

Large Language Models (LLMs) like [ChatGPT](#), the new “Sydney” mode in Bing (which still exists apparently), and Google’s Bard have completely taken over the news cycles. I’ll leave the speculation on whose jobs these are going to steal for other publications; *this* post is going to dive into how these models actually work from where they get their data to the math (well, the basics you need to know) that allows them to generate such weirdly “real” text.

Machine learning 101, a crash course

LLMs are a type of Machine Learning model like any other. So to understand how they work, we need to first understand how ML works in general. *Disclaimer:* there are some incredible visual resources on the web that explain how Machine Learning works in more depth, and probably better than me – I’d highly recommend checking them out! This section will give you the basics in a technically style.

The simplest way to understand basic ML models is through prediction: **given what I already know, what’s going to happen in a new, unknown situation?** This is pretty much how your brain works. Imagine you’ve got a friend who is *constantly* late. You’ve got a party coming up, so your expectation is that he’s going to, shocker, be late again. You don’t *know* that for sure, but given that he has always *been* late, you figure there’s a good chance he will be this time. And he shows up on time, you’re surprised, and you keep that new information in the back of your head; maybe next time you’ll adjust your expectations on the chance of him being late.

Your brain has millions of these models working all the time, but their actual internal mechanics are beyond our scientific understanding for now. So in the real world, we need to settle for algorithms – some crude, and some highly complex – that learn from data and extrapolate what’s going to happen in

unknown situations. Models are usually **trained** to work for **specific domains** (predicting stock prices, or generating an image) but increasingly they're becoming more general purpose.

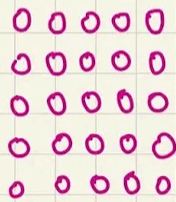
Logistically, a Machine Learning model is sort of like [an API](#): it takes in some inputs, and you teach it to give you some outputs. Here's how it works:

1. **Gather training data** – you gather a bunch of data about whatever you're trying to model
2. **Analyze training data** – you analyze that data to find patterns and nuance
3. **Pick a model** – you pick an algorithm (or a few) to learn that data and how works
4. **Training** – you run the algorithm, it learns, and stores what it has learned
5. **Inference** – you show new data to the model, and it spits out what it think

You design the model's *interface* – what kind of data it takes, and what kind of data it returns – to match whatever your task is.

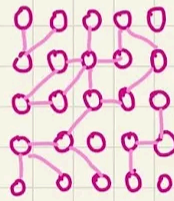
Machine Learning in a nutshell

gather some data



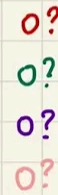
make sure it's good

model the data



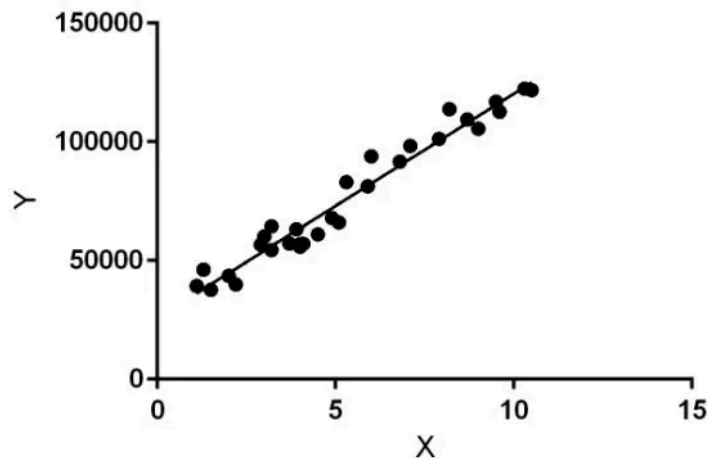
try different algorithms
and settings

understand unknown data



feed new data into
the model

So what is the algorithm actually doing? Basically, it's a really good analyst. It's finding the relationships between the data that you give it, which are often too subtle and complex for you to figure out manually. The data usually has some sort of X – characteristics, settings, details – and a Y – what ended up happening. If you're looking at this data:



You don't need ML to tell you that when X is 15, Y will be somewhere around 150,000. But what happens when there are 30 different X parameters? Or when the data is shaped all strange? Or when you're dealing with text? ML is fundamentally about modeling complex domains with complex data, where human manual ability falls sadly short. That's really all it is.

That's why ML algorithms can be as simple as [linear regression](#) – which you may have learned about in Statistics 101 – or as complex as a neural network with millions of nodes. The kinds of models that have made headlines recently are mind bogglingly complex, and took the work of hundreds of people (not to mention decades of collective research). But you'll often find Data Scientists at companies using pretty simple algorithms, and getting good results.

Deeper Look

Creating powerful ML models from scratch is an incredibly specialized discipline. While many Data Scientists and ML Engineers indeed do that with frameworks like [PyTorch](#) and [Tensorflow](#), others build on top of existing open source models and extend their functionality. And you can even outsource the entire model development process, and use someone else's right out of the box.

Model development is **iterative**: unless your data is super simple, you'll likely need to try different algorithms, and tweak them constantly before your model begins to make any sense. This is part science and math, part art, and part plain randomness.

Language models and generating text

When your data has a time component to it – say you want to predict stock prices in the future, or understand what's going to happen in an upcoming election – it's pretty easy to understand what a model is doing. It's using the p

to predict the future. But many ML models don't work with time series data and language models are a great example of that.

Language models are just ML models that work with text data. You train them on what's called a **corpus** (or just body) of text, and then use them for any number of different things, like:

- Answering questions
- Searching
- Summarizing
- Transcription

The concept of the language model has been around for ages, but deep learning with neural networks has been making a big wave recently; we'll cover both.

Probabilistic language models

Statistically speaking, a probabilistic language model is just a probability distribution over words, or groups of words. In English, that means that it looks at a body of text, and dives deep into what words appear, when they appear, how often they appear, what sequence they appear in, and things like that. All of this information gets represented statistically.

Let's create a quick language model ourselves. Here's two sentences that may or may not represent my actual opinion:

"The best Manhattan cocktail specification uses two ounces of Van Brunt Empire Rye, one ounce of Cocchi Di Torino Sweet Vermouth, one dash of Angostura Bitters, and one dash of Orange Bitters. I stir it in a mixing glass about 60 turns, pour into a chilled Nick and Nora glass, and serve garnished with a Maraschino cherry."

To build a basic probabilistic language model, we'll gather **n-grams**, which is a fancy statistics word for groups of words. Let's say $n=1$, so we're just going to count how often words appear:

Word	Count
of	4
one	3
Bitters	2
...	...

And if $n=2$:

Word Pair	Count
dash of	2
one dash	2
...	...

You can try this yourself, or use a calculator [like the one here](#). What the model does is create tons and tons of n-grams, even taking into account which words appear next to each other and in which direction.

 **Don't sweat the details** 

I like going through the n-gram exercise just to illustrate that what many models are doing under the hood is really not that complicated (for others, it's very complicated). So don't worry if you didn't catch all of the above.

With that information stored, you then **predict** which words might appear next. If we wanted to generate a new sentence based on our two cocktail sentences, we would combine words in a way that *looks* like the previous ones.

Neural networks and language models

Probabilistic language models have been around for decades. Recently though has become a lot more popular to use neural networks – a much more complex algorithm – for language models. These networks are able to learn what's going on in a much more meaningful way, using something called **embeddings**. It's hard for models to learn from words, but it's a lot easier for them to learn from mathematical representations of those words.

Embeddings are a way of [taking data with tons of dimensions](#) – like giant blobs of text, with tons and tons of discrete words and word combinations – and representing them mathematically with way *less* data, without losing much fidelity. It's very hard for an ML model to work with the text from 100 different 1,500 word blog posts about cocktail making – that's 150K words! But if we can distill that information into a bunch of numbers, then we're cooking with gas.

Armed with a more ergonomic representation of words and text, neural networks can learn all sorts of important stuff about text, like:

- Semantic relationships between words
- Bringing in more context (sentences before and after a word, or sentence)
- Figuring out which words are important and which aren't

This stuff gets really, really complex. But the goal is pretty simple: a model that's powerful enough to take *a lot* of context into account when guessing the next word, sentence, or paragraph. Just like our brains do.

Large Language Models, circa 2023

ChatGPT and its cousins are basically really, really big language models (hence the moniker). They're built on layers and layers of advancements from the past decade, including:

- [Word2Vec models](#)
- [LSTM \(long short term memory\) models](#)
- [RNN \(recurrent neural networks\)](#)
- [Transformers](#) (yes) (also known as “foundation models”)

You don't really need to know what any of these are. What's important is to realize that these LLMs are not some surprise scientific breakthrough. Researchers have been inching towards today's reality slowly but surely for years and years, and each new development played a critical role in getting us here. There was a huge hype cycle when LSTMs made headway in 2019 (although the concept was originally introduced in the 90s). And the same for all of these. Research is weird!

⌘ Related Concepts ⌘

I want to stress that a *major* reason LLMs are perceived as a major advance not only because of the models themselves, but also [because of how they're distributed](#). OpenAI built a chat interface for ChatGPT, and opened it to the public. Anyone can use it! This was never really the case before this wave of models. And when anyone can use it, anyone can talk about it.

The way that ChatGPT and LLMs generate these entire paragraphs of text is by playing the word guessing game, over and over and over again. Here's how it works:

1. You give the model a prompt (this is the “predict” phrase”)
2. It predicts a word based on the prompt

3. It predicts a 2nd word based on the 1st word
4. It predicts a 3rd word based on the first 2 words
5. ...

It's really very primitive when you get down to it. But it turns out that the word guessing game can be very powerful when your model is trained on **all of the text on the entire fucking internet**. Data Scientists have long said (about ML models) that "garbage in means garbage out" – in other words, your models are only as good as the data you've used to train them. With [OpenAI's partnership with Microsoft](#), they've been able to dedicate tremendous amounts of computing resources towards gathering this data and training these models on powerful servers.

With the entire written internet as context, LLMs can produce sentences that rarely deviate from something "normal" in a way that older models could rarely do. If the phrase "I Angostura my cocktail with Manhattan ice around glass twist" never appears anywhere on the web, the model is probably not going to generate it. And this simple truth is a big piece of why these models are so good





What's the perfect specification for a Manhattan cocktail?



The Manhattan cocktail is a classic cocktail that traditionally consists of whiskey, sweet vermouth, and bitters. Here's a recipe for a standard Manhattan:



Ingredients:

- 2 oz rye whiskey or bourbon
- 1 oz sweet vermouth
- 2 dashes Angostura bitters
- Maraschino cherry or lemon twist, for garnish
- Ice

Instructions:

1. Fill a mixing glass or cocktail shaker with ice.
2. Add the rye or bourbon, sweet vermouth, and bitters.
3. Stir well until the mixture is chilled and well combined.
4. Strain the mixture into a chilled cocktail glass.
5. Garnish with a maraschino cherry or lemon twist, if desired.

Note: Some people prefer their Manhattan cocktails served "up" (strained into a chilled cocktail glass without ice), while others prefer them served on the rocks (over ice). Adjust the recipe accordingly to your preference.

ChatGPT sourced this data from cocktail recipes across the web and utilized it when guessing which words to generate.

All of this begs the important question of whether these LLMs actually *understand* the answers they're giving you. Answering that is part math, part philosophy, and part semantics (what does "understand" mean?). I like [this Gradient article](#) that explores the question from an interesting perspective.