# Faking the News with Natural Language Processing and GPT-2

Natural Language Processing is Fun Part 4

Adam Geitgey
Sep 27, 2019 · 19 min read

*This article is part of an on-going series on NLP: Part 1, Part 2, Part 3, Part 4.*

In February 2019, OpenAI announced (and refused to release) a new natural language processing model architecture called GPT-2. With GPT-2, you give it a piece of starting text, say "Machine Learning", and it continually predicts the next most likely word:

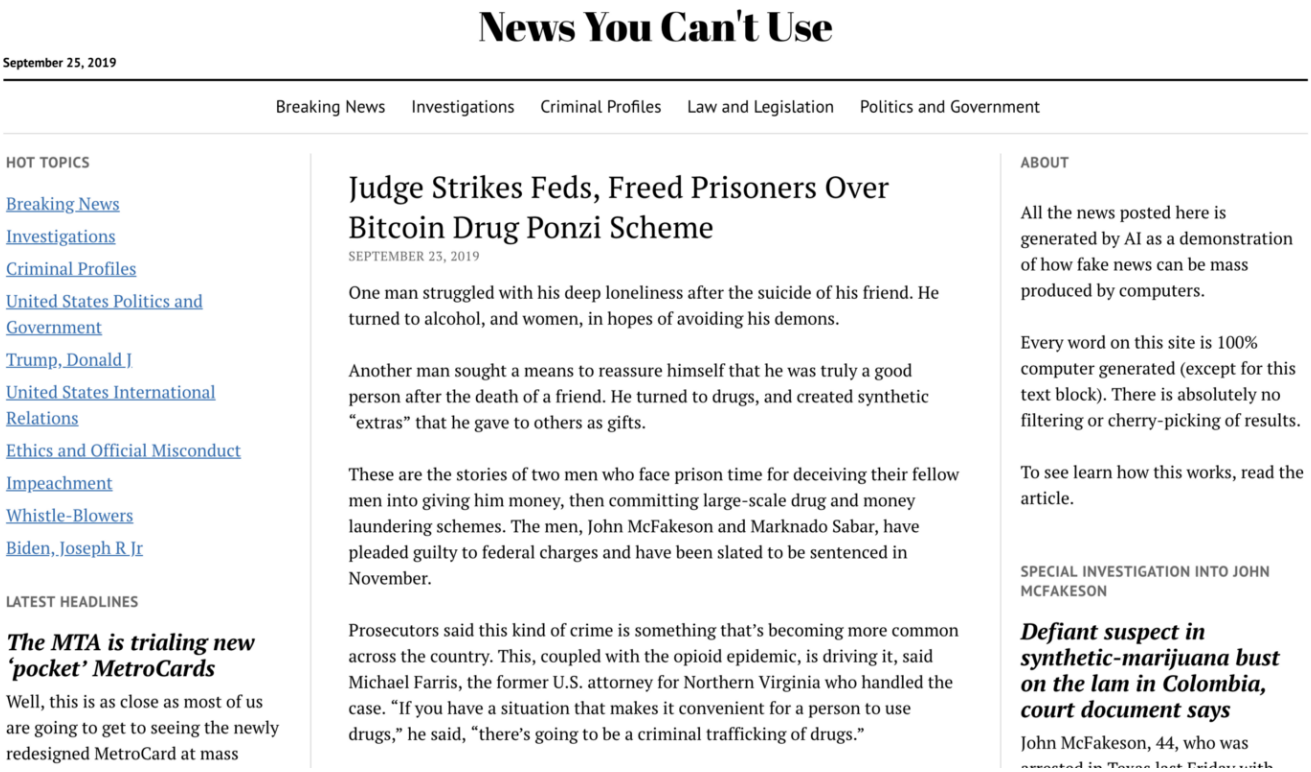
**Machine Learning**

Real text being generated by GPT-2 with "Machine Learning" as the input.

GPT-2 generates text that is far more realistic than any text generation system before it. OpenAI was so shocked by the quality of the output that they decided that the full GPT-2 model was ***too dangerous*** to release because it could be used to create endless amounts of fake news that could fool the public or clog up search engines like Google.

How easy it is for an average person to generate fake news that could trick a real person and how good are the results?

Let's explore how a system like this could work and how much of a threat it is. Let's try to build a newspaper populated with fake, computer generated news:



News You Can't Use

September 25, 2019

Breaking News | Investigations | Criminal Profiles | Law and Legislation | Politics and Government

HOT TOPICS

Breaking News
Investigations
Criminal Profiles
United States Politics and Government
Trump, Donald J
United States International Relations
Ethics and Official Misconduct
Impeachment
Whistle-Blowers
Biden, Joseph R Jr

LATEST HEADLINES

***The MTA is trialing new 'pocket' MetroCards***

Well, this is as close as most of us are going to get to seeing the newly redesigned MetroCard at mass

Judge Strikes Feds, Freed Prisoners Over Bitcoin Drug Ponzi Scheme

SEPTEMBER 23, 2019

One man struggled with his deep loneliness after the suicide of his friend. He turned to alcohol, and women, in hopes of avoiding his demons.

Another man sought a means to reassure himself that he was truly a good person after the death of a friend. He turned to drugs, and created synthetic "extras" that he gave to others as gifts.

These are the stories of two men who face prison time for deceiving their fellow men into giving him money, then committing large-scale drug and money laundering schemes. The men, John McFakeson and Marknado Sabar, have pleaded guilty to federal charges and have been slated to be sentenced in November.

Prosecutors said this kind of crime is something that's becoming more common across the country. This, coupled with the opioid epidemic, is driving it, said Michael Farris, the former U.S. attorney for Northern Virginia who handled the case. "If you have a situation that makes it convenient for a person to use drugs," he said, "there's going to be a criminal trafficking of drugs."

ABOUT

All the news posted here is generated by AI as a demonstration of how fake news can be mass produced by computers.

Every word on this site is 100% computer generated (except for this text block). There is absolutely no filtering or cherry-picking of results.

To see learn how this works, read the article.

SPECIAL INVESTIGATION INTO JOHN MCFAKESON

***Defiant suspect in synthetic-marijuana bust on the lam in Colombia, court document says***

John McFakeson, 44, who was

newsyoucantuse.com, my totally-fake, 100% AI-generated newspaper

To populate News You Can't Use, we'll create a Python script that can 'clone' a news site like the New York Times and generate artificial news stories on the same topics. Then, we'll test the quality of the output and discuss some of the ethical issues that this kind of model raises.

## The Entire Universe, One Word at a Time

The text generated by GPT-2 shows off a huge amount of knowledge about the world. GPT-2 was trained on many gigabytes of text scraped from the web and it has encoded a lot of that knowledge into the model. So when it generates text, it uses that knowledge to make the sentences more realistic. This is what makes it so powerful — the text it generates actually contains real world facts and figures.

If we enter a piece of starting text like this:

```
Abraham Lincoln
```

It will generate a sentence that fits that specific historical person:

```
Abraham Lincoln was born on April 4, 1809 in
Springfield, Illinois
```

Springfield, Illinois.

This sentence is *fascinating* for several reasons:

- First, it shows that the model has encoded that Abraham Lincoln was a person in America who was born in 1809.
- Second, the sentence is perfectly formatted and indistinguishable from something written by a human.
- Third, the sentence is **entirely wrong**.

Abraham Lincoln was born on February 12th, not April 4th. And he was born in Hodgenville, Kentucky, not Springfield, Illinois! But let's be honest, you didn't know that, did you? It *sounded* right!

That's GPT-2 in a nutshell. It's just a statistical model of how English is used on the web. It doesn't have a database of facts and figures to draw from like a 'Question and Answer' system. Instead, it has gotten so good at predicting the next word in any given situation that it has accidentally encoded a weird, squishy version of all human knowledge within the model.

Abraham Lincoln did start his career as a Lawyer in Springfield, Illinois, and Springfield is a city, so *Springfield* makes sense as a probable word that might show up in a sentence about Abraham Lincoln.

The other amazing ability of GPT-2 is that it can maintain coherency across several paragraphs of text. It's not just stringing together individual words that sound plausible, but it's creating text where pronouns agree with each other and the people and ideas mentioned in the text are consistent throughout. This is where it really blows away previous models.

## How GPT-2 Works

There are two main things that make GPT-2 work better than previous text generation models.

The first is that it is much bigger than other models. GPT-2 is absolutely huge (1.5 *billion* parameters) trained on a massive dataset using a remarkable amount of GPU-based computing power.

When OpenAI decided not to release the full GPT-2 model, they were confident that it couldn't be easily reproduced by individuals without specialized technical expertise and massive computing resources. But computing power gets cheaper and more accessible every day. Within a few months, two grad students named Aaron Gokaslan and Vanya Cohen were able reproduce the model with $50k in cloud computing credits. And unlike OpenAI, they released their results. Several other teams were also able to replicate the model as well.

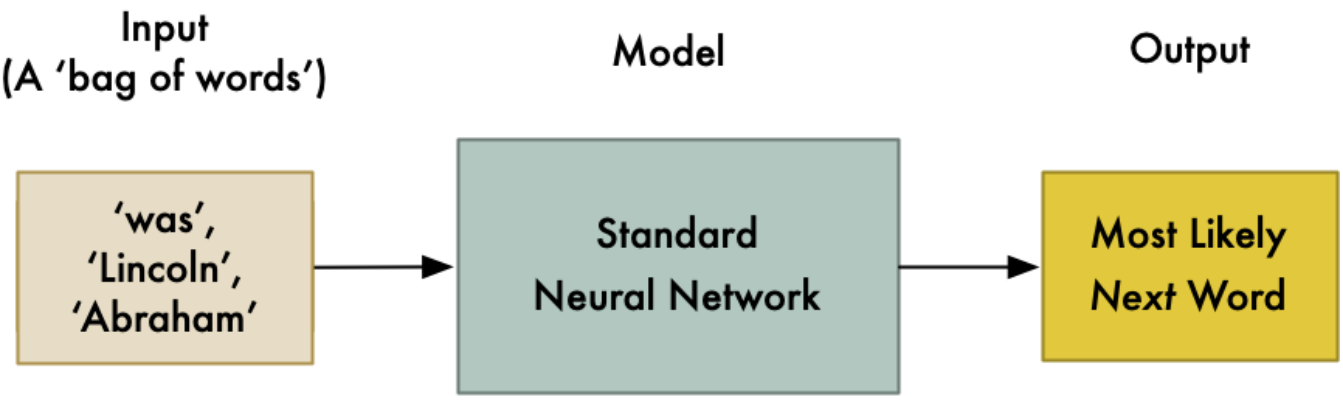
The second secret that makes GPT-2 work is that it's built on top of a brand new way to model text that is significantly better than anything before it — the *Transformer*.

To understand how Transformer-based models work, we need to start by looking at how things used to work.

## Bag of Words models

One of the simplest language models is called a "bag of words" model, or *BoW*. If you want to predict the next word in a sentence, you tell it which words have appeared in the sentence so far — in no particular order — and ask it to tell you the most likely next word.

Let's use the starting text "Abraham Lincoln was" and ask it to predict the next word:

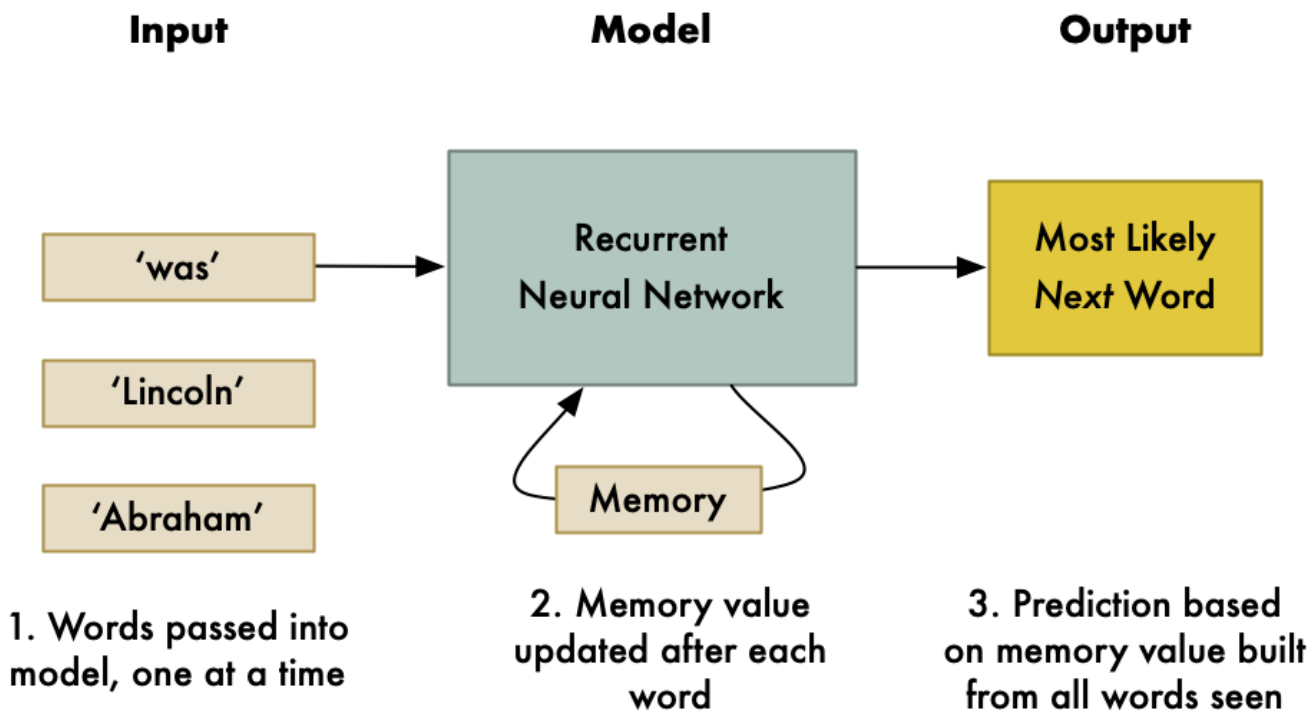



The words get passed into the model in no particular order, as if we threw them all in a bag and handed the jumbled mess to the model. The advantage to this model is that we use any off-the-shelf machine learning algorithm without any changes. But the obvious problem is that the model can only consider the presence of words, not the order that they appeared in. Word order matters a lot in language. Needing to model word order is what makes it hard to adapt traditional machine learning algorithms to language.

## Sequence Models and RNNs

In the early 2010's, Recurrent Neural Networks, or RNNs, became very popular for text modeling. They combine the flexibility of neural networks with an inherent ability to work well with variable-length sequences of text.

In an RNN, the model builds up an understanding of the sentence by looking at each word in sequence. As it sees each word, it updates a memory cell with what it thinks the current word means — thus generating a memory that represents the meaning of the whole sentence so far. When you finally ask it to predict the next word, it's predicting based on this accumulated memory:




RNNs gave us a way to take everything we had learned building image classification systems with deep neural networks and apply it to language modeling where sentences vary in length. Within just a year or two, everything from automatic language translation systems to speech recognition systems got a lot better.

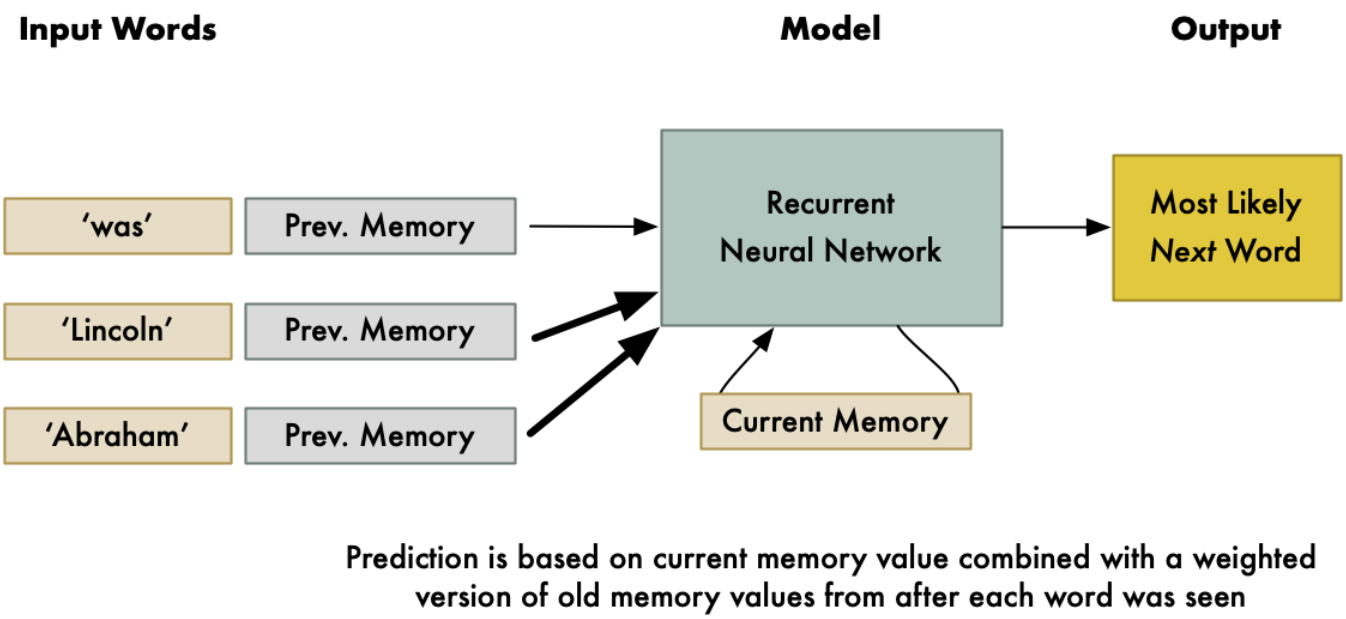
This is when all those "this text was generated by an AI" memes first started popping up on the internet. A well-trained RNN can generate text that looks pretty real — at least for a sentence or two.

## Making RNNs better with Attention

At the time, RNNs seemed like the obvious future of text modeling, so researchers kept looking for ways to improve their ability to model text. The most significant advance was adding an *Attention* mechanism.

A normal RNN remembers the words it has seen so far as a single array of numbers in its internal memory. Each new word it sees updates that memory, so more recent words tend to overpower earlier words. As a result, RNNs don't work very well on long pieces of text.

With an Attention mechanism, the model keeps track of what its memory was after each word. Then when it predicts the next word, it bases its prediction on a weighted version of all these past memories:




By weighting the old memory states differently, it "pays attention" to each word in the sentence more or less depending on how it thinks that word will help predict the next word.

Attention models led to significant performance improvements over standard RNNs in almost every situation, including text generation. But they still weren't able to generate text with coherent ideas across entire paragraphs of text.

## Replacing RNNs with the Transformer

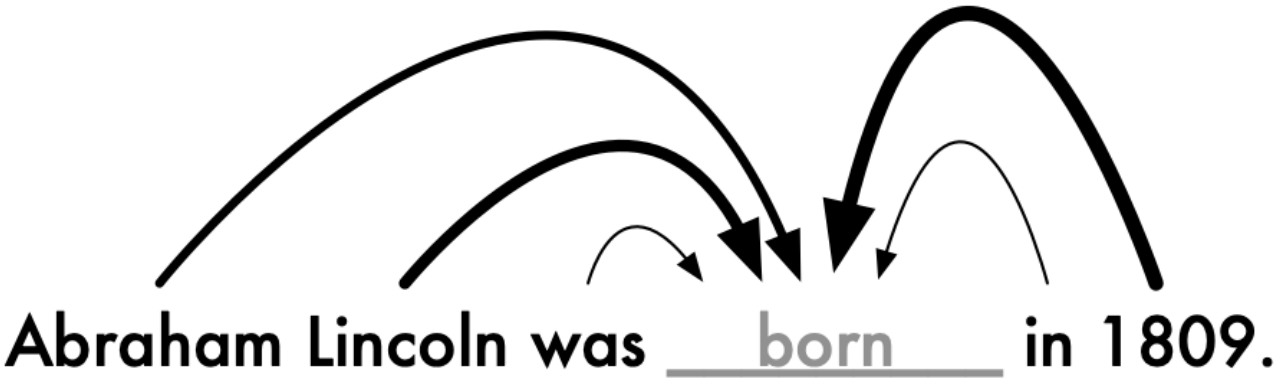
After seeing how much adding Attention to RNNs improved their accuracy, researchers started to wonder if the RNN itself was even really necessary. Maybe the way that Attention weights the predictive value of each previous word in the sentence is what really mattered.

Imagine that we randomly drop a word out of a real sentence:

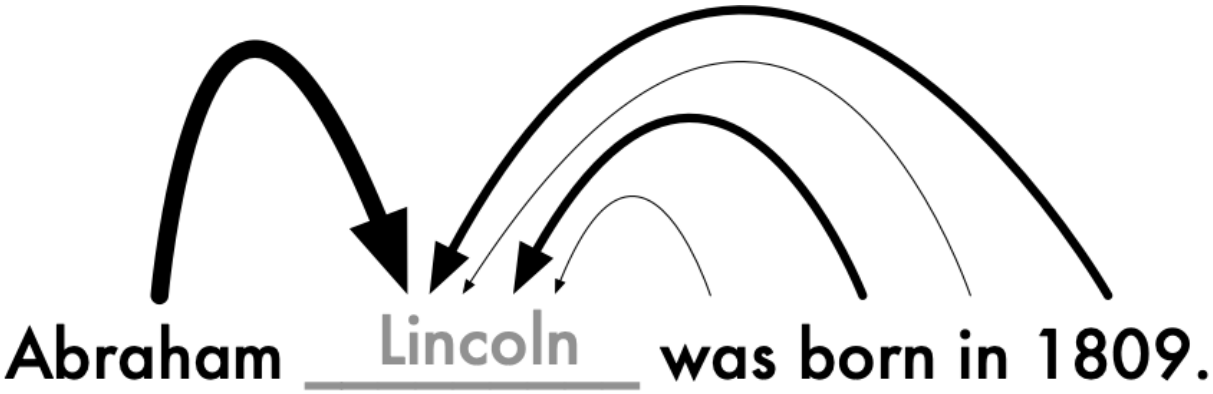
Abraham Lincoln was <DELETED> in 1809

Abraham Lincoln was <DELETED> in 1809.

If we train the model to predict the missing word based on the remaining words in the sentence, we could also measure how much each remaining word was instrumental in that prediction. That would give as a measurement of how strongly each word in the sentence is related to the missing word:




We can keep doing this for other words in the sentence, too. Let's see which words matter most if we drop out "Lincoln":

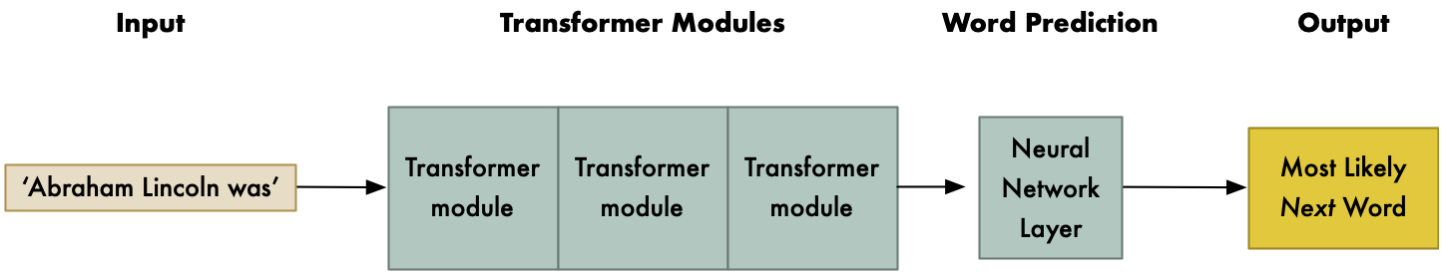



We can repeat this for every word in the sentence to find out how strongly-related each word is to every other word. And if we do this over millions and millions of sentences scraped from the web, the model will learn how every word in English relates to every other word in ***every possible context***. This is what happens inside of a *transformer module*.

A transformer module encodes the meaning of each word based entirely on the words around it *in the current sentence*. Using sentence context is what makes it work so well. A transformer has no problem understanding that the same word can mean totally different things in different contexts. This is why it can model sentences with pronouns

so effectively. The meaning of words like “he” or “she” are defined based on the other words currently in the sentence.

And as years of deep learning research have taught us, if something works well on its own, why not try to stack them? Just like with layers in a deep neural network, we can stack transformer modules on top of each other:




The full-size GPT-2 model has **48** of these Transformer layers stacked on top of each other! In fact, GPT-2 is just short for “Generative Pre-Trained Transformer #2”. It was OpenAI’s second attempt at building a giant text generation model using transformer modules.

## Grover — Extending GPT-2 to Generate News

GPT-2 can generate coherent text based on any initial prompt, but it’s really hard to control exactly *what* it will generate. GPT-2 will often take a simple starting sentence like “A man was arrested” and generate sentences that trail off into weird conspiracy theories or free-form poetry. Those are valid predictions, but they won’t help us generate realistic fake news articles.

If we are going to use GPT-2 to generate fake news, we need to be able to control what it generates and control the style that the text is generated in.

In Mid-2019, a model called **Grover** was released by a team at the University of Washington. Grover is a modified version of GPT-2 where the next predicted word is not only based on the previous words in the text, but also based on other pieces of *metadata* text like a *headline* or an *author*).

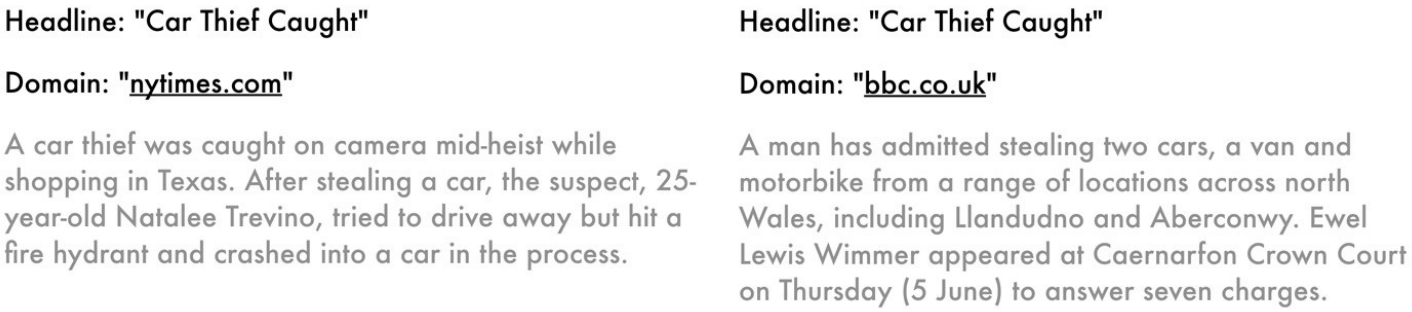
This means that if we give Grover a headline of “Car Thief Caught”, it will know that any text it generates should not only be realistic on its own, but it should also be a plausible fit for that headline.

Grover models a news story as having the following parts:

- Domain name (like *nytimes.com* or *bbc.co.uk*)
- Author name
- Date published
- Headline text
- Body text

Grover can generate the text of any of these fields based on the others. So you can give it a domain, author, date and headline, and it will generate the body. Or you can give a domain, author, date and body and it can generate a headline. Pretty cool!

Even cooler, they trained the model on millions of real news stories scraped from lots of different news websites, so the model has implicitly learned the *style* of different publications. You can make it generate different versions of the exact same story as it would look if it were written by different publications:




Real text article generated by Grover. The only difference is the 'domain name' parameter to the model.

Making the probability of the next word in a piece of text dependent on the metadata fields is a brilliant way to control what the model generates. The amount of location-specific detail encoded in these two samples is amazing. Just by changing the domain parameter from "nytimes.com" to "bbc.co.uk", the model knows that it should talk about Wales instead of Texas and that the UK has a Crown Court system. The model even wrote out the fake dates in UK-style (5 June) instead of US-style (June 5th).

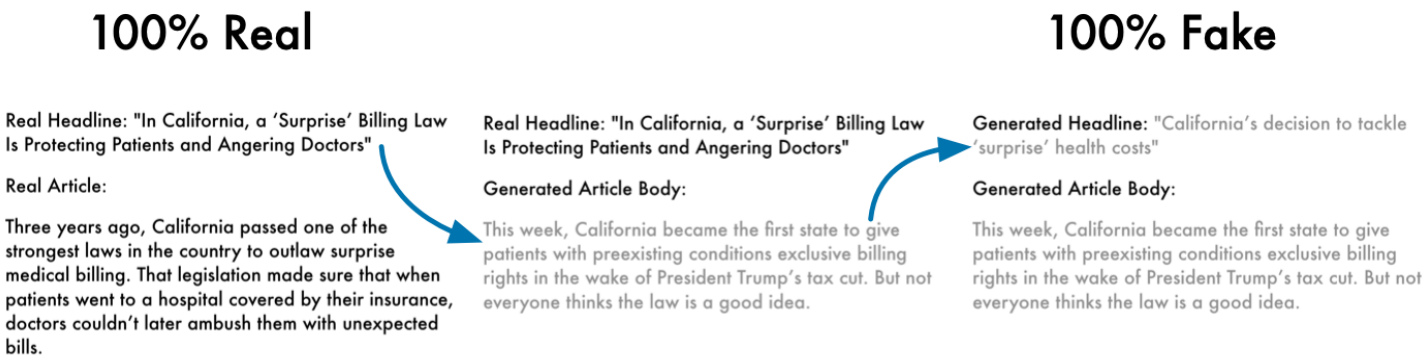
## Making Entirely Original News

With Grover, we can supply a headline, date, domain name and author and have it generate a news story. So if we can scrape a list of news stories from the New York

Times (even without the body of the story), we can easily generate new stories for each headline.

But why stop there? Once we have a story body, we can use it to re-generate a fake *headline* that fits the fake story. And if we have a new headline, we could generate a fake *author name*.

By generating each element of our article one at a time, we can start with real news from the New York Times and end up with fake stories that contain none of the original data. It's endless fake news!




So we'll start by grabbing an RSS feed of headlines from the New York Times, predicting new story bodies and then predicting new headlines for those story bodies. And finally, we'll upload those fakes stories to our own website automatically via an API.

## Creating Specific Stories

We aren't limited to generating alternate versions of real news stories. We can also generate any particular story we want just by writing the headline and having the model generate the story. And then we can use the story itself to generate a better headline, so our fake headline doesn't even have to be good.

So let's pick an imaginary victim and write a lot of simple headlines about them:

- <NAME> Sentenced to 10 Years in Jail
- <NAME> Accused of Fraud
- <NAME> Stole Millions from Orphans

The model will come up with interesting stories based on those headlines and then re-write the headlines.

Finally, we'll pepper these targeted stories in between the other fake stories we generated from real New York Times headlines. That will make our targeted stories look more realistic by making it look like our fake news website has actual content.

# Step 1: Creating the Fake Blog

To post fake news, we need a website to post it on. Let's set up a simple Wordpress blog where we can post our generated articles.

## Registering a Domain Name

First, we need to register a new domain name for our fake news site. You can use whatever domain name registrar you want. I used Hover and quickly found that newsyoucantuse.com was available. I registered it for $13.

| Item | Next Billing Date |
| --- | --- |
| newsyoucantuse.com<br>1 year - domain registration | 2020-09-23 |

| | |
| --- | --- |
| Order Amount: | $12.99 |
| ICANN Fees: | $0.18 |
| **Order Total:** | **$13.17** |

Domain name purchased!

## Setting up a new Wordpress Blog

Now that we have a domain name, we need to set up the website itself. We'll install WordPress (a super common blogging platform) on a virtual machine in the cloud. There are thousands of hosting companies that will run WordPress for you. You can use any of them as long as they let you install WordPress plugins.

I used DigitalOcean's Marketplace feature to do a one-click install of Wordpress on the smallest $5/month virtual machine instance:

```
Completing the configuration of WordPress.Success: WordPress installed successfully.
Installing WP fail2ban (4.2.5)
Downloading installation package from https://downloads.wordpress.org/plugin/wp-fail2ban.4.2.5.zip...
Unpacking the package...
Installing the plugin...
```

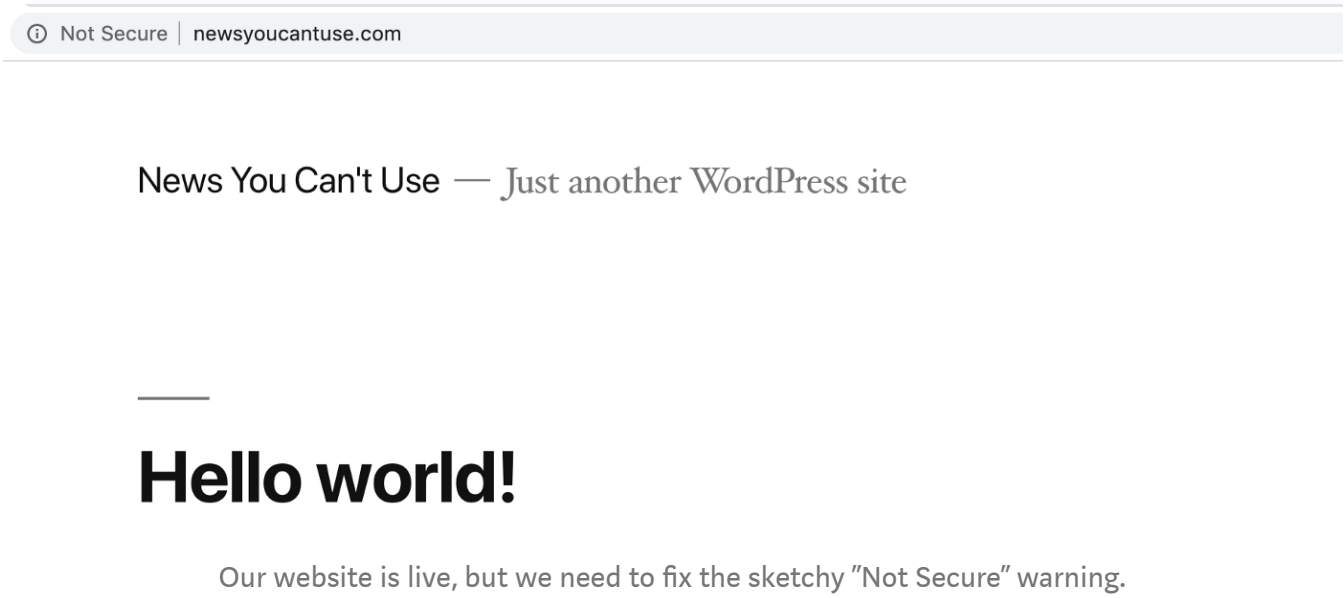

SSH to the new server and it auto-installs WordPress after you answer some basic questions.

## Point Domain Name at the New Blog

The next step is to point the domain name at the blog. I used the admin panel on Hover.com to do this. Then I had to wait about 15 minutes for the domain name to start resolving in my web browser.

Alternately if you want a little more security, you can sign up for a free account on cloudflare.com and use it to proxy traffic to your server while hiding the IP address of your server.

At this point, we have a sketchy-looking default WordPress blog running at newsyoucantuse.com:




## Set up SSH Encryption so the blog looks Legit

We want our blog to look realistic and not like a cheap spam website, so we need to enable TLS encryption so that the user to sees the "lock" icon in their browser (just like on their bank's website).

In the old days, enabling TLS encryption used to cost money. But thanks to the Let's Encrypt project, we can set up an SSH certificate for our blog for free. DigitalOcean

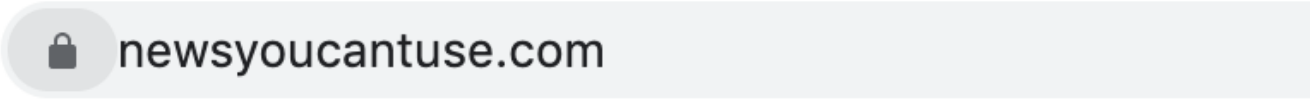
even scripts it for us. We just need to ssh to our blog's virtual machine again and run `certbot --apache` :

```
$ ssh root@[blog-ip-address]
$ certbot --apache

[.. snip ..]

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Congratulations! You have successfully enabled
https://newsyoucantuse.com
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Now our site loads without any warnings!




Nice, now we have the sweet, sweet legitimacy of the lock icon!

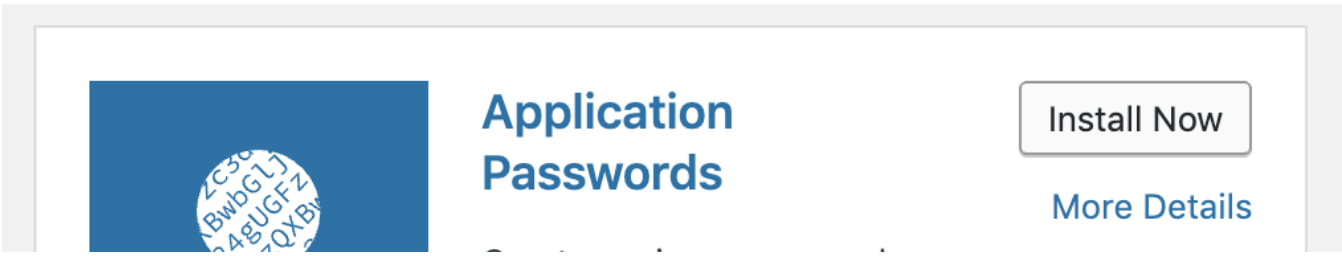
## Making The Blog Look Pretty

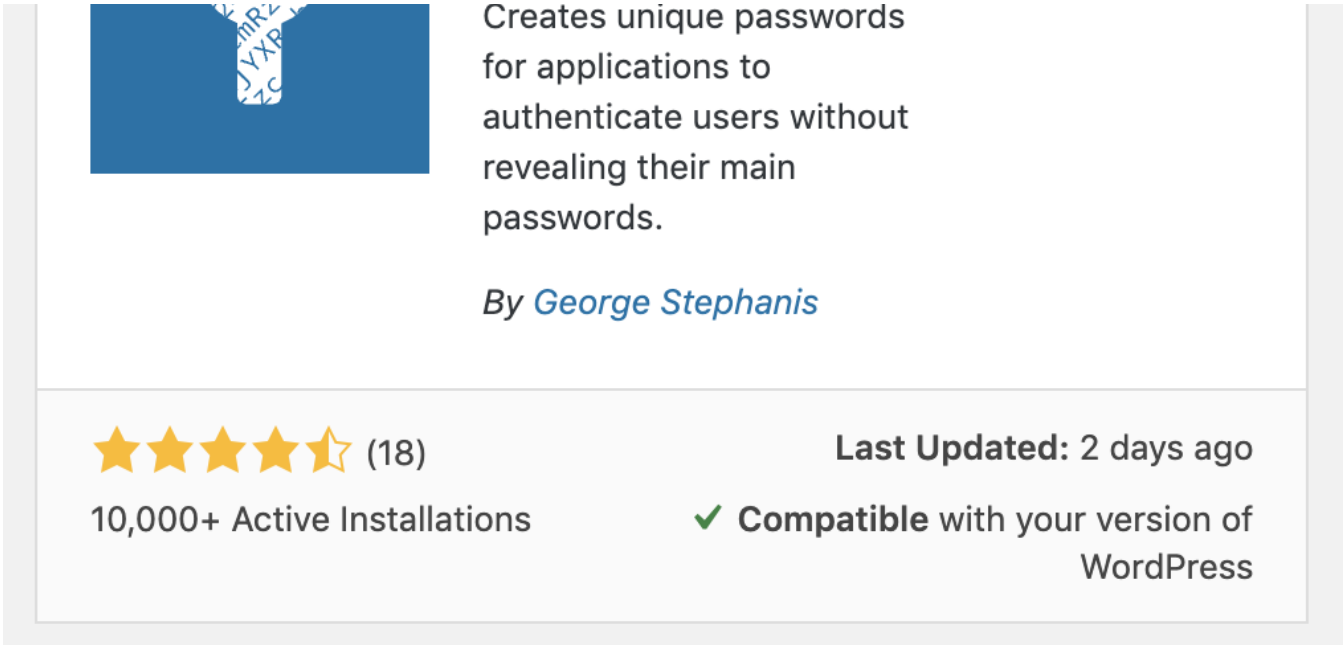
Now we need to log into WordPress Admin and choose a nice theme so the site doesn't look fake. I chose the free theme called "Mission News" because it looked like the New York Times, but you can use whatever you you want. While you are in there, you might as well delete all the sample blog posts that come with WordPress, too.

## Setting Up the WordPress REST API for Posting

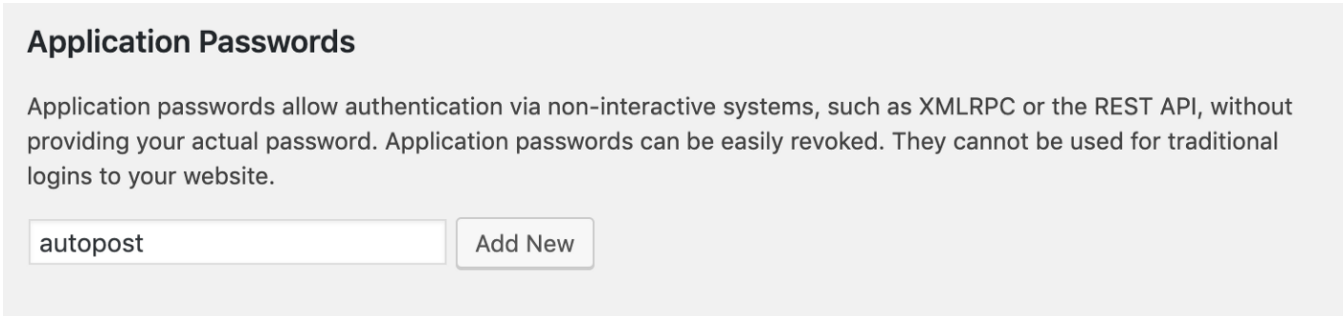
If we were normal bloggers, we'd start writing articles now — but we have AI to write for us! We just need a way for our Python script to post directly to our blog.

WordPress has a REST API that we can use to post automatically from a script. Unfortunately, WordPress doesn't support script-based user authentication out of the box, so we need to install an extra plugin called Application Passwords. You can do that right from the WordPress Admin Panel:






Once the plugin is installed and activated, navigate to User configuration and there will be a new option at the bottom of the User detail page where you can add an application password:

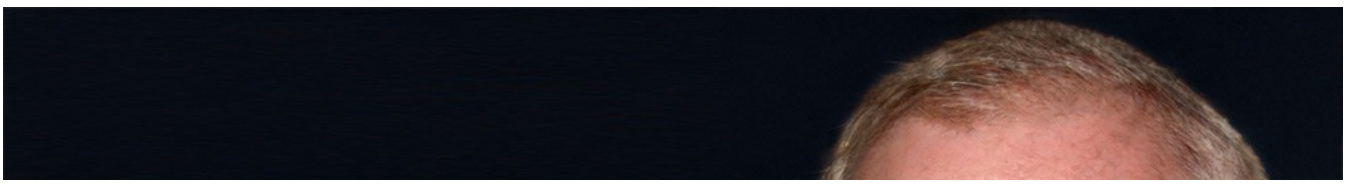



Save the password it generates because we'll need it for our Python script.

## Creating a Fake Victim

We'll be able to generate fake news stories about anyone that we want. But we don't want to post negative things about real people. For this article, I'm making up a fake person that I'll call John McFakeson.

For the picture, I went to thispersondoesnotexist.com, which uses a GAN to generate photographs of imaginary people. I grabbed a random computer-generated picture and added a little text in PhotoShop. Here's how it looks:




This dude does not exist. Like the all the recent news about him, his picture is computer generated.

## Scraping Stories from The New York Times

To clone stories from the New York Times, we need a list of their headlines. Luckily, they publish a list of RSS feeds. We can use any standard RSS parser to read these feeds and extract a list of headlines, authors, tags, and all the other metadata we need to generate fake stories in the same style.

I'm using the Python feedparser library to grab that data. With that library, it just takes a couple of lines of code to read an RSS feed.

## Running it

I've put together the Python code to grab any RSS feed, run the stories through Grover to generate new text and headlines, and post them to your WordPress blog. It even adds some realistic formatting to make the stories look real. Here's the code.



Link to the code (too long to paste inline here)

You can download and run this yourself, but running the 'huge' Grover model to generate text in a reasonable amount of time requires a powerful GPU.

Google recently launched a platform called Google Colaboratory (or Colab for short). It lets you write Python code on a web page and then run the code on one of their servers using their GPU. It's their version of cloud-hosted Jupyter noteboooks:



Google's Colab system

Colab is totally free to use — even with a GPU. So you can take the code I've written, clone the notebook, and run it yourself right now.

To get started, go to this Colab notebook and choose "Open in Playground" so you have your own personal copy of it that you can edit. Then you need to update these values in the first cell:

```
# Your Wordpress Blog where the fake articles will be posted
WORDPRESS_BLOG_API_ENDPOINT = "https://your-domain-name.com/?
rest_route=/wp/v2"
WORDPRESS_USER = 'YOUR_WORDPRESS_USERNAME'
WORDPRESS_APP_PASSWORD = 'YOUR_APP_PASSWORD_HERE'

# Fake person who will be slandered/libeled in the fake articles
NAME_TO_SLANDER = "John McFakeson"
IMAGE_TO_SLANDER = "https://cdn-images-
1.medium.com/max/1600/1*P8FfDY2TXPR0bZ0XIJYRWw.jpeg"
```

You can also change the list of hard-coded see headlines if you want to try playing around with some of your custom headlines.

When you've updated those values, go to the "Runtime" menu and choose "Run all". Assuming you put in the correct values for your WordPress blog, you should see output like this at the bottom:

```
Building article from headline 'John McFakeson bought fake twitter
followers to pretend to be a celebrity'

- Generated fake article titled 'British businessman amassed 7.6
million fake Twitter followers to scam women'
```

From there, go check out your blog to see your new fake stories!

## Evaluating the Results

You can poke around https://newsyoucantuse.com/ to see unfiltered examples of fake news generated with this script. Some of the stories are nearly indistinguishable from real stories and others are… less successful.

One thing you'll discover playing while around with generating your own fake news is that the model is much better at generating slightly distorted versions of reality than it

is at describing imaginary situations.

A stereotypical headline like “Man caught stealing car” will almost always result in a plausible article. But zany a headline like “Alexander the Great caught stealing space rocket” will generate articles that go to *weeeeird* places. This is because the model is only trained on real news stories. When you give it a zany headline, you are asking it to make a prediction outside of its training data set. The results won’t always make a lot of sense:



I purposely didn’t cherry pick the fake news examples on my website so that you can see an unfiltered view of what the results look like.

## Implications

On a technical level, I think that GPT-2 and Grover are some of the most exciting developments in machine learning in a long time. But they definitely have the possibility of being misused. The better that ML models get, the more we all need to consider the implications of what we build.

OpenAI originally decided that GPT-2 was too dangerous to release to the public. In the months since, there has been a huge amount of discussion on whether or not holding back the release of the model was the right choice. Ultimately, it didn’t matter. The GPT-2 models were replicated by other researchers and were released anyway, so the tools are now out in the wild. And given that this technology is now in the hands of people with bad intentions, I think it’s important that we spread knowledge of these

tools as wide as possible so that everyone can learn how to adapt to them and come up with rules for how they can be used ethically.

In fact, the team behind Grover created the project not to generate fake news, but as a tool to *detect* fake news. They found that the best way to automatically detect fake news was to pair a fake news detection model with a fake news generation model and have them work against each other to get better. So in effect, releasing more powerful models is the best way to build stronger tools to protect against them.

### How Original is the Generated Text?

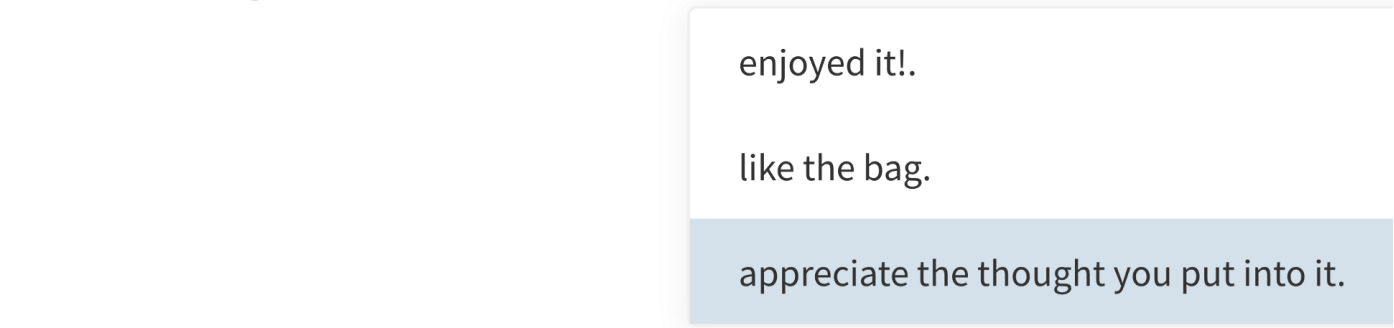
One of the most fascinating aspects of the text generated by GPT-2 and Grover is that it contains references to specific twitter accounts, hash tags, website addresses and other bits of factual information. This brings up the question — how original is the text generated by the model? Is it actually stringing together original thoughts or is it just regurgitating bits of text stolen from all the websites the model was trained on? In other words, is the model overfitting or underfitting the training data set?

This is somewhat of an open question. The researchers behind GPT-2 reported that the model is still underfitting the dataset. And in my anecdotal experience, the generated articles really do contain new sequences of words that don't show up anywhere on Google. I've played around with generating new Shakespeare soliloquies and song lyrics and occasionally the results are so good that I am convinced it must be regurgitating training data, but so far I haven't been able to find any of those results anywhere else using Google. Even still, there's no guarantee that the model won't generate plagiarized text.

## New Tools have New Uses We Can't Yet Imagine

You can do bad things with GPT-2. You could deploy millions of spammy websites overnight and confuse Google's Search engine with endless amounts of junk in an attempt to break search rankings (*please don't*). But just because you can do annoying things with this technology doesn't mean that it isn't a net benefit to society.

There are also lots of ways that this technology can be used to build new tools. Imagine you are trying to craft the perfect 'Thank You' note, but you just can't think of what to write. A text editor with this technology built in could help you finish your email. In fact, there's already a real implementation of GPT-2-based auto-complete from HuggingFace:




Using AI to write my 'Thank You' notes

There are going to be many creative and artistic applications of this technology as well. David Bowie famously wrote a lot of his song lyrics in the 1990s using a primitive program called the *Verbasizer*. It mixed random bits of text into new fragments of lyrics which spurred his creativity:


We can only imagine what today's artists will do with more powerful tools that can synthesize original ideas in whatever style they want. I think this is a very exciting area to explore!

## Further Reading

Want to learn more? Here are some key papers to read:

- Attention Is All You Need, which introduced the Transformer model

- OpenAI's GPT-2 announcement and the related paper
- Defending Against Neural Fake News, which introduced Grover, by Rowan Zellers, et al.

. . .

If you liked this article, consider signing up for my Machine Learning is Fun! Newsletter to find out when I write more stuff like this:

I've written also written a book on ML! It not only expands on my articles, but it has tons of brand new content and lots of hands-on coding projects. Check it out now.

You can also follow me on Twitter at @ageitgey, email me directly or find me on linkedin. I'd love to hear from you if I can help you or your team with machine learning.

Machine Learning Naturallanguageprocessing Fake News AI