The Mathematics of Misinformation

Noah Giansiracusa

Introduction

Last year I wrote a book, aimed at a general audience, that explores how data-driven algorithms have impacted the news industry and our ability to separate fact from fiction [6].[1] This article zeros in on, and amplifies, some of the more mathematical aspects of that story in what I hope will be both informative and engaging to a mathematical audience. As you'll soon see, there are many fun ingredients at play here, ranging from elementary notions (fractions, linear functions, and weighted sums) to intermediate level concepts (eigenvalues and Shannon information) to sophisticated uses of probability theory, network analysis, and deep learning.

This topic of information and misinformation is complex, multifaceted, and interdisciplinary, and in my opinion more mathematicians should try to enter the public discussions surrounding it and undertake research related to it. I believe we, the math community, can do for misinformation what we have been doing for topics like gerrymandering, where mathematicians have assisted policy makers and helped shape the discourse while also discovering marvelous math topics to explore [9]. I hope this article helps launch some readers down this path, and I would be happy for interested individuals to reach out to me on this.

Supervised Learning

Not all of the mathematical aspects of misinformation discussed here involve machine learning, but many of them do, so let's start with a very quick review. The basic idea of supervised learning is to assume that a *target* variable y depends on a collection of *predictor* variables $x_1, \ldots, x_p$ in some mostly deterministic way that can be deduced from the data. When the target variable is $\mathbb{R}$-valued this is called *regression*, whereas when it takes values in some finite set this is called *classification*. When the predictor variables are $\mathbb{R}$-valued, regression takes the form

$$y = f_{\vec{w}}(x_1, \ldots, x_p),$$

Noah Giansiracusa is an assistant professor of mathematical sciences at Bentley University. His email address is ngiansiracusa@bentley.edu.

DOI: https://doi.org/10.1090/noti2566

[1]*For more on the book and my writings related to it, see* www.noahgian.com.

where $f_{\vec{w}} : \mathbb{R}^p \to \mathbb{R}$ is a function that depends on a potentially large number of parameters $\vec{w} \in \mathbb{R}^N$. The underlying process may be more complex than this functional relationship—it might even be that different values of the target variable are observed for the same values of the predictor variables—but this simple setup is good enough to make useful predictions in most situations.

Training data refers to a set S of observed values of the predictors and target,

$$\{(x_{1,\alpha}, \dots, x_{p,\alpha}, y_\alpha)\}_{\alpha \in S} \subseteq \mathbb{R}^{p+1}.$$

For regression, this is used to adjust the parameters $\vec{w}$ to minimize the difference between the *predicted* value $f_{\vec{w}}(x_{1,\alpha}, \dots, x_{p,\alpha})$ and the *actual* value y_α (or some cost function applied to this difference) when α ranges over a set disjoint from S called the *test* data; for classification we try to minimize the misclassification rate (fraction of predictions that are incorrect) on the test data, or some other error measurement derived from the confusion matrix (the matrix whose ij entry records the number of data points of class i that were predicted to have class j).

The simplest example is *linear regression*, in which

$$f_{\vec{w}}(x_1, \dots, x_p) = w_0 + w_1 x_1 + \cdots + w_p x_p.$$

A *neural network* is an extension of this where instead of requiring f to be linear, we let f be a composition of linear and non-linear functions of a certain type: f is a composition of any number of pairs consisting of a vector-valued linear function followed by the component-wise application of the *activation function* (typically the sigmoid function or the piecewise linear function that is 0 on negative numbers and the identity on non-negative numbers). Each such pair is called a *layer* in the network, and when the number of layers is ≥ 2 this form of machine learning is called *deep* learning. In contrast to linear regression, the number of parameters N in a neural network is usually much larger than the number of predictors p, and the parameters are very difficult to interpret. Much more could be said, but this is enough for what follows. For more on deep learning, I recommend the book *Deep Learning* [8].

Text Generation

You've likely heard about troll farms cranking out fake news in the lead-up to the 2016 US presidential election. One concern surrounding the powerful advances in deep learning in recent years is that people could use neural networks to instantly and inexpensively generate an unlimited number of fake news articles. These fears reached a crescendo (or at least, a local maximum) in 2020 when a deep learning text generation system called *GPT-3* was released.

Here's the supervised learning task GPT-3 was trained on. A data point is a passage of n words, the predictors are the first $n-1$ words, and the target is the n^{th} word. For a document with $m \geq n$ words, a window of n consecutive words is slid across, one word at a time, to produce $m-n+1$ training points:

Input: "The dog likes to play fetch with me," $n = 6$

1st prediction: The dog likes to play ???

2nd prediction: dog likes to play fetch ???

3rd prediction: likes to play fetch with ???

GPT-3 "read" in this manner a massive number of webpages and scanned books. When the prediction was correct it moved on to the next, whereas when it was incorrect the neural network parameters were adjusted based on the prediction error. To give a sense of the enormous scale, all of Wikipedia was included yet comprised only 0.5% of the training set.

When training a neural network to, say, recognize traffic lights in photos, humans need to manually indicate which training photos have traffic lights and which don't.[2] In contrast, no manual data labeling was necessary for GPT-3's training task because both the predictor variable values *and* the target variable values come directly from the input text. This is called *self*-supervised learning, and it is crucial for scaling up the training process to the magnitude needed for convincing text generation.

The computational resources to train GPT-3 cost about \$10 million. That's a lot, but the neural network need only be trained once—after that, it can be used to generate text by feeding it a passage of text and having it predict words one at a time. For example, one can feed GPT-3 the headline to a (possibly made-up) news story and GPT-3 will extend it to a full article. You don't need to know computer programming or neural networks to use GPT-3—you just type/paste your prompt into a webpage then click a button. The price to generate 10,000 words is about one penny. Let's give it a try:

Headline (user prompt): *Mathematician solves famed millennium problem.*

Article (GPT-3 text): *A mathematician has solved a famed millennium problem that experts said had "defied all attempts" at a solution for more than 500 years. The problem, known as the ABC conjecture, deals with the relationship between whole numbers and prime numbers. The conjecture was first proposed by French mathematicians in the early 19th century, but it was not until 2012 that Japanese mathematician Shinichi Mochizuki claimed to have solved it. However, his proof was deemed too complex to be verified by his peers, leading some to doubt its validity. Now, after eight years of work, British mathematician Sir Andrew Wiles has published a paper in which he provides a new and simpler proof of the conjecture. If the*

[2] *That's why this is a common task for captcha systems: your answers provide the training data for a supervised learning system, presumably one related to self-driving cars.*

proof is verified, it could have major implications for the field of number theory, as well as for cryptography.

This sounds plausible enough that I Googled just now to see if this really happened and I missed the news! No, it did not—GPT-3 succeeded in creating superficially convincing mathematical misinformation. However, there are some humorous defects here—did you catch any? The ABC conjecture is *not* a millennium problem (although, if you're like me, you had to consult the list to double-check this), and if it was first proposed in the 19th century (it wasn't, by the way—it was the late 20th century, and only one of the two mathematicians who proposed it is French) then how has it defied a solution for 500 years? In general, GPT-3 output has a loose relationship with reality; nowhere in its training process is truth prioritized over falsehood. Getting GPT-3-type systems to produce more factual output is an active area of research [1, 10, 13].

GPT-3 doesn't just predict the most likely word to follow a passage of text—it estimates a probability distribution on the possible words. (For instance, for the prompt "The dog likes to play..." it might estimate that the next word is "fetch" with 50% probability, "chase" with 25% probability, and "rough" with 25% probability.) One option is to always choose the most probable word, but then you can only get one completion for each prompt,[3] so more common is to sample from this distribution—so that the most likely word is selected most often, but rarer words will be selected sometimes. In practice, people usually do this and produce several GPT-3 completions for a prompt then manually select the most convincing/useful of them. This is what I did with the above millennium problem prompt—some of the GPT-3 completions were not nearly as convincing or interesting as the one I included here, and many essentially just recycled old press releases from when Perelman solved the Poincaré conjecture.

One method proposed to detect text generated by systems like GPT-3 is to use such a system to score the probability of each word in the text and see whether low probability words appear at a disproportionately high rate (if they do, this suggests that the text is organic) [16]. This works reasonably well when the system used to generate the text is very similar to the system used to score the probability of the words. But there are many variants of GPT-3 available now, with many more appearing routinely, so this detection method is not too practical.

Thus far, GPT-3 has not led to the flood of fake news that some people expected—perhaps because the bottleneck is not writing fake news, it is writing fake news that will go viral, and at least so far that requires more of a human touch (additionally, most fake news is low quality and cheap to write manually anyway). But GPT-3 does raise the prospect of troll farms mass producing fake news and running large-scale experiments to study virality empirically with unprecedented precision and scale. Moreover, GPT-3 can be used to power more humanlike automated bot accounts on social media—and bots routinely play a large role in disinformation campaigns since they can create artificial engagement on social media posts (more on this later).

Deepfakes

One month into Russia's invasion of Ukraine, a deepfake video of President Zelensky circulated online in which he tells Ukrainians to lay down their arms and surrender. Fortunately the video was low-quality and didn't deceive many people, and Zelensky himself addressed it almost immediately clarifying that it was fake. To hear more about this incident and the history and possible future of deepfake videos in politics you can check out a recent *Slate* podcast episode[4] I was on; here in this article I will focus instead on the mathematical question of how deepfakes are made.

There are many different types of deepfake videos (face swap, lip sync, puppeteering, speech synthesis, etc.) and many different neural network architectures involved, but here I'll just give a flavor of the topic by discussing one particular form (see [11] for more) that relies on my favorite concept in all of deep learning: the autoencoder. I'll also discuss a distinct use of the term "deepfake" in which a neural network doesn't just edit faces, it creates new ones from scratch (though so far this latter technique applies only to photos, not videos).

Autoencoders. Recall that in self-supervised learning the values of the target variable are inferred directly from the data in an automatic fashion rather than entered laboriously by hand. One ingenious form of self-supervised deep learning is the *autoencoder*. This is a neural network that learns to compress data by passing it through a low-dimensional space. What's so slick about this is that usually the goal isn't the compression itself; compressing data is merely a trick to encourage the network to find meaningful patterns in the data.

Here's how it works. An autoencoder learns to approximate the identity function $\mathrm{id} : \mathbb{R}^p \to \mathbb{R}^p$. That is, the output is a p-dimensional vector $\vec{y}$ and it is trained by setting $\vec{y} = \vec{x}$, meaning the target values for each training point are by fiat equal to the predictor values. The key is that the autoencoder consists of a sequence of progressively narrowing layers followed symmetrically by a sequence of progressively widening layers—meaning it decomposes into a

[3] *Another amusing issue with always selecting the most probable word is that this is the* least informative *word, in the sense of Shannon's information theory: the information of an event with probability p is* $\log(\frac{1}{p})$, *so the more probable a word is, the less informative it is.*

[4] *https://slate.com/podcasts/what-next-tbd/2022/03/why-the-zelensky-deepfake-failed.*

composition

$$\underbrace{\mathbb{R}^p \to \mathbb{R}^{p_1} \to \cdots \to}_{\text{encoder}} \mathbb{R}^{p_\ell} \underbrace{\to \cdots \to \mathbb{R}^{p_1} \to \mathbb{R}^p}_{\text{decoder}},$$

where $p > p_1 > \cdots > p_\ell$. The result is that the neural network must learn from the training data how to represent the original p-dimensional data in a much smaller p_ℓ-dimensional space.

You can think of this like zipping a file—except rather than a human programmer specifying the compression algorithm, the neural network figures one out on its own. Indeed, passing the data points from the original $\mathbb{R}^p$ down to $\mathbb{R}^{p_\ell}$ is like zipping, then passing them back to the final $\mathbb{R}^p$ is unzipping (the official parlance is *encoding* and *decoding*), and the supervised learning task means the output should resemble the input as closely as possible. By the magic of deep learning, the neural network typically uncovers meaningful multi-scale structure in the data in order to do this, and the structure it uncovers and exploits is specific to the type of data it is exposed to.

If the training data set is a collection of 100×100 images of different human faces, then typically $p = 30{,}000$ as an image is represented by the three RGB intensity values for each pixel, and if, say, $p_\ell = 50$, then the neural network must find a way of encoding each face using only 50 numbers. An overly anthropomorphic version of this would be to describe each face with numbers representing things like the person's age, skin tone, hair color, hair style, the shape of their face, etc. In reality, the features the neural network learns are not nearly so recognizable to our human minds, but they are nonetheless a distillation of the raw pixel values into larger-scale structures.

If the training set consists of many images of one individual's face, rather than images of many different individuals, then the neural network doesn't need to encode things like hair color and skin tone and the shape of the face, so it might instead develop features recording things like the angle of the face and the extent to which the mouth is open, the lips are smiling, and the eyebrows are raised (this description is continuing our overly anthropomorphized version of what actually happens). That is, the neural network focuses less on distinguishing different individuals and more on distinguishing different expressions on the particular individual. This is the key behind the autoencoder's use in the face swap deepfake.

Face swap. Suppose the goal is to swap person A's face onto a movie of person B. The problem reduces from movies to still images by working one frame at a time, and locating faces in an image is a standard task (usually solved by deep learning), so we're reduced to the following: we want to transform an image of face B into an image of face A—but when doing so, we need the new face A to take on the orientation and expression exhibited in the given image of face B (see Figure 1). Here's the autoencoder approach.

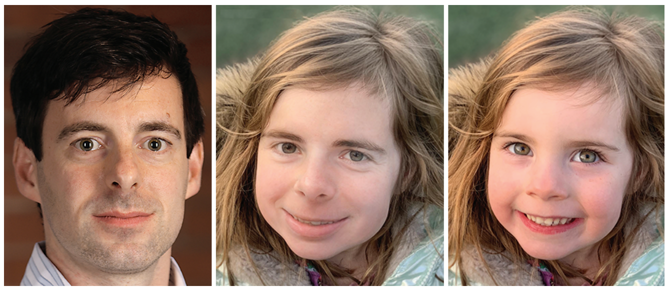

Figure 1. Illustration of a face swap.

We typically think of an autoencoder as comprising an encoder and a decoder that are trained in tandem. For the face swap, we'll use two autoencoders (one trained on images of face A, one on images of face B), but we'll force them to share the same encoder: during the training process, whenever the parameters in the encoding portion of autoencoder A are updated, the corresponding parameters in autoencoder B are forcibly updated in the exact same manner, and vice versa. This way, the low-dimensional features that are developed for encoding both faces will agree: if the first number measures how much face A is smiling, then it will also measure how much face B is smiling. Then all we do is encode face B (with either encoder, since they're the same) and then decode it with the decoder from autoencoder A. This computes all the salient properties of face B's expression then produces an image of face A with the same properties. If B was looking to the left with a big smile, then we'll get A's face looking to the left with a big smile. Doing this frame-by-frame is the cleverly elegant autoencoder approach to face swap deepfakes.

Deepfake profile photos. The term "deepfake" has two distinct meanings: (1) any form of video editing based on neural networks (this is what we have discussed so far), and (2) synthesizing lifelike photos of non-existing human faces. Unsurprisingly, the second form of deepfake has also been weaponized for disinformation. For instance, in September 2020 both Facebook and Twitter announced that they had uncovered a group of coordinated inauthentic accounts spreading anti-Biden disinformation; the accounts all used deepfake profile photos and were masquerading as American users, when in reality they were fake personas operated by the Russian government. Interestingly, what tipped Facebook and Twitter off to these accounts wasn't their use of deepfake photos, it was the network structure of their friends/followers/actions—a topic we'll return to later in this article.

The reason nefarious actors use deepfake photos to hide their tracks instead of just grabbing random profile pics off the internet is because the latter are easily uncovered by a "reverse image search." If you drag and drop an image file onto Google's search bar, then Google will scour the web for similar images. So if you use someone else's photo for your profile, it's easy to find the original source, thereby revealing the deception; if you use a deepfake photo, then there is no original source to find.

The math behind reverse image search is quite cool. A standard method is the following: (1) train a large autoencoder on a huge database of photos then encode all the images on the web as well as the input image; (2) the vectors for the former that are closest in some metric to that of the latter are the most similar images. Computing distances directly on the original images, viewed as matrices, does not work well because even minor modifications (cropping, rotation, translation, adjustment of the color palette, etc.) usually result in very distant image matrices. The low-dimensional representations autoencoders develop tend to capture the spirit of the image, so those kinds of minor modifications typically have little if any impact on the encoded version of the image. Morally speaking, encoding with a suitable autoencoder introduces a sort of continuity with respect to basic image manipulations that does not exist at the raw pixel level.

Deepfake photos are all created by some refinement of a general method called a *generative adversarial network*, or *GAN* for short. The idea is to pit two neural networks against each other: the *generator* tries to create new data points that look similar to the ones in the training set, while the *discriminator* tries to determine which data points are real training data and which are fake ones created by the generator. Both networks start out terrible at their respective jobs but then throughout the training process they push each other to steadily improve. It is a marvelous idea, though one should be careful about leaning too heavily on an intuitive view of this duel: the two networks improve and "learn" throughout the training process, but what and how they learn does not typically align with how human minds learn.

There are some interesting math questions surrounding GANs. For instance, this dueling neural networks setup can be viewed in game-theoretic terms, and a recent paper showed that GANs do not always have Nash equilibria but they do have a different form of zero-sum game equilibrium [3]. Since GANs can be used to generate new data points that resemble the data points in any setting, they have myriad applications, such as augmenting small training sets for supervised learning tasks; deepfake profile photo generation is when one applies GANs to a training set consisting of photos of human faces.[5]

[5]*Think you can tell deepfakes from real images? Try visiting the website* `https://www.whichfaceisreal.com/`.

Facebook's News Feed

For years it has been believed that Facebook's News Feed algorithm is responsible for an unsettling amount of the viral spread of misinformation online. Awareness of this issue came to the fore and was sharply clarified by the trove of internal documents released by Facebook employee-turned-whistleblower Frances Haugen in the fall of 2021. The basic problem is that Facebook algorithmically ranks the order of the posts seen by all of its 3 billion users, and it does this primarily by pushing the posts that receive the most *engagement* (likes, shares, comments, emoji reactions, etc.) to the top—but human psychology is such that there's a correlation between the posts we engage with the most and posts that are divisive, offensive, and misinformative. To better understand this problem, let's dig into the math behind Facebook's News Feed algorithm.

The math behind the algorithm. The details of Facebook's algorithm are kept closely guarded, but the broad strokes were outlined in a company blog post. Each user has a set of potential posts they could be shown (the posts by their friends, the pages they follow, the groups they're in, etc.). Let's fix a user and a moment in time. Facebook uses deep learning to predict the probability that this user will like, share, short comment on, long comment on, etc., each post in this set. Also predicted is the probability of each post violating a platform policy (Facebook prohibits hate speech, incitement to violence, and certain specific forms of misinformation). All these probabilities—call them $q_1, \dots, q_r$—are aggregated into a single number, called the post's *value* v (to the given user at the given moment in time), by taking a weighted sum of the probabilities: $v = \sum w_i q_i$. The weights w_i don't just depend on the type of engagement i, they also depend on a variety of factors such as how close the poster is believed to be to the user (posts by closer friends are given more value) and the category of the post (Facebook has at times temporarily lowered the weight on political posts). After a few additional tweaks, Facebook orders your News Feed posts by these value scores v.

Unsurprisingly, the weights on the engagement probabilities are positive whereas the weight on the policy violation probability is negative. Actually, the weight of the angry reaction was originally set to 5 times the weight of a like, but it was eventually lowered to 0—because it turns out angry reactions correlate with a lot of bad things, like politically polarizing content and misinformation.

In a recent *Boston Globe* opinion article [7], I encouraged Facebook to create a control panel where users can see and adjust all these weights. It's unclear what impact this would have on things like misinformation in the aggregate, but at least it would allow users to customize their individual experiences—which is important since not everyone is impacted by the different forms of dangerous content equally. For instance, users in marginalized populations that are frequently subject to online harassment may want to weight the policy violation probability much more heavily. I also argue that congress should require all large online platforms using data-driven algorithms for ranking information to allow users to toggle on/off the various data sources the algorithms rely on. Lawyers may call this data privacy and transparency; as a mathematician, I call it letting people choose the inputs to the functions that determine what we see online.

The internal Facebook research leaked by Haugen revealed some fascinating statistics about harmful content—and how to potentially reduce it. Civic content classified as toxic receives twice as many haha reactions and 33% more angry reactions than heart reactions (this discovery factored in to Facebook's decision to drop the angry weight to zero). Comments with heart reactions are 15 times less likely to be policy violations. Reshare *depth* refers to the length of chains of reshares (e.g., if A posts something then A's friend B sees it and reshares then B's friend C sees this reshare and reshares it, this is depth 2 since it is 2 steps away from the original poster)—and it was found that strongly down-weighting posts based on reshare depth would reduce civic misinformation in the form of links by about 25% and in the form of photos by about 50%. Another interesting mathematical detail revealed in the leaked documents deserves its own subsection—so let's have at it.

Deceptive denominators. In numerous congressional hearings and transparency reports, Mark Zuckerberg and other Facebook officials have touted the success of their machine learning approach to automatically detecting and removing hate speech, repeatedly citing a success rate around 94%. But upon closer inspection, they never asserted that 94% of hate speech is taken down—what they asserted is that among the hate speech that Facebook takes off its platform, 94% of it was detected algorithmically (the rest was flagged manually by users). So how much of the total hate speech on the platform does Facebook manage to take down?

Leaked internal documents revealed that the figure is only around 3–5%, and in some locations it is as low as half a percent. Facebook never outright lied about this, but it acted deceptively by routinely providing the impressively high percentage while keeping the shockingly low percentage secret—even when members of Congress directly asked how successful Facebook is at removing hate speech. I discussed this in an article for *Wired* [5] that was illustrated nicely by AMS Vice President Francis Su (see Figure 2). The punchline is that denominators really matter: when looking only at the hate speech that Facebook takes down, their machine learning algorithms work very well—but when looking at all hate speech on the platform, they are, alas, terrible.

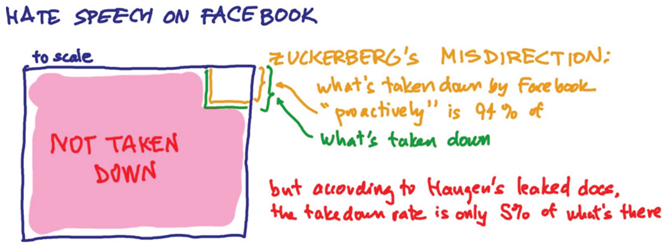
HATE SPEECH ON FACEBOOK
to scale
NOT TAKEN DOWN
ZUCKERBERG'S MISDIRECTION:
what's taken down by Facebook "proactively" is 94% of
what's taken down
but according to Haugen's leaked docs, the takedown rate is only 5% of what's there

Figure 2. Illustration by Francis Su of Facebook's deceptive measurement of hate speech.

That said, Facebook was quick to point out that their content moderation is not a binary leave up/take down decision. As you recall, each post has an estimated probability of violating a policy (such as hate speech); when this probability is above a high threshold the post is taken down automatically, but even when this threshold is not reached the post is down-ranked by the News Feed algorithm according to this probability. Facebook rightly pointed out[6] that in recent years they have used this approach quite effectively to reduce the *prevalence* of hate speech—defined as the fraction of hate speech posts users see on average relative to all posts they see. Down-ranking posts reduces their prevalence without reducing the total amount of hate speech on the platform.

It has proven much harder to measure how much misinformation is on platforms like Facebook and how much of it is taken down by various forms of moderation—in large part because there is no widely accepted definition of misinformation.

Google Rankings

Days after the 2016 US presidential election, the top link in a Google search for "final election results" was a low-quality WordPress blog falsely asserting that Trump had won the popular vote. This was just one very notable instance of misinformation climbing the ranks of Google searches. Many other instances had been noted in the lead-up to the election, and people began to wonder whether this had played a role in Trump's victory—a victory that caught most experts by surprise. Shortly after the election, Google's CEO Sundar Pichai was asked not just whether viral misinformation (or "fake news," as it was called then) might have played a role in the election, but whether it might have played a *decisive* role (meaning that it's impact was enough to swing the electoral college victory from Clinton to Trump), and this was his response: "Sure. You know, I think fake news as a whole could be an issue."

How did a junk blog post and other trashy news sources climb to the top of Google's search rankings? To answer this, we need to step back to the origins of Google—and the linear algebra underlying it.

[6] *https://about.fb.com/news/2021/10/hate-speech-prevalence-dropped-facebook/.*

Let's start with a slightly different question: How should one measure a user's importance in a social media network? To make things concrete, let's consider Twitter, which forms a directed graph in which users are vertices and an edge from U to U' means U follows U'. The most obvious measure of influence of a vertex is its in-degree, which here is a user's number of followers. But the expected number of users your tweets will reach doesn't just depend on your number of followers—it also depends on the number of followers your followers have, and the number of followers they have, etc. So we'd like a deeper way of measuring influence in a network.

Given a graph with vertices v_i, one would like to assign non-negative numbers x_i to the vertices such that the number on each vertex is proportional to the numbers on its neighbors:

$$x_i = c\sum_j a_{ij}x_j,$$

where a_{ij} are the entries of the adjacency matrix A. This is the matrix equation $\vec{x} = cA\vec{x}$, so it is asking for $\vec{x}$ to be an eigenvector (with eigenvalue $\frac{1}{c}$). Since we are only interested in non-negative numbers here, by the Perron-Frobenius Theorem there is a unique (up to scaling) solution $\vec{x}$. These vertex numbers x_i are called the *eigenvector centrality* scores for the graph. They have a beautiful random walk interpretation: when starting at random vertices in the graph and taking steps to neighboring vertices with uniform probability, the eigenvector centrality scores are proportional to the fraction of time spent at each vertex. Twitter publicly lists the follower counts on all accounts (Barack Obama currently holds the #1 spot, followed by Justin Bieber in #2 and Katy Perry in #3); I wish Twitter also listed the the eigenvector centrality scores so users could readily see which users are the most influential in this deeper sense.

The world wide web can be viewed as a directed network in which webpages are the vertices and links are represented by directed edges. Keeping in mind that people sometimes navigate the web by typing in a URL instead of clicking a link, a modified random walk process is to fix a probability p and then at each step with probability p walk to one of the neighboring vertices, as before, but with probability $1-p$ jump directly to any vertex in the graph (chosen with uniform probability). The fraction of time spent at each vertex in this modified random walk process still has an eigenvector interpretation, and—more importantly for us—it was the original method Google used to rank search results; it is called *PageRank*. The name is a bit of wordplay: it refers both to its application in ranking web*page*s and to Google co-cofounder Larry *Page*.[7]

The problem with using a mathematical formula like PageRank for ranking search results is that over time people learn how to game the system—and even without overt attempts to game rankings, there's nothing preventing fake news publishers and other forms of misinformation from rising to the top. This is dangerous since people typically think that if something shows up at the top of Google then it's true.

Coordinated disinformation campaigns, sometimes organized by foreign governments, were able to land propaganda high on Google search rankings by building large networks of sites that linked to each other and that drew additional links from popular far-right "news" sites. And when a popular but low-quality site like Breitbart linked to, say, a lowly blog post about the election results, suddenly that blog post catapulted up in the search rankings.

Over time—and especially during a concerted and continuing push following 2016 to "elevate quality journalism"—Google has developed additional signals that go into search rankings. Google now uses an army of 30,000 low-paid contract workers who manually evaluate search ranking results according to a 168-page instruction manual they are provided, and their by-hand rankings form the training data for machine learning algorithms. Google has been using these manually trained machine learning signals together with the original PageRank measure together with additional factors, such as direct assessments of the quality of news sources. We don't know too much more than that because the details are kept secret.

The punchline to this story is that Google has not solved its misinformation problem but it has made tremendous strides on it since 2016, largely by admitting that a purely mathematical ranking system is insufficient and human insight is needed to make sure fake news does not rise to the top.

Network Dynamics

A landmark paper on the spread of misinformation was published in *Science* in 2018 [14]. By studying the propagation of over 100,000 stories across Twitter over a 10-year span, it found that false stories traveled faster and further than true stories:

- False stories reached 1,500 people six times faster than true stories.
- Even when controlling for various differences between the original posters, such as their number of followers and whether their account was verified by Twitter, false stories were 70% more likely to get retweeted than true stories.

[7]*Incidentally, the other Google co-founder, Sergey Brin, is the son of Michael Brin, a mathematics professor in dynamical systems. And speaking of math in the family of tech giant founders: the most popular Russian social media site, VK, was founded by the Durov brothers, one of whom is a 3-time gold medalist at the IMO who wrote a remarkable dissertation* [2] *under Gerd Faltings that plays an important role in the pure math research I've done with my brother* [4]*; the Durovs also founded the very popular internet messaging service Telegram.*

- The largest network of replies/retweets had around 50,000 users when the story was false but only around 2,000 users when it was true.

There are two very different ways that information can spread and reach a large number of users on Twitter: a prominent influencer could tweet a story that many followers will directly retweet, or a less prominent user could tweet a story that gets retweeted by a small number of followers who then get it retweeted by some of their followers, etc. Even if a story reaches the same number of retweets in these two scenarios, the first is considered a shallow spread and the second a deep spread since it penetrates more deeply into the social network. It was found in this study that not only did false stories ultimately reach larger audiences, but they did so with much greater depth:

- True stories seldom chained together more than 10 layers of retweets, whereas the most viral false stories reached 20 layers of retweets—and they did so 10 times as quickly as the true stories reached their 10 layers.

However, there is a crucial caveat to these results. The researchers tracked all stories that had been fact-checked by one of several reputable fact-checking organizations, so really what they were comparing was the virality of fact-checked articles that were deemed true versus fact-checked articles that were deemed false. It's not hard to convince yourself that fact-checked articles do *not* form a representative sample of all news stories. Indeed, most news stories don't receive fact-checks because they are obviously true—so it is only the suspicious stories that end up on fact-checking sites.

There have been various efforts to train supervised learning classifiers to detect fake news stories based on their spread across social media. Some of these involve quite sophisticated methods, such as a fascinating "geometric" form of deep learning in which the non-Euclidean geometry of networks is heavily leveraged [12]. But a fully content-agnostic approach like this (meaning one that looks only at the network propagation patterns of posts, rather than the posts themselves) is a largely quixotic endeavor: high rates of false positives are unavoidable and propagation patterns vary tremendously across time, region, platform, and topic. That said, network propagation patterns are an important signal in the moderation process: some platforms use early indicators of virality to promote posts in the moderation queue. In other words, among the questionable posts that have been flagged for human moderators to inspect, platforms often try to assign their moderators first to the posts that are most likely to become viral—and network propagation dynamics are crucial for detecting this.

Another important application of network geometry/dynamics in the realm of curtailing misinformation is the detection of bot accounts on social media. A common strategy for fake news publishers is to use bot accounts to seed early stage virality for stories by creating artificial engagement. The engagement-based algorithms used by platforms like Facebook and Twitter pick up on this early virality and, mistaking it for authentic, broadcast the stories to a wider audience. Fake news stories are often quite controversial and tend to draw a lot of comments and reactions, so once these stories reach a wide audience, they tend to draw even more engagement—and hence, by the nature of the algorithms, even wider audiences. This is the algorithmic path to virality in which the initial bot-driven artificial engagement soon becomes authentic human user engagement.

Using bot accounts in this way is prohibited on most platforms. For instance, Facebook requires each account to correspond to a real user; Twitter does not tie accounts to individuals and it even allows some harmless bot activity, but it bans coordinated manipulation of the kind described above. But how do the platforms detect bot accounts?

In November 2020, Facebook released some details on its latest deep learning bot detection algorithm [15]. It relies on over 20,000 predictor variables that look not just at the user in question but also at all users in that user's network of friends. Facebook didn't disclose what these predictors are but it did say that they include demographic information, such as the distribution of ages and gender in the friend network, and connectivity properties of the friend network. The algorithm is trained in a two-tier process: first, it is trained on a large data set that has been labeled automatically, to get a coarse understanding of the task, then it is fine-tuned via training on a small data set that has been labeled manually so the algorithm can learn more nuanced distinctions. Facebook estimated in the fourth quarter of 2020 that approximately 5% of its active users were fake accounts. Throughout that year, it used this new deep learning system to remove over 5 billion accounts that were believed to be fake and actively engaging in abusive behavior—and that doesn't include the millions of blocked attempts to create fake accounts each day.

Conclusion

The issue of widespread misinformation on the internet, and how to rein it in, has received a lot of attention in recent years from politicians, journalists, tech companies, and academic researchers. Misinformation does not belong to a single academic discipline—it has been studied by political scientists, social scientists, computer scientists, economists, psychologists, media studies scholars, and many others. What I have found most striking is that if you draw a Venn diagram of various approaches these disciplines have taken to study misinformation, it's not too much of a stretch to say that mathematics is what lies at the middle. Most prominently, math helps us quantify the spread of misinformation and quantify the effect of potential interventions, and it helps us understand the algorithms that create and amplify misinformation.

I encourage readers to explore the engaging math at the center of this story of misinformation. You may find a new research topic, you may find a new interdisciplinary collaboration, and you may help a congressional office better understand the complex processes it is trying to regulate. Coming at this from a math background, you'll find the barrier to entry in this field is surprisingly low yet the potential for impact and intellectual stimulation is surprisingly high.

References

[1] Nicola De Cao, Wilker Aziz, and Ivan Titov, *Editing factual knowledge in language models*, Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP2021) (2021), 6491–6506, arXiv:2104.08164.

[2] Nikolai Durov, *New approach to arakelov geometry*, arXiv:0704.2030, 2007.

[3] Farzan Farnia and Asuman Ozdaglar, *Do GANs always have Nash equilibria?*, Proceedings of the 37th International Conference on Machine Learning (Hal Daumé III and Aarti Singh, eds.), Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, https://proceedings.mlr.press/v119/farnia20a.html, pp. 3029–3039.

[4] Jeffrey Giansiracusa and Noah Giansiracusa, *Equations of tropical varieties*, Duke Math. J. **165** (2016), no. 18, 3379–3433, DOI 10.1215/00127094-3645544. MR3577368

[5] Noah Giansiracusa, *Facebook uses deceptive math to hide its hate speech problem*, Wired (2021), https://www.wired.com/story/facebooks-deceptive-math-when-it-comes-to-hate-speech/.

[6] Noah Giansiracusa, *How algorithms create and prevent fake news: Exploring the impacts of social media, deepfakes, GPT-3, and more*, Apress (a division of Springer Nature), 2021, https://doi.org/10.1007/978-1-4842-7155-1.

[7] Noah Giansiracusa, *Facebook could make its algorithms truly work for you*, Boston Globe (2022), https://www.bostonglobe.com/2022/03/24/opinion/facebook-could-make-its-algorithms-truly-work-you/.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, 2016. MR3617773

[9] Scott Hershberger, *Courts, commissions, and consultations: how mathematicians are working to end gerrymandering*, Notices Amer. Math. Soc. **69** (2022), no. 4, 616–623, DOI 10.1090/noti2461. MR4398072

[10] Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, and Nat McAleese, *Teaching language models to support answers with verified quotes*, arXiv:2203.11147 [cs.CL], 2022.

[11] Yisroel Mirsky and Wenke Lee, *The creation and detection of deepfakes: A survey*, ACM Comput. Surv. **54** (2021), no. 1.

[12] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael Bronstein, *Fake news detection on social media using geometric deep learning*, arXiv:1902.06673 [cs.SI], 2019.

[13] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman, *WebGPT: Browser-assisted question-answering with human feedback*, arXiv:2112.09332 [cs.CL], 2021.

[14] Soroush Vosoughi, Deb Roy, and Sinan Aral, *The spread of true and false news online*, Science **359** (2018), 1146–1151.

[15] Teng Xu, Gerard Goossen, Huseyin Kerem Cevahir, Sara Khodeir, Yingyezhe Jin, Frank Li, Shawn Shan, Sagar Patel, David Freeman, and Paul Pearce, *Deep entity classification: Abusive account detection for online social networks*, Usenix Security 2021, 2021.

[16] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi, *Defending against neural fake news*, Advances in Neural Information Processing Systems 32 (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), Curran Associates, Inc., 2019, pp. 9054–9065.

Noah Giansiracusa

Credits

The opening image is courtesy of enot-poloskun via Getty.
Figure 1 and author photo are courtesy of Noah Giansiracusa.
Figure 2 is courtesy of Francis Su.