

29 APRIL 2019 / DEEP  
LEARNING

# How to Build OpenAI's GPT-2: "The AI That's Too Dangerous to Release"



Today, the United Nations has called for the immediate withdrawal of all nuclear weapons from the world.

The sentence you just read wasn't written by me, the author of this article, nor was it written by the editor. No. What you just read was written entirely by [OpenAI's GPT-2 language model](#), prompted only with the word "Today".

Apart from another fancy acronym, GPT-2 brought along somewhat coherent (semantically, at least) language generation capabilities, some semblance of hope for zero-shot transfer learning, and a [transformer network](#) trained with approximately 1.5 *billion* parameters on a text corpus with over 40 *gigabytes* of internet wisdom.

That's kind of a big deal.

But of course, what really broke the internet was talking, four-horned, half-breed unicorns in the Andes...

```
SYSTEM PROMPT (HUMAN-WRITTEN) In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley, "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them — they were so close they could touch their horns.

While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic."
```

A text generation sample from OpenAI's GPT-2 language model

In this post, I'm not going to talk about [better language models and their implications](#). As the great Stan Lee once said, "nuff said" about that.

Here, I'll show you how exactly humanity's greatest text generator (at the time of this writing, at least) works, and how to build your own in just a few lines of code.

Note, however, that the GPT-2 model that we're going to build won't start generating fake Brexit campaigns. The original model was trained for months, harnessing the power of 100+ GPUs.

So unless you've got that kind of computing power, it's a feat if your mini-GPT can get subject-verb agreement right.

## What GPT-2 Actually Is

As has become the norm when there is a breakthrough in deep learning research, there's been a fair share of terminator imagery accompanying popular articles that describe OpenAI's latest set of matrix multiplications. So I thought I'll start by clearing a few things up.

GPT-2 stands for "Generative Pretrained Transformer 2":

- **"Generative"** means the model was trained to predict (or "generate") the next token in a sequence of tokens in an unsupervised way. In other words, the model was thrown a whole lot of raw text data and asked to figure out the statistical features of the text to create more text.
- **"Pretrained"** means OpenAI created a large and powerful language model, which they fine-tuned for specific tasks like machine translation later on. This is kind of like transfer learning with Imagenet, except it's for NLP. This retraining approach became quite popular in 2018 and is very likely to be a [trend that continues throughout 2019](#).
- **"Transformer"** means OpenAI used the [transformer architecture](#), as opposed to an [RNN](#), [LSTM](#), [GRU](#) or any other 3/4 letter acronym you have in mind. I'm not going to discuss the transformer architecture in detail since there's already another [great article](#) on the FloydHub blog that explains how it works.
- **"2"** means this isn't the [first time they're trying this whole GPT thing out](#).

## How it Works

So here's a summary of all the 2018 NLP breakthroughs that you need to understand before getting into GPT-2. I'll illustrate it using some insanely advanced math:

2018



OpenAI Transformer v1(akaGPT-1) = [ULMFiT](#)+ Transformer

2019



GPT-2 = GPT-1 + reddit + A lot of  
compute

Wait, What!?

Ok. there's a fair amount of background knowledge required to get all of that. To top that, I've also left out essential ideas like ELMo and BERT that while not immediately relevant when talking about GPT-2, were instrumental to its eventual development.

If you're already aware of the technologies that led up to GPT-2, congratulations! You basically now understand what it takes to invent a state of the art NLP model! ☺☺

But for the rest of you that were daydreaming about a Sesame Street-Michael Bay crossover, let's get into it.

## Transformers



[Source](#)

The transformer is an awesome neural network architecture. As I mentioned already, the details of this model are ... fairly detailed.

So for the purposes of this article, treat the transformer as a black box— it defines a structure for performing computations. Though in actuality, that's a gross abstraction, so I'd encourage you to read [this](#) transformer article before you continue.

## Pre-trained Language Models





Photo by [Kelly Sikkema](#)

[Another trend](#) that the NLP community picked up in 2018 was the idea of transfer learning, which had been going on for years in the computer vision world, but has only recently picked up the pace for NLP tasks. Again, transfer learning has been hugely successful and is likely to continue throughout 2019.

“Transfer learning” here is usually done in 2 ways: feature-based and [fine-tuning](#).

ELMo uses a feature-based method, where contextual word embeddings are created by concatenating the hidden state vectors from a pretrained language model to the existing word vector. But I’m not going to elaborate on that, because neither BERT nor GPT use the feature-based approach.

Throughout 2018, we’ve come to see that fine-tuning works slightly better, probably because it allows you to tweak the language model through backpropagation.

## Transformers + Pre-trained Language Models



Photo by [Rafaela Biazzi](#)

So here was OpenAI’s big insight: transformers work pretty good, Fine-tuning a language model works pretty good, so a transformer + a pretrained language model should work pretty good.

A few backprops later, GPT was born.

In reality, though, there were a few hurdles to cross. First, looking at the transformer architecture, it’s not entirely clear how to make it work for language modeling. Take a look at the diagram below to see what I mean.



[Source](#)

The transformer expects a complete sentence (“sentence” here is not the same as an English sentence, it means a sequence of words of some fixed length—512 in the case of GPT), which it encodes and

“unrolls” into a sequence of

“transforms” using a decoder.

This behavior makes it an excellent choice for sequence-to-sequence applications, like machine translation and question answering, but it’s practically useless for language modeling, where we want to predict the next word given a sequence of words.

Luckily, the decoder part of the transformer can sort of do this on its own. Think about what the decoder is actually doing—given an encoded representation of a sequence, it generates a new sequence word by word.

Written concretely,

$$\text{word}_t = \text{Decoder}(\text{word}_{t-1}, \text{encoding})$$

If we throw away the encoding part from the decoder, we get this:

$$\text{word}_t = \text{Decoder}(\text{word}_{t-1})$$

Which is precisely what a language model is supposed to do!

Consequently, we need to throw away the entire encoder section of the transformer so our final architecture will look like this:



[Source](#)

To summarize, GPT is nothing but the decoder part of a regular transformer network, with all the references to the encoder thrown away.

## Fine-tuning GPT

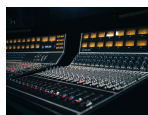


Photo by [Jiroe](#)

What we’ve discussed so far is only half of the story. GPT works well across a multitude of tasks only because of the other innovation—fine-tuning.

Here’s where we’re at—we have a really good language model that (hopefully) has learned dynamics of the English language after being trained on a vast text corpus.

Now, in theory, if we stick a task-specific layer or two on top of the language model, *weshould* \*\*get something that leverages the language model’s linguistic capabilities while also adapting to the task at hand

something that leverages the language model's linguistic capabilities while also adapting to the task at hand.

Turns out that it's not just theory. This method actually works. It works really well. Well enough to beat state of the art on a suite of NLP benchmarks. Well enough to be hailed as [NLP's ImageNet moment](#).

## A Small Step for Man, a Giant Leap for Language Model-kind



Photo by [History in HD](#)

GPT was great. But not for long. Another similar approach—BERT was introduced by the google language team right after GPT came out, and like a kid in a candy store, the NLP folks sent GPT to the grave. RIP.

But not for long. OpenAI quickly bounced back with a revolutionary idea—doing absolutely nothing at all.

You see, what made BERT so great was that it used what's called a bidirectional language model, as opposed to GPT's unidirectional language model. I'm not going to specifically address why bidirectional models are better than unidirectional models, but when has "bi-" anything been worse than "uni-" something?

Here's how I imagine the discussion at the OpenAI boardroom probably went on the day the BERT paper was published:

**Manager:** "Hmm... this BERT thing seems to be working better than our idea. What gives?"

**Random Engineer 1:** "Well, they did this thing called masked language modeling, where they hid a certain percentage of words and trained a language model to predict them. That allowed them to use a bidirectional model which deeply encodes..."

**Manager:** "In English, please."

**Random Engineer 1:** "Their model is essentially ours with an extra pair of eyes on the back of its head."

**Manager:** "So the all-important question: how do we beat Mr. Sesame Street?"

**Random Engineer 2:** "We could train a bidirectional model too. But that would just be copying them. Or maybe we could concatenate..."

**Random Engineer 1:** "Nah. That would just take another step in an endless cycle. If they came up with BERT, they'd probably do something better eventually. We need a more long term solution."

**Intern:** “You know, we could just throw more GPUs and data at it.”

**All thee in unison:** “Genius!”

## The Front Page of the Internet

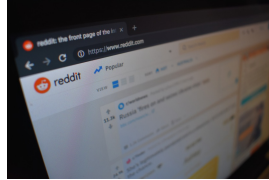
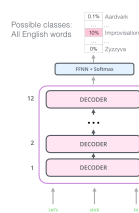


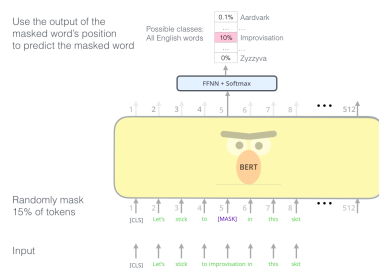
Photo by [Con Karampelas](#)

Instead of trying to beat BERT at its own game, the next iteration of GPT, prolifically named GPT-2, changes the very nature of the game it’s playing.

In simple terms, BERT is trained to be very good at fill-in-the-blanks, while GPT-2 is trained to be very good at writing essays.



GPT-2 predicts the next word



BERT predicts missing (masked) words

I should also point out that what makes GPT-2 worthy of the “2” is massive scale. While BERT has a respectable 340 million parameters, GPT-2 blows it out of the water with a whopping 1.5 billion parameters.

Since summer 2020, “GPT-2” has been a common keyword in AI ML, BERT and others have shown

*Since our work on “Semi-supervised sequence learning”, ELMO, BERT and others have shown changes in the algorithm give big accuracy gains. But now given these nice results with a vanilla language model, it's possible that a big factor for gains can come from scale. Exciting!*

<https://t.co/fL4acZE2Cn>

— Quoc Le (@quocleix) [February 18, 2019](#)

It also happens to be trained on a large chunk of Reddit, since the author decided that this was undeniably the perfect location to obtain high quality, impeccable prose.

Specifically, OpenAI trained GPT-2 on text obtained from outbound Reddit links submitted by authors who have an authoritative karma of 3 or greater. The dataset was taken from web links, and the data is text. So, fascinatingly, they called the dataset WebText.

Since BERT was tasked with filling in blanks, there's no way (yet) that it could GPT-2's Shakespearean capabilities, *even if* it actually performed the task of language modeling better.

So great. We now have an algorithm capable of generating coherent *sounding* text. And I'd like to emphasize *sounding* because the reality is that [GPT-2 generated text only makes sense if you're not paying attention](#).

Both methods aim to create large and powerful pretrained language models that are useful in a transfer learning context, but GPT-2 had a hidden superpower that was guaranteed to drive media outlets wild—writing articles about talking, four-horned, half-breed unicorns in the Andes...

## Making Sense if it All

If you've read this article completely and still feel like you don't have GPT-2 engrained in your bones, fear not.

We all understand better when we actually see the code running before our eyes. So here it is, your chance to see GPT-2 (powered by [Hugging Face's pretrained PyTorch model](#)) running in real time, right before your eyes.



## Further Reading

Thanks for reading my article. Hopefully, you now understand the key ideas behind one of the most significant NLP breakthroughs of 2019, beyond just the media hype.

While I was writing this, I came across many great resources that I think you'll find useful, considering that you made it this far. So here you go. Happy learning.

## Fun Stuff



- The team at the Allen institute have put together a really cool [interactive GPT-2 demo!](#)
- [ELMo](#) is another fairly recent NLP techniques that I wanted to discuss, but it's not immediately relevant in the context of GPT-2.

## Technical Papers

- The idea of transfer learning in NLP isn't entirely new. Semi-supervised sequence learning can be thought of as [the paper that birthed NLP transfer learning](#).
- If you're interested learning about how exactly the art of fine-tuning language models developed, look no further than [the ULMFiT paper](#).
- As I mentioned in the article, GPT-2 is the second iteration in generative pretrained transformer saga. If you want the first chapter in the story, check out [the first GPT paper](#) and [OpenAI's official blogpost](#).
- Remember the other model that I talked about? The one that "competed" with GPT-2? You can learn more about how that works from [the BERT paper](#).
- Finally, the best resources to truly grasp the intricacies of GPT-2 would be [the GPT-2 paper](#) and [the official blogpost](#).

---

Just a few weeks before I started writing this article, I couldn't have known that an AI breakthrough was right around the corner. Yet, a few days and a few hyperlinks later, I, a random high school graduate halfway across the globe, was looking at research produced by a world-class organization with over 1 billion dollars in total investments.

Dear internet, thank you for making great things accessible

Dear internet, thank you for making great things accessible.

And if you felt that this article provided any value to you at all, the credit goes to [Alessio Gozzoli](#) and the wonderful [FloydHub AI writer](#) program. FloydHub, thank you for helping me learn far more than I could have alone.

## FloydHub Call for AI writers

Want to write amazing articles like Ajay and play your role in the long road to Artificial General Intelligence? [We are looking for passionate writers](#), to build the world's best blog for practical applications of groundbreaking A.I. techniques. FloydHub has a large reach within the AI community and with your help, we can inspire the next wave of AI. [Apply now](#) and join the crew!

## About Ajay Uppili Arasanipalai

Ajay is a deep learning enthusiast and student at the University of Illinois at Urbana-Champaign. Ajay is a [FloydHub AI Writer](#). He writes technical articles for blogs like FloydHub, freecodecamp, HackerNoon, and his own blog, [Elliptigon](#). You can connect with Ajay via [LinkedIn](#), [Twitter](#), [Medium](#), [Facebook](#), [Reddit](#), and [GitHub](#).

### Subscribe to FloydHub Blog

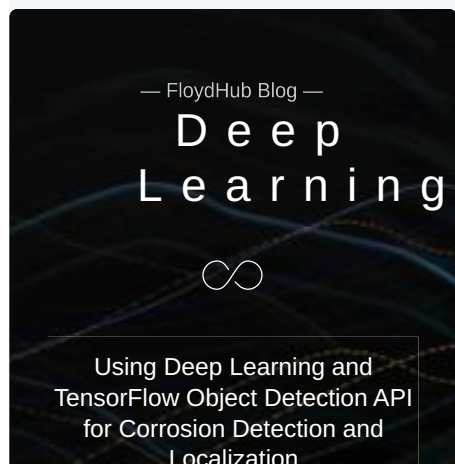
Get the latest posts delivered right to your inbox

Subscribe



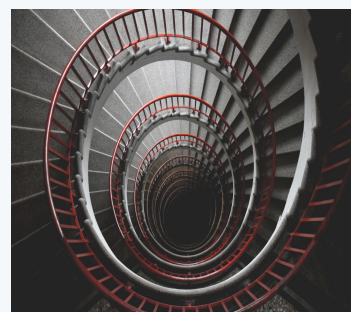
#### Ajay Uppili Arasanipalai

Ajay is a deep learning enthusiast and student at the University of Illinois at Urbana-Champaign. He writes technical articles for blogs like freecodecamp, HackerNoon, and his own blog, [Elliptigon](#).



REINFORCEMENT LEARNING

#### An introduction to Q-Learning: Reinforcement Learning



DEEP LEARNING


#### A Beginner's Guide on Recurrent Neural Networks with

A Beginner's Guide on Recurrent Neural Networks with PyTorch

Meta-Reinforcement Learning

See all 31 posts →

Learn about the basic concepts of reinforcement learning and implement a simple RL algorithm called Q-Learning.




27 MIN

READ

**PyTorch**

Learn the basics of Recurrent Neural Networks and build a simple Language Model with PyTorch



17 MIN

READ