimization and deployment of machine learning models. Here the infrastructural complexities of machine learning practices become visible. Understood as a mode of databased programming machine learning relies on large scale processing capacities that today are provided as cloud computing services—a market dominated by global technology corporations like Amazon, Microsoft, Google, IBM etc. (Flexera 2019). What is more, engaging with the rich diversity of educational and informational resources reveals the central role of programming libraries for coding cultures. Learning to machine learn is deeply intertwined with learning to make use of code

8 https://aws.amazon.com/de/sagemaker/

libraries and frameworks. While there are repositories dedicated to do "Machine Learning From Scratch"[9], educational resources promising information on how to do machine learning with library x or framework y are more common.

The variety of repositories with informational resources is reflected in the heterogeneity of code repositories. Today, by far the most popular repository in machine learning and artificial intelligence is the TensorFlow framework.[10] Initially developed by Google Brain for internal use the framework has been released in 2015 as open source by Google and is since maintained by the company, but it has attracted a large number of external contributors as well. TensorFlow is a framework for deep learning that mainly supports the training and deployment of neural networks and is, thus, dedicated to a specific paradigm of machine learning that has become dominant throughout the past decade. Other popular libraries such as *scikit-learn*[11] focus on other areas of machine learning and in consequence support a broader range of approaches.[12] While it is possible to implement basic neural networks with scikit-learn they play a somewhat marginal role in this library. This is underlined by the missing support for deep and reinforcement learning as well as the use of GPU hardware (scikit-learn 2019, 5 and 7). TensorFlow and scikit-learn, thus, are different articulations and materializations of machine learning. Yet, they are similar in that they can both be considered as general-purpose frameworks, i.e. they are not explicitly focused on specific application domains.

A number of special-purpose machine learning libraries have gained a relatively high popularity as well. Among them are the *Unity ML-Agents Toolkit*[13], OpenCV[14], and the *ChatterBot*[15] and *Rasa*[16] frameworks. The ML-Agents Toolkit allows the implementation of so-called machine learning agents in the Unity game engine. As a plugin for a development environment for games the toolkit is of course aimed at this application area, but also allows for the development of algorithms for robotics or the training of self-driving cars in virtual environments (cp. Unity n.d.). Chatterbot and Rasa are frameworks for the implementation of AI-based chatbots. Machine learning here articulates itself as the use of pretrained models for natural language understanding and dialogue management as well as the ongoing

9 https://aws.amazon.com/eriklindernoren/ML-From-Scratch

10 https://github.com/tensorflow/tensorflow

11 https://github.com/scikit-learn/scikit-learn

12 For a comprehensive overview of machine learning paradigms and approaches see Domingos (2015).

13 https://github.com/Unity-Technologies/ml-agents

14 https://github.com/opencv/opencv

15 https://github.com/gunthercox/ChatterBot

16 https://github.com/RasaHQ/rasa

improvement of such models based on training data gathered in beta tests and during real world deployment. By providing code resources for the development of domain specific software, such frameworks are not for machine learning in general, but have some element of machine learning build in. This raises the question how special-purpose frameworks structure machine learning practices in specific ways as well as how they prescribe the operational logics and performances of applications created with them. In the case of chatbots that is to ask how *conversationality* and *communicability* is inscribed by such frameworks.

In addition to general-purpose and special-purpose libraries at least a third type can be distinguished which could be called meta-libraries. A popular example for this kind of library is *Keras*[17]. It is built on top of the deep learning frameworks TensorFlow, Theano[18] and CNTK[19] and provides a unified interface to a multiplicity of different code libraries. Keras, thus adds an abstraction layer which serves as frontend to the backend of multiple deep learning libraries. Two of those libraries are developed by global corporations—Google in the case of TensorFlow and Microsoft in the case of CNTK (Cognitive Toolkit). The third framework was mainly developed in academic contexts and is maintained by the Montreal Institute for Learning Algorithms. The end of its active development has been announced in 2017 citing among other reasons that "strong industrial players are backing different software stacks in a stimulating competition" (Bengio 2017). Indeed, large global digital technology companies compete in the space of open source machine learning libraries with respective frameworks. *PyTorch*[20] (Facebook), *Neo-AI-DLR*[21] (Amazon), *Aerosolve*[22] (AirBnB) as well as the already mentioned frameworks TensorFlow (Google) and CNTK (Microsoft) are just a few examples. The release of machine learning libraries in open source by Google, Facebook, Amazon, Microsoft etc. can be understood as an effort towards the democratization of AI. At the same time, the open source coding culture has become a space of corporate intervention and competition. How this affects the future of machine learning technologies as well as the ways in which machine learning will be built into the fabric of our technological world are questions that remain unanswered. Critical research, thus, needs to attend even more closely to the logics and politics of code libraries and programming frameworks.

17 https://github.com/keras-team/keras

18 https://github.com/Theano/Theano

19 https://github.com/microsoft/CNTK

20 https://github.com/pytorch/pytorch

21 https://github.com/neo-ai/neo-ai-dlr

22 https://github.com/airbnb/aerosolve

Literature

Bengio, Yoshua (2017): "MILA and the Future of Theano." 2017. https://groups.google.com/forum/#!topic/theano-users/7Poq8BZutbY.

Berry, David M (2011): The Philosophy of Software: Code and Mediation in the Digital Age, Basingstoke: Palgrave Macmillan.

Burkhardt, Marcus (2019): "Version Control. Zur Softwarebasierten Koordination von Ko-Laboration." In: Gießmann, Sebastian/ Röhl, Tobias/Trischler, Ronja (eds.), Materialität Der Kooperation, Wiesbaden: Springer VS, pp. 91–117.

Burrell, J. (2016): "How the Machine 'Thinks: Understanding Opacity in Machine Learning Algorithms." In: Big Data & Society 3/1. https://doi.org/10.1177/2053951715622512.

Chun, Wendy Hui Kyong (2008): "On 'Sourcery,' or Code as Fetish." In: Configurations 16/3, pp. 299-324. https://doi.org/10.1353/con.0.0064.

Crawford, Kate/ Calo, Ryan (2016): "There Is a Blind Spot in AI Research." In: Nature News 538/7625, pp. 311-13. https://doi.org/10.1038/538311a.

Domingos, Pedro (2015): The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World, New York: Basic Books.

Flexera (2019): "RightScale State of the Cloud Report 2019." Accessed June 9, 2019. https://info.flexerasoftware.com/SLO-WP-State-of-the-Cloud-2019-DE.

GitHub (2019): "The State of the Octoverse 2018." Accessed June 11, 2019. https://octoverse.github.com/.

— (a): "About Stars." Accessed June 11, 2019. https://help.github.com/en/articles/about-stars.

— (b): "Fork a Repo." Accessed June 6, 2019. https://help.github.com/en/articles/fork-a-repo.

Goodwin, Charles (2018): Co-Operative Action. Learning in Doing: Social, Cognitive and Computational Perspectives, New York: Cambridge University Press.

IANS (2017): "We Want to Democratise Artificial Intelligence: Google." YourStory.Com. Accessed June 9, 2019. https://yourstory.com/2017/08/democratise-artificial-intelligence-google/.

Kelty, Christopher M. (2008): Two Bits: The Cultural Significance of Free Software. Experimental Futures, Durham: Duke University Press.

Kollanyi, Bence (2016): "Where Do Bots Come From? An Analysis of Bot Codes Shared on GitHub." In: International Journal of Communication 10/0, pp. 4932-4951.

Mackenzie, Adrian (2015): "The Production of Prediction: What Does Machine Learning Want?" In: European Journal of Cultural Studies 18/4-5, pp. 429-45. https://doi.org/10.1177/1367549415577384.

— (2017): Machine Learners: Archaeology of a Data Practice, Cambridge: MIT Press.

— (2017): "Infrastructures in Name Only?: Identifying Effects of Depth and Scale." In: Harvey, Penny/Jensen, Casper Brunn/Morita, Atsuro (eds.), Infrastructures and Social Complexity: A Companion, London: Routledge.

— (2018): "48 Million Configurations and Counting: Platform Numbers and Their Capitalization." In: Journal of Cultural Economy 11/1, pp. 36-53. https://doi.org/10.1080/17530350.2017.1393443.

Marino, Mark C (2014): "Field Report for Critical Code Studies" In: Computational Culture: A Journal of Software Studies 4. http://computationalculture.net/field-report-for-critical-code-studies-2014%e2%80%a8/.

Microsoft (n.d.): "Democratizing AI." *Stories* (blog). Accessed September 5, 2018. https://news.microsoft.com/features/democratizing-ai/.

Moore, Todd (2018): "Think 2018: The Democratization of Artificial Intelligence." IBM Code (blog). March 20, 2018. https://developer.ibm.com/code/2018/03/20/think-2018-democratization-artificial-intelligence/.

Pichai, Sundar (2017): Google I/O'17 Keynote. https://www.youtube.com/watch?v=Y2VF8tmLFHw.

—(2018): Google I/O'18 Keynote. https://www.youtube.com/watch?v=QzbpXC0oxLo.

Schmarzo, William (2017): "Democratizing Artificial Intelligence, Deep Learning, Machine Learning with Dell EMC Ready Solutions." In: InFocus Blog | Dell EMC Services. Accessed June 7, 2019. https://infocus.dellemc.com/william_schmarzo/democratizing-artificial-intelligence-deep-learning-machine-learning-with-dell-emc-ready-solutions/.

scikit-learn (2019): "Scikit-Learn User Guide (Release 0.22.Dev0)." Accessed June 7, 2019. https://scikit-learn.org/dev/_downloads/scikit-learn-docs.pdf.

Simonite, Tom (2017): "Apple Just Joined Tech's Great Race to Democratize AI." WIRED. https://www.wired.com/2017/06/apple-siri-ai/.

Turing, Alan M. (1992): "Intelligent Machinery." In: Mechanical Intelligence, edited by Darrel C. Ince. Collected Works of A.M. Turing, Amsterdam: North-Holland, pp. 107-27.

Unity (n.d.): "Machine Learning." Unity. Accessed June 7, 2019. https://unity3d.com/machine-learning.

Wulf, Josh (2017): "9 Resources to Get Started Coding with JavaScript." Opensource.Com. Accessed June 7, 2019. https://opensource.com/article/17/6/get-started-coding-javascript.