Biases & Algorithms

Pedro Goulart

Hello,

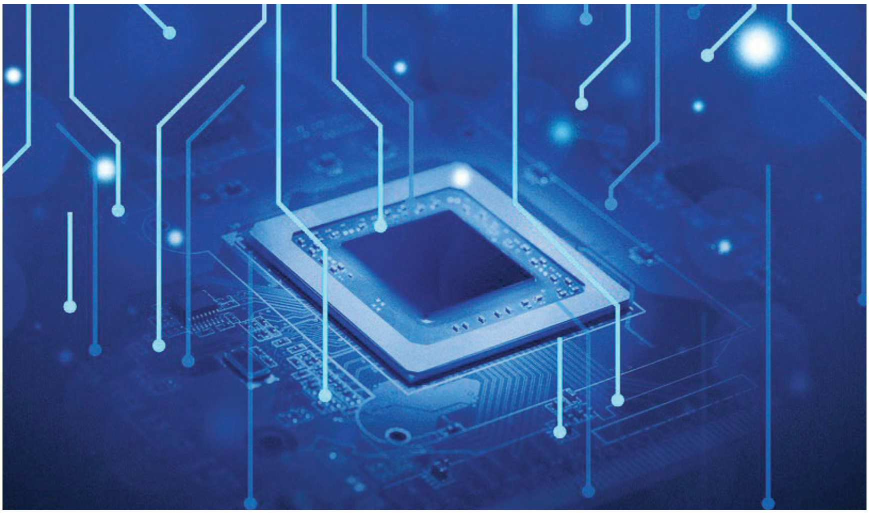
I AM PEDRO GOULART AND I'M THE AUTHOR OF THIS Zine. I am a Computer Science student at the University of Wisconsin - Madison, and I also have a certificate in digital studies, which aims at exploring the relationship between digital technologies and society. Furthermore, I am an international student from Brazil and currently the director of the Wisconsin Union Directorate Global Connections Committee, which aims at bridging the gap between international students and domestic students as well as raise awareness of different cultures through our student programming.

MY GOAL IN THIS ZINE IS DISCUSSING HOW THE underrepresentation of minorities in technology companies contributes to unintentional biases that carry to the design of algorithms. Those algorithms that have the power of affecting the lives of many and a fatal flaw product of biases can make a minority community even more underrepresented. This Zine further aims at giving the tools for leaders in the technology industry to remediate unintentional biases in algorithms.

Hope you like it!

Introduction: A Story of Power

TECHNOLOGICAL ADVANCES, SUCH AS THE INTERnet, have been frequently seen as instruments that approximate individuals and deconstruct prejudice. This has seldom been the case, however, since society tends to shape technology. According to Danah Boyd, the founder of the Data & Society Research Institute and a social media and technology scholar, technological advances do not create nor solve cultural divisions as people have previously come to believe[1]. Therefore, a better way to analyze technology is understanding it as a tool used by the society that mimics its behavior and existing social divisions. This means that if change does not spark from society itself, then no change is made at all.



UNDERSTANDING TECHNOLOGY AS A PRODUCT OF society allows us to better analyze how it tends to follow politics of power and mimic societal behavior. Social media is a good example of how society shapes technology because its content is mostly created by users; if you consider how information used to be often distributed by influential media companies in the 1800s, you might think that social media allows for a more representative and equal distribution of news sources. This, however, is unfortunately not the case.

AS POINTED BY MATTHEW HINDMAN, A PROFESSOR of media and public affairs at George Washington University and the author of the award-winning book The Myth of Digital Democracy, most-visited websites and consumed content tend to be those with an already existing offline presence, driven by elites[2]. Examples of the 100 most accessed websites include The New York Times, CNN, Fox News, Forbes, Vice, and many other well-known media outlets. This creates what is known as the 80-20 Rule, where about 20% of websites receive about 80% of worldwide internet traffic[2]. This shapes the information distributed in social media, and while it is true that user-generated stories (such as viral videos) can revert this relationship of power, the majority of the content created is still led by influential media companies. This exemplifies how new technologies, such as social media, reflect the politics of power that exists in society.

WHILE POLITICS OF POWER STILL SHAPE NEW technologies, another issue relating to user behavior also takes place. It is not difficult to find stories of people that have been oppressed on the internet such as Gamergate, a discussion centered around making the game industry more inclusive. The disparity between the sides of the debate could not be a better reflection of society, on one end there are mostly young white men who advocate for a gaming industry that continues to underrepresent and often misrepresent minorities and on the other side the rest of the people that believe that visibility and inclusion in the gaming industry is important[3]. This exemplifies a much broader cultural discussion around power politics and how technology is a product of its creators. The companies have, most likely unintentionally, underrepresented minorities in their games, which may have been caused by a lack of diversity in their development teams[3]. This has then affected a whole community of minorities that also used the technology. This further exemplifies how societal behavior, such as implicit biases, can get carried to the creation of new technologies, which then make them products of unintentional biases.

THIS LEADS US TO THE MAIN TOPIC OF THIS ZINE: How implicit biases are incorporated into new technologies, specifically those that require an algorithmic code. Now that we have defined how society shapes technology, we can better analyze examples of biases in technology and what might have contributed to that.

```
extract_num-
ber_and_incr (destination, source) int
*destination; unsigned char **source; {  extract_num-
ber (destination, *source); *source += 2; } #ifndef EXTRACT_MAC-
ROS #undef EXTRACT_NUMBER_AND_INCR #define EXTRACT_NUM-
BER_AND_INCR(dest, src) \ extract_number_and_incr (&dest, &src) #endif /*
not EXTRACT_MACROS */ #endif /* DEBUG */ ␌ /* If DEBUG is defined, Regex prints
many voluminous messages about what it is doing (if the variable `debug' is nonzero).  If
linked with the main program in `iregex.c', you can enter patterns and strings interactively.
And if linked with the main program in `main.c' and the other test files, you can run the al-
ready-written tests.  */ #ifdef DEBUG /* We use standard I/O for debugging.  */ #include <stdio.h>
/* It is useful to test things that ``must'' be true when debugging.  */ #include <assert.h> static int
debug = 0; #define DEBUG_STATEMENT(e) e #define DEBUG_PRINT1(x) if (debug) printf (x) #define
DEBUG_PRINT2(x1, x2) if (debug) printf (x1, x2) #define DEBUG_PRINT3(x1, x2, x3) if (debug) printf
(x1, x2, x3) #define DEBUG_PRINT4(x1, x2, x3, x4) if (debug) printf (x1, x2, x3, x4) #define DE-
BUG_PRINT_COMPILED_PATTERN(p, s, e)\ if (debug) print_partial_compiled_pattern (s, e) #define DE-
BUG_PRINT_DOUBLE_STRING(w, s1, sz1, s2, sz2)     \ if (debug) print_double_string (w, s1, sz1, s2, sz2)
extern void printchar(); /* Print the fastmap in human-readable form.  */ void print_fastmap (fastmap)
char *fastmap; { unsigned was_a_range = 0; unsigned i = 0;   while (i < (1 << BYTEWIDTH)) { if (fastmap[i++])
{ was_a_range = 0; printchar (i - 1); while (i < (1 << BYTEWIDTH)  &&  fastmap[i]) { was_a_range = 1; i++; } if
(was_a_range) { printf ("-"); printchar (i - 1); } } } putchar ('\n');  } /* Print a compiled pattern string in hu-
man-readable form, starting at the START pointer into it and ending just before the pointer END.  */ void
print_partial_compiled_pattern (start, end) unsigned char *start; unsigned char *end; { int mcnt, mcnt2; un-
signed char *p = start; unsigned char *pend = end; if (start == NULL) { printf ("(null)\n"); return; } /* Loop over
pattern commands.  */ while (p < pend) { switch ((re_opcode_t) *p++) { case no_op: printf ("/no_op");
break; case exactn: mcnt = *p++; printf ("/exactn/%d", mcnt); do { putchar ('/'); printchar (*p++); }
while (--mcnt); break; case start_memory: mcnt = *p++; printf ("/start_memory/%d/%d", mcnt,
*p++); break; case stop_memory: mcnt = *p++; printf ("/stop_memory/%d/%d", mcnt, *p++);
break; case duplicate: printf ("/duplicate/%d", *p++); break; case anychar: printf ("/anychar");
break; case charset: case charset_not: { register int c; printf ("/charset%s", (re_opcode_t) *(p -
1) == charset_not ? "_not" : ""); assert (p + *p < pend); for (c = 0; c < *p; c++) { unsigned bit;
unsigned char map_byte = p[1 + c]; putchar ('/'); for (bit = 0; bit < BYTEWIDTH; bit++) if
(map_byte & (1 << bit)) printchar (c * BYTEWIDTH + bit); } p += 1 + *p; break; } case beg-
line: printf ("/begline"); break; case endline: printf ("/endline"); break; case on_failure_-
jump: extract_number_and_incr (&mcnt, &p); printf ("/on_failure_jump/0/%d", mcnt);
break; case on_failure_keep_string_jump: extract_number_and_incr (&mcnt, &p); printf
("/on_failure_keep_string_jump/0/%d", mcnt); break; case dummy_failure_jump: ex-
tract_number_and_incr (&mcnt, &p); printf ("/dummy_failure_jump/0/%d", mcnt); break;
case push_dummy_failure: printf ("/push_dummy_failure"); break; case may-
be_pop_jump: extract_number_and_incr (&mcnt, &p); printf
("/maybe_pop_jump/0/%d", mcnt); break; case pop_failure_-
jump: extract_number_and_incr (&mcnt, &p); printf ("/pop_-
failure_jump/0/%d", mcnt); break;        case jump_past_alt:
extract_number_and_incr (&mcnt, &p); printf ("/-
```

Biases in Technology

BIASES IN THE TECHNOLOGY INDUSTRY AND algorithms come inherently from those that develop code: humans. While biases can be hard to overcome if we do not know about them, denying that humans may have unintentional biases is shutting off an extremely important conversation that can help people better identify biases and deconstruct them. The reason these unintentional biases are problematic is that they are commonly carried to the work environment and algorithmic code of applications.

ACCORDING TO YAËL EISENSTAT, A PAST CIA OFFIcer and former elections integrity operations head at Facebook, studies have shown that humans cannot fully avoid bias and denying it is counterproductive to find ing ways to combat biases[5]. While most of those biases are unintentional, not understanding how to counter biases in the technology industry can ultimately influence the workplace environment and lead to algorithmic bias.

A REPORT BY THE UNITED STATES EQUAL EMPLOYment opportunity commission has found that at bigger technology companies there is a serious issue of representation. Google and Facebook, for example, have about 1% of African-American individuals as part of their workforce, and the rest of the tech industry has a 1-2% average[6]. Furthermore, according to a study conducted by the Computing Research Association, out of the students that graduate with a bachelor's degrees in computer science, only 4.6% of them are African-Americans[6]. These figures exemplify the underrepresentation of minorities in technological companies and academia. Facebook, for example, used the fact that there were not a lot of African-American students graduating in Computer Science as the reason for their shortfall in hiring a more diverse group of people.

THE REPORT BY THE COMPUTING RESEARCH ASSOciation also verified that about 15.7% of Computer Science graduates were women, compared to 37% around 1985[6]. Maja Mataric, a professor of computer science at the University of Southern California, believes this decrease may have been caused due to the culture of male-dominated departments, which in some instances pushed women away or made them comply and sustain the male-dominance[6]. This shows that there is likely a deeper explanation for the lack of diversity in technology companies other than the subset of graduating students in computer science not being diverse enough.

WHILE THERE IS NOT ENOUGH SUPPORTING REsearch that analyzes possible biases by technology industry contractors, it is, given the data we have discussed, a possibility that cannot be discarded. These disparities in the workplace contribute to less diverse environments which then can contribute to algorithmic bias.

ACCORDING TO MIT GRAD STUDENT JOY BUOLAMwini, algorithmic biases can spread rapidly due to their nature, creating exclusionary experiences and discriminatory practices[4]. Joy experienced algorithmic bias when she downloaded an open source generic code for facial recognition which only recognized lighter skin tones. This issue repeated itself when she visited a start-up in Hong Kong that was using the same open source code that could not recognize her face. Based on how facial recognition works, the reason for this issue would be on a biased training set (which consist of examples created by programmers to train their algorithm in recognizing faces)[4]. A training set that is not very diverse can produce a biased algorithm that cannot recognize certain faces, thus exemplifying how unintentional biases can carry to software development and affect a large number of individuals.

ANOTHER EXAMPLE OF ALGORITHMIC INEQUALity is with an Artificial Intelligence recruiting technology created by Amazon. The software developed biases against women's resumes because the training set was mostly given men's resumes[7]. While Amazon ultimately removed the project, it still serves as a strong example of the consequences of a biased training set.

GOOGLE HAS ALSO HAD A FEW ISSUES WITH ITS own recognition and recommendation software. In 2015, Google Photos categorized an African-American user's pictures of herself and a friend as gorillas[7]. On another instance, YouTube appeared to be recommending radicalizing far-right content on to casual viewers[6]. Moreover, Google has been accused of having image search and autocomplete algorithms that rely on sexist and racist stereotypes [7]. These examples show that algorithmic bias can affect a myriad of services and users.

WHILE MOST OF THE EXAMPLES DISCUSSED HAVE focused on algorithmic biases that were likely unintentional, most of the times they appear as a reflection of a development team that is not equiped to analyze and deal with biases. When attempting to measure biases on Facebook's political advertisement policies for the 2018 election campaign, Yaël verified that there were problems with the team she was leading; essentially employees were not sure how to analyze biases and mischaracterized advertisements that would feed an artificial intelligence algorithm[5]. While there are steps being taken to reduce discrimination in training sets, it is also important to consider that humans themselves are the ones developing those algorithms and training sets.

ACCORDING TO JOY BUOLAMWINI, ALGORITHMS are starting to be used in many fields such as security, healthcare, and education, which means algorithmic bias can affect an even greater number of individuals. Therefore, action is needed to deconstruct biases.

Taking Action

Not just a necessity
but a responsibility

TO PREVENT ALGORITHMIC BIASES, THERE ARE A few steps that can create more inclusive code and employ more inclusive coding practices. According to Buolamwini, there are three major areas that we can focus on. Essentially, it all starts with people.

WHO CODES MATTER

As discussed previously, it is essential to have a diverse group of individuals on a team. According to Buolamwini, this for better checking of each other's blind spots where it related to biases, especially in training sets of data that influence machine learning[4]. Moreover, when there are more people coming from diverse backgrounds, more opinions tend to be represented.

HOW WE CODE MATTERS

Fairness is a critical factor that must be included when designing algorithms. Ensuring that the code created is not inherently biased to find a specific result or behave in a certain way that could exclude other individuals, or in other words, ensuring the algorithm gives fair results, is imperative to prevent algorithmic bias.

WHY WE CODE MATTERS

A priority should be placed in social change. Computational creation, according to Buolamwini, has produced great prosperity to humans, and using this potential to create more equal environments can contribute to giving even greater power to humans[4]. It is imperative, therefore, to consider social change a priority and not an afterthought.

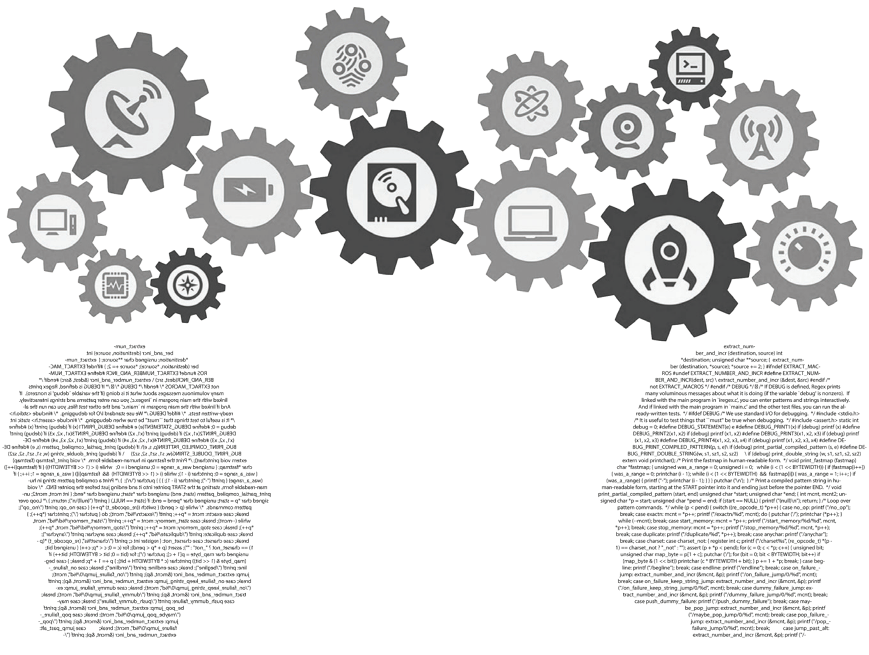
THESE THREE AREAS MAKE IT CLEAR THAT THE problem of algorithmic biases is directly related to the problem of diversity in the technology industry. It is not, as some might have argued, dependent solely in those that develop algorithms, but rather on the company team as a whole. Moreover, the idea of focusing on people rather than the data may also be unfamiliar or even counter-productive to some, however, it is important to remember where the biases come from.

FOCUSING ON MAKING THE DATA LESS BIASED rather than the team that made the algorithm or provided the data is one of the first steps that companies might take, though it is not effective. Becoming reliant on data is, according to Eisentat, a product of bias in itself[5]. The problem of trusting data and trying to make it less biased is that it often does not employ the necessary critical thinking to ensure that, in doing so, you are not influenced by confirmation, pattern, or other cognitive biases[5]. This means that individuals that do not understand their own biases will not truly make the data less biased.

HUMANS RELY ON COGNITIVE BIASES TO MAKE decisions, and so it becomes challenging to analyze them. Understanding your own biases and challenging them is, according to Eisentat, one of the most important steps in order to mitigate algorithmic biases[5]. It is, however, not an easy task and companies should encourage its employees to learn about their own biases through workshops or group discussions. Nevertheless, it should not be forced since people tend to reinforce their own biases if they feel threatened.

EDUCATING PEOPLE TO UNDERSTAND THEIR OWN biases requires a continued effort from all fields. Experts and insiders believe that efforts to improve diversity in the technology industry must be supported by different fields such as academia and universities, companies and nonprofits, and others[6]. The impact of such, according to Juan Gilbert, a researcher and educator in computer science, diverse teams provide better solutions to problems, and increasing participation is essential for countries to become even greater.

WHILE IT IS IMPORTANT FOR COMPANIES TO FOCUS on deconstructing biases on people, it may take a long time before the algorithms are revised. A platform created by Joy Buolamwini called The Algorithmic Justice League (available on **ajlunited.org**) can identify bias in algorithms and audit current software[4]. The platform helps identify bias, create more inclusive tests, and helps think about the social impact of the technology being developed[4]. This means that companies can focus on deconstructing biases from their employees and from their algorithmic code at the same time, which drastically reduces the impact of algorithmic bias and prevents future biases from appearing.



Final Remarks

THROUGH THIS ZINE, I HOPE YOU WERE ABLE TO better understand the issue of algorithmic bias and how unintentional biases in the technology industry can reflect on the lives of countless individuals. I also hope that you learned how important it is to focus on deconstructing biases from people, having a more diverse team of individuals in the industry, and using platforms to check for possible biases in algorithmic code or training sets can be helpful in preventing this issue.

EVEN THOUGH I FOCUSED ON THE TECHNOLOGY industry and the biases in algorithms, this issue is present in many workplaces throughout the world, and it is possible to utilize the steps listed on this zine to deconstruct biases. From my own experience, I have noticed that speaking up on a team can be very beneficial for the learning of others and so I would encourage you to be the spark of change and contribute to this conversation on biases. I believe that knowledge is one of the most important tools to prevent biases.

* * *

References

1. Boyd, D. (2014). *It's complicated: the social lives of networked teens*. New Haven, CT. Yale University Press.

2. Hindman, M. (2009). *The myth of digital democracy*. Princeton, NJ. Princeton University Press.

3. Dewey, C. (2014, October 14). The only guide to Gamergate you will ever need to read. *The Washington Post*. Retrieved from: http://washingtonpost.com

4. Buolamwini, J. (2016, November). How I'm fightingias in algorithms [Video file]. Retrieved from http://ted.com

5. Eisentat, Y. (2019, February 2). The real realson tech struggles with algorithmc bias. *Wired*. Retrieved from http://wired.com

6. Mone, G. (2017, January). Bias in technology. *Communications of the ACM*. 60 (1), 19-20. doi: 10.1145/3014388

7. Hicks, M. (2018, October 12). Why tech's gender problem is nothing new. *The Guardian*. Retrieved from: http://theguardian.com

Queer Emerging Leaders Program

University of Wisconsin - Madison