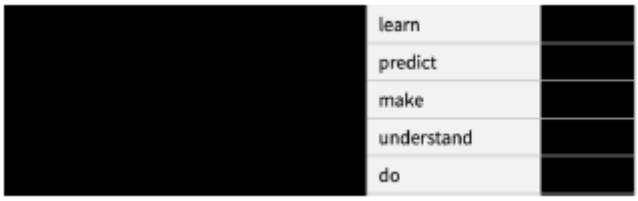
Co robi ChatGPT… i dlaczego działa?

To tylko dodawanie jednego słowa na raz

To, że ChatGPT może automatycznie generować coś, co brzmi nawet powierzchownie, jak tekst napisany przez człowieka, jest niezwykłe i nieoczekiwane. Ale jak to robi? I dlaczego to działa? Moim celem jest przedstawienie ogólnego zarysu tego, co dzieje się w ChatGPT, a następnie zbadanie, dlaczego tak dobrze radzi sobie z tworzeniem tego, co możemy uznać za znaczący tekst. Na wstępie powinienem powiedzieć, że skupię się na szerszym obrazie tego, co się dzieje — i chociaż wspomnę o kilku szczegółach technicznych, nie będę się w nie zagłębiał. (A istota tego, co powiem, odnosi się równie dobrze do innych obecnych „dużych modeli językowych" [LLM], jak do ChatGPT.) Pierwszą rzeczą do wyjaśnienia jest to, że to, co ChatGPT zawsze zasadniczo stara się zrobić, to stworzyć „ rozsądna kontynuacja" dowolnego tekstu, jaki do tej pory powstał, gdzie przez „rozsądny" rozumiemy „to, czego można się spodziewać po kimś, kto napisał po zobaczeniu, co ludzie napisali na miliardach stron internetowych itp." Powiedzmy, że mamy tekst „Najlepszą rzeczą w sztucznej inteligencji jest jej zdolność do". Wyobraź sobie skanowanie miliardów stron tekstu napisanego przez ludzi (np. zdigitalizowane książki) i znajdowanie wszystkich wystąpień tego tekstu — a następnie sprawdzanie, jakie słowo występuje dalej, w jakim ułamku czasu. ChatGPT skutecznie robi coś takiego, z tą różnicą, że (jak wyjaśnię) nie patrzy na tekst dosłowny; szuka rzeczy, które w pewnym sensie „pasują znaczeniowo". Ale końcowym rezultatem jest to, że tworzy uszeregowaną listę słów, które mogą wystąpić wraz z „prawdopodobieństwami":



Niezwykłą rzeczą jest to, że kiedy ChatGPT robi coś w rodzaju napisania eseju, w zasadzie robi tylko pytanie w kółko „biorąc pod uwagę dotychczasowy tekst, jakie powinno być następne słowo?" – i za każdym razem dodając słowo. (Dokładniej, jak wyjaśnię, jest to dodanie „znaku", który może być tylko częścią słowa, dlatego czasami może „wymyślać nowe słowa".) Ale OK, na każdym kroku dostaje lista słów z prawdopodobieństwem. Ale który właściwie powinien wybrać, aby dodać do eseju (lub czegokolwiek innego), który pisze? Można by pomyśleć, że powinno to być słowo „najwyżej ocenione" (tj. takie, któremu przypisano najwyższe „prawdopodobieństwo"). Ale tutaj zaczyna się wkradać odrobina voodoo. Ponieważ z jakiegoś powodu — być może pewnego dnia będziemy mieli naukową wiedzę — jeśli zawsze wybieramy najwyżej oceniane słowo, zazwyczaj otrzymujemy bardzo „płaski" esej, który nigdy nie wydaje się „wykazywać jakiejkolwiek kreatywności" (a czasem nawet powtarza słowo w słowo). Ale jeśli czasami (losowo) wybierzemy słowa o niższej randze, otrzymamy „bardziej interesujący" esej. Fakt, że występuje tutaj losowość, oznacza, że jeśli wielokrotnie użyjemy tego samego monitu, prawdopodobnie za każdym razem otrzymamy inny esej. Zgodnie z ideą voodoo istnieje szczególny tak zwany parametr „temperatury", który określa, jak często będą używane słowa o niższej randze, a do generowania esejów okazuje się, że „temperatura" wynosząca 0,8 wydaje się najlepsza. (Warto podkreślić, że nie ma tu żadnej „teorii"; chodzi tylko o to, co sprawdziło się w praktyce. I na przykład pojęcie „temperatury" istnieje, ponieważ rozkłady wykładnicze znane z fizyki statystycznej są używane , ale nie ma żadnego „fizycznego" połączenia — przynajmniej na tyle, na ile nam wiadomo). Zanim przejdziemy dalej, powinienem wyjaśnić, że dla celów informacyjnych przeważnie nie zamierzam używać pełnego systemu, który jest w ChatGPT; zamiast tego zwykle pracuję

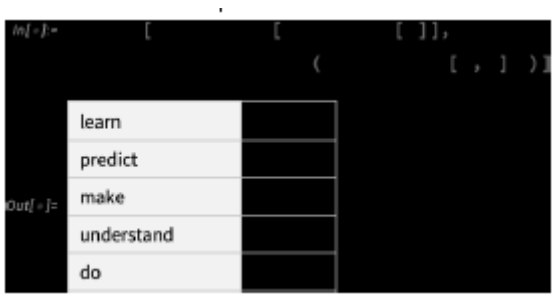
z prostszym systemem GPT-2, który ma tę fajną cechę, że jest wystarczająco mały, aby można go było uruchomić na standardowym komputerze stacjonarnym. I tak właściwie do wszystkiego, co pokażę, będę mógł dołączyć jawny kod Wolfram Language, który możesz natychmiast uruchomić na swoim komputerze. (Kliknij dowolne zdjęcie tutaj, aby skopiować kod za nim.) Na przykład, oto jak uzyskać powyższą tabelę prawdopodobieństw. Najpierw musimy odzyskać podstawową sieć neuronową „modelu języka”:

```
In[ ]:= model =
          NetModel[{"GPT2 Transformer Trained on WebText Data",
            "Task" → "LanguageModeling"}]

Out[ ]= NetChain[ Input port:  string
                  Output port: class ]
```

Później zajrzymy do wnętrza tej sieci neuronowej i porozmawiamy o jej działaniu. Ale na razie możemy po prostu zastosować ten „model netto” jako czarną skrzynkę do naszego dotychczasowego tekstu i poprosić o 5 najlepszych słów według prawdopodobieństwa, które zgodnie z modelem powinny być następujące:

To bierze ten wynik i zamienia go w jawnie sformatowany „zbiór danych”:



Oto co się stanie, jeśli wielokrotnie „zastosujemy model” — na każdym kroku dodamy słowo, które ma najwyższe prawdopodobieństwo (określone w tym kodzie jako „decyzja” z modelu):

Co się stanie, jeśli ktoś pojedzie dłużej? W tym przypadku („zero temperatury”) to, co pojawia się wkrótce, staje się raczej zagmatwane i powtarzalne:

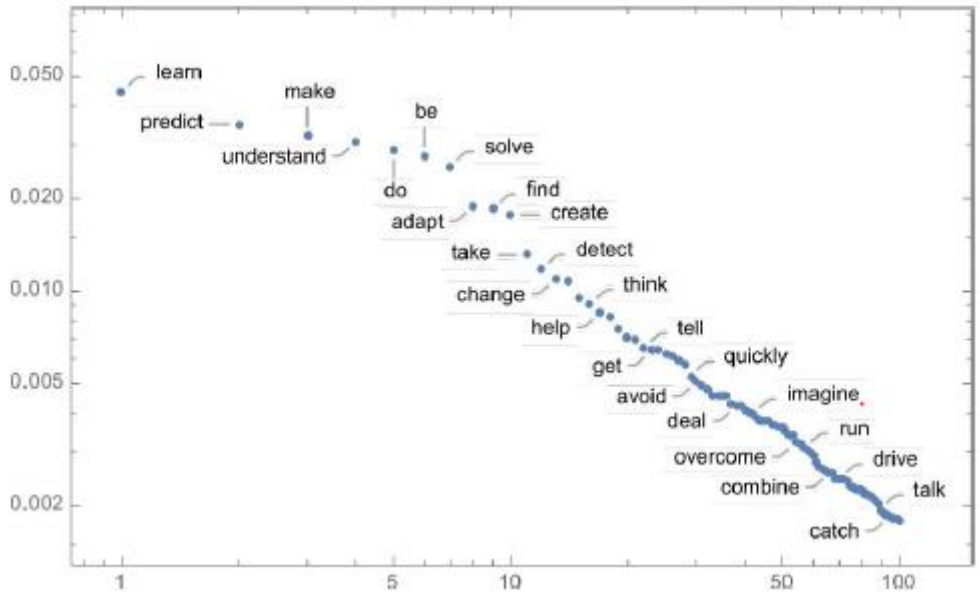
The best thing about AI is its ability to learn from experience. It's not just a matter of learning from experience, it's learning from the world around you. The AI is a very good example of this. It's a very good example of how to use AI to improve your life. It's a very good example of how to use AI to improve your life. The AI is a very good example of how to use AI to improve your life. It's a very good example of how to use AI to

Ale co, jeśli zamiast zawsze wybierać „górne" słowo, czasami losowo wybiera się słowa „nienajlepsze" (z „losowością" odpowiadającą „temperaturze" 0,8)? Ponownie można zbudować tekst:

{The best thing about AI is its ability to,
The best thing about AI is its ability to create,
The best thing about AI is its ability to create worlds,
The best thing about AI is its ability to create worlds that,
The best thing about AI is its ability to create worlds that are,
The best thing about AI is its ability to create worlds that are both,
The best thing about AI is its ability to create worlds that are both exciting,
The best thing about AI is its ability to create worlds that are both exciting,}

I za każdym razem, gdy ktoś to zrobi, zostaną dokonane różne losowe wybory, a tekst będzie inny - jak w tych 5 przykładach:

Warto zaznaczyć, że już na pierwszym etapie istnieje wiele możliwych „kolejnych słów" do wyboru (przy temperaturze 0,8), choć ich prawdopodobieństwa dość szybko spadają (i tak, prosta na tym wykresie loglog odpowiada rozpad „prawa potęgowego" n –1, który jest bardzo charakterystyczny dla ogólnej statystyki języka):



Co się stanie, jeśli ktoś pojedzie dłużej? Oto losowy przykład. Jest lepszy niż topowy przypadek (zero temperatury), ale nadal w najlepszym razie trochę dziwny:

The best thing about AI is its ability to see through, and make sense of, the world around us, rather than panicking and ignoring. This is known as AI "doing its job" or AI "run-of-the-mill." Indeed, taking an infinite number of steps, developing a machine that can be integrated with other systems, or controlling one system that's truly a machine, is one of the most fundamental processes of AI. Aside from the human-machine interaction, AI was also a big part of creativity

Dokonano tego za pomocą najprostszego modelu GPT-2 (z 2019 r.). Z nowszymi i większymi modelami GPT-3 wyniki są lepsze. Oto górne słowo (temperatura zerowa) utworzone z tym samym „monitem", ale z największym modelem GPT-3:

The best thing about AI is its ability to automate processes and make decisions quickly and accurately. AI can be used to automate mundane tasks, such as data entry, and can also be used to make complex decisions, such as predicting customer behavior or analyzing large datasets. AI can also be used to improve customer service, as it can quickly and accurately respond to customer inquiries. AI can also be used to improve the accuracy of medical diagnoses and to automate the process of drug discovery.

A oto losowy przykład w „temperaturze 0,8":

The best thing about AI is its ability to learn and develop over time, allowing it to continually improve its performance and be more efficient at tasks. AI can also be used to automate mundane tasks, allowing humans to focus on more important tasks. AI can also be used to make decisions and provide insights that would otherwise be impossible for humans to figure out.

Skąd się biorą prawdopodobieństwa?

OK, więc ChatGPT zawsze wybiera następne słowo na podstawie prawdopodobieństwa. Skąd jednak biorą się te prawdopodobieństwa? Zacznijmy od prostszego problemu. Rozważmy generowanie tekstu w języku angielskim po jednej literze (a nie słowie) na raz. Jak możemy ustalić, jakie powinno być prawdopodobieństwo dla każdej litery? Bardzo minimalną rzeczą, jaką możemy zrobić, jest po prostu pobranie próbki tekstu w języku angielskim i obliczenie, jak często występują w nim różne litery. Na przykład liczy się litery w artykule Wikipedii na temat „kotów":

```
In[ ]:= LetterCounts[WikipediaData["cats"]]

Out[ ]= ⟨| e → 4279, a → 3442, t → 3397, i → 2739, s → 2615, n → 2464, o → 2426,
  r → 2147, h → 1613, l → 1552, c → 1405, d → 1331, m → 989, u → 916,
  f → 760, g → 745, p → 651, y → 591, b → 511, w → 509, v → 395, k → 212,
  T → 114, x → 85, A → 81, C → 81, I → 68, S → 55, F → 42, z → 38, E → 36
```

I to samo dotyczy „psów":

```
In[ ]:= LetterCounts[WikipediaData["dogs"]]

Out[ ]= ⟨| e → 3911, a → 2741, o → 2608, i → 2562, t → 2528, s → 2406,
  n → 2340, r → 1866, d → 1584, h → 1463, l → 1355, c → 1083, g → 929,
  m → 859, u → 782, f → 662, p → 636, y → 500, b → 462, w → 409,
  v → 406, k → 151, T → 90, C → 85, I → 80, A → 74, x → 71, S → 65,
```

Wyniki są podobne, ale nie takie same („o" jest bez wątpienia bardziej powszechne w artykule „psy", ponieważ występuje przecież w samym słowie „pies"). Mimo to, jeśli weźmiemy wystarczająco dużą próbkę tekstu w języku angielskim, możemy spodziewać się ostatecznie uzyskania co najmniej dość spójnych wyników:

```
In[ ]:= English LANGUAGE [ character frequencies ]

Out[ ]= {e → 12.7%, t → 9.06%, a → 8.17%, o → 7.51%, i → 6.97%, n → 6.75%,
  s → 6.33%, h → 6.09%, r → 5.99%, d → 4.25%, l → 4.03%, c → 2.78%, u → 2.76%,
  m → 2.41%, w → 2.36%, f → 2.23%, g → 2.02%, y → 1.97%, p → 1.93%, b → 1.49%,
  v → 0.978%, k → 0.772%, j → 0.153%, x → 0.150%, q → 0.0950%, z → 0.0740%}
```

Oto próbka tego, co otrzymamy, jeśli po prostu wygenerujemy sekwencję liter z tymi prawdopodobieństwami:

```
rronoitadatcaeaesaotdoysaroiyiinnbantoioestlhddeocneooewceseciselnodrtrdgriscsatsepesdcnio
    uhoetsedeyhedslernevstothindtbmnaohngotannbthrdthtonsipieldn
```

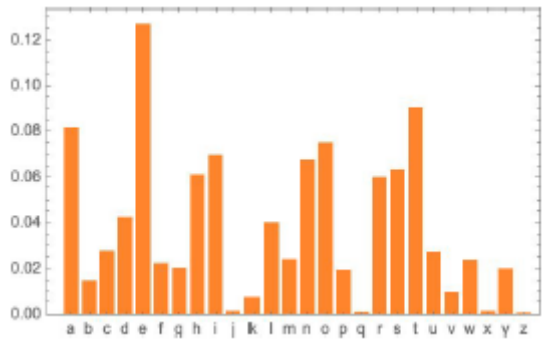
Możemy podzielić to na „słowa”, dodając spacje tak, jakby były literami z pewnym prawdopodobieństwem:

```
sd n oeiaim satnwhoo eer rtr ofiianordrenapwokom del oaas ill e h f
    rellptohltvoettseodtrncilntehtotrkthrslo hdaol n sriaefr hthehtn ld gpod a  h y oi
```

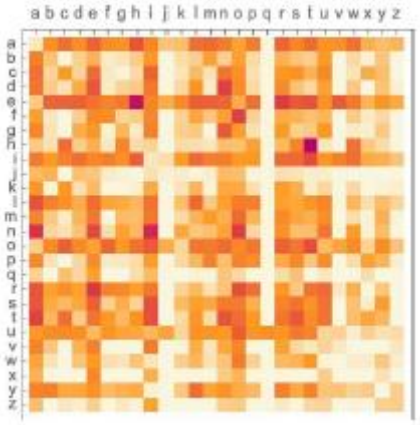
Możemy wykonać nieco lepszą robotę, tworząc „słowa”, zmuszając rozkład „długości słów”, aby zgadzał się z tym, co jest w języku angielskim:

```
ni hilwhuei kjtn isjd erogofnr n rwhwfao rcuw lis fahte uss cpnc
    nlu oe nusaetat llfo oeme rrhrtn xdses ohm oa tne ebedcon oarvthv ist
```

Nie otrzymaliśmy tutaj żadnych „rzeczywistych słów”, ale wyniki wyglądają nieco lepiej. Aby jednak pójść dalej, musimy zrobić coś więcej niż tylko losowo wybrać każdą literę z osobna. I na przykład wiemy, że jeśli mamy „q”, następną literą w zasadzie musi być „u”. Oto wykres prawdopodobieństw dla samych liter:



A oto wykres, który pokazuje prawdopodobieństwa par liter („2-gramów”) typowym angielskim tekstem. Możliwe pierwsze litery są pokazane w poprzek strony, drugie litery w dół strony:



Widzimy tutaj na przykład, że kolumna „q” jest pusta (prawdopodobieństwo zerowe), z wyjątkiem wiersza „u”. OK, więc teraz zamiast generować nasze „słowa” po jednej literze naraz, wygenerujmy je, patrząc na dwie litery na raz, używając tych „2-gramowych” prawdopodobieństw. Oto próbka wyniku — która zawiera kilka „rzeczywistych zamówień”:

on inguman men ise forerenoft weat iofobato buc ous corew ousesetiv
falle tinouco ryefo ra the ecederi pasuthrgr cuconom tra tesla wil tat pere thi

Przy wystarczającej ilości tekstu w języku angielskim możemy uzyskać całkiem dobre oszacowania nie tylko prawdopodobieństw pojedynczych liter lub par liter (2 gramy), ale także dłuższych ciągów liter. A jeśli generujemy „losowe słowa” z coraz dłuższymi n-gramowymi prawdopodobieństwami, widzimy, że stają się one stopniowo „bardziej realistyczne”:

Ale załóżmy teraz — mniej więcej tak, jak robi to ChatGPT — że mamy do czynienia z całymi słowami, a nie z literami. Istnieje około 40 000 powszechnie używanych słów w języku angielskim. A patrząc na duży zbiór tekstów w języku angielskim (powiedzmy kilka milionów książek, zawierających w sumie kilkaset miliardów słów), możemy oszacować, jak powszechne jest każde słowo. Korzystając z tego, możemy zacząć generować „zdania”, w których każde słowo jest niezależnie wybierane losowo, z takim samym prawdopodobieństwem, jak występuje w korpusie. Oto próbka tego, co otrzymujemy:

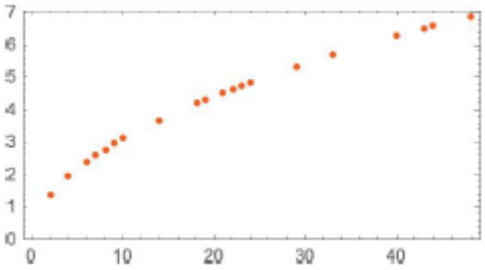
of program excessive been by was research rate not here of of other is men
were against are show they the different the half the the in any were leaved

Nic dziwnego, to nonsens. Jak więc możemy działać lepiej? Podobnie jak w przypadku liter, możemy zacząć brać pod uwagę nie tylko prawdopodobieństwa pojedynczych słów, ale także prawdopodobieństwa par lub dłuższych n-gramów słów. Robiąc to dla par, oto 5 przykładów tego, co otrzymujemy, we wszystkich przypadkach zaczynając od słowa „kot”:

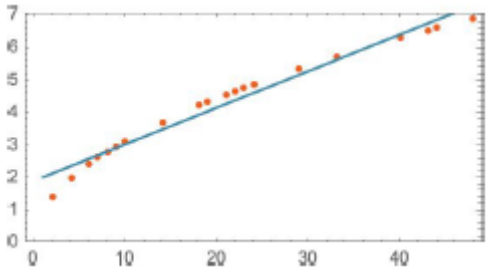
Robi się nieco bardziej „rozsądnie”. I możemy sobie wyobrazić, że gdybyśmy byli w stanie użyć wystarczająco długich n-gramów, w zasadzie „otrzymalibyśmy ChatGPT” — w tym sensie, że otrzymalibyśmy coś, co wygenerowałoby sekwencje słów o długości eseju z „poprawnym ogólnym esejem prawdopodobieństwa”. Ale oto problem: po prostu nie ma nawet wystarczająco dużo tekstu w języku angielskim, który kiedykolwiek został napisany, aby móc wydedukować te prawdopodobieństwa. Podczas przeszukiwania sieci może być kilkaset miliardów słów; w książkach, które zostały zdigitalizowane, może być jeszcze sto miliardów słów. Ale przy 40 000 pospolitych słów nawet liczba możliwych 2 gramów wynosi już 1,6 miliarda, a liczba możliwych 3 gramów wynosi 60 bilionów. Więc nie ma sposobu, abyśmy mogli oszacować prawdopodobieństwa nawet dla wszystkich z nich na podstawie tekstu, który tam jest. A zanim dojdziemy do „fragmentów eseju” składających się z 20 słów, liczba możliwości jest większa niż liczba cząstek we wszechświecie, więc w pewnym sensie nigdy nie udałoby się ich wszystkich zapisać. Więc co możemy zrobić? Główną ideą jest stworzenie modelu, który pozwoli nam oszacować prawdopodobieństwo, z jakim powinny wystąpić sekwencje — mimo że nigdy nie widzieliśmy wyraźnie tych sekwencji w korpusie tekstu, który przeglądaliśmy. Istotą ChatGPT jest właśnie tak zwany „duży model językowy” (LLM), który został zbudowany, aby dobrze oszacować te prawdopodobieństwa.

Co to jest model?

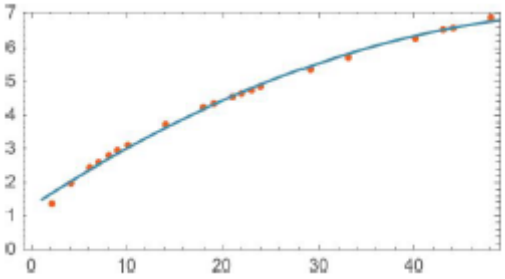
Załóżmy, że chcesz wiedzieć (tak jak Galileusz pod koniec XVI wieku), ile czasu zajmie uderzenie w ziemię kuli armatniej zrzuconej z każdego piętra wieży w Pizie. Cóż, można po prostu zmierzyć to w każdym przypadku i sporządzić tabelę wyników. Albo możesz zrobić to, co jest istotą nauk teoretycznych: stworzyć model, który daje jakąś procedurę obliczania odpowiedzi, a nie tylko mierzenie i zapamiętywanie każdego przypadku. Wyobraźmy sobie, że mamy (nieco wyidealizowane) dane dotyczące czasu spadania kuli armatniej z różnych pięter:



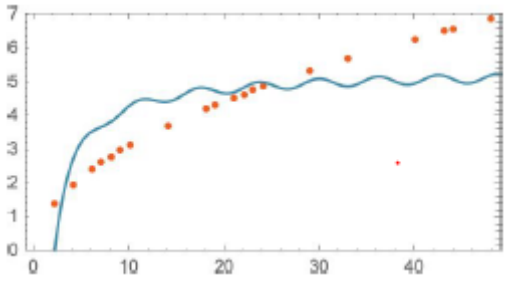
Jak dowiedzieć się, ile czasu zajmie upadek z podłogi, o której nie mamy wyraźnych danych? W tym konkretnym przypadku możemy wykorzystać znane prawa fizyki, aby to rozpracować. Ale powiedzmy, że wszystko, co mamy, to dane i nie wiemy, jakie prawa nim rządzą. Następnie możemy dokonać matematycznego przypuszczenia, na przykład powinniśmy użyć linii prostej jako modelu:



Moglibyśmy wybrać różne linie proste. Ale to jest ten, który jest średnio najbliższy danym, które otrzymujemy. I z tej prostej możemy oszacować czas upadku na dowolne piętro. Skąd wiedzieliśmy, że należy spróbować użyć tu linii prostej? Na pewnym poziomie nie. Jest to po prostu coś, co jest matematycznie proste i jesteśmy przyzwyczajeni do tego, że wiele danych, które mierzymy, okazuje się dobrze pasować dzięki matematycznie prostym rzeczom. Moglibyśmy spróbować czegoś bardziej skomplikowanego matematycznie — powiedzmy a + bx + cx^2 - i wtedy w tym przypadku zrobimy to lepiej:



Jednak sprawy mogą pójść całkiem nie tak. Oto najlepsze, co możemy zrobić z a + b /x + c sin(x):



Warto zrozumieć, że nigdy nie ma „modelu bez modelu”. Każdy model, którego używasz, ma określoną strukturę bazową — a następnie określony zestaw „pokręteł, którymi możesz obracać” (tj. parametry, które możesz ustawić), aby dopasować je do danych. A w przypadku ChatGPT używa się wielu takich „pokręteł” – w rzeczywistości jest ich 175 miliardów. Ale niezwykłe jest to, że podstawowa struktura ChatGPT — z „zaledwie” tak wieloma parametrami — jest wystarczająca do stworzenia modelu, który oblicza prawdopodobieństwa następnego słowa „wystarczająco dobrze”, aby dać nam rozsądne fragmenty tekstu o długości eseju.

Modele zadań podobnych do ludzkich

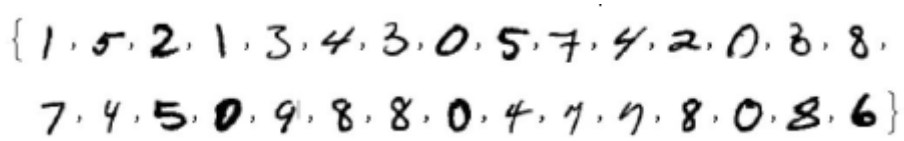
Przykład, który podaliśmy powyżej, polega na stworzeniu modelu danych liczbowych, który zasadniczo pochodzi z prostej fizyki – gdzie od kilku stuleci wiemy, że „stosuje się prosta matematyka”. Ale dla ChatGPT musimy stworzyć model tekstu w języku ludzkim, takiego jaki wytwarza ludzki mózg. A dla czegoś takiego nie mamy (przynajmniej na razie) czegoś takiego jak „prosta matematyka”. Jaki więc może być jej model? Zanim porozmawiamy o języku, porozmawiajmy o innym ludzkim zadaniu:

rozpoznawaniu obrazów. Jako prosty przykład rozważmy obrazy cyfr (i tak, to jest klasyczny przykład uczenia maszynowego):

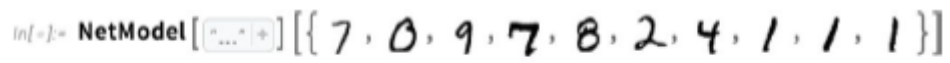
Jedną rzeczą, którą moglibyśmy zrobić, to pobrać kilka przykładowych obrazów dla każdej cyfry:

Następnie, aby dowiedzieć się, czy obraz, który otrzymaliśmy jako dane wejściowe, odpowiada a

konkretnej cyfry, moglibyśmy po prostu dokonać wyraźnego porównania piksel po pikselu z próbkami, które mamy. Ale jako ludzie z pewnością wydajemy się robić coś lepiej – ponieważ nadal możemy rozpoznawać cyfry, nawet jeśli są na przykład pisane odręcznie i mają wszelkiego rodzaju modyfikacje i zniekształcenia:

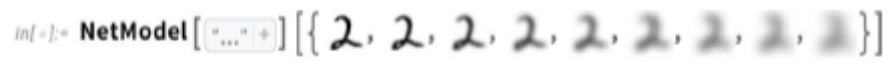


Kiedy tworzyliśmy model dla naszych danych liczbowych powyżej, byliśmy w stanie przyjąć otrzymaną wartość liczbową x i po prostu obliczyć a + b x dla poszczególnych a i b . Jeśli więc potraktujemy tutaj wartość poziomu szarości każdego piksela jako jakąś zmienną xi, czy istnieje jakaś funkcja wszystkich tych zmiennych, która — po ocenie — mówi nam, z jakiej cyfry pochodzi obraz? Okazuje się, że można skonstruować taką funkcję. Nic dziwnego, nie jest to jednak szczególnie proste. A typowy przykład może obejmować około pół miliona operacji matematycznych. Ale wynik końcowy jest taki, że jeśli podamy zbiór wartości pikseli dla obrazu do tej funkcji, wyjdzie liczba określająca, której cyfry mamy obraz. Później porozmawiamy o tym, jak można skonstruować taką funkcję, oraz o idei sieci neuronowych. Ale na razie potraktujmy tę funkcję jako czarną skrzynkę, do której wprowadzamy obrazy, powiedzmy, odręcznie zapisanych cyfr (jako tablice wartości pikseli) i uzyskujemy odpowiadające im liczby:



Out[ ]= {7, 0, 9, 7, 8, 2, 4, 1, 1, 1}

Ale co tak naprawdę się tutaj dzieje? Powiedzmy, że stopniowo zamazujemy cyfrę. Jeszcze przez chwilę nasza funkcja „rozpoznaje” to, tutaj jako „2”. Ale wkrótce „gubi się” i zaczyna dawać „zły” wynik:



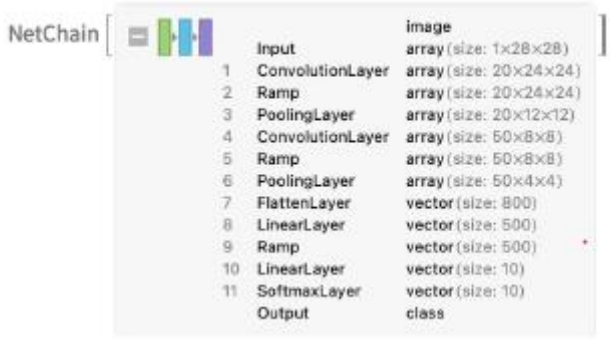
Out[ ]= {2, 2, 2, 1, 1, 1, 1, 1, 1}

Ale dlaczego mówimy, że to „zły” wynik? W tym przypadku wiemy, że uzyskaliśmy wszystkie obrazy, rozmywając „2”. Ale jeśli naszym celem jest stworzenie modelu tego, co ludzie mogą zrobić w rozpoznawaniu obrazów, prawdziwym pytaniem, które należy zadać, jest to, co człowiek zrobiłby, gdyby pokazano mu jeden z tych niewyraźnych obrazów, nie wiedząc, skąd on pochodzi. I mamy „dobry

model”, jeśli wyniki, które uzyskujemy z naszej funkcji, zazwyczaj zgadzają się z tym, co powiedziałby człowiek. A nietrywialnym faktem naukowym jest to, że dla zadania rozpoznawania obrazu takiego jak to, teraz w zasadzie wiemy, jak konstruować funkcje, które to robią. Czy możemy „udowodnić matematycznie”, że działają? Więc nie. Ponieważ aby to zrobić, musielibyśmy mieć matematyczną teorię tego, co my, ludzie, robimy. Weź obraz „2” i zmień kilka pikseli. Możemy sobie wyobrazić, że mając tylko kilka pikseli „nie na miejscu”, nadal powinniśmy uważać obraz za „2”. Ale jak daleko to powinno zajść? To kwestia ludzkiej percepcji wzrokowej. I tak, odpowiedź bez wątpienia byłaby inna w przypadku pszczół lub ośmiornic — i potencjalnie zupełnie inna w przypadku domniemanych kosmitów.

Sieci neuronowe

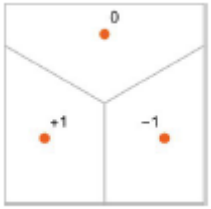
OK, więc jak właściwie działają nasze typowe modele do zadań takich jak rozpoznawanie obrazu? Najbardziej popularne i odnoszące sukcesy obecnie podejście wykorzystuje sieci neuronowe. Sieci neuronowe, wynalezione w latach czterdziestych XX wieku w formie niezwykle zbliżonej do ich dzisiejszego zastosowania, można traktować jako proste idealizacje tego, jak wydaje się działać mózg. W ludzkim mózgu znajduje się około 100 miliardów neuronów (komórek nerwowych), z których każda jest w stanie wytworzyć impuls elektryczny z częstotliwością nawet tysiąca razy na sekundę. Neurony są połączone w skomplikowaną sieć, a każdy neuron ma gałęzie przypominające drzewo, co pozwala mu przekazywać sygnały elektryczne do prawdopodobnie tysięcy innych neuronów. W przybliżeniu to, czy dany neuron wytworzy impuls elektryczny w danym momencie, zależy od tego, jakie impulsy otrzyma od innych neuronów — z różnymi połączeniami o różnych „wagach”. Kiedy „widzimy obraz”, dzieje się tak, że kiedy fotony światła z obrazu padają na komórki („fotoreceptory”) z tyłu naszych oczu, wytwarzają sygnały elektryczne w komórkach nerwowych. Te komórki nerwowe są połączone z innymi komórkami nerwowymi i ostatecznie sygnały przechodzą przez całą sekwencję warstw neuronów. I właśnie w tym procesie „rozpoznajemy” obraz, ostatecznie „formując myśl”, że „widzimy 2” (i być może w końcu robimy coś w rodzaju wypowiedzenia słowa „dwa” na głos). Funkcja „czarnej skrzynki” z poprzedniej sekcji jest „zmatematyzowaną” wersją takiej sieci neuronowej. Tak się składa, że ma 11 warstw (choć tylko 4 „warstwy podstawowe”):



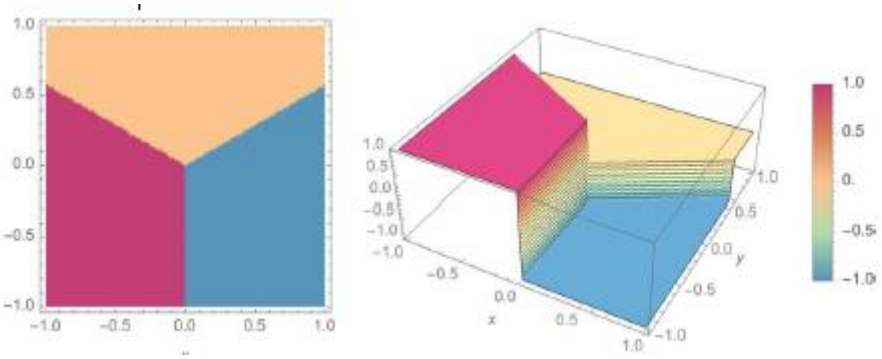
W tej sieci neuronowej nie ma nic szczególnie „wyprowadzonego teoretycznie”; to po prostu coś, co — jeszcze w 1998 roku — zostało skonstruowane jako element inżynierii i okazało się, że działa. (Oczywiście niewiele różni się to od tego, jak moglibyśmy opisać nasze mózgi jako wytworzone w procesie ewolucji biologicznej). OK, ale w jaki sposób taka sieć neuronowa „rozpoznaje rzeczy”? Kluczem jest pojęcie atraktorów. Wyobraź sobie, że mamy odręcznie napisane obrazy 1 i 2:

W jakiś sposób chcemy, aby wszystkie jedynki „były przyciągane do jednego miejsca”, a wszystkie dwójki „były przyciągane do innego miejsca”. Innymi słowy, jeśli obraz jest w jakiś sposób „bliższy 1” niż 2, chcemy, aby znalazł się na „1 miejscu” i odwrotnie. Jako prostą analogię, powiedzmy, że mamy pewne pozycje na płaszczyźnie, oznaczone kropkami (w prawdziwym życiu mogą to być pozycje kawiarni). Wtedy możemy sobie wyobrazić, że zaczynając od dowolnego punktu na płaszczyźnie, zawsze chcielibyśmy skończyć w najbliższym punkcie (tj. zawsze byśmy szli do najbliższej kawiarni). Możemy to przedstawić, dzieląc płaszczyznę na regiony („baseny atraktorów”) oddzielone wyidealizowanymi „działami wodnymi”:

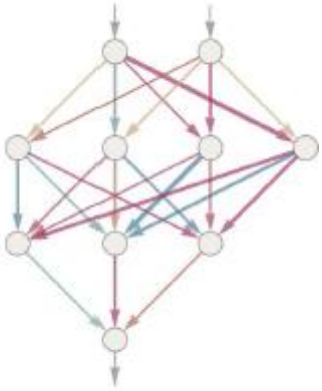
Możemy myśleć o tym jako o wykonaniu pewnego rodzaju „zadania rozpoznawania”, w którym nie robimy czegoś w rodzaju identyfikowania, która cyfra danego obrazu „wygląda najbardziej” — ale raczej po prostu, całkiem bezpośrednio, widzimy, jaka kropka na danym obrazie punkt jest najbliżej. (Konfiguracja „diagramu Woronoja”, którą tutaj pokazujemy, oddziela punkty w dwuwymiarowej przestrzeni euklidesowej; zadanie rozpoznawania cyfr można traktować jako wykonywanie czegoś bardzo podobnego — ale w 784-wymiarowej przestrzeni utworzonej z poziomów szarości wszystkich pikseli w każdego obrazu.) Jak więc sprawić, by sieć neuronowa „wykonała zadanie rozpoznawania”? Rozważmy ten bardzo prosty przypadek:



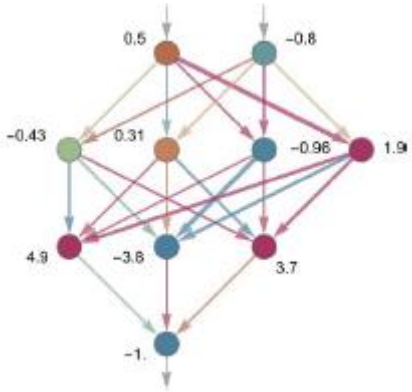
Naszym celem jest pobranie „danych wejściowych” odpowiadających pozycji {x ,y } - a następnie „rozpoznanie” ich jako dowolnego z trzech punktów, do których jest najbliżej. Innymi słowy, chcemy, aby sieć neuronowa obliczała funkcję {x ,y } taką jak:



Jak więc to zrobić za pomocą sieci neuronowej? Ostatecznie sieć neuronowa jest połączonym zbiorem wyidealizowanych „neuronów” — zwykle ułożonych warstwami — na przykład:

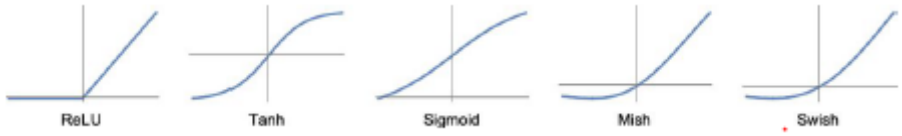


Każdy „neuron” jest skutecznie skonfigurowany do oceny prostej funkcji numerycznej. Aby „korzystać” z sieci, po prostu wprowadzamy liczby (takie jak nasze współrzędne x i y) na górze, a następnie neurony w każdej warstwie „oceniają swoje funkcje” i przesyłają wyniki dalej przez sieć — ostatecznie generując końcowy wynik na dnie:



W tradycyjnym (inspirowanym biologicznie) układzie każdy neuron faktycznie ma określony zestaw „połączeń przychodzących” z neuronów na poprzedniej warstwie, przy czym każde połączenie ma przypisaną określoną „wagę” (która może być liczbą dodatnią lub ujemną). Wartość danego neuronu określa się, mnożąc wartości „poprzednich neuronów” przez odpowiadające im wagi, a następnie dodając je i dodając stałą, a na końcu stosując funkcję „progowania” (lub „aktywacji”). W kategoriach matematycznych, jeśli neuron ma wejścia x = {x 1 , x 2 …}, to obliczamy f [w . x + b], gdzie wagi w i stała b są generalnie dobierane inaczej dla każdego neuronu w sieci; funkcja f jest zwykle taka sama. Informatyka W . x + b to tylko kwestia mnożenia i dodawania macierzy. „Funkcja aktywacji” f

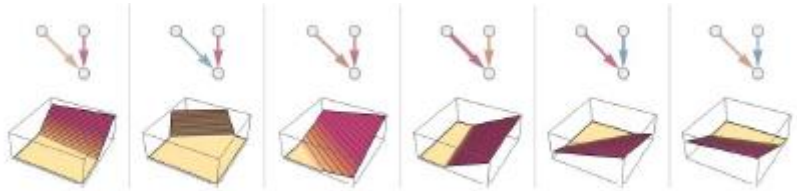
wprowadza nieliniowość (i ostatecznie prowadzi do nietrywialnego zachowania). Często używane są różne funkcje aktywacji; tutaj użyjemy Ramp (lub ReLU):



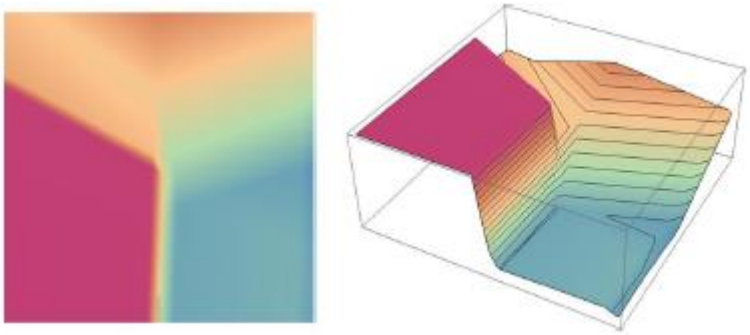
Dla każdego zadania, które ma wykonać sieć neuronowa (lub, równoważnie, dla każdej ogólnej funkcji, którą chcemy, aby oceniała) będziemy mieć różne wagi do wyboru. (I — jak omówimy to później — te wagi są zwykle określane przez „uczenie" sieci neuronowej przy użyciu uczenia maszynowego na przykładach żądanych danych wyjściowych). Ostatecznie każda sieć neuronowa odpowiada po prostu jakiejś ogólnej funkcji matematycznej — choć może być niechlujnym w pisaniu. Dla powyższego przykładu byłoby to:

$$\begin{aligned}
&w_{511} f(w_{311} f(b_{11} + x\,w_{111} + y\,w_{112}) + w_{312} f(b_{12} + x\,w_{121} + y\,w_{122}) + \\
&\quad w_{313} f(b_{13} + x\,w_{131} + y\,w_{132}) + w_{314} f(b_{14} + x\,w_{141} + y\,w_{142}) + b_{31}) + \\
&w_{512} f(w_{321} f(b_{11} + x\,w_{111} + y\,w_{112}) + w_{322} f(b_{12} + x\,w_{121} + y\,w_{122}) + \\
&\quad w_{323} f(b_{13} + x\,w_{131} + y\,w_{132}) + w_{324} f(b_{14} + x\,w_{141} + y\,w_{142}) + b_{32}) + \\
&w_{513} f(w_{331} f(b_{11} + x\,w_{111} + y\,w_{112}) + w_{332} f(b_{12} + x\,w_{121} + y\,w_{122}) + \\
&\quad w_{333} f(b_{13} + x\,w_{131} + y\,w_{132}) + w_{334} f(b_{14} + x\,w_{141} + y\,w_{142}) + b_{33}) + b_{51}
\end{aligned}$$

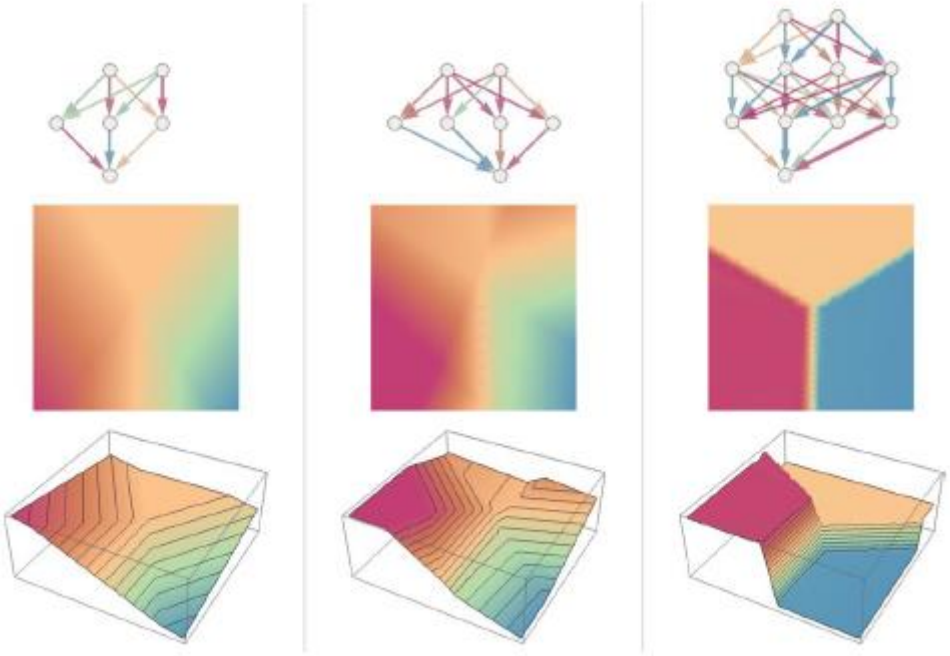
Sieć neuronowa ChatGPT również odpowiada funkcji matematycznej takiej jak ta — ale w rzeczywistości z miliardami terminów. Wróćmy jednak do poszczególnych neuronów. Oto kilka przykładów funkcji, które neuron z dwoma wejściami (reprezentującymi współrzędne x i y) może obliczyć z różnymi wyborami wag i stałych (oraz Ramp jako funkcja aktywacji):



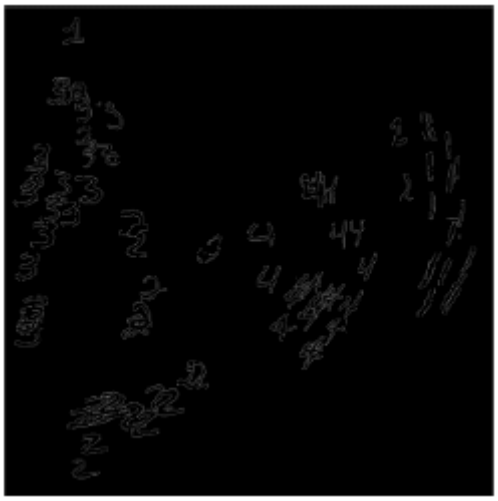
Ale co z większą siecią widzianą z góry? Cóż, oto co oblicza:



Nie jest to całkiem „właściwe", ale jest zbliżone do funkcji „najbliższego punktu", którą pokazaliśmy powyżej. Zobaczmy, co dzieje się z innymi sieciami neuronowymi. W każdym przypadku, jak wyjaśnimy później, używamy uczenia maszynowego, aby znaleźć najlepszy wybór wag. Następnie pokazujemy tutaj, co oblicza sieć neuronowa z tymi wagami:



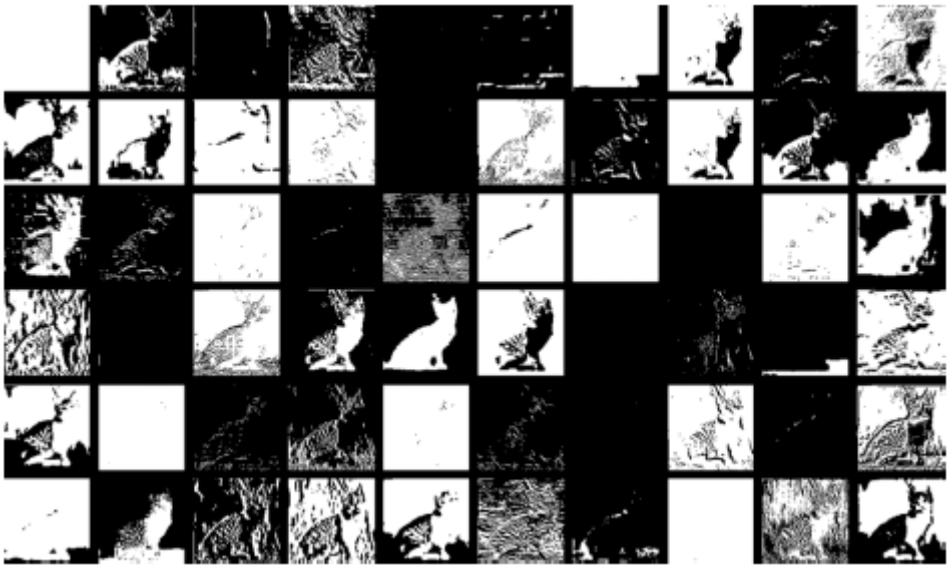
Większe sieci generalnie lepiej radzą sobie z aproksymacją funkcji, do której dążymy. A w „środku każdego basenu atraktora” zwykle otrzymujemy dokładnie taką odpowiedź, jakiej oczekujemy. Ale na granicach – gdzie sieć neuronowa „ma trudności z podjęciem decyzji” – sprawy mogą być bardziej chaotyczne. Dzięki temu prostemu „zadaniu rozpoznawania” w stylu matematycznym jasne jest, jaka jest „właściwa odpowiedź”. Ale w problemie rozpoznawania cyfr pisanych odręcznie nie jest to takie oczywiste. Co jeśli ktoś napisał „2” tak źle, że wyglądało to jak „7” itp.? Mimo to możemy zapytać, w jaki sposób sieć neuronowa rozróżnia cyfry – i to daje wskazówkę:



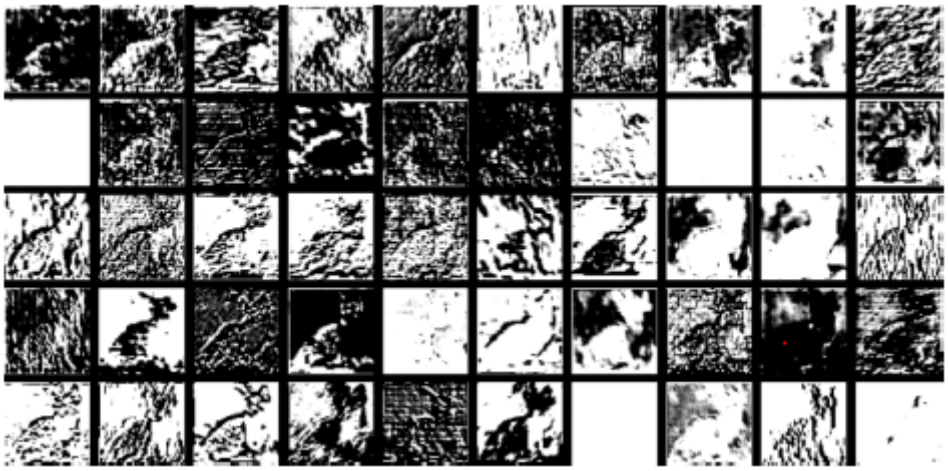
Czy możemy powiedzieć „matematycznie”, w jaki sposób sieć dokonuje rozróżnień? Nie bardzo. Po prostu „robi to, co robi sieć neuronowa”. Ale okazuje się, że zwykle wydaje się to dość dobrze zgadzać z rozróżnieniami, które czynimy my, ludzie. Weźmy bardziej rozbudowany przykład. Załóżmy, że mamy obrazy kotów i psów. Mamy też sieć neuronową, która została wyszkolona, by je rozróżniać. Oto, co może zrobić na kilku przykładach:

Teraz jest jeszcze mniej jasne, jaka jest „właściwa odpowiedź”. Co powiesz na psa przebranego w strój kota? Itd. Bez względu na to, jakie dane wejściowe zostaną podane, sieć neuronowa generuje odpowiedź. I, jak się okazuje, robić to w sposób, który jest w miarę spójny z tym, co mogą zrobić ludzie. Jak powiedziałem powyżej, nie jest to fakt, który możemy „wyprowadzić z pierwszych zasad”. To po prostu coś, co zostało empirycznie potwierdzone, przynajmniej w niektórych dziedzinach. Ale jest to kluczowy powód, dla którego sieci neuronowe są przydatne: że w jakiś sposób wychwytują „ludzki” sposób robienia rzeczy. Pokaż sobie zdjęcie kota i zapytaj „Dlaczego to jest kot?”. Może zacząłbyś mówić: „Cóż, widzę jego spiczaste uszy itp.”. Ale nie jest łatwo wyjaśnić, w jaki sposób rozpoznałeś obraz jako kota. Po prostu twój mózg w jakiś sposób to rozgryzł. Ale dla mózgu nie ma możliwości (przynajmniej na razie) „wejść do środka” i zobaczyć, jak to rozgryzł. A co z (sztuczną) siecią neuronową? Cóż, łatwo zobaczyć, co robi każdy „neuron”, gdy pokazujesz zdjęcie kota. Ale nawet uzyskanie podstawowej wizualizacji jest zwykle bardzo trudne. W ostatecznej sieci, której użyliśmy do rozwiązania problemu „najbliższego punktu” powyżej, znajduje się 17 neuronów. W sieci do rozpoznawania cyfr odręcznych jest ich 2190. A w sieci, w której rozpoznajemy psy i koty, jest ich 60 650. Normalnie byłoby dość trudno wyobrazić sobie, co odpowiada przestrzeni 60 650 wymiarów. Ale ponieważ jest to sieć skonfigurowana do obsługi obrazów, wiele jej warstw neuronów jest zorganizowanych w tablice, takie jak tablice pikseli, na które patrzy. A jeśli weźmiemy typowy obraz kota

wtedy możemy przedstawić stany neuronów w pierwszej warstwie za pomocą zbioru pochodnych obrazów — z których wiele możemy z łatwością zinterpretować jako rzeczy takie jak „kot bez tła” lub „zarys kota”:



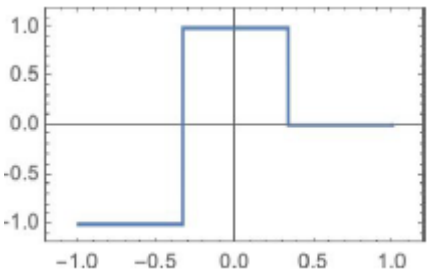
Przy 10. warstwie trudniej jest zinterpretować, co się dzieje:



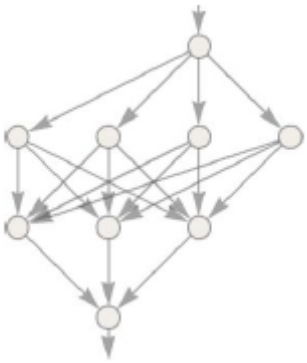
Ale ogólnie możemy powiedzieć, że sieć neuronowa „wyłapuje pewne cechy” (być może są wśród nich spiczaste uszy) i używa ich do określenia, z czego jest obraz. Ale czy te cechy są tymi, dla których mamy nazwy, takie jak „spiczaste uszy”? Przeważnie nie. Czy nasze mózgi używają podobnych funkcji? Najczęściej nie wiemy. Warto jednak zauważyć, że kilka pierwszych warstw sieci neuronowej, takiej jak ta, którą tutaj pokazujemy, wydaje się wybierać aspekty obrazów (takie jak krawędzie obiektów), które wydają się być podobne do tych, o których wiemy, że są wybierane przez pierwszy poziom przetwarzanie wizualne w mózgu. Ale powiedzmy, że chcemy „teorii rozpoznawania kotów” w sieciach neuronowych. Możemy powiedzieć: „Spójrz, ta konkretna sieć to robi” — i od razu daje nam to poczucie „jak trudny jest to problem” (i na przykład, ile neuronów lub warstw może być potrzebnych). Ale przynajmniej na razie nie mamy sposobu na „narracyjny opis” tego, co robi sieć. A może dlatego, że naprawdę jest obliczeniowo nieredukowalny i nie ma ogólnego sposobu, aby dowiedzieć się, co robi, z wyjątkiem wyraźnego śledzenia każdego kroku. A może po prostu nie „rozgryzliśmy nauki” i nie zidentyfikowaliśmy „praw naturalnych”, które pozwalają nam podsumować to, co się dzieje. Napotkamy te same problemy, gdy będziemy mówić o generowaniu języka za pomocą ChatGPT. I znowu nie jest jasne, czy istnieją sposoby na „podsumowanie tego, co robi”. Ale bogactwo i szczegółowość języka (oraz nasze doświadczenie z nim) może pozwolić nam zajść dalej niż za pomocą obrazów.

Uczenie maszynowe i trenowanie sieci neuronowych

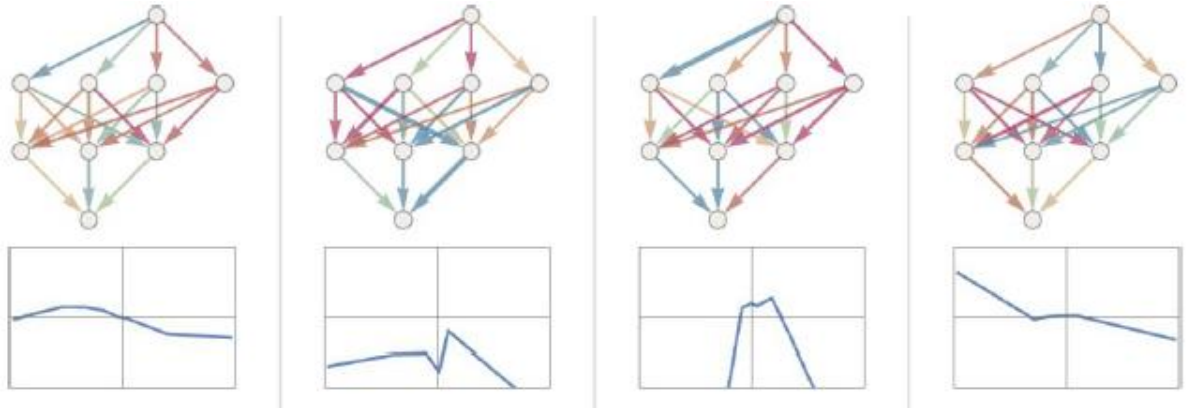
Do tej pory mówiliśmy o sieciach neuronowych, które „już wiedzą”, jak wykonywać określone zadania. Ale to, co sprawia, że sieci neuronowe są tak przydatne (prawdopodobnie także w mózgach), polega na tym, że nie tylko mogą one zasadniczo wykonywać wszelkiego rodzaju zadania, ale można je stopniowo „szkolić na podstawie przykładów”, aby wykonywać te zadania. Kiedy tworzymy sieć neuronową do odróżniania kotów od psów, nie musimy skutecznie pisać programu, który (powiedzmy) wyraźnie znajduje wąsy; zamiast tego po prostu pokazujemy wiele przykładów tego, czym jest kot, a czym jest pies, a następnie każemy sieci „uczyć się” na ich podstawie, jak je rozróżnić. Chodzi o to, że wyszkolona sieć „uogólnia” na podstawie konkretnych przykładów, które pokazuje. Jak widzieliśmy powyżej, nie chodzi tylko o to, że sieć rozpoznaje konkretny wzór piksela przykładowego obrazu kota, który został wyświetlony; chodzi raczej o to, że sieć neuronowa w jakiś sposób rozróżnia obrazy na podstawie tego, co uważamy za swego rodzaju „ogólną kociątość”. Jak więc właściwie działa trening sieci neuronowej? Zasadniczo zawsze staramy się znaleźć wagi, które sprawią, że sieć neuronowa pomyślnie odtworzy podane przez nas przykłady. Następnie polegamy na sieci neuronowej, aby „interpolować” (lub „uogólniać”) „między” tymi przykładami w „rozsądny” sposób. Spójrzmy na problem jeszcze prostszy niż najbliższy punkt powyżej. Spróbujmy po prostu uzyskać sieć neuronową, aby nauczyć się funkcji:



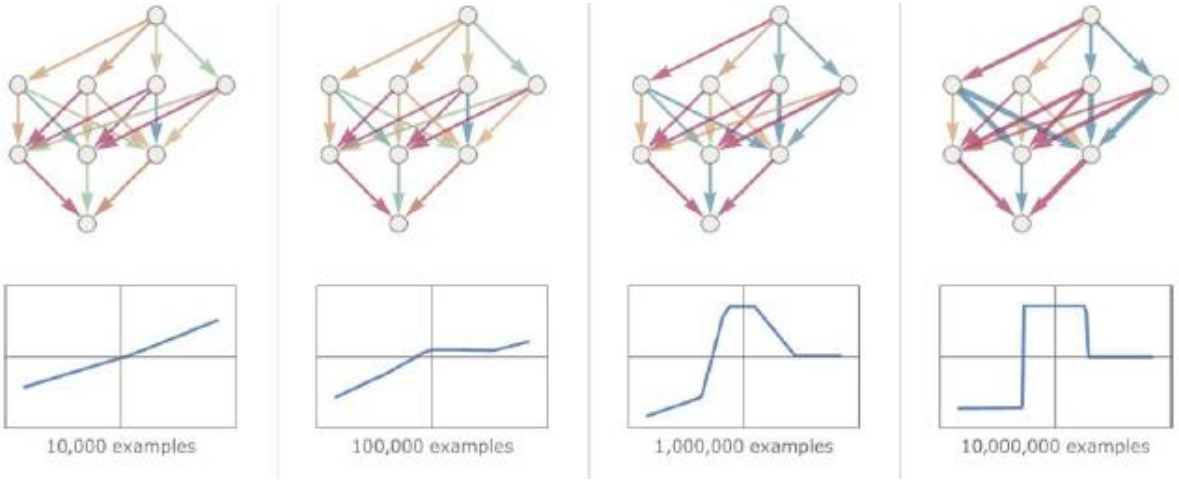
Do tego zadania będziemy potrzebować sieci, która ma tylko jedno wejście i jedno wyjście, na przykład:



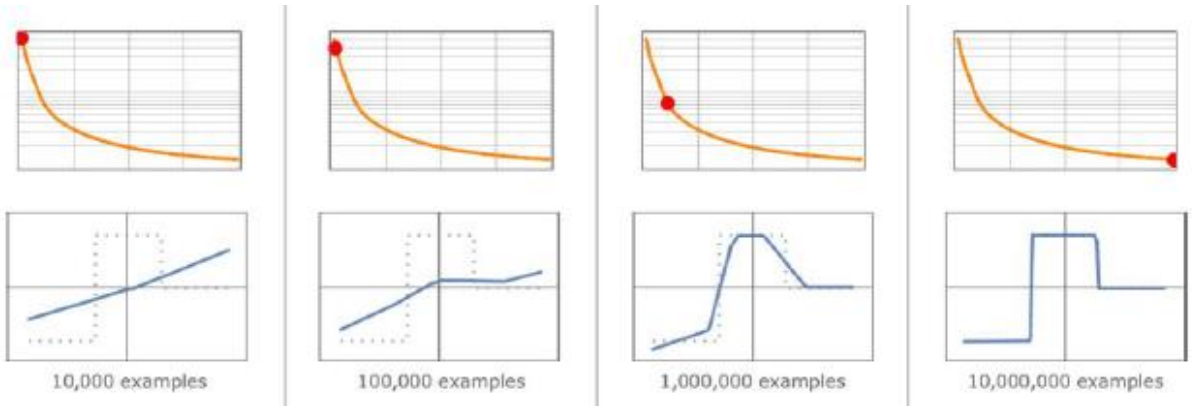
Ale jakich ciężarów itp. powinniśmy używać? Przy każdym możliwym zestawie wag sieć neuronowa obliczy jakąś funkcję. I na przykład oto, co robi z kilkoma losowo wybranymi zestawami ciężarów:



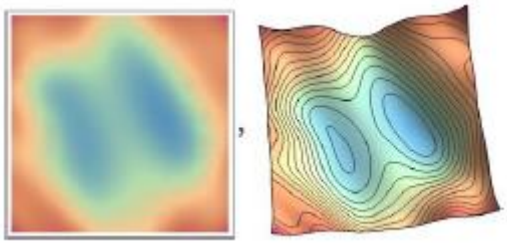
I tak, możemy wyraźnie zobaczyć, że w żadnym z tych przypadków nie zbliża się nawet do odtworzenia pożądanej funkcji. Jak więc znaleźć wagi, które będą odtwarzać funkcję? Podstawową ideą jest dostarczenie wielu przykładów „wejście → wyjście", z których można „uczyć się" – a następnie próba znalezienia wag, które odtworzą te przykłady. Oto wynik tego działania z coraz większą liczbą przykładów:



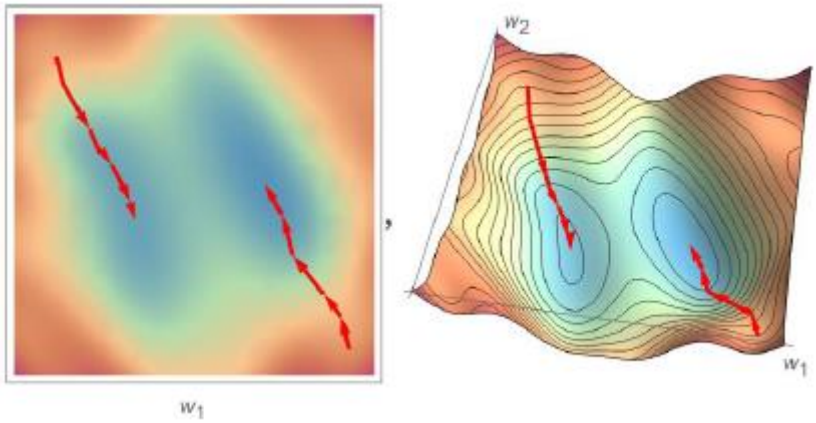
Na każdym etapie tego „treningu" wagi w sieci są stopniowo dostosowywane — i widzimy, że w końcu otrzymujemy sieć, która z powodzeniem odtwarza pożądaną funkcję. Jak więc dostosować wagę? Podstawową ideą jest, aby na każdym etapie zobaczyć, „jak daleko jesteśmy" od uzyskania pożądanej funkcji — a następnie zaktualizować wagi w taki sposób, aby się do nich zbliżyć. Aby dowiedzieć się, „jak daleko jesteśmy", obliczamy coś, co zwykle nazywa się „funkcją straty" (lub czasami „funkcją kosztu"). Tutaj używamy prostej funkcji straty (L2), która jest po prostu sumą kwadratów różnic między wartościami, które otrzymujemy, a wartościami rzeczywistymi. Widzimy, że w miarę postępu procesu uczenia funkcja utraty stopniowo maleje (zgodnie z pewną „krzywą uczenia się", która jest różna dla różnych zadań) — aż osiągniemy punkt, w którym sieć (przynajmniej w dobrym przybliżeniu) z powodzeniem się odtwarza funkcja, którą chcemy:



W porządku, więc ostatnim istotnym elementem do wyjaśnienia jest sposób dostosowywania ciężarków w celu zmniejszenia funkcji straty. Jak już powiedzieliśmy, funkcja straty daje nam „odległość" między wartościami, które mamy, a wartościami rzeczywistymi. Ale „wartości, które mamy" są określane na każdym etapie przez aktualną wersję sieci neuronowej — i przez jej wagi. Ale teraz wyobraź sobie, że wagi są zmiennymi — powiedzmy w_i . Chcemy dowiedzieć się, jak dopasować wartości tych zmiennych, aby zminimalizować zależne od nich straty. Wyobraźmy sobie np. (w niewiarygodnym uproszczeniu typowych sieci neuronowych stosowanych w praktyce), że mamy tylko dwie wagi w_1 i w_2 . Wtedy możemy mieć stratę, która jako funkcja w 1 i w 2 wygląda następująco:



Analiza numeryczna zapewnia różnorodne techniki znajdowania minimum w takich przypadkach. Ale typowym podejściem jest po prostu stopniowe podążanie ścieżką najbardziej stromego spadku od tego, co mieliśmy poprzednio w_1 , w_2 :



Podobnie jak woda spływająca z góry, wszystko, co jest gwarantowane, to to, że ta procedura zakończy się na jakimś lokalnym minimum powierzchni („górskie jezioro"); może równie dobrze nie osiągnąć ostatecznego globalnego minimum. Nie jest oczywiste, czy możliwe byłoby znalezienie ścieżki najbardziej stromego zejścia na „krajobrazie wagowym". Ale rachunek różniczkowy przychodzi na ratunek. Jak wspomnieliśmy powyżej, zawsze można myśleć o sieci neuronowej jako o obliczaniu

funkcji matematycznej — która zależy od danych wejściowych i wag. Ale teraz rozważ zróżnicowanie w odniesieniu do tych wag. Okazuje się, że reguła łańcuchowa rachunku różniczkowego w efekcie pozwala „rozwikłać" operacje wykonywane przez kolejne warstwy w sieci neuronowej. W rezultacie możemy — przynajmniej w pewnym lokalnym przybliżeniu — „odwrócić" działanie sieci neuronowej i stopniowo znaleźć wagi, które minimalizują straty związane z wyjściem. Powyższy rysunek pokazuje rodzaj minimalizacji, której możemy potrzebować w nierealistycznie prostym przypadku z zaledwie 2 wagami. Okazuje się jednak, że nawet przy znacznie większej liczbie wag (ChatGPT wykorzystuje 175 miliardów) nadal można dokonać minimalizacji, przynajmniej do pewnego poziomu przybliżenia. W rzeczywistości wielki przełom w „głębokim uczeniu się", który nastąpił około 2011 r., wiązał się z odkryciem, że w pewnym sensie łatwiej jest wykonać (przynajmniej w przybliżeniu) minimalizację, gdy zaangażowanych jest wiele wag niż wtedy, gdy jest ich stosunkowo mało. Innymi słowy – nieco wbrew intuicji – rozwiązywanie bardziej skomplikowanych problemów za pomocą sieci neuronowych może być łatwiejsze niż prostszych. Zgrubnym powodem tego wydaje się być to, że kiedy ma się wiele „zmiennych wagi", ma się przestrzeń wielowymiarową z „mnóstwem różnych kierunków", które mogą doprowadzić do minimum – podczas gdy przy mniejszej liczbie zmiennych łatwiej utknąć w lokalnym minimum („górskie jezioro"), z którego nie ma „kierunku wyjścia". Warto zauważyć, że w typowych przypadkach istnieje wiele różnych kolekcji wag, z których wszystkie dają sieci neuronowe o prawie takiej samej wydajności. I zwykle w praktycznym treningu sieci neuronowych dokonuje się wielu przypadkowych wyborów

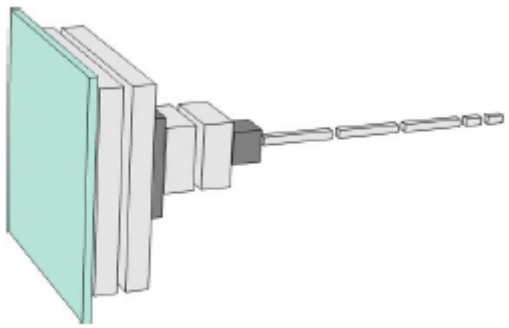
Ale każde takie „inne rozwiązanie" będzie miało przynajmniej trochę inne zachowanie. A jeśli poprosimy, powiedzmy, o „ekstrapolację" poza region, w którym podaliśmy przykłady treningowe, możemy uzyskać diametralnie różne wyniki:

Ale który z nich jest „właściwy"? Naprawdę nie ma sposobu, aby powiedzieć. Wszystkie są „zgodne z obserwowanymi danymi". Ale wszystkie odpowiadają różnym „wrodzonym" sposobom „myślenia" o tym, co robić „nieszablonowo". A niektóre mogą wydawać się nam, ludziom, „bardziej rozsądne" niż inne.

Praktyka i tradycja treningu sieci neuronowych

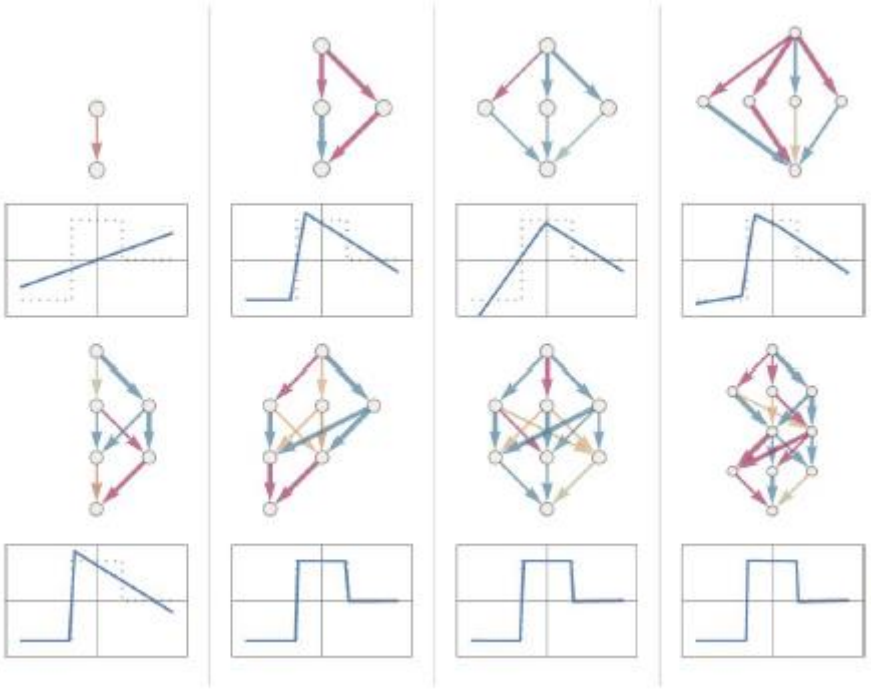
Szczególnie w ciągu ostatniej dekady dokonano wielu postępów w sztuce uczenia sieci neuronowych. I tak, to w zasadzie sztuka. Czasami — szczególnie z perspektywy czasu — można dostrzec przynajmniej przebłysk „naukowego wyjaśnienia" czegoś, co się dzieje. Ale większość rzeczy odkrywano metodą prób i błędów, dodając pomysły i sztuczki, które stopniowo budowały znaczącą wiedzę na temat pracy z sieciami neuronowymi. Jest kilka kluczowych części. Po pierwsze, jest kwestia tego, jaką architekturę sieci neuronowej należy zastosować do konkretnego zadania. Następnie pojawia się kluczowa kwestia, w jaki sposób uzyskać dane, na których można trenować sieć neuronową. I coraz częściej nie

zajmujemy się trenowaniem sieci od zera: zamiast tego nowa sieć może albo bezpośrednio zawierać inną już wytrenowaną sieć, albo przynajmniej może użyć tej sieci do generowania dla siebie większej liczby przykładów treningowych. Można by pomyśleć, że do każdego zadania potrzebna będzie inna architektura sieci neuronowej. Ale okazało się, że ta sama architektura często wydaje się działać nawet w przypadku pozornie całkiem różnych zadań. Na pewnym poziomie przypomina to ideę obliczeń uniwersalnych (i moją zasadę równoważności obliczeniowej), ale, jak omówię to później, myślę, że jest to raczej odzwierciedlenie faktu, że zadania, które zwykle próbujemy uzyskać neuronowe sieci do zrobienia są „podobne do ludzi", a sieci neuronowe mogą uchwycić całkiem ogólne „procesy podobne do ludzi". We wcześniejszych czasach sieci neuronowych istniała idea, że należy „sprawić, by sieć neuronowa robiła jak najmniej". Na przykład przy konwersji mowy na tekst uważano, że należy najpierw przeanalizować dźwięk mowy, podzielić go na fonemy itp. Ale okazało się, że — przynajmniej w przypadku „zadań podobnych do człowieka" — zwykle lepiej jest po prostu spróbować wytrenować sieć neuronową w zakresie „problemu od końca do końca", pozwalając jej „odkryć" dla siebie niezbędne cechy pośrednie, kodowanie itp. Pojawił się również pomysł, aby do sieci neuronowej wprowadzać skomplikowane poszczególne komponenty, aby w efekcie „jasno realizowała określone algorytmiczne idee". Ale po raz kolejny okazało się to w większości nieopłacalne; zamiast tego lepiej po prostu zająć się bardzo prostymi komponentami i pozwolić im „zorganizować się" (choć zwykle w sposób, którego nie możemy zrozumieć), aby osiągnąć (prawdopodobnie) odpowiednik tych algorytmicznych pomysłów. Nie oznacza to, że nie ma „pomysłów strukturalnych", które są istotne dla sieci neuronowych. Na przykład posiadanie dwuwymiarowych tablic neuronów z lokalnymi połączeniami wydaje się co najmniej bardzo przydatne na wczesnych etapach przetwarzania obrazów. A posiadanie wzorców łączności, które koncentrują się na „patrzeniu wstecz w sekwencjach", wydaje się przydatne — jak zobaczymy później — w radzeniu sobie z rzeczami takimi jak ludzki język, na przykład w ChatGPT. Ale ważną cechą sieci neuronowych jest to, że – podobnie jak ogólnie komputery – ostatecznie zajmują się tylko danymi. A obecne sieci neuronowe — przy obecnym podejściu do uczenia sieci neuronowych — zajmują się w szczególności tablicami liczb. Ale w trakcie przetwarzania te tablice można całkowicie zmienić i zmienić ich kształt. Na przykład sieć, której użyliśmy do identyfikacji cyfr powyżej, zaczyna się od „podobnej do obrazu" tablicy 2D, szybko „zagęszczając" do wielu kanałów, ale następnie „koncentrując się" w tablicy 1D, która ostatecznie będzie zawierała elementy reprezentujące różne możliwe cyfry wyjściowe:



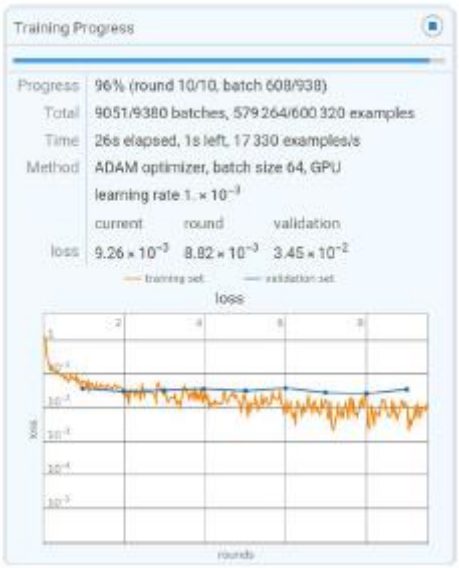
Ale OK, jak można określić, jak duża sieć neuronowa będzie potrzebna do wykonania określonego zadania? To coś w rodzaju sztuki. Na pewnym poziomie kluczową rzeczą jest wiedzieć „jak trudne jest to zadanie". Ale w przypadku zadań podobnych do ludzkich jest to zazwyczaj bardzo trudne do oszacowania. Tak, może istnieć systematyczny sposób wykonywania zadania bardzo „mechanicznie" za pomocą komputera. Ale trudno powiedzieć, czy istnieją sztuczki lub skróty, które pozwalają znacznie łatwiej wykonać zadanie przynajmniej na „poziomie ludzkim". Aby „mechanicznie" zagrać w określoną grę, może zająć wyliczenie gigantycznego drzewa gier; ale może istnieć znacznie łatwiejszy („heurystyczny") sposób osiągnięcia „gry na poziomie człowieka". Kiedy ma się do czynienia z małymi

sieciami neuronowymi i prostymi zadaniami, można czasami wyraźnie zauważyć, że „stąd nie można się tam dostać”. Na przykład, oto najlepsze, co wydaje się być w stanie wykonać w zadaniu z poprzedniej sekcji z kilkoma małymi sieciami neuronowymi:



Widzimy, że jeśli sieć jest zbyt mała, po prostu nie może odtworzyć pożądanej funkcji. Ale powyżej pewnego rozmiaru nie ma problemu - przynajmniej jeśli trenuje się go wystarczająco długo, z wystarczającą liczbą przykładów. Nawiasem mówiąc, te zdjęcia ilustrują fragment wiedzy o sieciach neuronowych: często można uciec z mniejszą siecią, jeśli istnieje „ściskanie” w środku, które zmusza wszystko do przejścia przez mniejszą pośrednią liczbę neuronów. (Warto również wspomnieć, że sieci „bez warstwy pośredniej” — lub tak zwany „perceptron” — mogą uczyć się tylko zasadniczo funkcji liniowych — ale gdy tylko istnieje choćby jedna warstwa pośrednia, zawsze można w zasadzie dowolnie dobrze przybliżyć dowolną funkcję, przy przynajmniej jeśli ktoś ma wystarczającą liczbę neuronów, chociaż aby można było go wyszkolić, zazwyczaj stosuje się jakąś regularyzację lub normalizację. OK, powiedzmy, że osiedliliśmy się na określonej architekturze sieci neuronowej. Teraz pojawia się problem z uzyskaniem danych do trenowania sieci. A wiele praktycznych wyzwań związanych z sieciami neuronowymi — i ogólnie z uczeniem maszynowym — koncentruje się na pozyskiwaniu lub przygotowywaniu niezbędnych danych szkoleniowych. W wielu przypadkach („nauczanie nadzorowane”) chce się uzyskać wyraźne przykłady danych wejściowych i oczekiwanych wyników. Na przykład można chcieć oznaczyć obrazy tagami według tego, co w nich jest, lub według innego atrybutu. A może ktoś będzie musiał wyraźnie przejść - zwykle z wielkim wysiłkiem - i wykonać tagowanie. Ale bardzo często okazuje się, że można wykorzystać coś, co już zostało zrobione, lub użyć tego jako swego rodzaju proxy. I tak, na przykład, można użyć tagów alt, które zostały dostarczone dla obrazów w Internecie. Lub w innej domenie można użyć napisów, które zostały utworzone dla filmów. Lub — w przypadku szkoleń z zakresu tłumaczeń językowych — można korzystać z równoległych wersji stron internetowych lub innych dokumentów, które istnieją w różnych językach. Ile danych potrzebujesz, aby pokazać sieć neuronową, aby wyszkolić ją do określonego zadania? Ponownie, trudno jest oszacować na podstawie pierwszych zasad. Z pewnością wymagania można radykalnie zmniejszyć, stosując „uczenie się transferu” do „przeniesienia” rzeczy, takich jak listy ważnych funkcji, które zostały już nauczone w innej sieci. Ale generalnie sieci neuronowe muszą „zobaczyć wiele przykładów”, aby dobrze trenować. I przynajmniej w przypadku niektórych zadań ważnym elementem

wiedzy o sieciach neuronowych jest to, że przykłady mogą być niewiarygodnie powtarzalne. I rzeczywiście, standardową strategią jest po prostu pokazywanie sieci neuronowej wszystkich posiadanych przykładów, w kółko. W każdej z tych „koledzy treningowych" (lub „epoek") sieć neuronowa będzie znajdować się w co najmniej nieco innym stanie, a „przypomnienie jej" konkretnego przykładu jest przydatne, aby „zapamiętać ten przykład". (I tak, być może jest to analogiczne do przydatności powtarzania w zapamiętywaniu człowieka.) Ale często powtarzanie tego samego przykładu w kółko nie wystarcza. Konieczne jest również pokazanie odmian sieci neuronowej przykładu. Cechą wiedzy o sieciach neuronowych jest to, że te odmiany „zwiększania danych" nie muszą być wyrafinowane, aby były użyteczne. Niewielka modyfikacja obrazów za pomocą podstawowego przetwarzania obrazu może sprawić, że będą one zasadniczo „tak dobre jak nowe" do uczenia sieci neuronowych. I podobnie, gdy zabraknie rzeczywistego wideo itp. do szkolenia samojezdnych samochodów, można kontynuować i po prostu uzyskać dane z uruchomionych symulacji w modelowym środowisku przypominającym grę wideo, bez wszystkich szczegółów rzeczywistych scen. Co powiesz na coś takiego jak ChatGPT? Cóż, ma tę fajną cechę, że może „uczyć się bez nadzoru", co znacznie ułatwia zdobywanie przykładów do trenowania. Przypomnij sobie, że podstawowym zadaniem ChatGPT jest wymyślenie, jak kontynuować otrzymany fragment tekstu. Aby więc uzyskać „przykłady szkoleniowe", wystarczy pobrać fragment tekstu i zamaskować jego koniec, a następnie użyć go jako „danych wejściowych do uczenia" — przy czym „wyjście" jest kompletne, niezamaskowany fragment tekstu. Omówimy to później, ale najważniejsze jest to, że — w przeciwieństwie do, powiedzmy, uczenia się, co jest na obrazach — nie ma potrzeby „wyraźnego oznaczania"; ChatGPT może w efekcie po prostu uczyć się bezpośrednio z dowolnych przykładów tekstu, które mu podano. OK, a co z faktycznym procesem uczenia się w sieci neuronowej? Ostatecznie wszystko sprowadza się do ustalenia, jakie ciężary najlepiej oddają podane przykłady treningowe. Istnieje wiele szczegółowych wyborów i „ustawień hiperparametrów" (tak zwanych, ponieważ wagi można traktować jako „parametry"), których można użyć do dostosowania sposobu, w jaki to się robi. Istnieją różne możliwości wyboru funkcji straty (suma kwadratów, um wartości bezwzględnych itp.). Istnieją różne sposoby minimalizacji strat (jak daleko w przestrzeni wagowej należy się poruszać na każdym kroku itp.). A potem pojawiają się pytania, jak dużą „partię" przykładów pokazać, aby uzyskać każde kolejne oszacowanie straty, którą próbuje się zminimalizować. I tak, można zastosować uczenie maszynowe (jak to robimy na przykład w Wolfram Language) do zautomatyzowania uczenia maszynowego — i automatycznego ustawiania takich rzeczy jak hiperparametry. Ale w końcu cały proces szkolenia można scharakteryzować, obserwując, jak stopniowo zmniejsza się strata (jak w tym monitorze postępu Wolfram Language dla małego szkolenia):

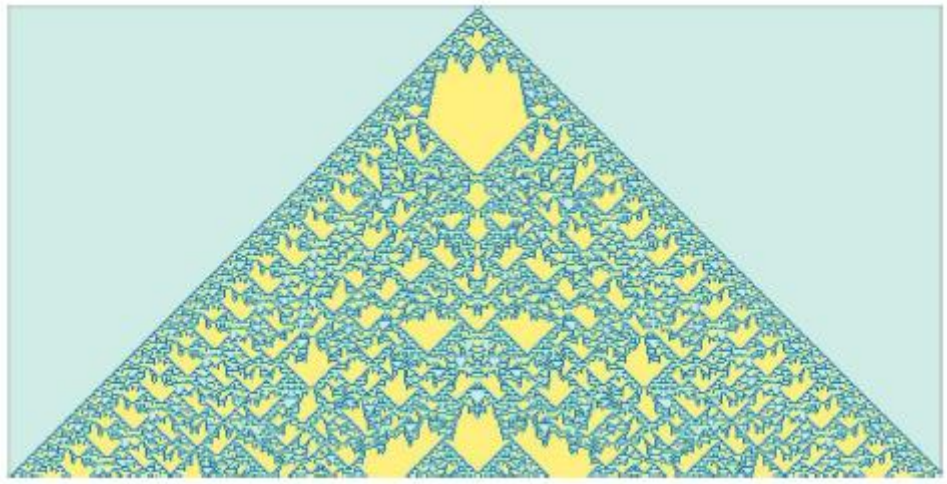


Zazwyczaj widać, że strata maleje przez jakiś czas, ale ostatecznie spłaszcza się do pewnej stałej wartości. Jeśli ta wartość jest wystarczająco mała, to szkolenie można uznać za udane; w przeciwnym razie jest to prawdopodobnie znak, że należy spróbować zmienić architekturę sieci. Czy można powiedzieć, ile czasu powinno zająć spłaszczenie „krzywej uczenia się”? Podobnie jak w przypadku wielu innych rzeczy, wydaje się, że istnieją przybliżone relacje skalowania oparte na prawie potęgowym, które zależą od rozmiaru sieci neuronowej i ilości danych, z których się korzysta. Ale ogólny wniosek jest taki, że uczenie sieci neuronowej jest trudne i wymaga dużego wysiłku obliczeniowego. Z praktycznego punktu widzenia ogromna większość tego wysiłku jest poświęcona wykonywaniu operacji na tablicach liczb, w czym GPU są dobre — dlatego szkolenie sieci neuronowych jest zwykle ograniczone dostępnością procesorów graficznych. Czy w przyszłości będą zasadniczo lepsze sposoby trenowania sieci neuronowych — lub ogólnie robienia tego, co robią sieci neuronowe? Myślę, że prawie na pewno. Podstawową ideą sieci neuronowych jest stworzenie elastycznej „tkaniny obliczeniowej” z dużej liczby prostych (zasadniczo identycznych) komponentów – i posiadanie tej „tkaniny”, którą można stopniowo modyfikować, aby uczyć się na przykładach. W obecnych sieciach neuronowych zasadniczo wykorzystuje się idee rachunku różniczkowego — zastosowane do liczb rzeczywistych — do wykonania tej przyrostowej modyfikacji. Ale coraz bardziej oczywiste jest, że posiadanie precyzyjnych liczb nie ma znaczenia; 8 bitów lub mniej może wystarczyć nawet przy obecnych metodach. W przypadku systemów obliczeniowych, takich jak automaty komórkowe, które w zasadzie działają równolegle na wielu pojedynczych bitach, nigdy nie było jasne, jak wykonać tego rodzaju przyrostową modyfikację, ale nie ma powodu sądzić, że nie jest to możliwe. I w rzeczywistości, podobnie jak w przypadku „przełomu głębokiego uczenia się z 2012 roku” może się okazać, że taka stopniowa modyfikacja będzie faktycznie łatwiejsza w bardziej skomplikowanych przypadkach niż w prostych. Sieci neuronowe – być może trochę jak mózgi – są skonfigurowane tak, aby mieć zasadniczo stałą sieć neuronów, z modyfikowaną siłą („wagą”) połączeń między nimi. (Być może w przynajmniej młodych mózgach może również rosnąć znaczna liczba zupełnie nowych połączeń.) Chociaż może to być wygodna konfiguracja dla biologii, wcale nie jest jasne, czy jest to nawet bliskie najlepszemu sposobowi osiągnięcia potrzebnej nam funkcjonalności. A coś, co obejmuje odpowiednik progresywnego przepisywania sieci (być może przypominające nasz projekt fizyczny), może ostatecznie być lepsze. Ale nawet w ramach istniejących sieci neuronowych istnieje obecnie kluczowe ograniczenie: uczenie sieci neuronowej w obecnym kształcie jest zasadniczo sekwencyjne, a efekty każdej partii przykładów są propagowane z powrotem w celu aktualizacji wag. I rzeczywiście, przy obecnym sprzęcie komputerowym — nawet biorąc pod uwagę procesory graficzne — większość sieci

neuronowej jest „bezczynna” przez większość czasu podczas szkolenia, a tylko jedna część jest aktualizowana na raz. W pewnym sensie dzieje się tak dlatego, że nasze obecne komputery mają zwykle pamięć oddzieloną od ich procesorów (lub procesorów graficznych). Ale w mózgach jest prawdopodobnie inaczej – każdy „element pamięci” (tj. neuron) jest również potencjalnie aktywnym elementem obliczeniowym. A gdybyśmy mogli skonfigurować nasz przyszły sprzęt komputerowy w ten sposób, mogłoby się okazać, że szkolenie będzie znacznie wydajniejsze.

„Z pewnością wystarczająco duża sieć może zrobić wszystko!”

Możliwości czegoś takiego jak ChatGPT wydają się tak imponujące, że można by sobie wyobrazić, że gdyby można było po prostu „kontynuować” i trenować coraz większe sieci neuronowe, to ostatecznie byłoby w stanie „robić wszystko”. A jeśli ktoś zajmuje się rzeczami, które są łatwo dostępne dla bezpośredniego ludzkiego myślenia, jest całkiem możliwe, że tak jest. Ale lekcja z ostatnich kilkuset lat nauki jest taka, że istnieją rzeczy, które można zrozumieć za pomocą procesów formalnych, ale nie są one łatwo dostępne dla bezpośredniego ludzkiego myślenia. Wielkim przykładem jest matematyka nietrywialna. Ale ogólny przypadek to tak naprawdę obliczenia. Ostatecznie problemem jest zjawisko obliczeniowej nieredukowalności. Istnieją pewne obliczenia, które można by pomyśleć, że wymagałyby wielu kroków, ale które w rzeczywistości można „zredukować” do czegoś całkiem natychmiastowego. Ale odkrycie obliczeniowej nieredukowalności oznacza, że nie zawsze to działa. Zamiast tego istnieją procesy — prawdopodobnie takie jak ten poniżej — w przypadku których ustalenie, co się nieuchronnie dzieje, wymaga zasadniczo prześledzenia każdego kroku obliczeniowego:



Rodzaje rzeczy, które normalnie robimy z naszymi mózgami, są przypuszczalnie specjalnie wybrane, aby uniknąć obliczeniowej nieredukowalności. Wykonanie obliczeń w mózgu wymaga szczególnego wysiłku. W praktyce jest w dużej mierze niemożliwe „przemyślenie” kroków działania jakiegokolwiek nietrywialnego programu tylko w mózgu. Ale oczywiście do tego mamy komputery. A dzięki komputerom możemy z łatwością wykonywać długie, obliczeniowo nieredukowalne rzeczy. Kluczową kwestią jest to, że generalnie nie ma dla nich skrótu. Tak, moglibyśmy zapamiętać wiele konkretnych przykładów tego, co dzieje się w określonym systemie obliczeniowym. A może nawet moglibyśmy zobaczyć pewne („redukowalne obliczeniowo”) wzorce, które pozwoliłyby nam na małe uogólnienie. Ale chodzi o to, że nieredukowalność obliczeniowa oznacza, że nigdy nie możemy zagwarantować, że nie wydarzy się coś nieoczekiwanego — i tylko poprzez jawne wykonanie obliczeń można stwierdzić, co faktycznie dzieje się w każdym konkretnym przypadku. I w końcu istnieje tylko fundamentalne napięcie między możliwością uczenia się a obliczeniową nieredukowalnością. Uczenie się polega w efekcie na kompresji danych poprzez wykorzystanie regularności. Ale nieredukowalność obliczeniowa oznacza, że ostatecznie istnieje granica tego, jakie mogą być prawidłowości. W praktyce można sobie

wyobrazić budowanie małych urządzeń obliczeniowych — takich jak automaty komórkowe lub maszyny Turinga — w dające się trenować systemy, takie jak sieci neuronowe. I rzeczywiście, takie urządzenia mogą służyć jako dobre „narzędzia" dla sieci neuronowej – tak jak Wolfram|Alpha może być dobrym narzędziem dla ChatGPT. Ale nieredukowalność obliczeniowa oznacza, że nie można oczekiwać, że „dostaniemy się do środka" tych urządzeń i sprawimy, że się nauczą. Innymi słowy, istnieje ostateczny kompromis między możliwościami a możliwością szkolenia: im bardziej chcesz, aby system „prawdziwie wykorzystywał" swoje możliwości obliczeniowe, tym bardziej będzie wykazywał nieredukowalność obliczeniową i tym mniej będzie można go trenować. A im bardziej można go zasadniczo wyszkolić, tym mniej będzie w stanie wykonać wyrafinowanych obliczeń. (W przypadku ChatGPT w obecnej sytuacji sytuacja jest w rzeczywistości znacznie bardziej ekstremalna, ponieważ sieć neuronowa używana do generowania każdego tokena wyjściowego jest czystą siecią „feed-forward", bez pętli, a zatem nie ma możliwości wykonywania jakichkolwiek obliczenia z nietrywialnym „przepływem sterowania".) Oczywiście można by się zastanawiać, czy umiejętność wykonywania nieredukowalnych obliczeń jest rzeczywiście ważna. I rzeczywiście przez większą część historii ludzkości nie było to szczególnie ważne. Ale nasz nowoczesny świat technologiczny został zbudowany na inżynierii, która korzysta przynajmniej z obliczeń matematycznych – a coraz częściej także z bardziej ogólnych obliczeń. A jeśli spojrzymy na świat przyrody, jest on pełen nieredukowalnych obliczeń — które powoli rozumiemy, jak naśladować i wykorzystywać do naszych celów technologicznych. Tak, sieć neuronowa z pewnością może zauważyć rodzaje prawidłowości w świecie przyrody, które moglibyśmy również łatwo zauważyć przy „niewspomaganym ludzkim myśleniu". Ale jeśli chcemy opracować rzeczy, które wchodzą w zakres nauk matematycznych lub obliczeniowych, sieć neuronowa nie będzie w stanie tego zrobić – chyba że skutecznie „użyje jako narzędzia" „zwykłego" systemu obliczeniowego. Ale jest w tym wszystkim coś potencjalnie mylącego. W przeszłości było wiele zadań — w tym pisanie esejów — które, jak zakładaliśmy, były „zasadniczo zbyt trudne" dla komputerów. A teraz, gdy widzimy je wykonane przez takich jak ChatGPT, nagle myślimy, że komputery musiały stać się znacznie potężniejsze - w szczególności przewyższające rzeczy, które już zasadniczo były w stanie zrobić (jak stopniowe obliczanie zachowania systemów obliczeniowych, takich jak automaty komórkowe) . Ale to nie jest właściwy wniosek do wyciągnięcia. Obliczeniowo nieredukowalne procesy są nadal nieredukowalne obliczeniowo i nadal są zasadniczo trudne dla komputerów - nawet jeśli komputery mogą z łatwością obliczyć swoje indywidualne kroki. Zamiast tego powinniśmy dojść do wniosku, że zadania – takie jak pisanie esejów – które my, ludzie, moglibyśmy wykonać, ale nie sądziliśmy, że komputery mogą to zrobić, są w rzeczywistości w pewnym sensie łatwiejsze obliczeniowo, niż myśleliśmy. Innymi słowy, powodem, dla którego sieć neuronowa może odnieść sukces w pisaniu eseju, jest to, że napisanie eseju okazuje się być problemem „płytszym obliczeniowo", niż myśleliśmy. I w pewnym sensie zbliża nas to do „posiadania teorii" na temat tego, jak my, ludzie, radzimy sobie z takimi rzeczami, jak pisanie esejów lub ogólnie radzenie sobie z językiem. Gdybyś miał wystarczająco dużą sieć neuronową, to tak, mógłbyś robić wszystko, co ludzie mogą z łatwością zrobić. Ale nie uchwyciłbyś tego, co ogólnie świat przyrody może zrobić – ani tego, co mogą zrobić narzędzia, które stworzyliśmy ze świata przyrody. I to użycie tych narzędzi — zarówno praktycznych, jak i koncepcyjnych — pozwoliło nam w ostatnich stuleciach przekroczyć granice tego, co jest dostępne dla „czystej, niewspomaganej ludzkiej myśli" i uchwycić dla ludzkich celów więcej tego, co istnieje w fizycznym i obliczeniowym wszechświat

Koncepcja osadzania

Sieci neuronowe – przynajmniej w obecnej formie – są zasadniczo oparte na liczbach. Więc jeśli zamierzamy ich użyć do pracy nad czymś takim jak tekst, będziemy potrzebować sposobu na przedstawienie naszego tekstu za pomocą liczb. I z pewnością moglibyśmy zacząć (zasadniczo tak, jak robi to ChatGPT) od przypisania numeru do każdego słowa w słowniku. Istnieje jednak ważny pomysł

— na przykład kluczowy dla ChatGPT — który wykracza poza to. I to jest idea „osadzania”. O osadzeniu można myśleć jako o sposobie przedstawienia „istoty” czegoś za pomocą tablicy liczb — z tą właściwością, że „rzeczy znajdujące się w pobliżu” są reprezentowane przez liczby znajdujące się w pobliżu. I tak, na przykład, możemy myśleć o osadzeniu słowa jako próbie ułożenia słów w swego rodzaju „przestrzeni znaczeniowej”, w której słowa, które są w jakiś sposób „bliskie znaczeniem”, pojawiają się w pobliżu osadzania. Rzeczywiste osadzenie, które jest używane — powiedzmy w ChatGPT — zwykle obejmuje duże listy liczb. Ale jeśli rzutujemy do 2D, możemy pokazać przykłady układania słów przez osadzenie:

I tak, to, co widzimy, znakomicie radzi sobie z uchwyceniem typowych codziennych wrażeń. Ale jak możemy skonstruować takie osadzenie? Z grubsza chodzi o to, aby spojrzeć na duże ilości tekstu (tutaj 5 miliardów słów z sieci), a następnie zobaczyć „jak podobne” są „środowiska”, w których pojawiają się różne słowa. Na przykład „aligator” i „krokodyl” często pojawiają się prawie zamiennie w podobnych zdaniach, co oznacza, że zostaną umieszczone w pobliżu osadzania. Ale „rzepa” i „orzeł” nie będą pojawiać się w innych podobnych zdaniach, więc zostaną umieszczone daleko od siebie podczas osadzania. Ale jak właściwie zaimplementować coś takiego za pomocą sieci neuronowych? Zacznijmy od osadzenia nie dla słów, ale dla obrazów. Chcemy znaleźć sposób na scharakteryzowanie obrazów za pomocą list liczb w taki sposób, aby „obrazom, które uważamy za podobne”, przypisano podobne listy liczb. Jak stwierdzić, czy powinniśmy „uważać obrazy za podobne”? Cóż, jeśli nasze obrazy są, powiedzmy, odręcznymi cyframi, możemy „uznać dwa obrazy za podobne”, jeśli mają tę samą cyfrę. Wcześniej omawialiśmy sieć neuronową, która została wyszkolona do rozpoznawania odręcznych cyfr. I możemy myśleć o tej sieci neuronowej jako skonfigurowanej tak, że na wyjściu umieszcza obrazy w 10 różnych pojemnikach, po jednym dla każdej cyfry. Ale co, jeśli „przechwycimy” to, co dzieje się w sieci neuronowej, zanim zostanie podjęta ostateczna decyzja „to jest„ 4 ””? Można by się spodziewać, że wewnątrz sieci neuronowej znajdują się liczby, które charakteryzują obrazy jako „przeważnie 4-podobne, ale trochę podobne do 2” lub coś w tym rodzaju. Chodzi o to, aby wybrać takie liczby, aby użyć ich jako elementów osadzania. Oto koncepcja. Zamiast bezpośrednio próbować scharakteryzować „który obraz znajduje się w pobliżu innego obrazu”, zamiast tego rozważamy dobrze zdefiniowane zadanie (w tym przypadku rozpoznawanie cyfr), dla którego możemy uzyskać wyraźne dane treningowe – a następnie wykorzystać fakt, że wykonując to zadanie sieć neuronowa pośrednio

musi podejmować „decyzje dotyczące bliskości”. Więc zamiast kiedykolwiek otwarcie mówić o „bliskości obrazów”, mówimy po prostu o konkretnym pytaniu, jaką cyfrę reprezentuje obraz, a następnie „pozostawiamy to sieci neuronowej”, aby pośrednio określić, co to oznacza o „bliskości obrazów”. Jak więc bardziej szczegółowo działa to w przypadku sieci rozpoznawania cyfr? Możemy myśleć o sieci jako składającej się z 11 kolejnych warstw, które możemy podsumować ikonicznie w ten sposób (z funkcjami aktywacji pokazanymi jako osobne warstwy):

Na początku wprowadzamy do pierwszej warstwy rzeczywiste obrazy, reprezentowane przez dwuwymiarowe tablice wartości pikseli. I na koniec — z ostatniej warstwy — uzyskujemy tablicę 10 wartości, które możemy sobie wyobrazić mówiąc „jak pewna” jest sieć, że obraz odpowiada każdej z cyfr od 0 do 9.

Podaj obraz,  a wartości neuronów w tej ostatniej warstwie to:

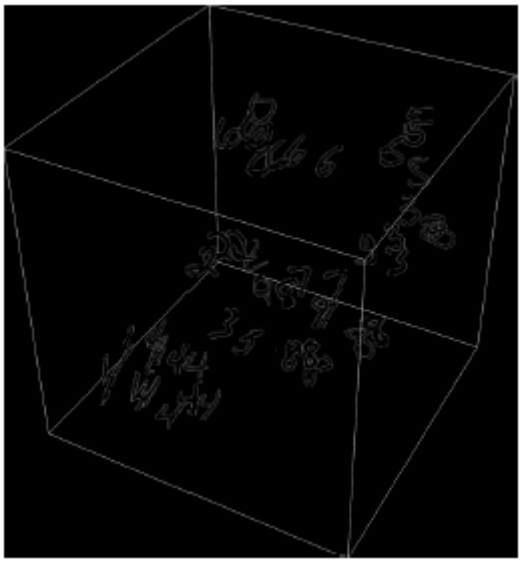
$$\{1.42071\times10^{-22},\ 7.69857\times10^{-14},\ 1.9653\times10^{-16},\ 5.55229\times10^{-21},\ 1.,$$
$$8.33841\times10^{-14},\ 6.89742\times10^{-17},\ 6.52282\times10^{-19},\ 6.51465\times10^{-12},\ 1.97509\times10^{-14}\}$$

Innymi słowy, sieć neuronowa jest w tym momencie „niesamowicie pewna”, że ten obraz to 4 – i aby rzeczywiście uzyskać wynik „4”, wystarczy wybrać pozycję neuronu o największej wartości. Ale co, jeśli spojrzymy o krok wcześniej? Ostatnią operacją w sieci jest tzw. softmax, który próbuje „wymusić pewność”. Ale zanim to zostało zastosowane, wartości neuronów to:

$$\{-26.134,\ -6.02347,\ -11.994,\ -22.4684,$$
$$24.1717,\ -5.94363,\ -13.0411,\ -17.7021,\ -1.58528,\ -7.38389\}$$

Neuron reprezentujący „4” nadal ma najwyższą wartość liczbową. Ale są też informacje w wartościach innych neuronów. I można się spodziewać, że ta lista liczb może być w pewnym sensie wykorzystana do scharakteryzowania „istoty” obrazu – a tym samym do dostarczenia czegoś, co możemy wykorzystać jako osadzenie. I tak na przykład każda z 4 tutaj ma nieco inny „podpis” (lub „osadzenie funkcji”) — wszystkie bardzo różnią się od 8:

Tutaj zasadniczo używamy 10 liczb do scharakteryzowania naszych obrazów. Ale często lepiej jest użyć znacznie więcej. I na przykład w naszej sieci rozpoznawania cyfr możemy uzyskać tablicę 500 liczb, stukając w poprzednią warstwę. I jest to prawdopodobnie rozsądna tablica do wykorzystania jako „osadzanie obrazu”. Jeśli chcemy stworzyć wyraźną wizualizację „przestrzeni obrazu” dla odręcznie zapisanych cyfr, musimy „zredukować wymiar”, skutecznie rzutując 500-wymiarowy wektor, który mamy, powiedzmy, w przestrzeń 3D:



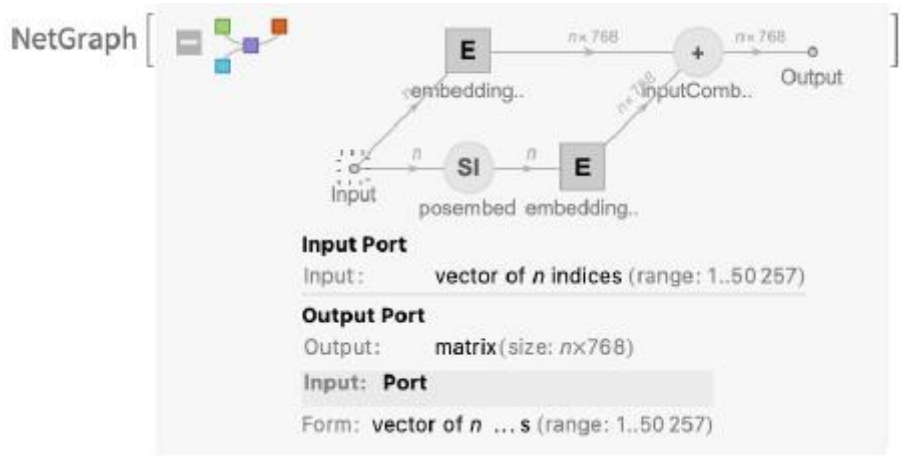
Właśnie rozmawialiśmy o tworzeniu charakterystyki (a tym samym osadzaniu) dla obrazów w oparciu o skuteczną identyfikację podobieństwa obrazów poprzez określenie, czy (zgodnie z naszym zestawem treningowym) odpowiadają one tej samej odręcznej cyfrze. I możemy zrobić to samo znacznie bardziej ogólnie dla obrazów, jeśli mamy zestaw uczący, który identyfikuje, powiedzmy, które z 5000 powszechnych typów obiektów (kot, pies, krzesło, …) przedstawia każdy obraz. W ten sposób możemy wykonać osadzanie obrazu, które jest „zakotwiczone" przez naszą identyfikację typowych obiektów, ale następnie „uogólnia to" zgodnie z zachowaniem sieci neuronowej. Chodzi o to, że o ile to zachowanie jest zgodne z tym, jak my, ludzie, postrzegamy i interpretujemy obrazy, skończy się to osadzeniem, które „wydaje nam się właściwe" i jest przydatne w praktyce w wykonywaniu zadań „podobnych do ludzkiego osądu". OK, więc jak zastosować to samo podejście, aby znaleźć osadzenie słów? Kluczem jest rozpoczęcie od zadania dotyczącego słów, dla których możemy z łatwością trenować. A standardowym takim zadaniem jest „przewidywanie słów". Wyobraź sobie, że dostaliśmy „kota ___". Jakie jest prawdopodobieństwo, że różne słowa „wypełnią lukę" w oparciu o duży zbiór tekstów (np. treść sieci)? Albo, alternatywnie, biorąc pod uwagę „___ czarne ___", jakie są prawdopodobieństwa dla różnych „słów flankujących"? Jak ustawić ten problem dla sieci neuronowej? Ostatecznie musimy sformułować wszystko w kategoriach liczb. Jednym ze sposobów na to jest po prostu przypisanie unikalnego numeru każdemu z około 50 000 popularnych słów w języku angielskim. Na przykład „the" może oznaczać 914, a „cat" (ze spacją przed) 3542. (I to są rzeczywiste liczby używane przez GPT-2). Tak więc w przypadku problemu „the ___ cat" , nasze dane wejściowe mogą być {914, 3542}. Jakie powinno być wyjście? Cóż, powinna to być lista około 50 000 liczb, które skutecznie podają prawdopodobieństwa dla każdego z możliwych słów „wypełniających". I jeszcze raz, aby znaleźć osadzenie, chcemy „przechwycić" „wnętrza" sieci neuronowej tuż przed „dojściem do końca" — a następnie wybrać listę liczb, które tam występują i które możemy wymyślić jako „charakteryzujące każde słowo". OK, więc jak wyglądają te charakterystyki? W ciągu ostatnich 10 lat opracowano sekwencję różnych systemów (word2vec , GloVe , BERT , GPT , …), z których każdy opiera się na innym podejściu do sieci neuronowych. Ale ostatecznie wszyscy biorą słowa i charakteryzują je listami od setek do tysięcy liczb. W swojej surowej postaci te „wektory osadzające" są dość mało pouczające. Na przykład oto, co GPT-2 tworzy jako surowe wektory osadzania dla trzech określonych słów:

Jeśli zrobimy takie rzeczy, jak zmierzymy odległości między tymi wektorami, możemy znaleźć takie rzeczy, jak „bliskości" słów. Później omówimy bardziej szczegółowo, co możemy uznać za „poznawcze" znaczenie takich osadzeń. Ale na razie najważniejsze jest to, że mamy sposób na użyteczne przekształcanie słów w zbiory liczb „przyjazne dla sieci neuronowych". Ale w rzeczywistości możemy pójść dalej niż tylko charakteryzować słowa za pomocą zbiorów liczb; możemy to również zrobić dla sekwencji słów, a nawet całych bloków tekstu. A wewnątrz ChatGPT właśnie tak sobie radzi. Bierze tekst, który ma do tej pory, i generuje wektor osadzania, aby go reprezentować. Następnie jego celem jest znalezienie prawdopodobieństwa dla różnych słów, które mogą wystąpić jako następne. I przedstawia swoją odpowiedź na to pytanie jako listę liczb, które zasadniczo dają prawdopodobieństwa dla każdego z około 50 000 możliwych słów. (Ściśle mówiąc, ChatGPT nie zajmuje się słowami, ale raczej „tokenami" — wygodnymi jednostkami językowymi, które mogą być całymi słowami lub po prostu fragmentami, takimi jak „pre", „ing" lub „ized". Praca z tokenami ułatwia aby ChatGPT obsługiwał rzadkie, złożone i nieanglojęzyczne słowa, a czasami, na dobre i na złe, wymyślał nowe słowa).

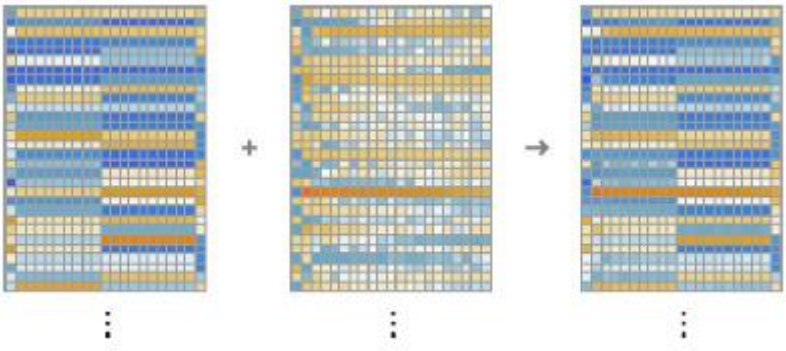
Wewnątrz ChatGPT

OK, więc w końcu jesteśmy gotowi do dyskusji o tym, co jest w ChatGPT. I tak, ostatecznie jest to gigantyczna sieć neuronowa – obecnie wersja tak zwanej sieci GPT-3 ze 175 miliardami wag. Pod wieloma względami jest to sieć neuronowa bardzo podobna do innych, które omówiliśmy. Ale jest to sieć neuronowa, która jest specjalnie skonfigurowana do radzenia sobie z językiem. A jego najbardziej zauważalną cechą jest fragment architektury sieci neuronowej zwany „transformatorem". W pierwszych sieciach neuronowych, które omówiliśmy powyżej, każdy neuron w danej warstwie był w zasadzie połączony (przynajmniej z pewną wagą) z każdym neuronem w warstwie poprzedniej. Ale tego rodzaju w pełni połączona sieć jest (prawdopodobnie) przesadą, jeśli pracuje się z danymi o określonej, znanej strukturze. I tak na przykład na wczesnych etapach pracy z obrazami typowe jest stosowanie tak zwanych konwolucyjnych sieci neuronowych („convnets"), w których neurony są efektywnie rozmieszczone na siatce analogicznej do pikseli na obrazie — i połączone tylko do neuronów znajdujących się w pobliżu sieci. Ideą transformatorów jest zrobienie czegoś przynajmniej trochę podobnego dla sekwencji tokenów, które składają się na fragment tekstu. Ale zamiast po prostu definiować stały region w sekwencji, w którym mogą występować połączenia, transformatory zamiast tego wprowadzają pojęcie „uwagi " — i pomysł „zwracania uwagi" bardziej na niektóre części w kolejności niż inne. Może pewnego dnia sensowne będzie po prostu uruchomienie ogólnej sieci neuronowej i dokonanie wszystkich dostosowań poprzez szkolenie. Ale przynajmniej na razie wydaje się, że w praktyce kluczowe znaczenie ma „modularyzacja" rzeczy — tak jak robią to transformatory i prawdopodobnie tak, jak robią to nasze mózgi. OK, więc co właściwie robi ChatGPT (a raczej sieć GPT-3, na której jest oparta)? Przypomnij sobie, że jego ogólnym celem jest kontynuacja tekstu w „rozsądny" sposób, w oparciu o to, co zobaczył ze szkolenia, które przeszedł (polegającego na przejrzeniu miliardów stron tekstu z web itp.) Tak więc w dowolnym momencie ma określoną ilość tekstu — a jego celem jest znalezienie odpowiedniego wyboru do dodania następnego tokena. Działa w trzech podstawowych etapach. Najpierw bierze sekwencję tokenów, która odpowiada dotychczasowemu tekstowi, i znajduje osadzenie (tj. tablicę liczb), które je reprezentuje. Następnie działa na tym osadzeniu — w „standardowy sposób sieci neuronowej", z wartościami „falującymi"

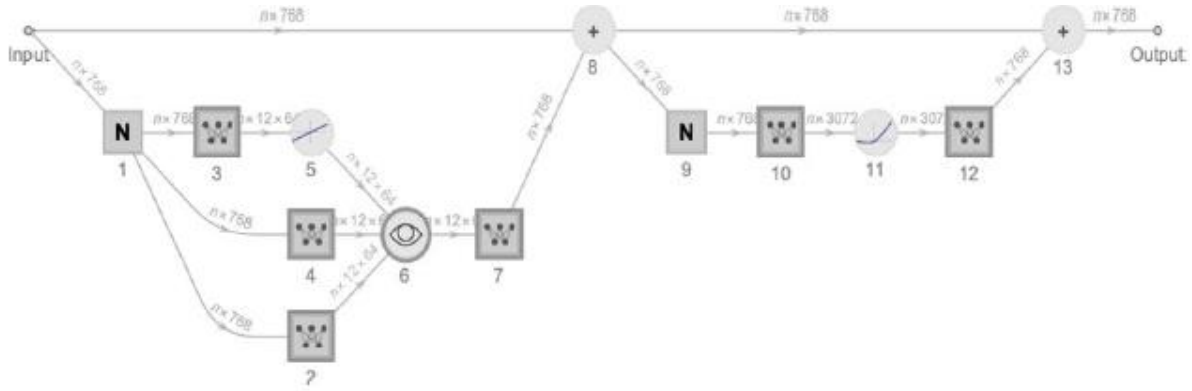
przez kolejne warstwy w sieci — w celu wytworzenia nowego osadzania (tj. nowej tablicy liczb). Następnie bierze ostatnią część tej tablicy i generuje z niej tablicę około 50 000 wartości, które zamieniają się w prawdopodobieństwa dla różnych możliwych kolejnych tokenów. (I tak, tak się składa, że używa się mniej więcej tej samej liczby tokenów, co powszechnych słów w języku angielskim, chociaż tylko około 3000 tokenów to całe słowa, a reszta to fragmenty). Punktem krytycznym jest to, że każdy część tego potoku jest realizowana przez sieć neuronową, której wagi są określane przez kompleksowe uczenie sieci. Innymi słowy, w efekcie nic poza ogólną architekturą nie jest „wyraźnie zaprojektowane"; wszystko jest po prostu „wyuczone" z danych treningowych. Istnieje jednak wiele szczegółów w sposobie konfiguracji architektury — odzwierciedlających wszelkiego rodzaju doświadczenia i wiedzę o sieci neuronowej. I – chociaż to zdecydowanie idzie w błoto – myślę, że warto porozmawiać o niektórych z tych szczegółów, nie tylko po to, aby zrozumieć, co składa się na zbudowanie czegoś takiego jak ChatGPT. Najpierw jest moduł osadzania. Oto schematyczna reprezentacja Wolfram Language dla GPT-2:



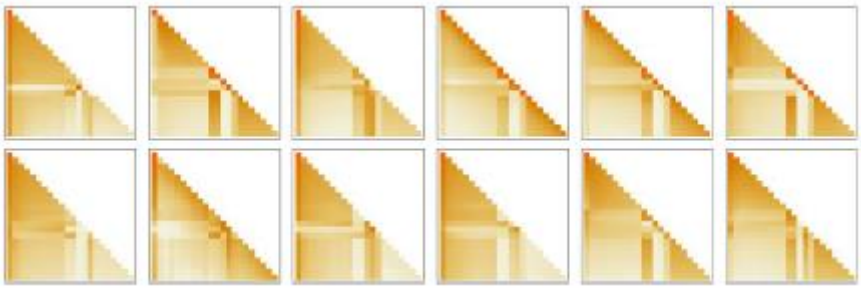
Dane wejściowe to wektor złożony z n tokenów (reprezentowanych jak w poprzedniej sekcji przez liczby całkowite od 1 do około 50 000). Każdy z tych tokenów jest konwertowany (przez jednowarstwową sieć neuronową) na wektor osadzania (o długości 768 dla GPT-2 i 12 288 dla GPT-3 ChatGPT). Tymczasem istnieje „ścieżka wtórna", która pobiera sekwencję pozycji (całkowitych) dla tokenów iz tych liczb całkowitych tworzy kolejny wektor osadzania. Na koniec wektory osadzania z wartości tokena i pozycji tokena są sumowane — w celu utworzenia ostatecznej sekwencji wektorów osadzania z modułu osadzania. Dlaczego po prostu dodaje się wektory osadzania token-value i token-position? Myślę, że nie ma w tym żadnej szczególnej nauki. Po prostu próbowano różnych rzeczy, a ta wydaje się działać. Częścią wiedzy o sieciach neuronowych jest to, że — w pewnym sensie — tak długo, jak konfiguracja jest „z grubsza właściwa", zazwyczaj można skupić się na szczegółach, po prostu przeprowadzając odpowiednie szkolenie, bez potrzeby „rozumienia w poziomie inżynieryjnym", w jaki sposób sieć neuronowa ostatecznie się skonfigurowała. Oto, co robi moduł osadzania, operując na łańcuchu znaków hello hello hello hello hello hello hello hello hello bye bye bye bye bye bye bye bye bye bye :



Elementy wektora osadzania dla każdego tokena są pokazane na dole strony, a na całej stronie widzimy najpierw serię osadzeń „cześć”, a następnie serię osadzeń „pa”. Druga tablica powyżej to osadzenie pozycyjne — z nieco losowo wyglądającą strukturą, która „tak się złożyło, że została nauczona” (w tym przypadku w GPT-2). OK, więc po module osadzania następuje „główne zdarzenie” transformatora: sekwencja tak zwanych „bloków uwagi” (12 dla GPT-2, 96 dla GPT-3 ChatGPT). To wszystko jest dość skomplikowane - i przypomina typowe duże, trudne do zrozumienia systemy inżynieryjne lub, jeśli o to chodzi, systemy biologiczne. W każdym razie oto schematyczna reprezentacja pojedynczego „bloku uwagi” (dla GPT-2):

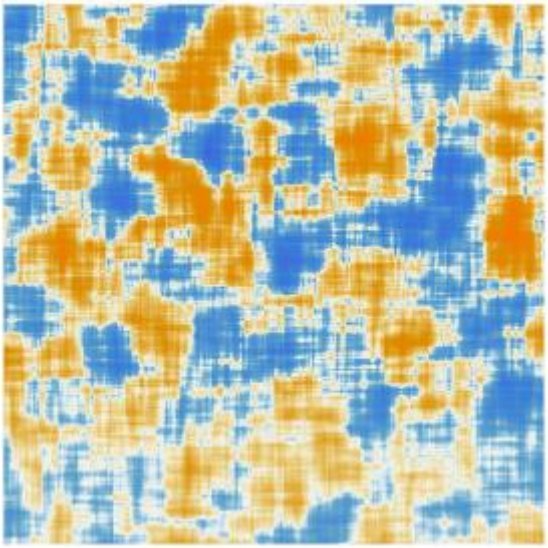


W każdym takim bloku uwagi znajduje się zbiór „głowy uwagi” (12 dla GPT-2, 96 dla GPT-3 ChatGPT) - z których każdy działa niezależnie na różnych fragmentach wartości w wektorze osadzania. (I tak, nie znamy żadnego konkretnego powodu, dla którego dobrym pomysłem jest podzielenie wektora osadzania ani tego, co „znaczą” jego różne części; to tylko jedna z tych rzeczy, które „odkryto, że działają” .) OK, więc co robią głowy uwagi? Zasadniczo są sposobem na „spojrzenie wstecz” w sekwencję tokenów (tj. w dotychczas powstały tekst) i „opakowanie przeszłości” w formę przydatną do znalezienia następnego tokena. W pierwszej sekcji powyżej mówiliśmy o używaniu 2-gramowych prawdopodobieństw do wybierania słów na podstawie ich bezpośrednich poprzedników. To, co robi mechanizm „uwagi” w transformatorach, polega na umożliwieniu „uwagi” nawet na znacznie wcześniejsze słowa – potencjalnie przechwytując w ten sposób sposób, w jaki, powiedzmy, czasowniki mogą odnosić się do rzeczowników, które pojawiają się w zdaniu o wiele słów przed nimi. Na bardziej szczegółowym poziomie głowa uwagi polega na rekombinacji fragmentów w wektorach osadzania powiązanych z różnymi tokenami o określonych wagach. I tak, na przykład, 12 głów uwagi w pierwszym bloku uwagi (w GPT-2) ma następujące („spojrzenie-cały-w-całą-drogę-do-początku-sekwencji-tokenów”) wzorce „wag rekombinacji” dla powyższego ciągu „hello , bye ”:



Po przetworzeniu przez głowice uwagi, wynikowy „przeważony wektor osadzania” (o długości 768 dla GPT-2 i długości 12 288 dla GPT-3 ChatGPT) jest przepuszczany przez standardową „w pełni połączoną” warstwę sieci neuronowej. Trudno zrozumieć, co robi ta warstwa. Ale oto wykres macierzy wag 768 × 768, której używa (tutaj dla GPT-2):

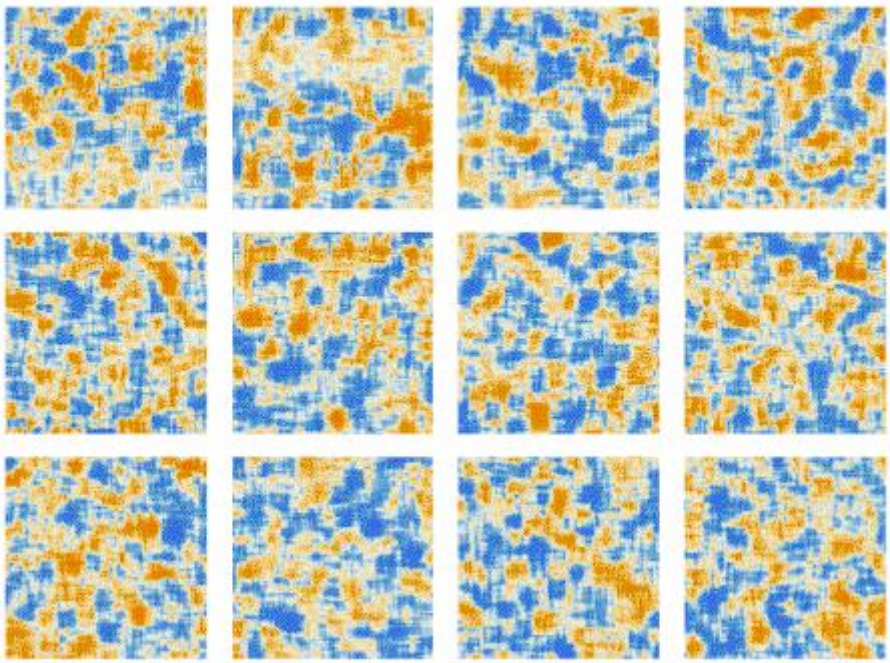
Biorąc pod uwagę średnie ruchome 64 × 64, zaczyna się wyłaniać pewna struktura (w stylu losowego spaceru):



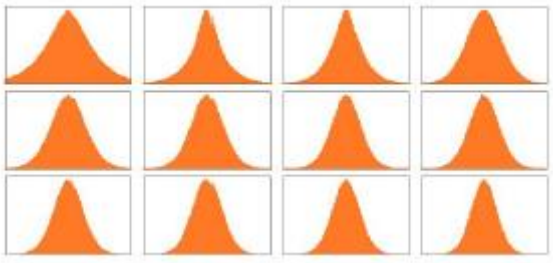
Co decyduje o tej strukturze? Ostatecznie jest to prawdopodobnie jakieś „kodowanie w sieci neuronowej” cech ludzkiego języka. Ale na razie, jakie mogą być te funkcje, jest całkiem nieznane. W

efekcie „otwieramy mózg ChatGPT” (lub przynajmniej GPT-2) i odkrywamy, że tak, jest tam skomplikowany i nie rozumiemy go — mimo że w końcu tworzy rozpoznawalny ludzki język . OK, więc po przejściu przez jeden blok uwagi mamy nowy wektor osadzania — który jest następnie sukcesywnie przepuszczany przez dodatkowe bloki uwagi (łącznie 12 dla GPT-2; 96 dla GPT-3). Każdy blok uwagi ma swój własny wzór „uwagi” i „w pełni połączonych” wag. Oto dla GPT-2 sekwencja wag uwagi dla wejścia „cześć, pa” dla pierwszej głowy uwagi:

A oto (uśrednione ruchome) „macierze” dla w pełni połączonych warstw:



Co ciekawe, chociaż te „macierze wag” w różnych blokach uwagi wyglądają dość podobnie, rozkłady rozmiarów wag mogą być nieco inne (i nie zawsze są gaussowskie):



Więc po przejściu przez te wszystkie bloki uwagi, jaki jest efekt netto transformatora? Zasadniczo ma to na celu przekształcenie oryginalnej kolekcji osadzeń dla sekwencji tokenów w ostateczną kolekcję. A szczególny sposób działania ChatGPT polega na wybraniu ostatniego osadzania w tej kolekcji i

„zdekodowaniu” go w celu utworzenia listy prawdopodobieństw dla tego, który token powinien być następny. To jest w zarysie to, co jest w ChatGPT. Może się to wydawać skomplikowane (nie tylko ze względu na wiele nieuchronnie arbitralnych „wyborów inżynieryjnych”), ale w rzeczywistości ostateczne elementy są niezwykle proste. Ponieważ ostatecznie mamy do czynienia z siecią neuronową wykonaną z „sztucznych neuronów”, z których każdy wykonuje proste czynności operacja polegająca na pobraniu zbioru liczbowych danych wejściowych, a następnie połączeniu ich z określonymi wagami. Oryginalne dane wejściowe do ChatGPT to tablica liczb (wektory osadzania dotychczasowych tokenów), a to, co dzieje się, gdy ChatGPT „uruchamia się” w celu wytworzenia nowego tokena, polega na tym, że te liczby „przechodzą” przez warstwy sieci neuronowej, z każdym neuronem „robiącym swoje” i przekazującym wynik do neuronów w następnej warstwie. Nie ma pętli ani „powrotu”. Wszystko po prostu „przekazuje dalej” przez sieć. Jest to bardzo odmienna konfiguracja od typowego systemu obliczeniowego — takiego jak maszyna Turinga — w którym wyniki są wielokrotnie „ponownie przetwarzane” przez te same elementy obliczeniowe. Tutaj - przynajmniej przy generowaniu danego tokena wyjścia - używany jest każdy element obliczeniowy (tj. Neuron) tylko raz. Ale w pewnym sensie nadal istnieje „zewnętrzna pętla”, która ponownie wykorzystuje elementy obliczeniowe nawet w ChatGPT. Ponieważ kiedy ChatGPT ma zamiar wygenerować nowy token, zawsze „odczytuje” (tj. przyjmuje jako dane wejściowe) całą sekwencję tokenów, które pojawiają się przed nim, w tym tokeny, które sam ChatGPT „zapisał” wcześniej. I możemy myśleć o tej konfiguracji w ten sposób, że ChatGPT – przynajmniej na najbardziej zewnętrznym poziomie – obejmuje „pętlę sprzężenia zwrotnego”, chociaż taką, w której każda iteracja jest wyraźnie widoczna jako token pojawiający się w generowanym przez nią tekście. Ale wróćmy do rdzenia ChatGPT: sieci neuronowej, która jest wielokrotnie używana do generowania każdego tokena. Na pewnym poziomie jest to bardzo proste: cała kolekcja identycznych sztucznych neuronów. A niektóre części sieci składają się po prostu z („w pełni połączonych”) warstw neuronów, w których każdy neuron w danej warstwie jest połączony (z pewną wagą) z każdym neuronem w warstwie poprzedzającej. Ale szczególnie ze swoją architekturą transformatora, ChatGPT ma części o większej strukturze, w których połączone są tylko określone neurony na różnych warstwach. (Oczywiście nadal można powiedzieć, że „wszystkie neurony są połączone” — ale niektóre mają po prostu zerową wagę.) Ponadto istnieją aspekty sieci neuronowej w ChatGPT, o których nie należy w naturalny sposób myśleć jako składających się wyłącznie z „homogenicznych ” warstwy. I na przykład – jak wskazuje powyższe ikoniczne podsumowanie – wewnątrz bloku uwagi znajdują się miejsca, w których „wykonuje się wiele kopii” przychodzących danych, z których każda przechodzi następnie inną „ścieżkę przetwarzania”, potencjalnie obejmującą inną liczbę warstw i tylko później rekombinacja. Ale chociaż może to być dogodna reprezentacja tego, co się dzieje, zawsze przynajmniej w zasadzie można pomyśleć o „gęstym wypełnieniu” warstw, ale po prostu o zerowych wagach. Jeśli spojrzeć na najdłuższą ścieżkę przez ChatGPT, zaangażowanych jest około 400 (rdzeniowych) warstw — pod pewnymi względami nie jest to ogromna liczba. Istnieją jednak miliony neuronów — w sumie 175 miliardów połączeń, a zatem 175 miliardów wag. Należy zdać sobie sprawę, że za każdym razem, gdy ChatGPT generuje nowy token, musi wykonać obliczenia obejmujące każdą z tych wag. Implementacyjnie obliczenia te można nieco uporządkować „według warstw” w wysoce równoległe operacje tablicowe, które można wygodnie wykonywać na procesorach graficznych. Ale dla każdego wyprodukowanego tokena nadal trzeba wykonać 175 miliardów obliczeń (i ostatecznie trochę więcej) — więc tak, nie jest zaskakujące, że wygenerowanie długiego fragmentu tekstu za pomocą ChatGPT może zająć trochę czasu. Ale w ostatecznym rozrachunku niezwykłe jest to, że wszystkie te operacje – pojedynczo tak proste, jak są – mogą w jakiś sposób razem wykonać tak dobrą „ludzką” pracę polegającą na generowaniu tekstu. Należy jeszcze raz podkreślić, że (przynajmniej o ile nam wiadomo) nie ma „ostatecznego teoretycznego powodu”, dla którego coś takiego miałoby działać. I w rzeczywistości, jak będziemy omawiać, myślę, że musimy postrzegać to jako - potencjalnie zaskakujące - odkrycie naukowe: że w jakiś sposób w sieci neuronowej, takiej jak

ChatGPT, możliwe jest uchwycenie istoty tego, co ludzkie mózgi potrafią zrobić w generowaniu języka .

Szkolenie ChatGPT

OK, więc przedstawiliśmy teraz zarys działania ChatGPT po skonfigurowaniu. Ale jak to się ułożyło? W jaki sposób określono wszystkie te 175 miliardów wag w jego sieci neuronowej? Zasadniczo są wynikiem szkolenia na bardzo dużą skalę, opartego na ogromnym zbiorze tekstów — w Internecie, w książkach itp. — napisanych przez ludzi. Jak powiedzieliśmy, nawet biorąc pod uwagę wszystkie te dane treningowe, z pewnością nie jest oczywiste, że sieć neuronowa byłaby w stanie z powodzeniem tworzyć tekst „podobny do ludzkiego". I znowu wydaje się, że potrzebne są szczegółowe elementy inżynierii, aby tak się stało. Ale wielką niespodzianką — i odkryciem — ChatGPT jest to, że jest to w ogóle możliwe. I że – w efekcie – sieć neuronowa o „zaledwie" 175 miliardach wag może stworzyć „rozsądny model" tekstu pisanego przez ludzi. W dzisiejszych czasach istnieje wiele tekstów napisanych przez ludzi, które są dostępne w formie cyfrowej. Publiczna sieć ma co najmniej kilka miliardów stron napisanych przez ludzi, zawierających w sumie może bilion słów tekstu. A jeśli uwzględni się niepubliczne strony internetowe, liczby mogą być co najmniej 100 razy większe. Do tej pory udostępniono ponad 5 milionów zdigitalizowanych książek (spośród około 100 milionów, które kiedykolwiek opublikowano), co daje kolejne 100 miliardów słów tekstu. I to nawet nie wspominając o tekście pochodzącym z mowy w filmach itp. (Dla osobistego porównania, mój całkowity dorobek opublikowanych materiałów to nieco mniej niż 3 miliony słów, a przez ostatnie 30 lat napisałem około 15 milionów słów e-maili i w sumie napisałem około 50 milionów słów – a w ciągu ostatnich kilku lat wypowiedziałem ponad 10 milionów słów w transmisjach na żywo . I tak, wyszkolę z tego wszystkiego bota). , biorąc pod uwagę wszystkie te dane, w jaki sposób można z nich wytrenować sieć neuronową? Podstawowy proces jest bardzo podobny do omówionego powyżej w prostych przykładach. Przedstawiasz partię przykładów, a następnie dostosowujesz wagi w sieci, aby zminimalizować błąd („stratę"), jaki sieć popełnia na tych przykładach. Najważniejszą rzeczą, która jest kosztowna w przypadku „powrotnej propagacji" błędu, jest to, że za każdym razem, gdy to robisz, każda waga w sieci zazwyczaj zmienia się przynajmniej odrobinę, a jest tylko wiele wag, z którymi trzeba sobie poradzić. (Rzeczywiste „obliczanie wsteczne" jest zwykle tylko o mały stały czynnik trudniejsze niż obliczenie do przodu.) Dzięki nowoczesnemu sprzętowi GPU łatwo jest obliczyć wyniki z partii tysięcy przykładów równolegle. Ale jeśli chodzi o faktyczne aktualizowanie wag w sieci neuronowej, obecne metody wymagają, aby robić to zasadniczo partiami. (I tak, prawdopodobnie właśnie w tym miejscu rzeczywiste mózgi — z ich połączonymi elementami obliczeniowymi i pamięciowymi — mają na razie przynajmniej przewagę architektoniczną). Nawet w pozornie prostych przypadkach uczenia się funkcji numerycznych, o których mówiliśmy wcześniej, stwierdziliśmy, że często musieli używać milionów przykładów, aby pomyślnie wytrenować sieć, przynajmniej od zera. Ile zatem przykładów będziemy potrzebować, aby wytrenować model „języka podobnego do ludzkiego"? Wydaje się, że nie ma żadnego podstawowego „teoretycznego" sposobu poznania. Ale w praktyce ChatGPT został pomyślnie przeszkolony na kilkuset miliardach słów tekstu. Część tekstu była podana kilka razy, część tylko raz. Ale w jakiś sposób „dostał to, czego potrzebował" z tekstu, który zobaczył. Ale biorąc pod uwagę ilość tekstu do nauki, jak dużej sieci potrzeba, aby „dobrze się tego nauczyć"? Ponownie, nie mamy jeszcze fundamentalnego teoretycznego sposobu, aby powiedzieć. Ostatecznie — jak omówimy to poniżej — prawdopodobnie istnieje pewna „całkowita zawartość algorytmiczna" ludzkiego języka i tego, co ludzie zazwyczaj w nim mówią. Ale następne pytanie dotyczy tego, jak wydajna będzie sieć neuronowa we wdrażaniu modelu opartego na tej algorytmicznej treści. I znowu nie wiemy — chociaż sukces ChatGPT sugeruje, że jest dość wydajny. I na koniec możemy po prostu zauważyć, że ChatGPT robi to, co robi, używając kilkuset miliardów wag — porównywalnych pod względem liczby do całkowitej liczby słów (lub tokenów) danych treningowych, które mu podano. W pewnym sensie jest to być może zaskakujące

(choć obserwowane empirycznie również w mniejszych analogach ChatGPT), że „rozmiar sieci", która wydaje się działać dobrze, jest tak porównywalny z „rozmiarem danych treningowych". W końcu z pewnością nie jest tak, że w jakiś sposób „wewnątrz ChatGPT" cały ten tekst z sieci, książek i tak dalej jest „bezpośrednio przechowywany". Ponieważ to, co faktycznie znajduje się w ChatGPT, to zbiór liczb — z dokładnością nieco mniejszą niż 10 cyfr — które są rodzajem rozproszonego kodowania zagregowanej struktury całego tego tekstu. Innymi słowy, możemy zapytać, jaka jest „skuteczna treść informacyjna" ludzkiego języka i co zazwyczaj się w nim mówi. Istnieje surowy korpus przykładów języka. A potem jest reprezentacja w sieci neuronowej ChatGPT. Ta reprezentacja jest bardzo prawdopodobnie daleka od reprezentacji „algorytmicznie minimalnej" (co omówimy poniżej). Ale jest to reprezentacja, którą można łatwo wykorzystać w sieci neuronowej. I w tej reprezentacji wydaje się, że w końcu jest raczej niewielka „kompresja" danych treningowych; wydaje się, że do przeniesienia „zawartości informacyjnej" słowa danych treningowych potrzeba średnio tylko nieco mniej niż jednej wagi sieci neuronowej. Kiedy uruchamiamy ChatGPT do generowania tekstu, zasadniczo musimy użyć każdej wagi raz. Więc jeśli istnieje n wag, mamy do wykonania n kroków obliczeniowych — chociaż w praktyce wiele z nich można zazwyczaj wykonać równolegle w procesorach graficznych. Ale jeśli potrzebujemy około n słów danych treningowych, aby ustawić te wagi, to z tego, co powiedzieliśmy powyżej, możemy wywnioskować, że będziemy potrzebować około n 2 kroków obliczeniowych, aby przeprowadzić szkolenie sieci - dlatego z obecnych metod, w końcu trzeba mówić o wysiłkach szkoleniowych wartych miliardy dolarów.

Poza szkoleniem podstawowym

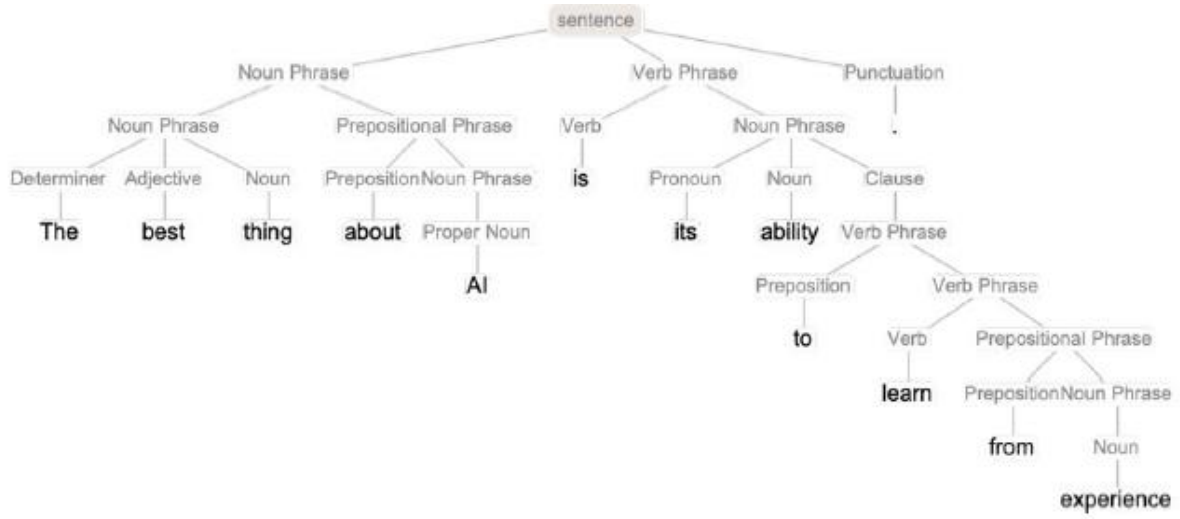
Większość wysiłku włożonego w szkolenie ChatGPT pochłania „pokazywanie" dużych ilości istniejącego tekstu z sieci, książek itp. Ale okazuje się, że jest jeszcze jedna – najwyraźniej dość ważna – część. Gdy tylko zakończy „surowe szkolenie" z oryginalnego korpusu tekstu, który został pokazany, sieć neuronowa w ChatGPT jest gotowa do rozpoczęcia generowania własnego tekstu, kontynuując podpowiedzi itp. Ale chociaż wyniki z tego mogą często wydawać się rozsądne , mają tendencję – szczególnie w przypadku dłuższych fragmentów tekstu – do „oddalania się" w sposób często nie przypominający człowieka. Nie jest to coś, co można łatwo wykryć, powiedzmy, przeprowadzając tradycyjne statystyki tekstu. Ale jest to coś, co prawdziwi ludzie czytający tekst łatwo zauważają. Kluczową ideą przy konstruowaniu ChatGPT było zrobienie kolejnego kroku po „pasywnym czytaniu" rzeczy takich jak sieć: aby prawdziwi ludzie aktywnie wchodzili w interakcję z ChatGPT, widzieli, co produkuje, i w efekcie przekazywali mu informacje zwrotne na temat „jak być dobry chatbot". Ale w jaki sposób sieć neuronowa może wykorzystać tę informację zwrotną? Pierwszym krokiem jest po prostu poproszenie ludzi o ocenę wyników z sieci neuronowej. Ale potem budowany jest inny model sieci neuronowej, który próbuje przewidzieć te oceny. Ale teraz ten model przewidywania można uruchomić — zasadniczo jak funkcję straty — na oryginalnej sieci, w efekcie pozwalając na „dostrojenie" tej sieci na podstawie przekazanych informacji zwrotnych od ludzi. A wyniki w praktyce wydają się mieć duży wpływ na powodzenie systemu w tworzeniu wyników „podobnych do człowieka". Ogólnie rzecz biorąc, interesujące wydaje się, jak mało „szturchania" „pierwotnie wyszkolonej" sieci wydaje się potrzebować, aby z pożytkiem podążała w określonych kierunkach. Można by pomyśleć, że aby sieć zachowywała się tak, jakby „nauczyła się czegoś nowego", trzeba by wejść i uruchomić algorytm uczący, dostosować wagi i tak dalej. Ale tak nie jest. Zamiast tego wydaje się, że wystarczy powiedzieć coś ChatGPT jeden raz — w ramach podanego monitu — a następnie może on z powodzeniem wykorzystać to, co mu powiedziałeś, podczas generowania tekstu. I jeszcze raz, fakt, że to działa, jest, jak sądzę, ważną wskazówką do zrozumienia, co ChatGPT „naprawdę robi" i jak odnosi się to do struktury ludzkiego języka i myślenia. Z pewnością jest w nim coś raczej ludzkiego: przynajmniej po tym całym wstępnym szkoleniu możesz mu coś powiedzieć tylko raz i może to „zapamiętać" — przynajmniej „wystarczająco długo", aby wygenerować fragment tekstu, używając

tego . Więc co się dzieje w takim przypadku? Może być tak, że „wszystko, co możesz mu powiedzieć, już gdzieś tam jest" — a ty po prostu prowadzisz to we właściwe miejsce. Ale to nie wydaje się prawdopodobne. Zamiast tego bardziej prawdopodobne wydaje się to, że tak, elementy już tam są, ale szczegóły są określone przez coś w rodzaju „trajektorii między tymi elementami" i właśnie to wprowadzasz, kiedy coś mówisz. I rzeczywiście, podobnie jak w przypadku ludzi, jeśli powiesz mu coś dziwacznego i nieoczekiwanego, co całkowicie nie pasuje do znanej mu struktury, nie wydaje się, aby udało mu się to „zintegrować". Może go „zintegrować" tylko wtedy, gdy w zasadzie jeździ w dość prosty sposób na ramach, które już ma. Warto również ponownie podkreślić, że nieuchronnie istnieją „algorytmiczne ograniczenia" tego, co sieć neuronowa może „wychwycić". Powiedz mu „płytkie" reguły w postaci „to idzie do tamtego" itp., a sieć neuronowa najprawdopodobniej będzie w stanie je reprezentować i reprodukować – i rzeczywiście to, co „już wie" z języka, da mu natychmiastowy wzór do naśladowania. Ale spróbuj podać reguły dla rzeczywistych „głębokich" obliczeń, które obejmują wiele potencjalnie nieredukowalnych obliczeniowo kroków, a to po prostu nie zadziała. (Pamiętaj, że na każdym kroku zawsze po prostu „przekazuje dane dalej" w swojej sieci, nigdy nie zapętlając się, z wyjątkiem generowania nowych tokenów.) Oczywiście sieć może nauczyć się odpowiedzi na określone „nieredukowalne" obliczenia. Ale gdy tylko pojawią się kombinatoryczne liczby możliwości, żadne takie podejście „styl wyszukiwania w tabeli" nie będzie działać. I tak, tak, podobnie jak ludzie, nadszedł czas, aby sieci neuronowe „sięgnęły" i używały rzeczywistych narzędzi obliczeniowych. (I tak, Wolfram|Alpha i Wolfram Language są wyjątkowo odpowiednie, ponieważ zostały stworzone do „rozmawiania o rzeczach na świecie", podobnie jak sieci neuronowe oparte na modelu języka).

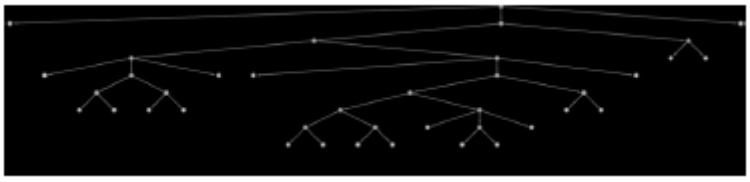
Co naprawdę umożliwia działanie ChatGPT?

Ludzki język – i procesy myślowe związane z jego generowaniem – zawsze wydawał się reprezentować swego rodzaju szczyt złożoności. I rzeczywiście wydawało się nieco niezwykłe, że ludzki mózg – z siecią „zaledwie" około 100 miliardów neuronów (i może 100 bilionów połączeń) może być za to odpowiedzialny. Być może, jak można by sobie wyobrazić, mózgi to coś więcej niż ich sieci neuronów - jak jakaś nowa warstwa nieodkrytej fizyki. Ale teraz dzięki ChatGPT mamy nową ważną informację: wiemy, że czysta, sztuczna sieć neuronowa z mniej więcej tyloma połączeniami, ile mózg ma neuronów, jest w stanie wykonać zaskakująco dobrą robotę, generując ludzki język. I tak, to wciąż duży i skomplikowany system — z mniej więcej taką liczbą wag sieci neuronowych, ile jest obecnie dostępnych słów tekstu na świecie. Ale na pewnym poziomie nadal trudno jest uwierzyć, że całe bogactwo języka i rzeczy, o których może mówić, można zamknąć w tak skończonym systemie. Częściowo to, co się dzieje, jest bez wątpienia odzwierciedleniem wszechobecnego zjawiska (które po raz pierwszy stało się oczywiste na przykładzie reguły 30), że procesy obliczeniowe mogą w efekcie znacznie zwiększyć pozorną złożoność systemów, nawet jeśli ich podstawowe zasady są proste. Ale w rzeczywistości, jak omówiliśmy powyżej, sieci neuronowe w rodzaju używanych w ChatGPT są zwykle konstruowane specjalnie w celu ograniczenia efektu tego zjawiska - i związanej z nim nieredukowalności obliczeniowej - w interesie uczynienia ich szkolenia bardziej dostępnym. Więc jak to się dzieje, że coś takiego jak ChatGPT może zajść tak daleko, jak to robi z językiem? Podstawowa odpowiedź, jak sądzę, jest taka, że język jest na podstawowym poziomie prostszy, niż się wydaje. A to oznacza, że ChatGPT — nawet ze swoją ostatecznie prostą strukturą sieci neuronowej — jest w stanie z powodzeniem „uchwycić istotę" ludzkiego języka i stojącego za nim myślenia. Co więcej, podczas szkolenia ChatGPT w jakiś sposób „pośrednio odkrył" wszelkie prawidłowości w języku (i myśleniu), które to umożliwiają. Sukces ChatGPT polega, jak sądzę, na dostarczeniu nam dowodu na istnienie fundamentalnego i ważnego odkrycia naukowego: sugeruje, że możemy spodziewać się nowych głównych „praw języka" — a właściwie „praw myślenia" — do odkrycia . W ChatGPT — zbudowanym jako sieć neuronowa — te prawa są w najlepszym razie ukryte. Ale gdybyśmy mogli w jakiś sposób jasno określić prawa, istnieje potencjał, aby robić rzeczy, które robi ChatGPT w znacznie bardziej

bezpośredni, wydajny i przejrzysty sposób. Ale OK, więc jakie mogą być te prawa? Ostatecznie muszą dać nam jakąś receptę na to, jak język – i to, co nim mówimy – są składane. Później omówimy, w jaki sposób „zajrzenie do środka ChatGPT" może dać nam pewne wskazówki na ten temat i jak to, co wiemy z budowania języka obliczeniowego, sugeruje ścieżkę naprzód. Ale najpierw omówmy dwa od dawna znane przykłady tego, czym są „prawa języka" — i jak odnoszą się one do działania ChatGPT. Pierwsza to składnia języka. Język to nie tylko przypadkowy zbiór słów. Zamiast tego istnieją (dość) określone reguły gramatyczne określające, w jaki sposób można łączyć różne rodzaje słów: na przykład w języku angielskim rzeczowniki mogą być poprzedzone przymiotnikami, a po nich czasowniki, ale zazwyczaj dwa rzeczowniki nie mogą znajdować się tuż obok każdego Inny. Taką strukturę gramatyczną można (przynajmniej w przybliżeniu) ująć za pomocą zestawu reguł, które określają, w jaki sposób suma „drzew parsowania" może zostać zebrana razem:



ChatGPT nie ma żadnej wyraźnej „wiedzy" o takich zasadach. Ale w jakiś sposób podczas swojego szkolenia domyślnie je „odkrywa" – a potem wydaje się być dobry w podążaniu za nimi. Jak to działa? Na poziomie „dużego obrazu" nie jest to jasne. Ale aby uzyskać pewien wgląd, być może pouczające jest spojrzenie na znacznie prostszy przykład. Rozważmy „język" utworzony z sekwencji ('s i)'s, z gramatyką, która określa, że nawiasy powinny być zawsze zrównoważone, co jest reprezentowane przez drzewo analizy, takie jak:



Czy możemy wytrenować sieć neuronową, aby tworzyła „poprawne gramatycznie" sekwencje nawiasów? Istnieją różne sposoby obsługi sekwencji w sieciach neuronowych, ale użyjmy sieci transformatorów, tak jak robi to ChatGPT. A mając prostą sieć transformatorów, możemy zacząć karmić ją poprawnymi gramatycznie sekwencjami nawiasów jako przykładami szkoleniowymi. Subtelność (która faktycznie pojawia się również w generacji języka ludzkiego ChatGPT) polega na tym, że oprócz naszych „tokenów treści" (tutaj „(" i „)") musimy dołączyć token „Koniec", który jest generowany w celu wskazania, że dane wyjściowe nie powinny być kontynuowane (tj. w przypadku ChatGPT osiągnięto „koniec historii"). Jeśli skonfigurujemy sieć transformatora z tylko jednym blokiem uwagi z 8 głowami i wektorami cech o długości 128 (ChatGPT również używa wektorów cech o długości

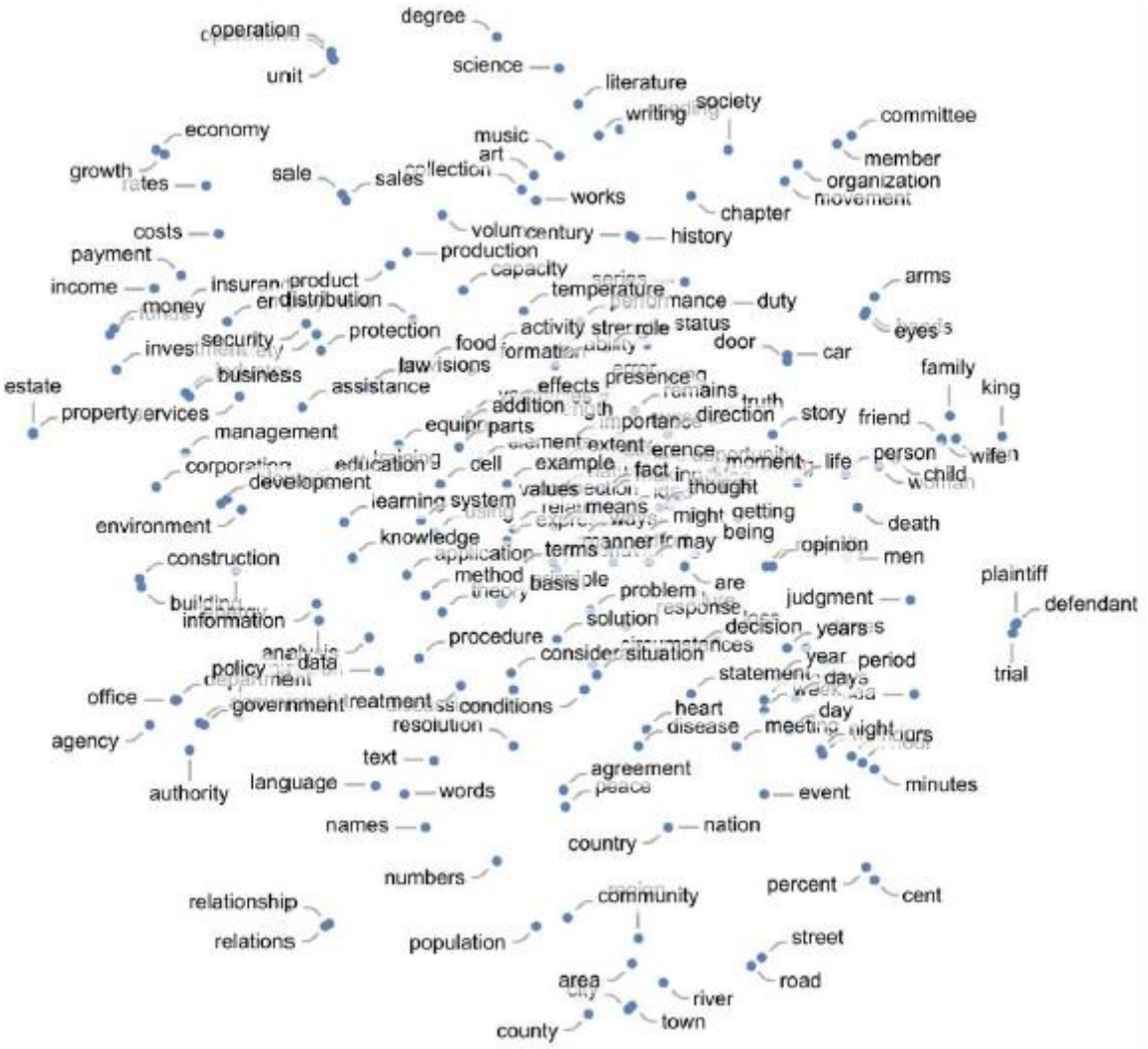
128, ale ma 96 bloków uwagi, każdy z 96 głowami), to nie wydaje się możliwe naucz się dużo o języku nawiasów. Ale przy 2 blokach uwagi proces uczenia się wydaje się być zbieżny — przynajmniej po podaniu około 10 milionów przykładów (i, jak to zwykle bywa z sieciami transformatorów, pokazywanie jeszcze większej liczby przykładów po prostu obniża jego wydajność). Dzięki tej sieci możemy zrobić analogię do tego, co robi ChatGPT, i zapytać o prawdopodobieństwo tego, jaki powinien być następny token — w sekwencji nawiasów:

I tak, do pewnej długości sieć działa dobrze. Ale potem to zaczyna zawodzić. To dość typowe zjawisko w „dokładnej" sytuacji takiej jak ta z siecią neuronową (lub z uczeniem maszynowym w ogólny). Przypadki, które człowiek „może rozwiązać w mgnieniu oka", może rozwiązać również sieć neuronowa. Ale przypadki, które wymagają zrobienia czegoś „bardziej algorytmicznego" (np. jawnego liczenia nawiasów, aby sprawdzić, czy są zamknięte), sieć neuronowa jest w jakiś sposób „zbyt płytka obliczeniowo", aby niezawodnie to zrobić. (Nawiasem mówiąc, nawet w pełni aktualny ChatGPT ma trudności z poprawnym dopasowaniem nawiasów w długich sekwencjach.) Co to oznacza dla rzeczy takich jak ChatGPT i składnia języka takiego jak angielski? Język nawiasów jest „surowy" – i znacznie bardziej przypomina „historię algorytmiczną". Ale w języku angielskim o wiele bardziej realistyczne jest „odgadnięcie", co gramatycznie będzie pasować na podstawie lokalnych wyborów słów i innych wskazówek. I tak, sieć neuronowa jest w tym znacznie lepsza – nawet jeśli może przegapić jakiś „formalnie poprawny" przypadek, który, no cóż, ludzie również mogą przegapić. Ale najważniejsze jest to, że fakt, że język ma ogólną strukturę składniową – z całą wynikającą z tego regularnością – w pewnym sensie ogranicza „ile" sieć neuronowa musi się nauczyć. A kluczową obserwacją „podobną do nauk przyrodniczych" jest to, że architektura transformatorów sieci neuronowych, takich jak ta w ChatGPT, wydaje się z powodzeniem być w stanie nauczyć się rodzaju zagnieżdżonej struktury składniowej przypominającej drzewo, która wydaje się istnieć (przynajmniej w pewnym przybliżeniu) we wszystkich ludzkich językach. Składnia zapewnia jeden rodzaj ograniczeń dla języka. Ale jest ich

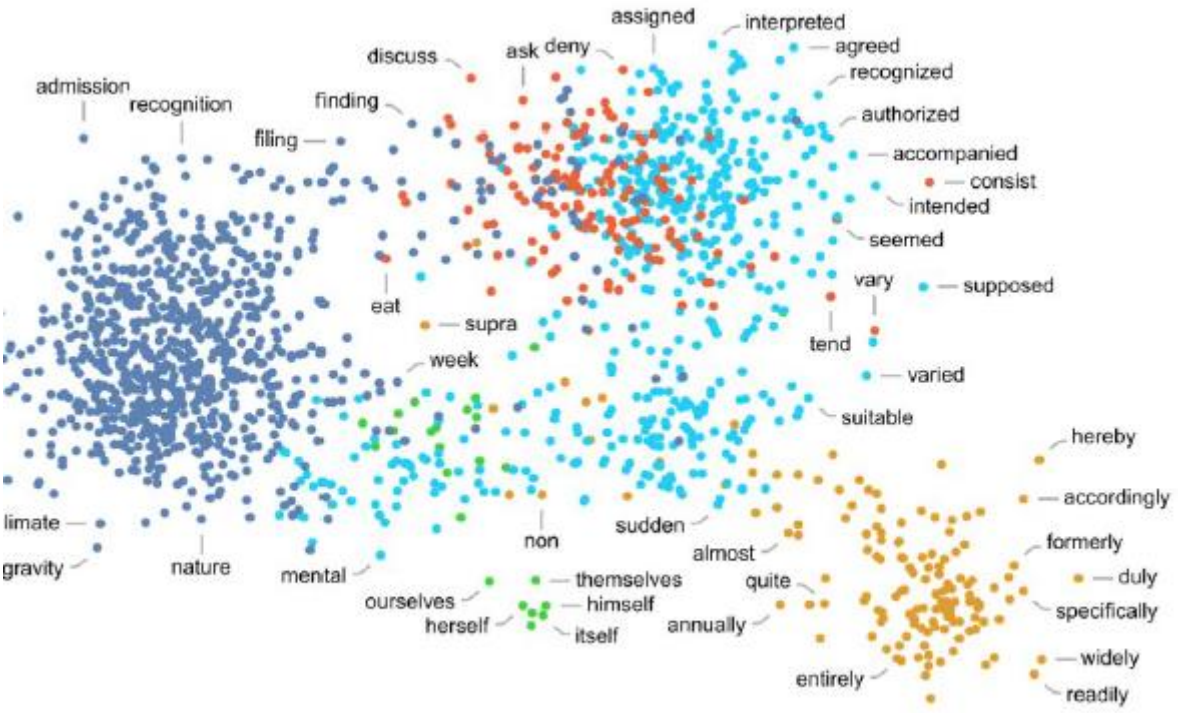
wyraźnie więcej. Zdanie takie jak „Dociekliwe elektrony zjadają niebieskie teorie dla ryb” jest poprawne gramatycznie, ale nie jest czymś, czego normalnie można by się spodziewać i nie byłoby uważane za sukces, gdyby wygenerował je ChatGPT — ponieważ, cóż, z normalnymi znaczeniami dla słowa w nim, to w zasadzie bez sensu. Ale czy istnieje ogólny sposób na stwierdzenie, czy zdanie ma sens? Nie ma na to tradycyjnej ogólnej teorii. Ale jest to coś, o czym można pomyśleć, że ChatGPT pośrednio „opracował teorię” po przeszkoleniu z miliardami (prawdopodobnie znaczących) zdań z sieci itp. Jaka może być ta teoria? Cóż, jest jeden mały zakątek, który jest w zasadzie znany od dwóch tysiącleci i jest to logika. I z pewnością w formie sylogistyki, w jakiej ją odkrył Arystoteles, logika jest zasadniczo sposobem mówienia, że zdania, które są zgodne z pewnymi wzorcami, są rozsądne, podczas gdy inne nie. Na przykład rozsądne jest powiedzenie „Wszystkie X to Y. To nie jest Y, więc to nie jest X” (jak w przypadku „Wszystkie ryby są niebieskie. To nie jest niebieskie, więc to nie jest ryba”). I tak jak można nieco kapryśnie wyobrazić sobie, że Arystoteles odkrył logikę sylogistyczną, przechodząc („w stylu uczenia maszynowego”) przez wiele przykładów retoryki, tak też można sobie wyobrazić, że podczas szkolenia ChatGPT będzie w stanie „odkryć logika sylogistyczna”, przeglądając dużo tekstu w Internecie itp. (I tak, chociaż można oczekiwać, że ChatGPT utworzy tekst zawierający „poprawne wnioski” oparte na rzeczach takich jak logika sylogistyczna, to zupełnie inna historia, jeśli chodzi do bardziej wyrafinowanej logiki formalnej — i myślę, że można się spodziewać, że zawiedzie tutaj z tych samych powodów, co zawodzi przy dopasowywaniu nawiasów.) Ale poza wąskim przykładem logiki, co można powiedzieć o tym, jak systematycznie konstruować (lub rozpoznawać) chociaż sensowny tekst? Tak, istnieją rzeczy takie jak Mad Libs&reg, które używają bardzo specyficznych „szablonów frazowych”. Ale w jakiś sposób ChatGPT pośrednio ma na to znacznie bardziej ogólny sposób. I być może nie ma nic do powiedzenia na temat tego, jak można to zrobić poza „jakoś to się dzieje, gdy masz 175 miliardów mas sieci neuronowych”. Ale mocno podejrzewam, że istnieje o wiele prostsza i silniejsza historia

Znaczenie przestrzeni i semantyczne prawa ruchu

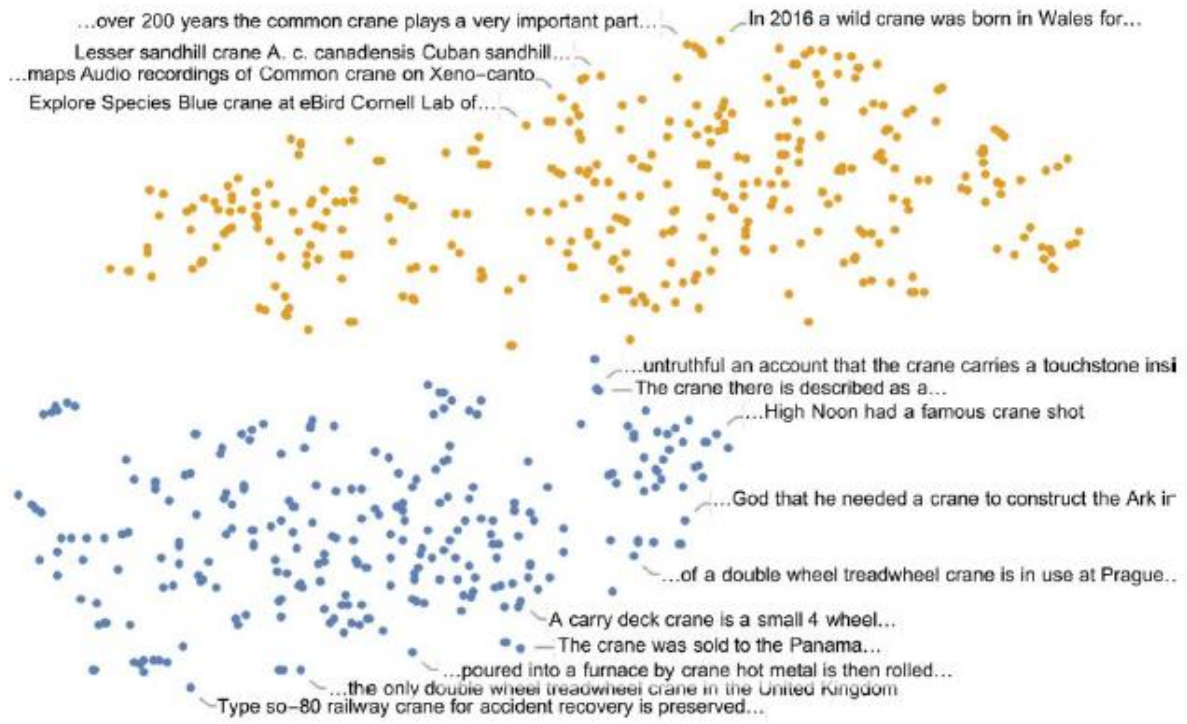
Omówiliśmy powyżej, że wewnątrz ChatGPT każdy fragment tekstu jest skutecznie reprezentowany przez tablicę liczb, które możemy traktować jako współrzędne punktu w jakiejś „językowej przestrzeni cech”. Kiedy więc ChatGPT kontynuuje fragment tekstu, odpowiada to śledzeniu trajektorii w językowej przestrzeni cech. Ale teraz możemy zapytać, co sprawia, że ta trajektoria odpowiada tekstowi, który uważamy za znaczący. A może istnieją jakieś „semantyczne prawa ruchu”, które definiują – lub przynajmniej ograniczają – sposób, w jaki punkty w językowej przestrzeni cech mogą się przemieszczać, zachowując „sensowność”? Jak więc wygląda ta lingwistyczna przestrzeń cech? Oto przykład, w jaki sposób pojedyncze słowa (tutaj rzeczowniki pospolite) mogą zostać ułożone, jeśli rzutujemy taką przestrzeń cech do 2D:



Powyżej widzieliśmy inny przykład oparty na słowach reprezentujących rośliny i zwierzęta. Ale w obu przypadkach chodzi o to, że obok siebie znajdują się „podobne semantycznie słowa”. Jako kolejny przykład, oto jak układają się słowa odpowiadające różnym częściom mowy:

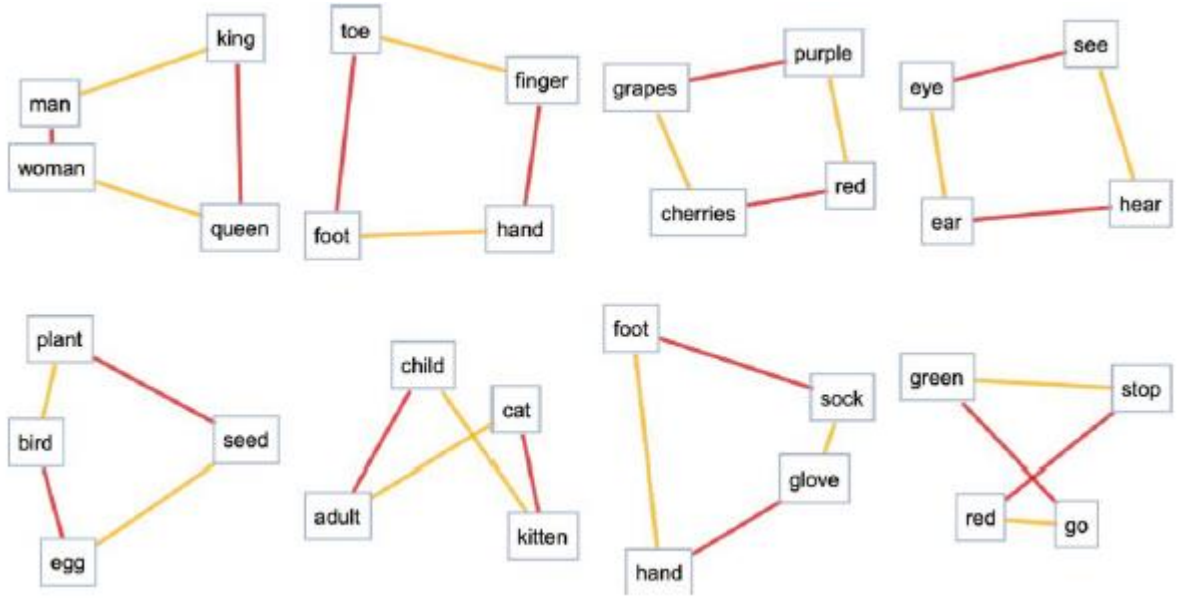


Oczywiście, dane słowo na ogół nie ma tylko „jednego znaczenia” (lub niekoniecznie odpowiada tylko jednej części mowy). A patrząc na to, jak zdania zawierające słowo układają się w przestrzeni cech, często można „oddzielić” różne znaczenia — jak w przykładzie dla słowa „żuraw” (ptak czy maszyna?):

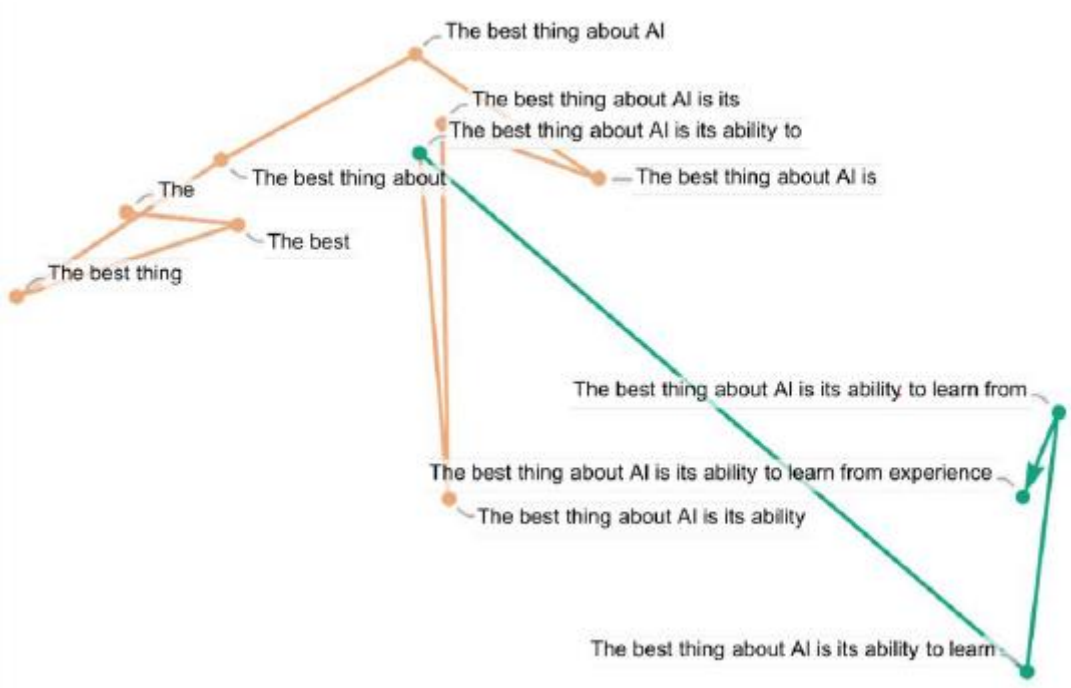


OK, więc jest przynajmniej prawdopodobne, że możemy myśleć o tej przestrzeni cech jako o umieszczaniu „słów w pobliżu znaczenia” blisko tej przestrzeni. Ale jakiego rodzaju dodatkową strukturę możemy zidentyfikować w tej przestrzeni? Czy istnieje na przykład pojęcie „transportu

równoległego", które odzwierciedlałoby „płaskość" w przestrzeni? Jednym ze sposobów, aby sobie z tym poradzić, jest spojrzenie na analogie:

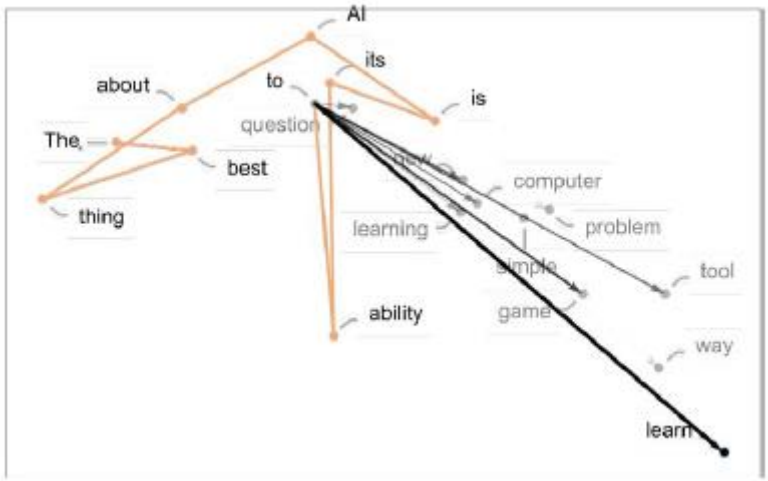


I tak, nawet gdy projektujemy do 2D, często pojawia się przynajmniej „odrobina płaskości", choć z pewnością nie jest to powszechnie widoczne. A co z trajektoriami? Możemy spojrzeć na trajektorię, jaką podąża monit o ChatGPT w przestrzeni funkcji — a następnie możemy zobaczyć, jak ChatGPT kontynuuje:

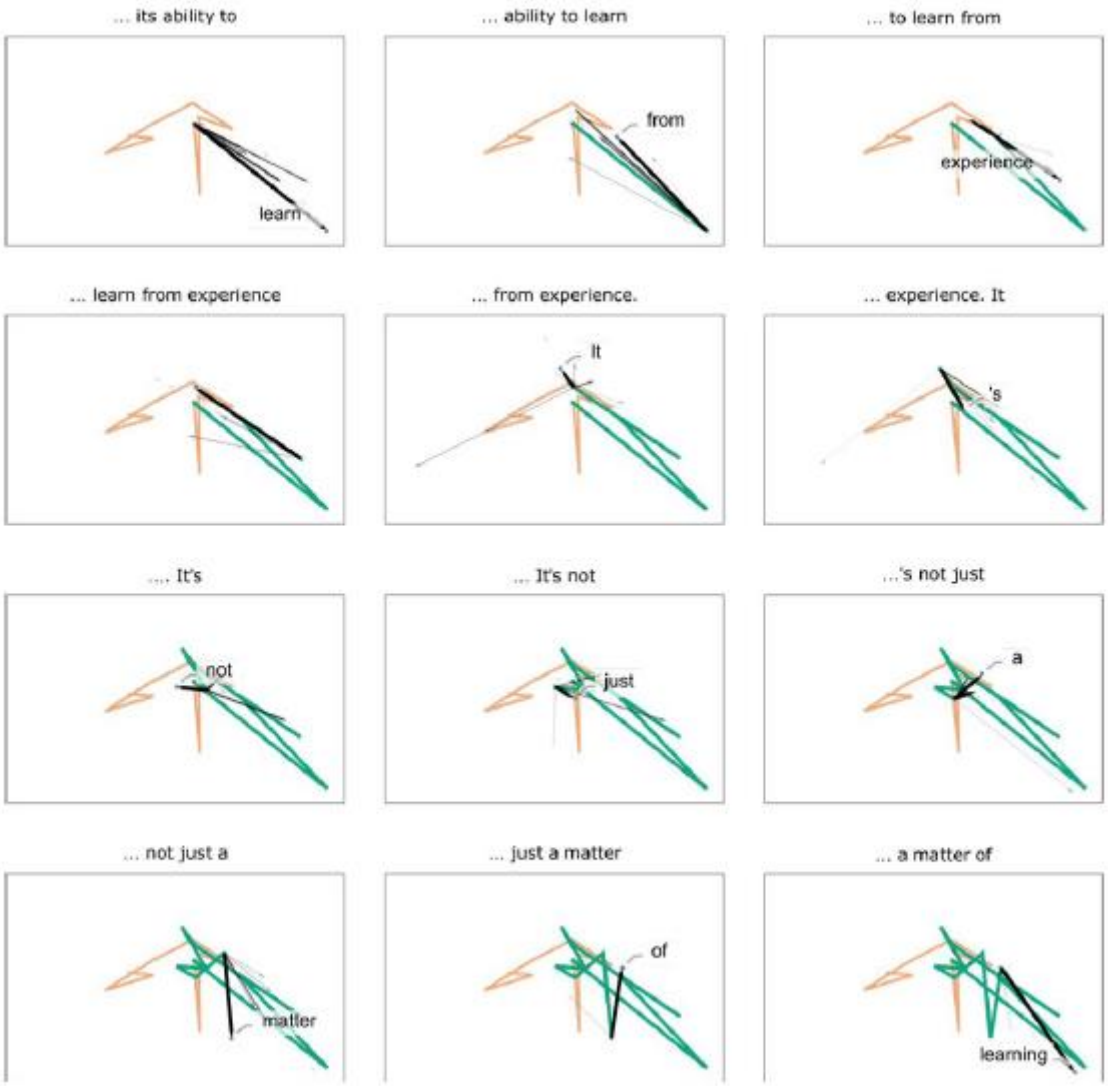


Z pewnością nie ma tu „geometrycznie oczywistego" prawa ruchu. I wcale nie jest to zaskakujące; w pełni spodziewamy się, że będzie to znacznie bardziej skomplikowana historia. I na przykład wcale nie jest oczywiste, że nawet jeśli istnieje „semantyczne prawo ruchu", w jakim rodzaju osadzania (lub w efekcie w jakich „zmiennych") będzie ono najbardziej naturalnie określone. na powyższym obrazku pokazujemy kilka kroków w „trajektorii" — gdzie na każdym etapie wybieramy słowo, które ChatGPT

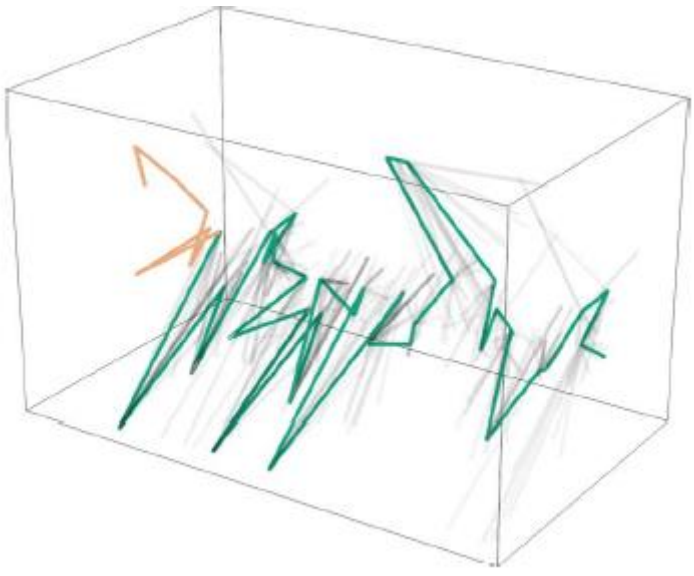
uważa za najbardziej prawdopodobne (przypadek „zerowej temperatury”). Ale możemy też zapytać, jakie słowa mogą „być następne” z jakim prawdopodobieństwem w danym momencie:



W tym przypadku widzimy, że istnieje „fan” słów o wysokim prawdopodobieństwie, który wydaje się zmierzać w mniej lub bardziej określonym kierunku w przestrzeni cech. Co się stanie, jeśli pójdziemy dalej? Oto kolejne „wentylatory”, które pojawiają się, gdy „posuwamy się” po trajektorii:



Oto reprezentacja 3D obejmująca łącznie 40 kroków:



I tak, wygląda to na bałagan - i nie robi nic, by szczególnie zachęcać do pomysłu, że można spodziewać się zidentyfikowania „matematyczno-fizycznych" „semantycznych praw ruchu" poprzez empiryczne badanie „co ChatGPT robi w środku" . Ale być może po prostu patrzymy na „niewłaściwe zmienne" (lub niewłaściwy układ współrzędnych) i gdybyśmy tylko spojrzeli na właściwy, od razu zauważylibyśmy, że ChatGPT robi coś „matematyczno-fizycznego" prostego, jak podążanie za geodezją. Ale na razie nie jesteśmy gotowi do „empirycznego dekodowania" z jego „wewnętrznego zachowania" tego, co ChatGPT „odkrył" na temat tego, jak ludzki język jest „składany".

Gramatyka semantyczna i potęga języka obliczeniowego

Co trzeba zrobić, aby stworzyć „zrozumiały ludzki język"? W przeszłości moglibyśmy przypuszczać, że może to być ludzki mózg. Ale teraz wiemy, że można to zrobić całkiem przyzwoicie za pomocą sieci neuronowej ChatGPT. Mimo to może to wszystko, co możemy zrobić, i nie będzie nic prostszego – ani bardziej zrozumiałego dla ludzi – co zadziała. Ale mam silne podejrzenie, że sukces ChatGPT pośrednio ujawnia ważny „naukowy" fakt: że w rzeczywistości sensowny ludzki język ma o wiele więcej struktury i prostoty, niż kiedykolwiek wiedzieliśmy - i że w końcu mogą istnieć nawet całkiem proste zasady, które opisz, jak taki język można złożyć. Jak wspomnieliśmy powyżej, gramatyka składniowa podaje reguły dotyczące tego, jak słowa odpowiadające rzeczom takim jak różne części mowy mogą być łączone w ludzkim języku. Ale aby poradzić sobie ze znaczeniem, musimy pójść dalej. Jedną z wersji tego, jak to zrobić, jest myślenie nie tylko o gramatyce składniowej języka, ale także o gramatyce semantycznej. Dla celów składni identyfikujemy takie rzeczy jak rzeczowniki i czasowniki. Ale dla celów semantycznych potrzebujemy „delikatniejszych stopni". Na przykład możemy zidentyfikować pojęcie „przemieszczania się" i pojęcie „obiektu", który „zachowuje swoją tożsamość niezależnie od lokalizacji". Istnieje nieskończona liczba konkretnych przykładów każdego z tych „pojęć semantycznych". Ale dla celów naszej gramatyki semantycznej będziemy mieć po prostu ogólną regułę, która zasadniczo mówi, że „obiekty" mogą się „poruszać". Jest wiele do powiedzenia na temat tego, jak to wszystko może działać (niektóre z nich powiedziałem wcześniej). Ale zadowolę się tutaj tylko kilkoma uwagami, które wskazują część potencjalnej ścieżki naprzód. Warto wspomnieć, że nawet jeśli zdanie jest całkowicie OK zgodnie z gramatyką semantyczną, nie oznacza to, że zostało zrealizowane (lub nawet mogłoby zostać zrealizowane) w praktyce. „Słoń udał się na Księżyc" bez wątpienia „przeszedłby" przez naszą gramatykę semantyczną, ale z pewnością nie został zrealizowany (przynajmniej jeszcze) w naszym rzeczywistym świecie – choć jest to absolutnie uczciwa gra dla fikcyjnego świata. Kiedy zaczynamy mówić o „gramatyce semantycznej", wkrótce zaczynamy pytać „Co się pod nią kryje?" Jaki „model świata" zakłada? Gramatyka składniowa polega tak naprawdę na konstruowaniu języka ze słów. Ale gramatyka semantyczna z konieczności angażuje się w pewnego rodzaju „model świata" – coś, co służy jako „szkielet", na szczycie którego język utworzony z rzeczywistych słów może być warstwowy. Do niedawna mogliśmy sobie wyobrazić, że (ludzki) język będzie jedynym ogólnym sposobem opisania naszego „modelu świata". Już kilka wieków temu zaczęto formalizować określone rodzaje rzeczy, opierając się zwłaszcza na matematyce. Ale teraz istnieje znacznie bardziej ogólne podejście do formalizacji: język obliczeniowy. I tak, to był mój wielki projekt realizowany przez ponad cztery dekady (jak teraz jest ucieleśniony w języku Wolframa): opracować precyzyjną reprezentację symboliczną, która może mówić tak szeroko, jak to tylko możliwe, o rzeczach na świecie, a także abstrakcyjnie rzeczy, na których nam zależy. I tak na przykład mamy symboliczne reprezentacje miast, molekuł, obrazów i sieci neuronowych oraz mamy wbudowaną wiedzę o tym, jak obliczać te rzeczy. Po dziesięcioleciach pracy omówiliśmy w ten sposób wiele obszarów. Ale w przeszłości nie zajmowaliśmy się specjalnie „codziennym dyskursem". W „Kupiłem dwa funty jabłek" możemy z łatwością przedstawić (i wykonać obliczenia żywieniowe i inne) „dwa funty jabłek". Ale nie mamy (jeszcze całkiem) symbolicznej reprezentacji „kupiłem". Wszystko to jest związane z ideą gramatyki semantycznej — i celem posiadania ogólnego symbolicznego „zestawu konstrukcyjnego" dla

pojęć, który dałby nam reguły dotyczące tego, co może pasować do czego, a tym samym dla „przepływu" tego, co możemy obrócić na ludzki język. Ale powiedzmy, że mamy ten „symboliczny język dyskursu". Co byśmy z tym zrobili? Moglibyśmy zacząć od generowania „lokalnie znaczącego tekstu". Ale ostatecznie prawdopodobnie będziemy chcieli uzyskać bardziej „globalnie znaczące" wyniki – co oznacza „obliczanie" więcej informacji o tym, co faktycznie może istnieć lub wydarzyć się na świecie (lub być może w jakimś spójnym fikcyjnym świecie). W tej chwili w Wolfram Language mamy ogromną ilość wbudowanej wiedzy obliczeniowej o wielu rodzajach rzeczy. Ale dla kompletnego symbolicznego języka dyskursu musielibyśmy wbudować dodatkowe „rachunki" dotyczące ogólnych rzeczy na świecie: jeśli przedmiot porusza się z A do B iz B do C, to jest przenoszony z A do C itd. Biorąc pod uwagę symboliczny język dyskursu, którego możemy używać do formułowania „samodzielnych wypowiedzi". Ale możemy go również użyć do zadawania pytań o świat w stylu „Wolfram|Alpha". Lub możemy go użyć do stwierdzenia rzeczy, które „chcemy zrobić", prawdopodobnie za pomocą jakiegoś zewnętrznego mechanizmu uruchamiającego. Możemy też użyć go do sformułowania twierdzeń — być może na temat rzeczywistego świata, a może na temat jakiegoś konkretnego świata, który rozważamy, fikcyjnego lub innego. Język ludzki jest z gruntu nieprecyzyjny, między innymi dlatego, że nie jest „powiązany" z konkretną implementacją obliczeniową, a jego znaczenie jest w zasadzie określone jedynie przez „umowę społeczną" między jego użytkownikami. Ale język obliczeniowy ze swej natury ma pewną fundamentalną precyzję — bo ostatecznie to, co określa, zawsze da się „jednoznacznie wykonać na komputerze". Ludzkiemu językowi zwykle udaje się uniknąć pewnej niejasności. (Kiedy mówimy „planeta", czy obejmuje to egzoplanety, czy nie, itp.?) Ale w języku obliczeniowym musimy być precyzyjni i jasno określić wszystkie rozróżnienia, które wprowadzamy. Często wygodnie jest wykorzystać zwykły ludzki język do tworzenia nazw w języku obliczeniowym. Ale znaczenia, jakie mają w języku obliczeniowym, są z konieczności precyzyjne - i mogą, ale nie muszą, obejmować pewne szczególne konotacje w typowym użyciu ludzkiego języka. Jak znaleźć podstawową „ontologię" odpowiednią dla ogólnego symbolicznego języka dyskursu? Cóż, to nie jest łatwe. Być może dlatego niewiele zrobiono w nich od prymitywnych początków, które Arystoteles stworzył ponad dwa tysiące lat temu. Ale naprawdę pomaga to, że dzisiaj wiemy tak dużo o tym, jak myśleć o świecie w sposób obliczeniowy (i nie zaszkodzi mieć „fundamentalną metafizykę" z naszego projektu Fizyka i ideę ruliady). Ale co to wszystko oznacza w kontekście ChatGPT? Dzięki szkoleniu ChatGPT skutecznie „poskładał" pewną (raczej imponującą) ilość gramatyki semantycznej. Ale sam jego sukces daje nam powód, by sądzić, że możliwe będzie zbudowanie czegoś bardziej kompletnego w formie języka obliczeniowego. I w przeciwieństwie do tego, co do tej pory odkryliśmy na temat wnętrza ChatGPT, możemy spodziewać się zaprojektowania języka obliczeniowego w taki sposób, aby był łatwo zrozumiały dla ludzi. Kiedy mówimy o gramatyce semantycznej, możemy narysować analogię do logiki sylogistycznej. Początkowo logika sylogistyczna była zasadniczo zbiorem reguł dotyczących zdań wyrażanych ludzkim językiem. Ale (tak, dwa tysiące lat później), kiedy rozwinęła się logika formalna, oryginalne podstawowe konstrukcje logiki sylogistycznej można było teraz wykorzystać do zbudowania ogromnych „formalnych wież", które obejmują na przykład działanie nowoczesnych obwodów cyfrowych. I tak, możemy się spodziewać, będzie z bardziej ogólną gramatyką semantyczną. Na początku może po prostu radzić sobie z prostymi wzorami, wyrażonymi, powiedzmy, jako tekst. Ale kiedy zbudujemy cały jego szkielet języka obliczeniowego, możemy spodziewać się, że będzie można go używać do wznoszenia wysokich wież „uogólnionej logiki semantycznej", które pozwolą nam pracować w precyzyjny i formalny sposób z wszelkiego rodzaju rzeczami, które nigdy wcześniej nie były dla nas dostępne, z wyjątkiem „poziomu parteru" poprzez ludzki język, z całą jego niejasnością. Możemy myśleć o konstrukcji języka obliczeniowego – i gramatyki semantycznej – jako reprezentującej rodzaj ostatecznej kompresji w przedstawianiu rzeczy. Ponieważ pozwala nam mówić o istocie tego, co możliwe, bez zajmowania się na przykład wszystkimi „zwrotami frazeologicznymi", które istnieją w zwykłym ludzkim języku. I możemy postrzegać wielką siłę ChatGPT jako coś trochę podobnego:

ponieważ on również w pewnym sensie „przewiercił się” do punktu, w którym może „połączyć język w semantycznie znaczący sposób” bez troski o różne możliwe zwroty wyrażenie. Co by się stało, gdybyśmy zastosowali ChatGPT do bazowego języka obliczeniowego? Język obliczeniowy może opisać, co jest możliwe. Ale to, co można jeszcze dodać, to poczucie „tego, co jest popularne” – oparte na przykład na czytaniu wszystkich tych treści w sieci. Ale potem — pod spodem — operowanie językiem obliczeniowym oznacza, że coś takiego jak ChatGPT ma natychmiastowy i fundamentalny dostęp do tego, co jest równoznaczne z ostatecznymi narzędziami do wykorzystania potencjalnie nieredukowalnych obliczeń. A to sprawia, że jest to system, który może nie tylko „wygenerować rozsądny tekst”, ale może oczekiwać, że wypracuje wszystko, co można wypracować, czy ten tekst rzeczywiście zawiera „poprawne” stwierdzenia o świecie – lub o czym ma mówić.

Więc… Co robi ChatGPT i dlaczego działa?

Podstawowa koncepcja ChatGPT jest na pewnym poziomie raczej prosta. Zacznij od ogromnej próbki tekstu stworzonego przez człowieka z sieci, książek itp. Następnie wytrenuj sieć neuronową, aby generowała tekst „taki jak ten”. A w szczególności spraw, aby mógł zaczynać od „podpowiedzi”, a następnie kontynuować tekst, który „jest taki, z jakim został przeszkolony”. Jak widzieliśmy, rzeczywista sieć neuronowa w ChatGPT składa się z bardzo prostych elementów — choć są ich miliardy. A podstawowa operacja sieci neuronowej jest również bardzo prosta, polegająca zasadniczo na przekazywaniu danych wejściowych pochodzących z dotychczas wygenerowanego tekstu „raz przez jego elementy” (bez żadnych pętli itp.) dla każdego nowego słowa (lub części słowa), które generuje. Ale niezwykłe — i nieoczekiwane — jest to, że ten proces może z powodzeniem tworzyć tekst, który z powodzeniem „podobny” jest do tego, co jest w sieci, w książkach itp. I nie tylko jest to spójny ludzki język, ale także „mówi rzeczy”, które „podążają za jego monit” wykorzystując treść, którą „przeczytał”. Nie zawsze mówi rzeczy, które „globalnie mają sens” (lub odpowiadają poprawnym obliczeniom) – ponieważ (bez, na przykład, dostępu do „supermocy obliczeniowych” Wolfram|Alpha) po prostu mówi rzeczy, które „brzmią dobrze” w oparciu o jak to „brzmiało” w materiałach szkoleniowych. Specyficzna inżynieria ChatGPT uczyniła go całkiem atrakcyjnym. Ale ostatecznie (przynajmniej dopóki nie będzie mógł korzystać z narzędzi zewnętrznych) ChatGPT „tylko” wyciąga „spójny wątek tekstu” z „statystyk konwencjonalnej wiedzy”, które zgromadził. Ale to niesamowite, jak bardzo ludzkie są wyniki. I jak już omówiłem, sugeruje to coś, co jest przynajmniej bardzo ważne z naukowego punktu widzenia: że ludzki język (i stojące za nim wzorce myślenia) są w jakiś sposób prostsze i bardziej „prawopodobne” w swojej strukturze, niż myśleliśmy. ChatGPT niejawnie go odkrył. Ale potencjalnie możemy to wyraźnie ujawnić za pomocą gramatyki semantycznej, języka obliczeniowego itp. To, co ChatGPT robi w zakresie generowania tekstu, jest bardzo imponujące — a wyniki są zwykle bardzo podobne do tego, co stworzylibyśmy my, ludzie. Czy to oznacza, że ChatGPT działa jak mózg? Jego podstawowa struktura sztucznej sieci neuronowej została ostatecznie wzorowana na idealizacji mózgu. I wydaje się całkiem prawdopodobne, że kiedy my, ludzie, tworzymy język, wiele aspektów tego, co się dzieje, jest dość podobnych. Jeśli chodzi o szkolenie (inaczej uczenie się), inny „sprzęt” mózgu i obecnych komputerów (a także być może niektóre nierozwinięte pomysły algorytmiczne) zmusza ChatGPT do zastosowania strategii, która prawdopodobnie jest raczej inna (i pod pewnymi względami znacznie mniej wydajniejszy) niż mózg. Jest też coś jeszcze: w przeciwieństwie nawet do typowych obliczeń algorytmicznych, ChatGPT nie ma wewnętrznie „pętli” ani „ponownych obliczeń na danych”. A to nieuchronnie ogranicza jego możliwości obliczeniowe – nawet w odniesieniu do obecnych komputerów, ale zdecydowanie w odniesieniu do mózgu. Nie jest jasne, jak to „naprawić” i nadal zachować zdolność uczenia systemu z rozsądną wydajnością. Ale zrobienie tego prawdopodobnie pozwoli przyszłemu ChatGPT robić jeszcze więcej „rzeczy podobnych do mózgu”. Oczywiście jest wiele rzeczy, których mózgi nie radzą sobie tak dobrze – szczególnie w przypadku nieredukowalnych obliczeń. A dla tych zarówno mózgów, jak i rzeczy takich jak ChatGPT, muszą szukać „narzędzi

zewnętrznych” - takich jak Wolfram Language. Ale na razie ekscytujące jest zobaczenie, co ChatGPT był już w stanie zrobić. Na pewnym poziomie jest to doskonały przykład fundamentalnego faktu naukowego, że duża liczba prostych elementów obliczeniowych może robić niezwykłe i nieoczekiwane rzeczy. Ale dostarcza też być może najlepszego impulsu, jaki mieliśmy od dwóch tysięcy lat, aby lepiej zrozumieć, jaki może być fundamentalny charakter i zasady tej centralnej cechy ludzkiej kondycji, jaką jest ludzki język i stojące za nim procesy myślowe.