AI - Wat zijn LLM's (zoals bijv. Ch × Wat doet ChatGPT ... en waarom × +

writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/?utm_source=www.the...

Gmail Maps YouTube Nieuws Vertalen

≡ STEPHEN WOLFRAM | Writings

RECENT | CATEGORIEËN | 🔍

Inhoud

Nu beschikbaar in paperback en op Kindle:

Wat doet ... en waarom werkt het?

Bestel nu

Zie ook:

"Wolfram|Alpha als de manier om computationele kennis-superkrachten naar ChatGPT te brengen"

»

Zoek geschriften

Recente geschr

Introdu
Noteb
in het I
8 juni 20

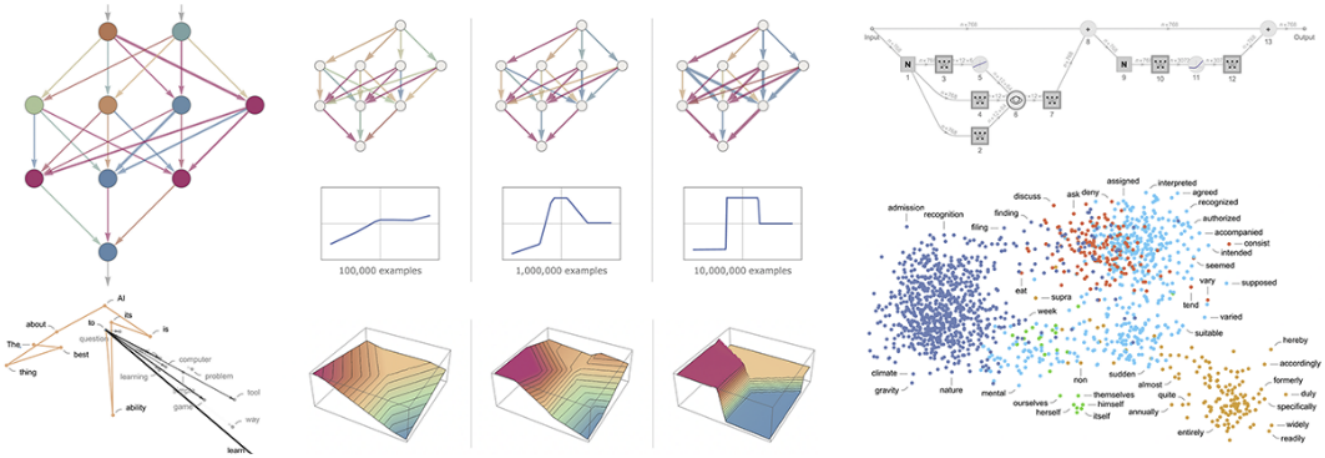
Promp
ontspa
Promp
7 juni 20

De nieu
functie
integre
23 mei 2

Een discussie over de geschiedenis van neurale netwerken »

Wat doet ChatGPT … en waarom werkt het?

14 februari 2023



Het is gewoon één woord per keer toevoegen

Dat ChatGPT automatisch iets kan genereren dat zelfs oppervlakkig leest als door mensen geschreven tekst is opmerkelijk en onverwacht. Maar hoe doet het dat? En waarom werkt het? Mijn doel hier is om een globaal overzicht te geven van wat er gaande is in ChatGPT - en vervolgens te onderzoeken waarom het zo goed kan presteren bij het produceren van wat we zouden kunnen beschouwen als zinvolle tekst. Ik zou om te beginnen moeten zeggen dat ik me ga concentreren op het grote geheel van wat er aan de hand is - en hoewel ik enkele technische details zal noemen, zal ik er niet diep op ingaan. (En de essentie van wat ik ga zeggen is net zo goed van

Directe introdu ChatGI 27 april

ChatGI superk 23 maar

Allemaal op da

populaire categ

- Kunstmatige intel
- Grote foto
- Bedrijven en Zake
- Computationele w
- Computationeel d
- Gegevenswetensc
- Onderwijs
- Toekomstperspec
- Historische perspe
- Taal en communic
- Leven en Tijden
- Levenswetenscha
- Wiskunde
- Wiskunde
- Nieuw soort wete
- Nieuwe technolog
- Persoonlijke analy
- Filosofie
- Natuurkunde

toepassing op andere huidige "grote taalmodellen" [LLM's] als op ChatGPT.)

Het eerste dat moet worden uitgelegd, is dat wat ChatGPT in wezen altijd probeert te doen, een "redelijke voortzetting" is van de tekst die het tot nu toe heeft gekregen, waarbij we met "redelijk" bedoelen "wat men zou verwachten dat iemand schrijft nadat hij heeft gezien wat mensen hebben geschreven op miljarden webpagina's, enz."

Dus laten we zeggen dat we de tekst hebben " *Het beste van AI is het vermogen om* ". Stel je voor dat je miljarden pagina's met door mensen geschreven tekst scant (bijvoorbeeld op internet en in gedigitaliseerde boeken) en alle exemplaren van deze tekst vindt - en dan ziet welk woord er in welk deel van de tijd komt. ChatGPT doet effectief zoiets, behalve dat (zoals ik zal uitleggen) het niet naar letterlijke tekst kijkt; het zoekt naar dingen die in zekere zin "overeenkomen in betekenis". Maar het eindresultaat is dat het een gerangschikte lijst met woorden oplevert die kunnen volgen, samen met "waarschijnlijkheden":

The best thing about AI is its ability to

learn	4.5%
predict	3.5%
make	3.2%
understand	3.1%
do	2.9%

En het opmerkelijke is dat wanneer ChatGPT zoiets doet als het schrijven van een essay, het in wezen gewoon steeds weer opnieuw vraagt "gezien de tekst tot dusver, wat zou het volgende woord moeten zijn?" - en elke keer een woord toevoegen. (Meer

- Ruliologie
- Software-ontwerp
- Wolfram|Alfa
- Wolfram|Een
- Wolfram-taal
- Ander

Geschriften per

2023 | 2022 | 2021
2016 | 2015 | 2014
2009 | 2008 | 2007

Explore the s
inspired by A

precies, zoals ik zal uitleggen, voegt het een "token" toe, wat slechts een deel van een woord kan zijn, en daarom kan het soms "nieuwe woorden verzinnen".)

Maar goed, bij elke stap krijgt het een lijst met woorden met waarschijnlijkheden. Maar welke moet het eigenlijk kiezen om toe te voegen aan het essay (of wat dan ook) dat het aan het schrijven is? Je zou kunnen denken dat dit het "hoogst gerangschikte" woord zou moeten zijn (dwz het woord waaraan de hoogste "waarschijnlijkheid" was toegekend). Maar hier begint een beetje voodoo naar binnen te sluipen. Want om de een of andere reden - waar we misschien op een dag een wetenschappelijk begrip van zullen hebben - als we altijd het hoogst gerangschikte woord kiezen, krijgen we meestal een heel "plat" essay, dat nooit "enige creativiteit" lijkt te tonen (en soms zelfs woord voor woord herhaalt). Maar als we soms (willekeurig) woorden met een lagere rangorde kiezen, krijgen we een 'interessanter' essay.

Het feit dat er hier willekeur is, betekent dat als we dezelfde prompt meerdere keren gebruiken, we waarschijnlijk elke keer verschillende essays krijgen. En, in overeenstemming met het idee van voodoo, is er een bepaalde zogenaamde 'temperatuur'-parameter die bepaalt hoe vaak woorden met een lagere rangorde worden gebruikt, en voor het genereren van essays blijkt dat een 'temperatuur' van 0,8 het beste lijkt. (Het is de moeite waard om te benadrukken dat er hier geen "theorie" wordt gebruikt; het is gewoon een kwestie van wat in de praktijk blijkt te werken. En het concept van "temperatuur" is er bijvoorbeeld omdat exponentiële verdelingen die bekend zijn uit de statistische fysica toevallig worden gebruikt , maar er is geen "fysieke" verbinding - althans voor zover we weten.)

Voordat we verder gaan, moet ik uitleggen dat ik voor expositiedoeleinden meestal

Science and backstory of was created

ORDER

niet het volledige systeem ga gebruiken dat in ChatGPT zit ; in plaats daarvan werk ik meestal met een eenvoudiger GPT-2-systeem , dat de leuke eigenschap heeft dat het klein genoeg is om op een standaard desktopcomputer te kunnen draaien. En dus kan ik voor vrijwel alles wat ik laat zien expliciete Wolfram- taalcode opnemen die je onmiddellijk op je computer kunt uitvoeren. (Klik hier op een afbeelding om de code erachter te kopiëren.)

Hier ziet u bijvoorbeeld hoe u de tabel met kansen hierboven krijgt. Eerst moeten we het onderliggende neurale netwerk van het "taalmodel" ophalen :

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

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

Later zullen we in dit neurale net kijken en praten over hoe het werkt. Maar voor nu kunnen we dit "netmodel" tot nu toe als een zwarte doos toepassen op onze tekst, en vragen naar de top 5 woorden naar waarschijnlijkheid die volgens het model zouden moeten volgen:

```
In[ ]:= model["The best thing about AI is its ability to", {"TopProbabilities", 5}]

Out[ ]= {do → 0.0288508, understand → 0.0307805,
           make → 0.0319072, predict → 0.0349748, learn → 0.0445305}
```

Dit neemt dat resultaat en maakt er een expliciet opgemaakte " dataset " van:

```
In[ ]:= Dataset[ReverseSort[Association[%]],
          ItemDisplayFunction → (PercentForm[#, 2] &)]
```

Out[ ]=

learn	4.5%
predict	3.5%
make	3.2%
understand	3.1%
do	2.9%

Dit is wat er gebeurt als iemand herhaaldelijk "het model toepast" - bij elke stap het woord toevoegen met de hoogste waarschijnlijkheid (in deze code gespecificeerd als de "beslissing" van het model):

```
In[ ]:= NestList[StringJoin[#, model[#, "Decision"]] &,
          "The best thing about AI is its ability to", 7]
```

```
Out[ ]= {The best thing about AI is its ability to,
         The best thing about AI is its ability to learn,
         The best thing about AI is its ability to learn from,
         The best thing about AI is its ability to learn from experience,
         The best thing about AI is its ability to learn from experience.,
         The best thing about AI is its ability to learn from experience. It,
         The best thing about AI is its ability to learn from experience. It's,
         The best thing about AI is its ability to learn from experience. It's not}
```

Wat gebeurt er als men langer doorgaat? In dit ("zero temperature") geval wordt wat eruit komt al snel nogal verward en repetitief:

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

Maar wat als in plaats van altijd het "top"-woord te kiezen, men soms willekeurig "niet-top"-woorden kiest (waarbij de "willekeurigheid" overeenkomt met "temperatuur" 0,8)? Wederom kan men tekst opbouwen:

{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,}

En elke keer dat iemand dit doet, worden er verschillende willekeurige keuzes gemaakt en zal de tekst anders zijn, zoals in deze 5 voorbeelden:

The best thing about AI is its ability to learn. I've always liked the

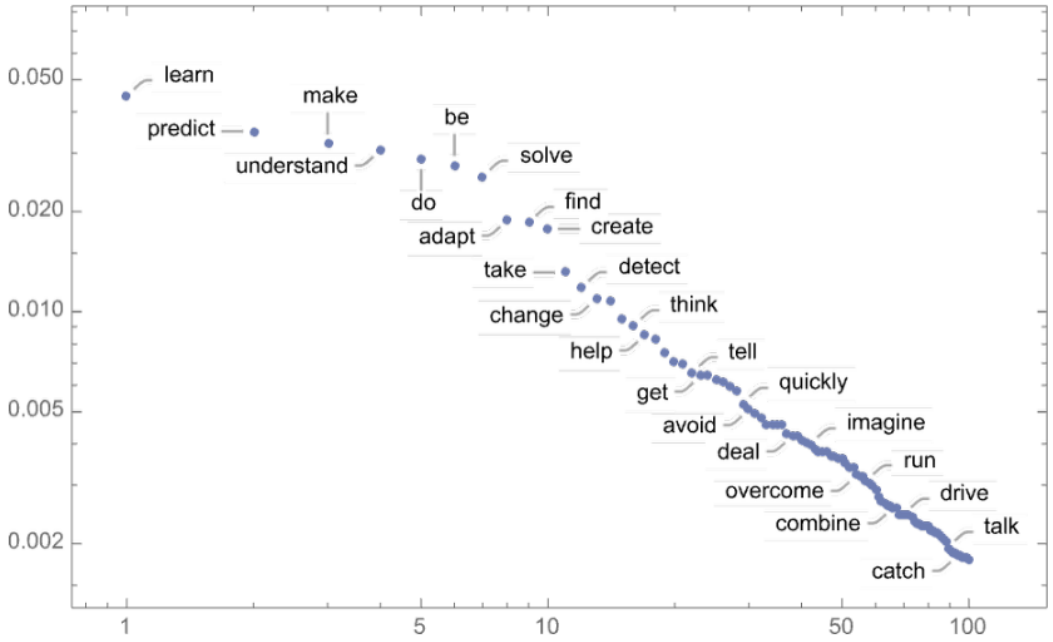
The best thing about AI is its ability to really come into your world and just

The best thing about AI is its ability to examine human behavior and the way it

The best thing about AI is its ability to do a great job of teaching us

The best thing about AI is its ability to create real tasks, but you can

Het is de moeite waard erop te wijzen dat er zelfs bij de eerste stap veel mogelijke "volgende woorden" zijn om uit te kiezen (bij een temperatuur van 0,8), hoewel hun waarschijnlijkheid vrij snel afneemt (en ja, de rechte lijn op deze log-log plot komt overeen met een n^{-1} "machtswet" verval dat zeer kenmerkend is voor de algemene taalstatistieken):



Dus wat gebeurt er als men langer doorgaat? Hier is een willekeurig voorbeeld. Het is beter dan het geval met het hoogste woord (temperatuur nul), maar op zijn best een beetje raar:

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

Dit is gedaan met het eenvoudigste GPT-2 model (uit 2019). Met de nieuwere en grotere GPT-3-modellen zijn de resultaten beter. Hier is de tekst met het hoogste woord (temperatuur nul) geproduceerd met dezelfde "prompt", maar met het grootste GPT-3-model:

> 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.

En hier is een willekeurig voorbeeld bij "temperatuur 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.

Waar komen de kansen vandaan?

OK, dus ChatGPT kiest altijd het volgende woord op basis van kansen. Maar waar komen die kansen vandaan? Laten we beginnen met een eenvoudiger probleem. Laten we eens kijken naar het genereren van Engelse tekst letter voor letter (in plaats

van woord). Hoe kunnen we berekenen wat de waarschijnlijkheid voor elke letter zou moeten zijn?

Een heel minimaal ding dat we zouden kunnen doen, is gewoon een voorbeeld van Engelse tekst nemen en berekenen hoe vaak verschillende letters erin voorkomen. Dus dit telt bijvoorbeeld de letters in het Wikipedia-artikel over "katten":

```
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
```

En dit doet hetzelfde voor "honden":

```
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,
```

De resultaten zijn vergelijkbaar, maar niet hetzelfde ("o" komt ongetwijfeld vaker voor in het artikel "honden", omdat het tenslotte voorkomt in het woord "hond" zelf). Maar als we een voldoende grote steekproef van Engelse tekst nemen, kunnen we uiteindelijk op zijn minst redelijk consistente resultaten verwachten:

```
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%}
```

Hier is een voorbeeld van wat we krijgen als we gewoon een reeks letters genereren met deze kansen:

```
rronoitadatcaeaesaotdoysaroiyiinnbantoioestlhddeocneooewceseciselnodrtrdgriscsatsepesdcnio
    uhoetsedeyhedslernevstothindtbmnaohngotannbthrdthtonsipieldn
```

We kunnen dit opsplitsen in "woorden" door spaties toe te voegen alsof het letters zijn met een bepaalde waarschijnlijkheid:

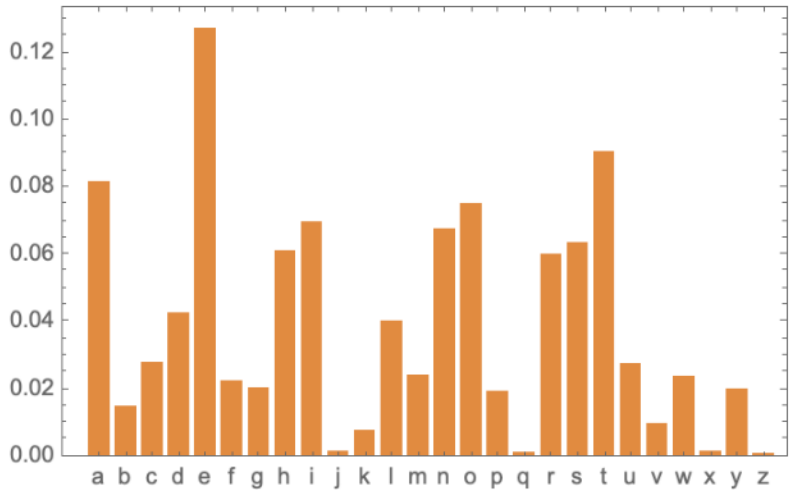
```
sd n oeiaim satnwhoo eer rtr ofiianordrenapwokom del oaas ill e h f
    rellptohltvoettseodtrncilntehtotrkthrslo hdaol n sriaefr hthehtn ld gpod a  h y oi
```

We kunnen "woorden" iets beter maken door de verdeling van "woordlengtes" te forceren om overeen te komen met wat het in het Engels is:

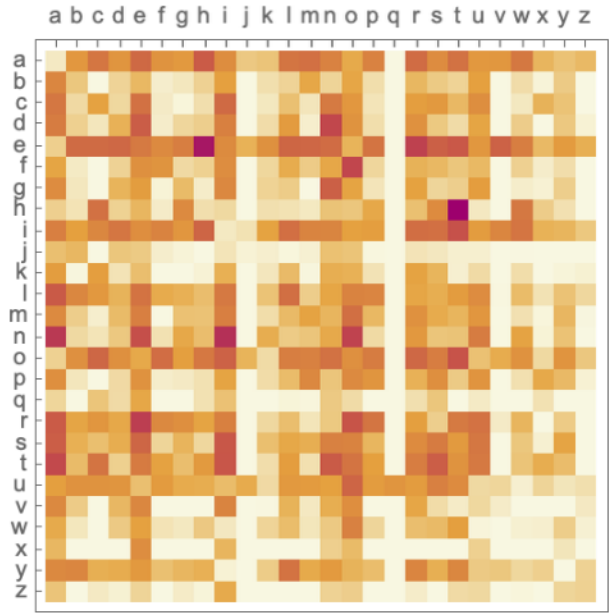
```
ni hilwhuei kjtn isjd erogofnr n rwhwfao rcuw lis fahte uss cpnc
    nlu oe nusaetat llfo oeme rrhrtn xdses ohm oa tne ebedcon oarvthv ist
```

We hebben hier toevallig geen "echte woorden" gekregen, maar de resultaten zien er iets beter uit. Om verder te gaan, moeten we echter meer doen dan alleen willekeurig elke letter afzonderlijk kiezen. En we weten bijvoorbeeld dat als we een "q" hebben, de volgende letter eigenlijk een "u" moet zijn.

Hier is een plot van de kansen voor letters op zichzelf:



En hier is een grafiek die de kansen laat zien van letterparen ("2-grams") in typisch Engelse tekst. De mogelijke eerste letters staan verspreid over de pagina, de tweede letters onderaan de pagina:



En we zien hier bijvoorbeeld dat de "q"-kolom leeg is (nul waarschijnlijkheid) behalve op de "u"-rij. OK, dus in plaats van onze "woorden" letter voor letter te genereren, laten we ze nu genereren door naar twee letters tegelijk te kijken, gebruikmakend van deze "2-gram" kansen. Hier is een voorbeeld van het resultaat, dat toevallig een paar "werkelijke woorden" bevat:

```
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
```

Met voldoende Engelse tekst kunnen we redelijk goede schattingen krijgen, niet alleen voor de waarschijnlijkheid van enkele letters of letterparen (2 gram), maar ook voor langere reeksen letters. En als we "willekeurige woorden" genereren met steeds langere *n* -gram kansen, zien we dat ze steeds "realistischer" worden:

0	on gxeeetowmt tsifhy ah aufnsoc ior oia itlt bnc tu ih uls
1	ri io os ot timumumoi gymyestit ate bshe abol viowr wotybeat mecho
2	wore hi usinallistin hia ale warou pothe of premetra bect upo pr
3	qual musin was witherins wil por vie surgedygua was suchinguary outheydays theresist
4	stud made yello adenced through theirs from cent intous wherefo proteined screa
5	special average vocab consumer market prepara injury trade consa usually speci utility

Maar laten we nu aannemen - min of meer zoals ChatGPT doet - dat we te maken hebben met hele woorden, niet met letters. Er zijn ongeveer 40.000 redelijk veelgebruikte woorden in het Engels . En door naar een groot corpus Engelse tekst te kijken (zeg maar een paar miljoen boeken, met in totaal een paar honderd miljard

woorden), kunnen we een schatting krijgen van hoe vaak elk woord voorkomt . En hiermee kunnen we beginnen met het genereren van "zinnen", waarin elk woord onafhankelijk willekeurig wordt gekozen, met dezelfde waarschijnlijkheid dat het in het corpus voorkomt. Hier is een voorbeeld van wat we krijgen:

```
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
```

Geen wonder dat dit onzin is. Dus hoe kunnen we het beter doen? Net als bij letters kunnen we niet alleen rekening houden met waarschijnlijkheden voor afzonderlijke woorden, maar ook met kansen voor paren of langere n -grammen woorden. Als we dit voor paren doen, zijn hier 5 voorbeelden van wat we krijgen, in alle gevallen beginnend met het woord "kat":

```
cat through shipping variety is made the aid emergency can the
cat for the book flip was generally decided to design of
cat at safety to contain the vicinity coupled between electric public
cat throughout in a confirmation procedure and two were difficult music
cat on the theory an already from a representation before a
```

Het wordt iets "verstandiger". *En we zouden ons kunnen voorstellen dat als we voldoende lange n* -grammen zouden kunnen gebruiken, we in feite "een ChatGPT" zouden krijgen - in die zin dat we iets zouden krijgen dat reeksen woorden van essaylengte zou genereren met de "juiste algehele essay" waarschijnlijkheden". Maar hier is het probleem: er is gewoon niet eens in de buurt genoeg Engelse tekst die ooit

is geschreven om die waarschijnlijkheden te kunnen afleiden.

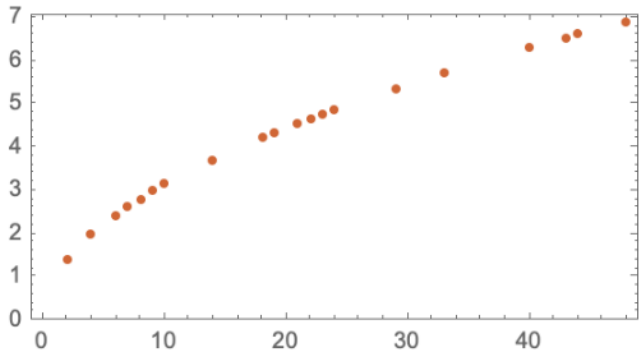
In een crawl van het web kunnen er een paar honderd miljard woorden zijn; in boeken die gedigitaliseerd zijn, staan misschien nog eens honderd miljard woorden. Maar met 40.000 gewone woorden is zelfs het aantal mogelijke 2 gram al 1,6 miljard – en het aantal mogelijke 3 gram is 60 biljoen. We kunnen dus op geen enkele manier de kansen schatten, zelfs niet voor al deze, op basis van beschikbare tekst. En tegen de tijd dat we bij "essayfragmenten" van 20 woorden komen, is het aantal mogelijkheden groter dan het aantal deeltjes in het universum, dus in zekere zin kunnen ze nooit allemaal worden opgeschreven.

Dus, wat kunnen we doen? Het grote idee is om een model te maken waarmee we de waarschijnlijkheid kunnen inschatten waarmee reeksen zouden moeten voorkomen - ook al hebben we die reeksen nooit expliciet gezien in het tekstcorpus dat we hebben bekeken. En de kern van ChatGPT is precies een zogenaamd "groot taalmodel" (LLM) dat is gebouwd om die kansen goed in te schatten.

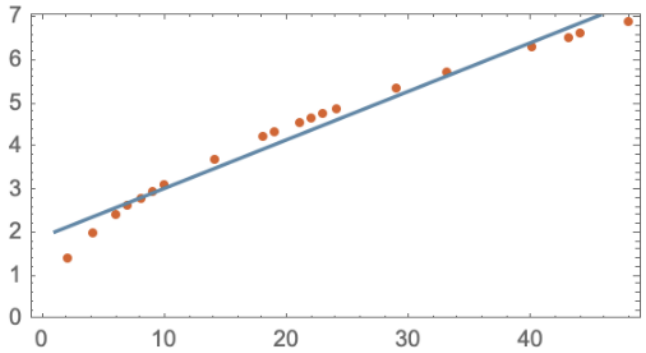
Wat is een model?

Stel dat u wilt weten (zoals Galileo eind 1500 deed) hoe lang het duurt voordat een kanonskogel die van elke verdieping van de Toren van Pisa valt, de grond raakt. Nou, je zou het gewoon in elk geval kunnen meten en een tabel maken van de resultaten. Of je zou kunnen doen wat de essentie is van theoretische wetenschap: een model maken dat een soort procedure geeft om het antwoord te berekenen in plaats van alleen maar elke casus te meten en te onthouden.

Laten we ons voorstellen dat we (enigszins geïdealiseerde) gegevens hebben over hoe lang het duurt voordat de kanonskogel van verschillende verdiepingen valt:



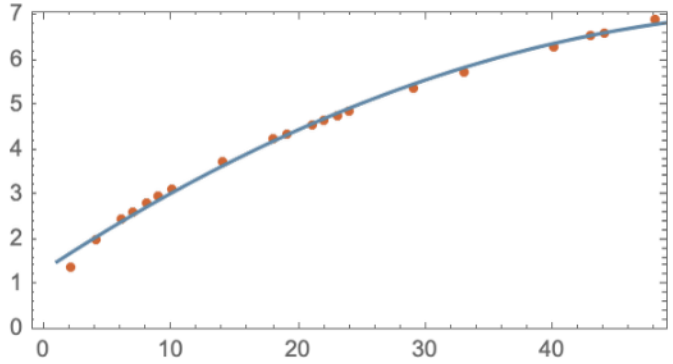
Hoe komen we erachter hoe lang het duurt om van een vloer te vallen waar we geen expliciete gegevens over hebben? In dit specifieke geval kunnen we bekende natuurwetten gebruiken om het uit te werken. Maar stel dat alles wat we hebben de gegevens zijn en dat we niet weten welke onderliggende wetten daarop van toepassing zijn. Dan zouden we een wiskundige gok kunnen doen, zo zouden we misschien een rechte lijn als model moeten gebruiken:



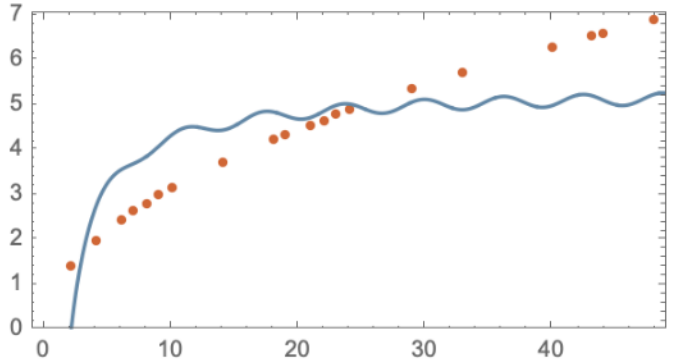
We kunnen verschillende rechte lijnen kiezen. Maar dit is degene die gemiddeld het dichtst in de buurt komt van de gegevens die we krijgen. En aan de hand van deze

rechte lijn kunnen we de valtijd voor elke verdieping schatten.

Hoe wisten we dat we hier een rechte lijn moesten gebruiken? Op een bepaald niveau deden we dat niet. Het is gewoon iets dat wiskundig eenvoudig is, en we zijn eraan gewend dat veel gegevens die we meten goed blijken te passen bij wiskundig eenvoudige dingen. We zouden iets wiskundig gecompliceerder kunnen proberen, bijvoorbeeld $a + b\,x + c\,x^2$, en in dit geval doen we het beter:



Het kan echter behoorlijk misgaan. Dit is het beste wat we kunnen doen met $a + b / x + c \sin(x)$:



Het is de moeite waard om te begrijpen dat er nooit een "modelloos model" is. Elk

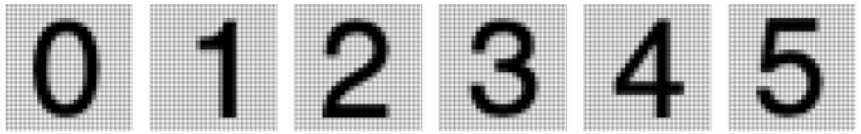
model dat u gebruikt, heeft een bepaalde onderliggende structuur - vervolgens een bepaalde set "knoppen waaraan u kunt draaien" (dwz parameters die u kunt instellen) om bij uw gegevens te passen. En in het geval van ChatGPT worden veel van dergelijke "knoppen" gebruikt - eigenlijk 175 miljard.

Maar het opmerkelijke is dat de onderliggende structuur van ChatGPT - met "slechts" zoveel parameters - voldoende is om een model te maken dat de waarschijnlijkheid van het volgende woord "goed genoeg" berekent om ons redelijke stukken tekst van essaylengte te geven.

Modellen voor mensachtige taken

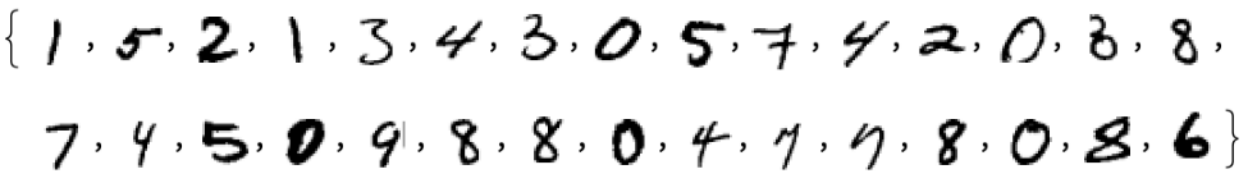
Het voorbeeld dat we hierboven gaven, betreft het maken van een model voor numerieke gegevens dat in wezen afkomstig is uit eenvoudige natuurkunde - waar we al eeuwenlang weten dat 'eenvoudige wiskunde van toepassing is'. Maar voor ChatGPT moeten we een model maken van tekst in menselijke taal, zoals geproduceerd door een menselijk brein. En voor zoiets hebben we (althans nog) niet zoiets als 'eenvoudige wiskunde'. Dus hoe zou een model ervan eruit kunnen zien?

Voordat we het over taal hebben, laten we het hebben over een andere menselijke taak: het herkennen van beelden. En laten we als eenvoudig voorbeeld hiervan afbeeldingen van cijfers bekijken (en ja, dit is een klassiek voorbeeld van machine learning):



Een ding dat we zouden kunnen doen, is een aantal voorbeeldafbeeldingen voor elk cijfer krijgen:

Om erachter te komen of een afbeelding die we als invoer krijgen overeenkomt met een bepaald cijfer, kunnen we een expliciete pixel-voor-pixel vergelijking maken met de voorbeelden die we hebben. Maar als mensen lijken we zeker iets beters te doen - omdat we nog steeds cijfers kunnen herkennen, zelfs als ze bijvoorbeeld met de hand zijn geschreven, en allerlei wijzigingen en vervormingen hebben:



Toen we een model maakten voor onze numerieke gegevens hierboven, waren we in staat om een numerieke waarde x te nemen die we kregen, en gewoon $a + bx$ berekenen voor bepaalde a en b . Dus als we de grijswaarde van elke pixel hier behandelen als een variabele x_i , is er dan een functie van al die variabelen die ons, na evaluatie, vertelt van welk cijfer de afbeelding is? Het blijkt dat het mogelijk is om zo'n functie te construeren. Het is niet verrassend dat het niet bijzonder eenvoudig is. En een typisch voorbeeld zou misschien een half miljoen wiskundige bewerkingen kunnen omvatten.

Maar het eindresultaat is dat als we de verzameling pixelwaarden voor een afbeelding in deze functie invoeren, het nummer tevoorschijn komt dat aangeeft van welk cijfer we een afbeelding hebben. Later zullen we het hebben over hoe zo'n functie kan worden geconstrueerd en over het idee van neurale netwerken. Maar laten we de functie nu beschouwen als een zwarte doos, waar we afbeeldingen invoeren van bijvoorbeeld handgeschreven cijfers (als arrays van pixelwaarden) en we halen de nummers eruit waarmee deze overeenkomen:

```
In[ ]:= NetModel["..."][{ 7, 0, 9, 7, 8, 2, 4, 1, 1, 1 }]

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

Maar wat is hier echt aan de hand? Laten we zeggen dat we een cijfer geleidelijk vervagen. Onze functie 'herkent' het nog even, hier als een '2'. Maar al snel "verliest het het" en begint het het "verkeerde" resultaat te geven:

```
In[ ]:= NetModel["..."][{ 2, 2, 2, 2, 2, 2, 2, 2, 2 }]

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

Maar waarom zeggen we dat het het "verkeerde" resultaat is? In dit geval weten we dat we alle afbeeldingen hebben gekregen door een "2" te vervagen. Maar als het ons doel is om een model te produceren van wat mensen kunnen doen bij het herkennen van beelden, is de echte vraag die gesteld moet worden wat een mens zou hebben gedaan als hij een van die wazige beelden zou zien, zonder te weten waar het vandaan kwam.

En we hebben een 'goed model' als de resultaten die we uit onze functie halen doorgaans overeenkomen met wat een mens zou zeggen. En het niet-triviale wetenschappelijke feit is dat we voor een beeldherkenningstaak als deze nu in wezen weten hoe we functies moeten construeren die dit doen.

Kunnen we 'wiskundig bewijzen' dat ze werken? Welnee. Want om dat te doen zouden we een wiskundige theorie moeten hebben van wat wij mensen doen. Neem de afbeelding "2" en verander een paar pixels. We kunnen ons voorstellen dat we met slechts een paar pixels "niet op hun plaats" de afbeelding nog steeds als een "2" zouden moeten beschouwen. Maar hoe ver moet dat gaan? Het is een kwestie van menselijke visuele perceptie . En ja, het antwoord zou ongetwijfeld anders zijn voor bijen of octopussen - en mogelijk heel anders voor vermeende buitenaardse wezens .

Neurale netten

OK, dus hoe werken onze typische modellen voor taken als beeldherkenning eigenlijk? De meest populaire en succesvolle huidige aanpak maakt gebruik van neurale netwerken . Uitgevonden - in een vorm die opmerkelijk dicht bij hun huidige gebruik ligt - in de jaren 1940 , kunnen neurale netwerken worden gezien als eenvoudige idealiseringen van hoe hersenen lijken te werken .

In de menselijke hersenen bevinden zich ongeveer 100 miljard neuronen (zenuwcellen), elk in staat om een elektrische puls te produceren tot misschien wel duizend keer per seconde. De neuronen zijn verbonden in een gecompliceerd net, waarbij elk neuron boomachtige takken heeft waardoor het elektrische signalen kan

doorgeven aan misschien wel duizenden andere neuronen. En of een bepaald neuron op een bepaald moment een elektrische puls produceert, hangt bij grove benadering af van de pulsen die het ontvangt van andere neuronen - met verschillende verbindingen die bijdragen met verschillende "gewichten".

Wanneer we "een afbeelding zien", gebeurt er dat wanneer fotonen van het licht van de afbeelding vallen op ("fotoreceptor") cellen aan de achterkant van onze ogen, ze elektrische signalen produceren in zenuwcellen. Deze zenuwcellen zijn verbonden met andere zenuwcellen en uiteindelijk gaan de signalen door een hele reeks lagen neuronen. En het is tijdens dit proces dat we het beeld 'herkennen' en uiteindelijk 'de gedachte vormen' dat we 'een 2 zien' (en misschien uiteindelijk zoiets doen als het hardop uitspreken van het woord 'twee').

De "black-box"-functie uit de vorige sectie is een "gewiskundige" versie van zo'n neuraal net. Het heeft toevallig 11 lagen (hoewel slechts 4 "kernlagen"):

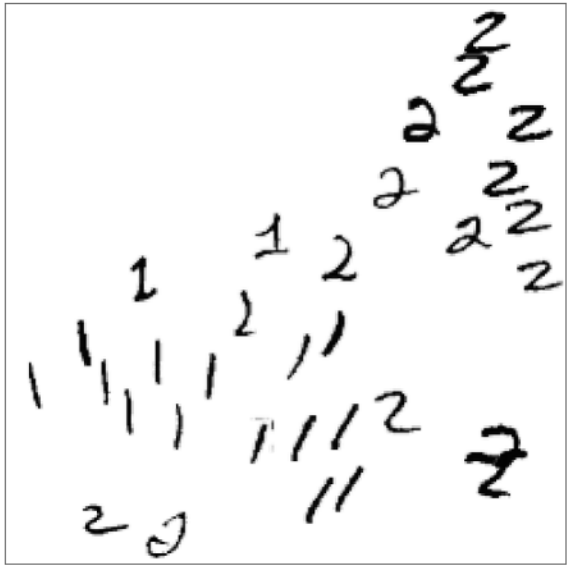
NetChain

		image
	Input	array (size: 1×28×28)
1	ConvolutionLayer	array (size: 20×24×24)
2	Ramp	array (size: 20×24×24)
3	PoolingLayer	array (size: 20×12×12)
4	ConvolutionLayer	array (size: 50×8×8)
5	Ramp	array (size: 50×8×8)
6	PoolingLayer	array (size: 50×4×4)
7	FlattenLayer	vector (size: 800)
8	LinearLayer	vector (size: 500)
9	Ramp	vector (size: 500)
10	LinearLayer	vector (size: 10)
11	SoftmaxLayer	vector (size: 10)
	Output	class

Er is niets bijzonders "theoretisch afgeleid" aan dit neurale net; het is gewoon iets dat

- in 1998 - werd gebouwd als een stuk techniek en bleek te werken. (Dat verschilt natuurlijk niet veel van hoe we onze hersenen zouden omschrijven als zijnde geproduceerd door het proces van biologische evolutie.)

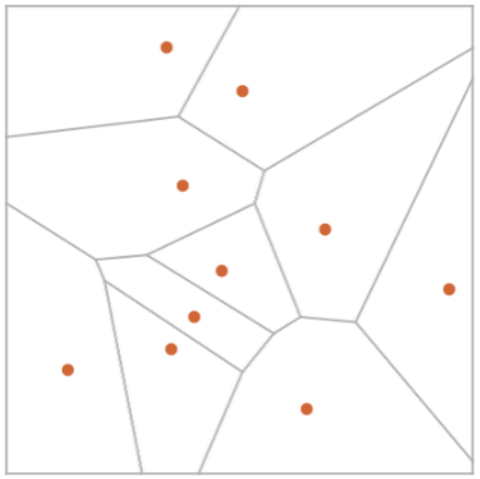
Oké, maar hoe "herkent" zo'n neuraal netwerk dingen? De sleutel is het begrip 'aantrekkers' . Stel je voor dat we handgeschreven afbeeldingen hebben van 1-en en 2-en:



Op de een of andere manier willen we dat alle 1-en "naar één plaats worden aangetrokken" en alle 2-en "naar een andere plaats worden aangetrokken". Of anders gezegd, als een afbeelding op de een of andere manier " dichter bij een 1 is " dan bij een 2, willen we dat het op de "1-plaats" terechtkomt en vice versa.

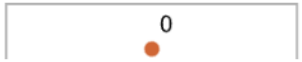
Als een eenvoudige analogie, laten we zeggen dat we bepaalde posities in het vlak hebben, aangegeven door stippen (in de praktijk kunnen dit de posities van

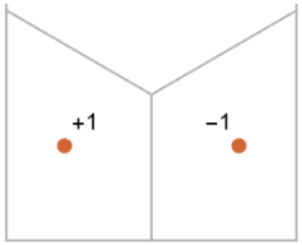
coffeeshops zijn). Dan kunnen we ons voorstellen dat we vanaf elk punt in het vliegtuig altijd bij de dichtstbijzijnde stip willen eindigen (dwz we gaan altijd naar de dichtstbijzijnde coffeeshop). We kunnen dit weergeven door het vlak te verdelen in regio's ("attractorbassins") gescheiden door geïdealiseerde "waterscheidingen":



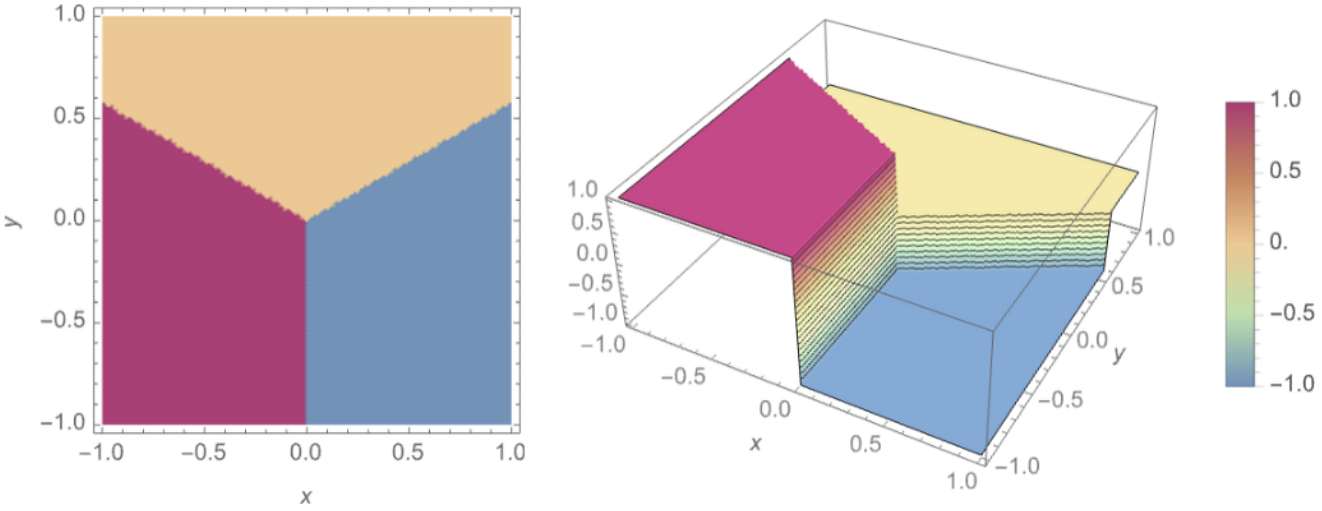
We kunnen dit zien als het implementeren van een soort 'herkenningstaak' waarbij we niet zoiets doen als identificeren op welk cijfer een bepaald beeld 'het meest lijkt', maar we zien gewoon heel direct op welke stip een gegeven punt het dichtst bij is. (De "Voronoi-diagram"-opstelling die we hier laten zien, scheidt punten in de 2D Euclidische ruimte; de cijferherkenningstaak kan worden gezien als iets dat erg op elkaar lijkt, maar in een 784-dimensionale ruimte gevormd uit de grijsniveaus van alle pixels in elke afbeelding.)

Dus hoe zorgen we ervoor dat een neuraal netwerk "een herkenningstaak doet"? Laten we dit heel eenvoudige geval eens bekijken:

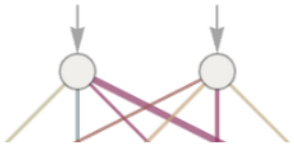


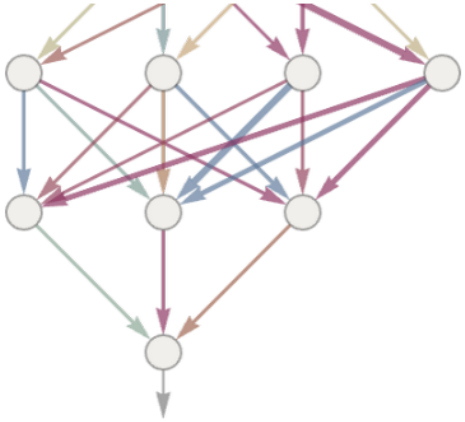


Ons doel is om een "invoer" te nemen die overeenkomt met een positie $\{x, y\}$ - en deze vervolgens te "herkennen" als een van de drie punten waar het het dichtst bij is. Of, met andere woorden, we willen dat het neurale net een functie berekent van $\{x, y\}$ zoals:

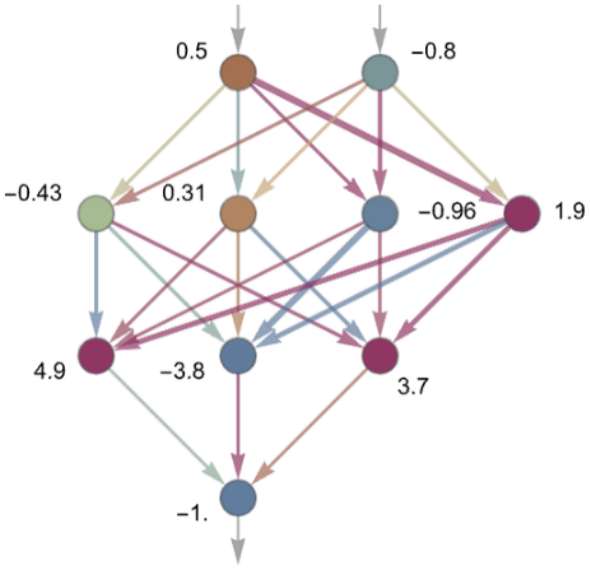


Dus hoe doen we dit met een neuraal net? Uiteindelijk is een neuraal net een verbonden verzameling van geïdealiseerde "neuronen" - meestal gerangschikt in lagen - met als eenvoudig voorbeeld:





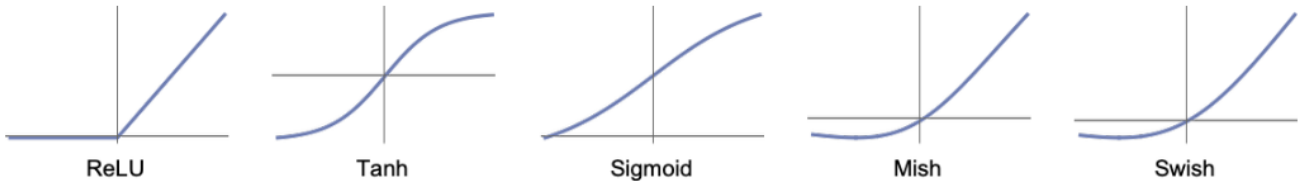
Elk "neuron" is effectief ingesteld om een eenvoudige numerieke functie te evalueren. En om het netwerk te "gebruiken", voeren we eenvoudig getallen in (zoals onze coördinaten x en y) bovenaan, laten vervolgens neuronen op elke laag "hun functies evalueren" en voeren de resultaten door het netwerk - uiteindelijk producerend het eindresultaat aan de onderkant:



In de traditionele (biologisch geïnspireerde) opstelling heeft elk neuron in feite een

bepaalde set "inkomende verbindingen" van de neuronen op de vorige laag, waarbij aan elke verbinding een bepaald "gewicht" wordt toegekend (dat een positief of negatief getal kan zijn). De waarde van een bepaald neuron wordt bepaald door de waarden van "vorige neuronen" te vermenigvuldigen met hun overeenkomstige gewichten, deze vervolgens op te tellen en een constante toe te voegen - en ten slotte een "drempelfunctie" (of "activerings") toe te passen. In wiskundige termen, als een neuron invoer heeft $x = \{ x_1 , x_2 ...\}$ dan berekenen we $f [w . x + b]$, waarbij de gewichten $w$en constante b worden over het algemeen anders gekozen voor elk neuron in het netwerk; de functie f is meestal hetzelfde.

Berekenen $w . x + b$ is gewoon een kwestie van matrixvermenigvuldiging en optelling. De "activeringsfunctie" f introduceert niet-lineariteit (en leidt uiteindelijk tot niet-triviaal gedrag). Verschillende activeringsfuncties worden vaak gebruikt; hier gebruiken we gewoon Ramp (of ReLU):



Voor elke taak die we willen dat het neurale net uitvoert (of, equivalent, voor elke algemene functie die we willen dat het evalueert), hebben we verschillende keuzes van gewichten. (En - zoals we later zullen bespreken - deze gewichten worden normaal gesproken bepaald door het neurale net te "trainen" met behulp van machine learning uit voorbeelden van de output die we willen.)

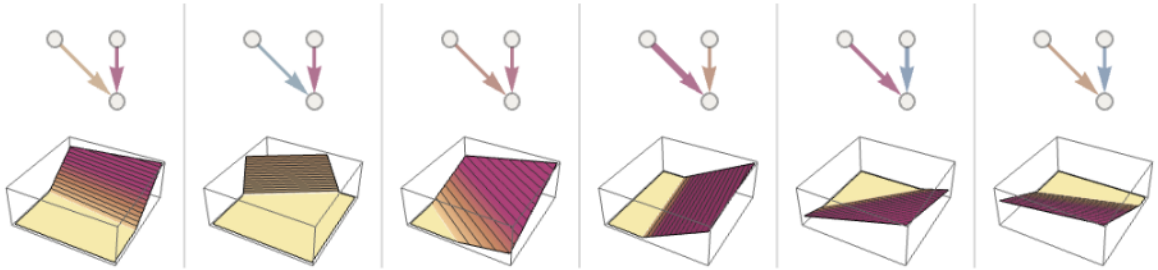
Uiteindelijk komt elk neuraal net gewoon overeen met een algemene wiskundige

functie, hoewel het rommelig kan zijn om uit te schrijven. Voor het bovenstaande voorbeeld zou het zijn:

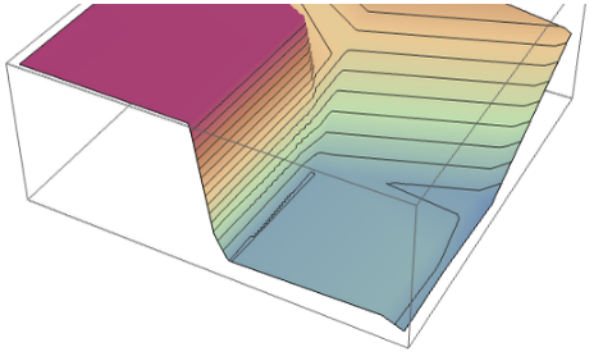
$$\begin{aligned}
&w_{511}f(w_{311}f(b_{11}+xw_{111}+yw_{112})+w_{312}f(b_{12}+xw_{121}+yw_{122})+\\
&\quad w_{313}f(b_{13}+xw_{131}+yw_{132})+w_{314}f(b_{14}+xw_{141}+yw_{142})+b_{31})+\\
&w_{512}f(w_{321}f(b_{11}+xw_{111}+yw_{112})+w_{322}f(b_{12}+xw_{121}+yw_{122})+\\
&\quad w_{323}f(b_{13}+xw_{131}+yw_{132})+w_{324}f(b_{14}+xw_{141}+yw_{142})+b_{32})+\\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

Het neurale netwerk van ChatGPT komt ook gewoon overeen met een wiskundige functie als deze, maar in feite met miljarden termen.

Maar laten we teruggaan naar individuele neuronen. Hier zijn enkele voorbeelden van de functies die een neuron met twee ingangen (die de coördinaten x en y vertegenwoordigen) kan berekenen met verschillende keuzes van gewichten en constanten (en `Ramp` als activeringsfunctie):

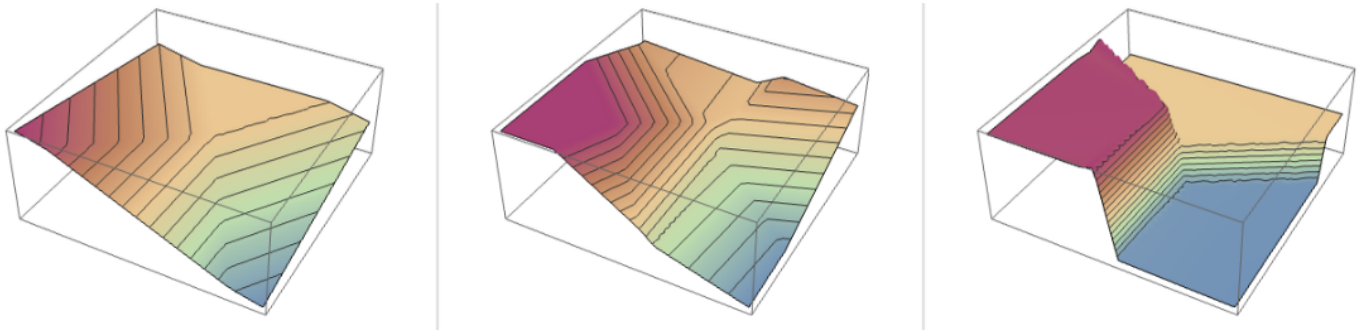


Maar hoe zit het met het grotere netwerk van bovenaf? Nou, dit is wat het berekent:



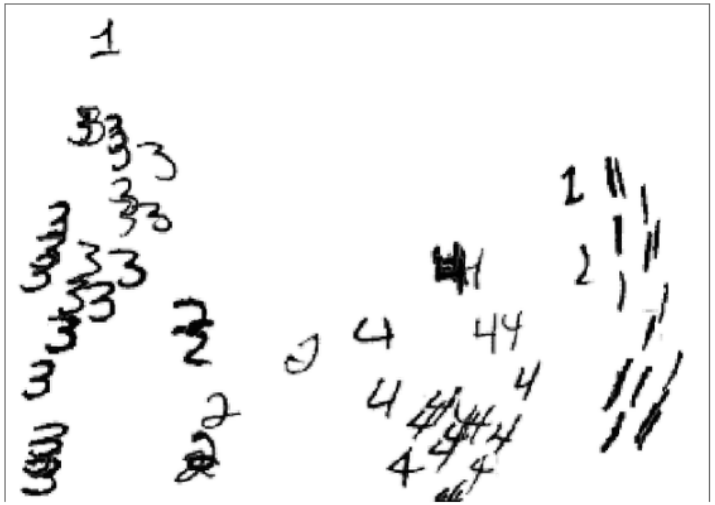
Het is niet helemaal "juist", maar het komt dicht in de buurt van de functie "dichtstbijzijnde punt" die we hierboven lieten zien.

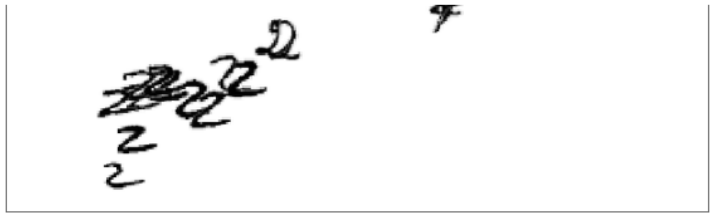
Laten we eens kijken wat er gebeurt met andere neurale netwerken. Zoals we later zullen uitleggen, gebruiken we in elk geval machine learning om de beste gewichtskeuze te vinden. Dan laten we hier zien wat het neurale net met die gewichten berekent:



Grotere netwerken zijn over het algemeen beter in het benaderen van de functie waarnaar we streven. En in het "midden van elk attractorbassin" krijgen we meestal precies het antwoord dat we willen. Maar aan de grenzen - waar het neurale net "moeite heeft om een besluit te nemen" - kan het rommeliger zijn.

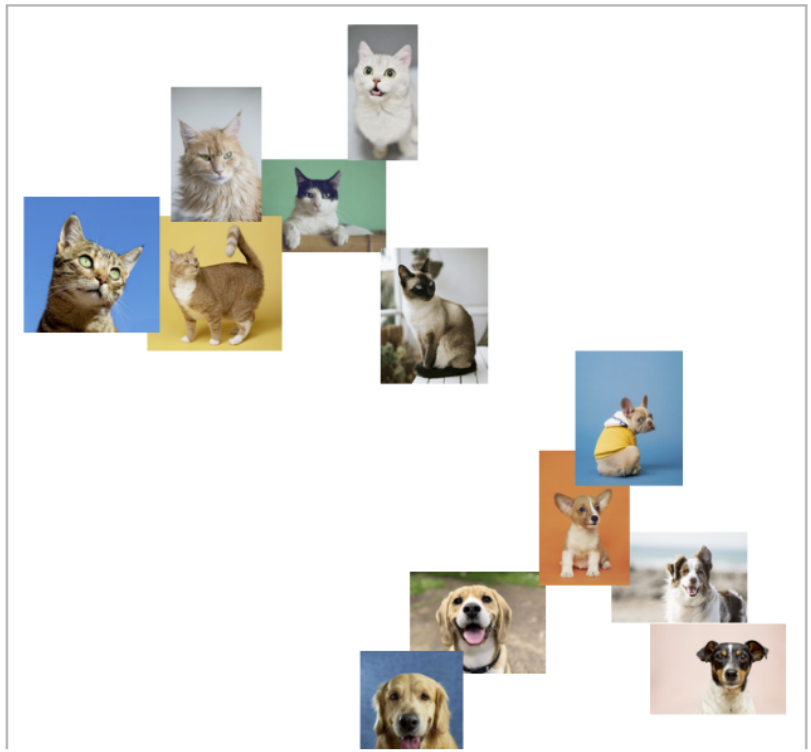
Met deze eenvoudige 'herkenningstaak' in wiskundige stijl is het duidelijk wat het 'juiste antwoord' is. Maar bij het herkennen van handgeschreven cijfers is het niet zo duidelijk. Wat als iemand een "2" zo slecht schreef dat het op een "7" leek, enz.? Toch kunnen we ons afvragen hoe een neuraal net cijfers onderscheidt - en dit geeft een indicatie:





Kunnen we "wiskundig" zeggen hoe het netwerk zijn onderscheidingen maakt? Niet echt. Het is gewoon "doen wat het neurale net doet". Maar het blijkt dat dat normaal gesproken vrij goed lijkt overeen te komen met het onderscheid dat wij mensen maken.

Laten we een uitgebreider voorbeeld nemen. Laten we zeggen dat we afbeeldingen hebben van katten en honden. En we hebben een neuraal net dat is getraind om ze te onderscheiden . Hier is wat het zou kunnen doen op enkele voorbeelden:



Nu is het nog minder duidelijk wat het “juiste antwoord” is. Hoe zit het met een hond gekleed in een kattenpak? Enz. Welke input het ook krijgt, het neurale net zal een antwoord genereren, en op een manier die redelijk consistent is met hoe mensen dat zouden kunnen. Zoals ik hierboven heb gezegd, is dat geen feit dat we kunnen "afleiden uit de eerste beginselen". Het is gewoon iets waarvan empirisch is vastgesteld dat het waar is, althans in bepaalde domeinen. Maar het is een belangrijke reden waarom neurale netwerken nuttig zijn: dat ze op de een of andere manier een "menselijke" manier van doen vastleggen.

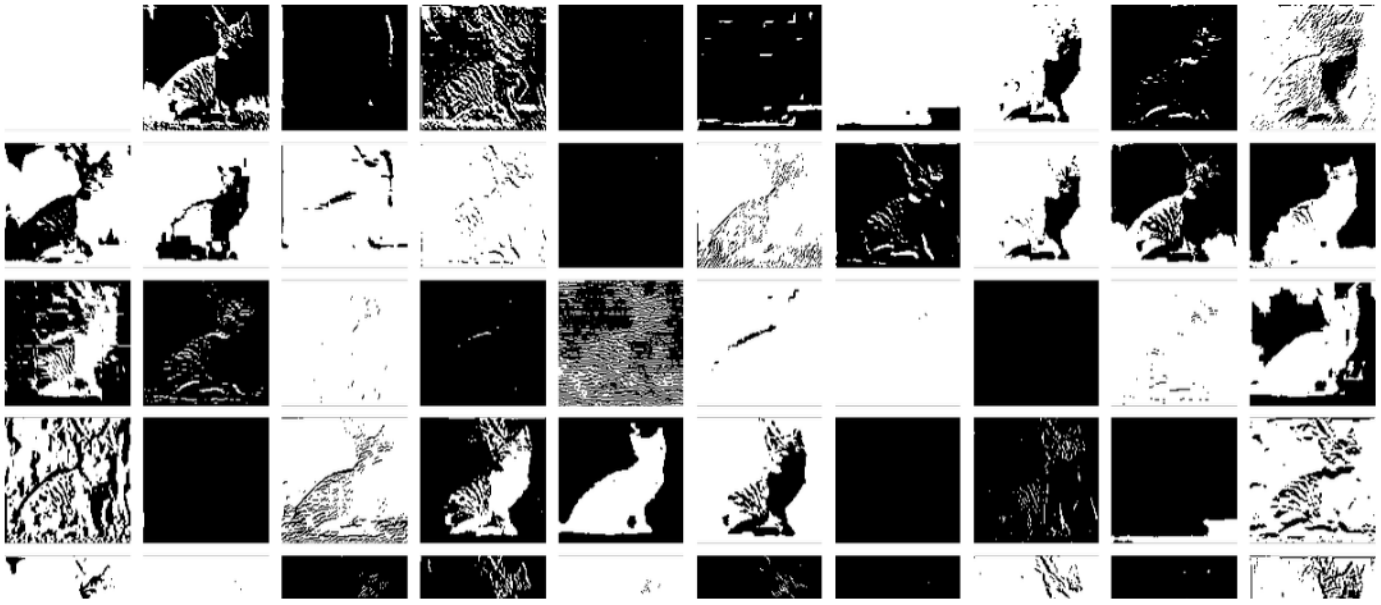
Laat jezelf een foto van een kat zien en vraag "Waarom is dat een kat?". Misschien zou je beginnen te zeggen: "Nou, ik zie zijn puntige oren, enz." Maar het is niet zo eenvoudig om uit te leggen hoe je de afbeelding als kat herkende. Het is gewoon dat je hersenen dat op de een of andere manier hebben bedacht. Maar voor een brein is er (althans nog) geen manier om "naar binnen te gaan" en te zien hoe het het heeft uitgezocht. Hoe zit het met een (kunstmatig) neuraal net? Nou, het is eenvoudig om te zien wat elk "neuron" doet als je een foto van een kat laat zien. Maar zelfs een basisvisualisatie krijgen is meestal erg moeilijk.

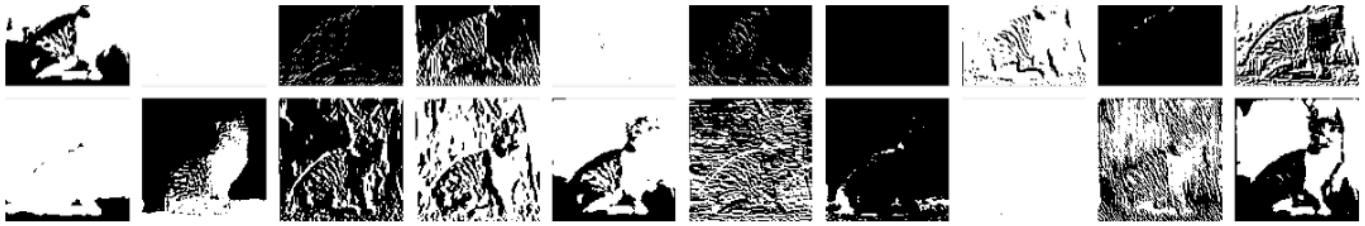
In het laatste net dat we gebruikten voor het "dichtstbijzijnde punt"-probleem hierboven zijn er 17 neuronen. In het net voor het herkennen van handgeschreven cijfers zijn er 2190. En in het net dat we gebruiken om katten en honden te herkennen zijn er 60.650. Normaal gesproken zou het behoorlijk moeilijk zijn om te visualiseren wat neerkomt op 60.650-dimensionale ruimte. Maar omdat dit een netwerk is dat is

opgezet om met afbeeldingen om te gaan, zijn veel van zijn neuronenlagen georganiseerd in arrays, zoals de arrays van pixels waar het naar kijkt.

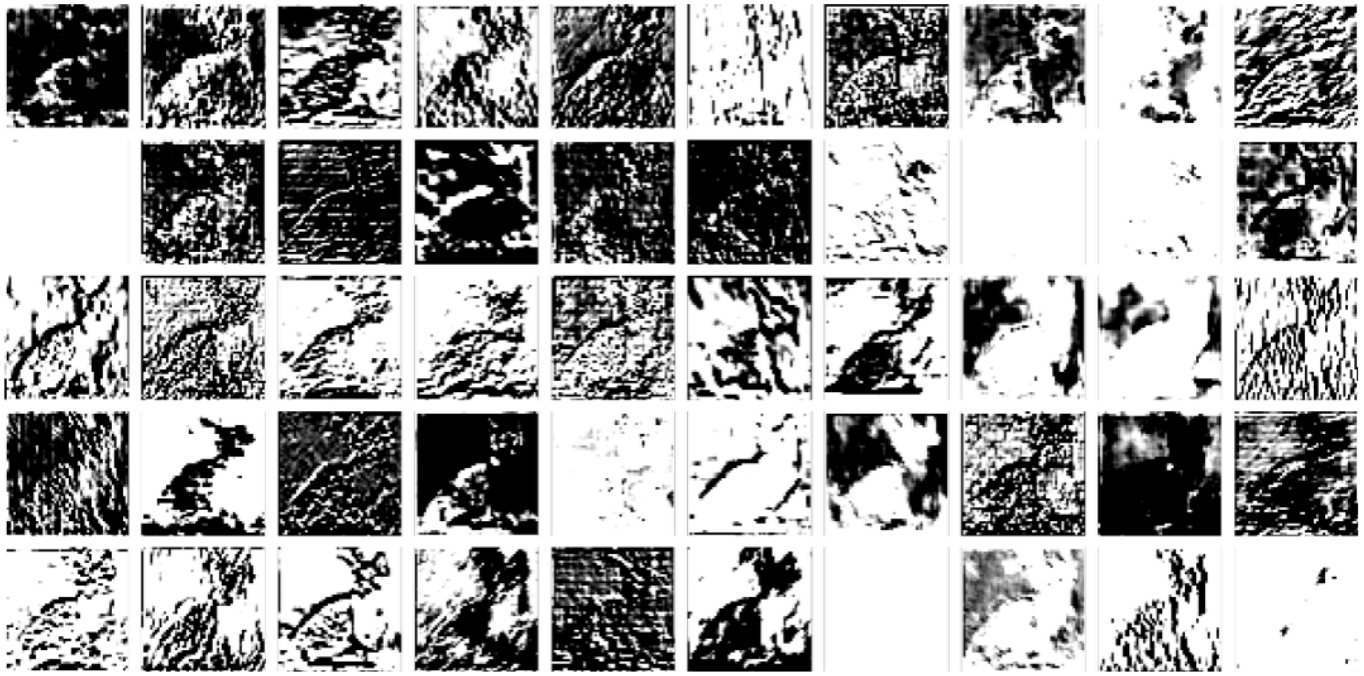
En als we een typisch kattenbeeld nemen

dan kunnen we de toestanden van neuronen op de eerste laag weergeven door een verzameling afgeleide afbeeldingen - waarvan we er vele gemakkelijk kunnen interpreteren als dingen als "de kat zonder zijn achtergrond" of "de omtrek van de kat":





Bij de 10e laag is het moeilijker te interpreteren wat er aan de hand is:



Maar in het algemeen zouden we kunnen zeggen dat het neurale net "bepaalde kenmerken eruit pikt" (misschien zijn er puntige oren), en deze gebruikt om te bepalen waar het beeld van is. Maar zijn dat kenmerken waarvoor we namen hebben, zoals "puntige oren"? Meestal niet.

Gebruiken onze hersenen vergelijkbare functies? Meestal weten we het niet. Maar het is opmerkelijk dat de eerste paar lagen van een neuraal net, zoals degene die we hier

laten zien, aspecten van afbeeldingen (zoals randen van objecten) lijken op te pikken die lijken op degene waarvan we weten dat ze worden uitgekozen door het eerste niveau van visuele verwerking in de hersenen.

Maar laten we zeggen dat we een "theorie van kattenherkenning" in neurale netwerken willen. We kunnen zeggen: "Kijk, dit specifieke net doet het" - en dat geeft ons meteen een idee van "hoe moeilijk een probleem" het is (en bijvoorbeeld hoeveel neuronen of lagen er nodig kunnen zijn). Maar vanaf nu hebben we tenminste geen manier om "een verhalende beschrijving te geven" van wat het netwerk aan het doen is. En misschien is dat omdat het echt rekenkundig onherleidbaar is, en er is geen algemene manier om erachter te komen wat het doet, behalve door elke stap expliciet te traceren. Of misschien is het gewoon dat we 'de wetenschap niet hebben doorgrond' en de 'natuurwetten' niet hebben geïdentificeerd die ons in staat stellen samen te vatten wat er aan de hand is.

We zullen dezelfde soorten problemen tegenkomen als we het hebben over het genereren van taal met ChatGPT. En nogmaals, het is niet duidelijk of er manieren zijn om "samen te vatten wat het doet". Maar door de rijkdom en detaillering van taal (en onze ervaring ermee) kunnen we wellicht verder komen dan met beelden.

Machine Learning en de training van neurale netwerken

We hebben het tot nu toe gehad over neurale netwerken die 'al weten' hoe ze bepaalde taken moeten uitvoeren. Maar wat neurale netwerken zo nuttig maakt (vermoedelijk ook in de hersenen) is dat ze niet alleen in principe allerlei taken kunnen uitvoeren,

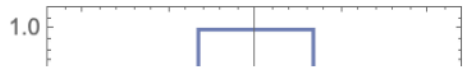
maar dat ze stapsgewijs kunnen worden "getraind aan de hand van voorbeelden" om die taken uit te voeren.

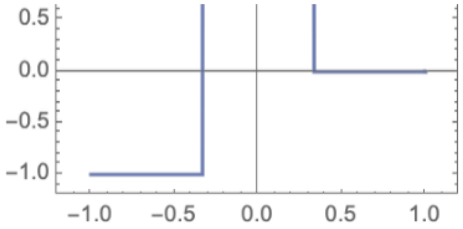
Als we een neuraal net maken om katten van honden te onderscheiden, hoeven we niet effectief een programma te schrijven dat (laten we zeggen) expliciet snorharen vindt; in plaats daarvan laten we gewoon veel voorbeelden zien van wat een kat is en wat een hond is, en laten we het netwerk hieruit "machine leren" hoe ze te onderscheiden.

En het punt is dat het getrainde netwerk "generaliseert" van de specifieke voorbeelden die het laat zien. Zoals we hierboven hebben gezien, is het niet alleen zo dat het netwerk het specifieke pixelpatroon herkent van een voorbeeld van een kattenafbeelding die het werd getoond; het is eerder dat het neurale net er op de een of andere manier in slaagt om beelden te onderscheiden op basis van wat wij beschouwen als een soort van "algemene katheid".

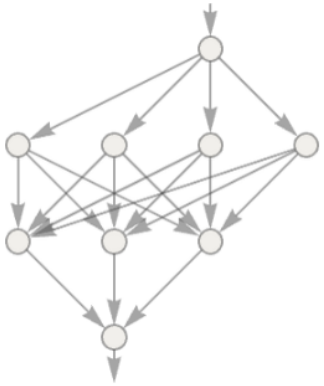
Dus hoe werkt neurale nettraining eigenlijk? In wezen proberen we altijd gewichten te vinden die ervoor zorgen dat het neurale net de voorbeelden die we hebben gegeven, met succes reproduceert. En dan vertrouwen we op het neurale net om op een "redelijke" manier te "interpoleren" (of "generaliseren") tussen deze voorbeelden.

Laten we eens kijken naar een probleem dat nog eenvoudiger is dan het dichtstbijzijnde punt hierboven. Laten we gewoon proberen een neuraal net te krijgen om de functie te leren:

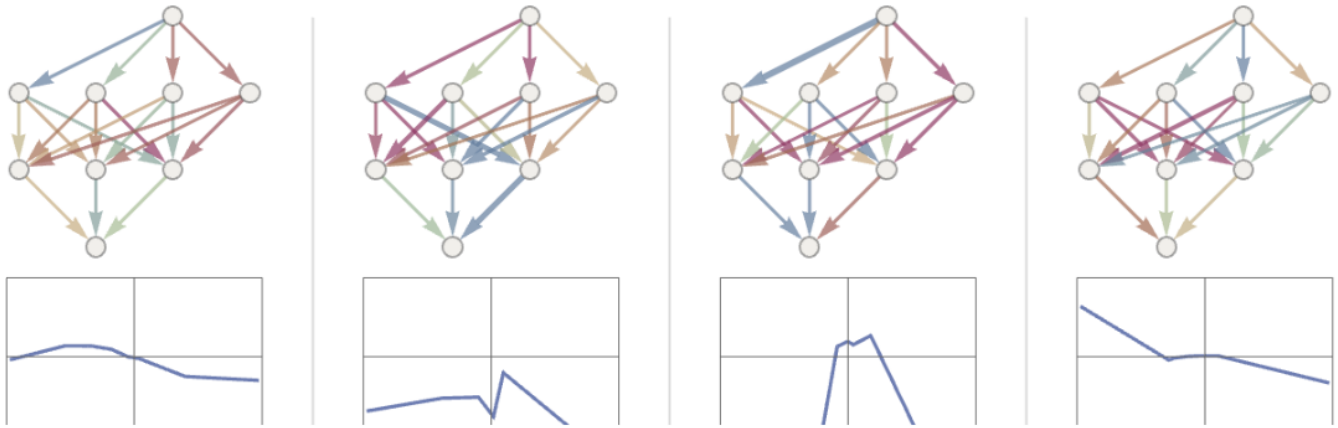




Voor deze taak hebben we een netwerk nodig dat slechts één ingang en één uitgang heeft, zoals:

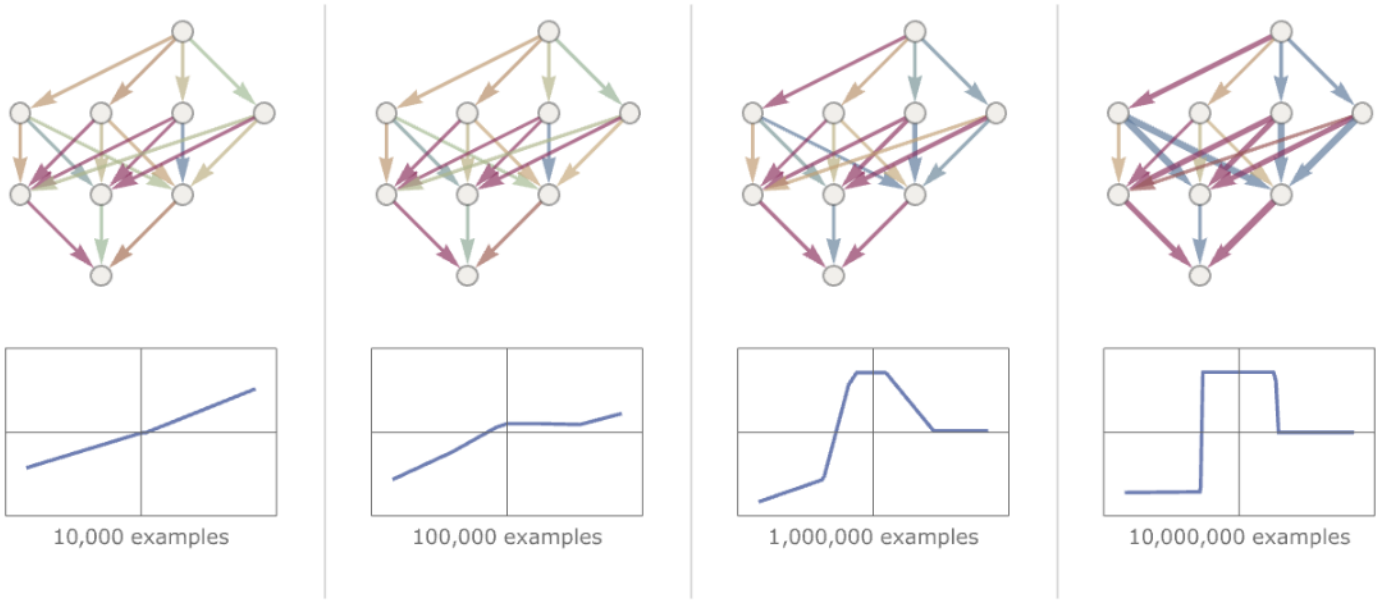


Maar welke gewichten enz. moeten we gebruiken? Met elke mogelijke set gewichten zal het neurale net een functie berekenen. En dit is bijvoorbeeld wat het doet met een paar willekeurig gekozen reeksen gewichten:



En ja, we kunnen duidelijk zien dat het in geen van deze gevallen ook maar in de buurt komt van het reproduceren van de functie die we willen. Dus hoe vinden we gewichten die de functie reproduceren?

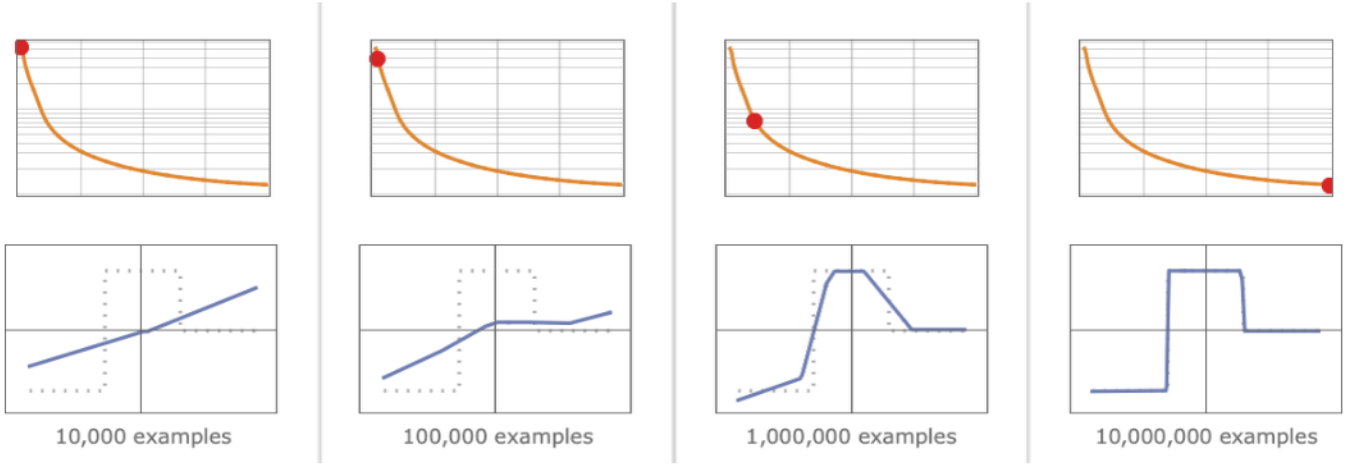
Het basisidee is om veel "input → output"-voorbeelden te geven om "van te leren" - en vervolgens te proberen gewichten te vinden die deze voorbeelden kunnen reproduceren. Dit is het resultaat van dat doen met steeds meer voorbeelden:



In elke fase van deze "training" worden de gewichten in het netwerk geleidelijk aangepast - en we zien dat we uiteindelijk een netwerk krijgen dat met succes de functie reproduceert die we willen. Dus hoe passen we de gewichten aan? Het basisidee is om in elke fase te zien "hoe ver we zijn" van het verkrijgen van de functie die we willen - en vervolgens de gewichten zodanig bij te werken dat ze dichterbij

komen.

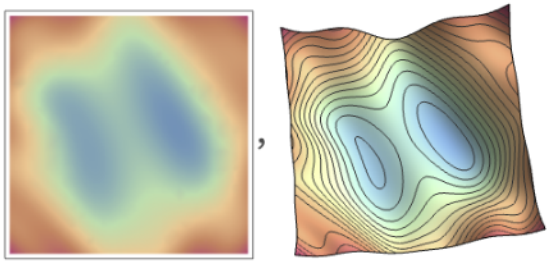
Om erachter te komen "hoe ver we zijn" berekenen we wat gewoonlijk een "verliesfunctie" wordt genoemd (of soms "kostenfunctie"). Hier gebruiken we een eenvoudige (L2) verliesfunctie die gewoon de som is van de kwadraten van de verschillen tussen de waarden die we krijgen en de werkelijke waarden. En wat we zien is dat naarmate ons trainingsproces vordert, de verliesfunctie geleidelijk afneemt (volgens een bepaalde "leercurve" die voor verschillende taken anders is) - totdat we een punt bereiken waarop het netwerk (tenminste tot een goede benadering) met succes reproduceert de functie die we willen:



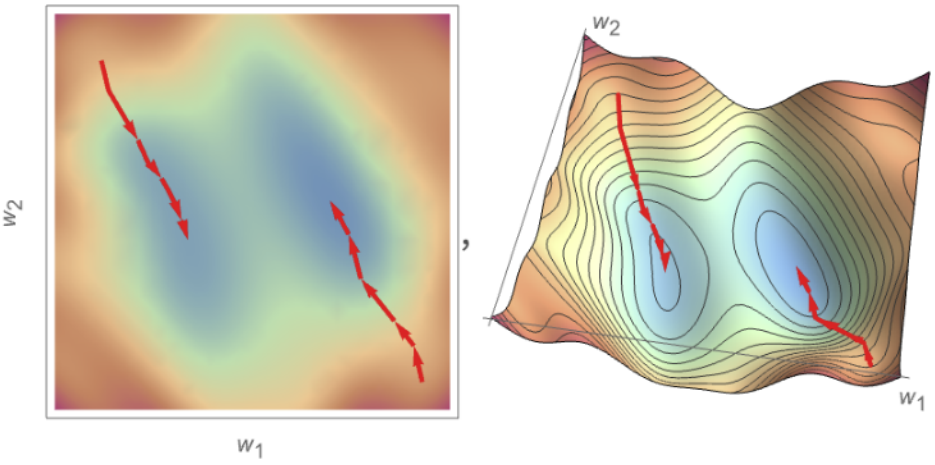
Oké, dus het laatste essentiële stuk om uit te leggen is hoe de gewichten worden aangepast om de verliesfunctie te verminderen. Zoals we hebben gezegd, geeft de verliesfunctie ons een "afstand" tussen de waarden die we hebben en de werkelijke waarden. Maar de 'waarden die we hebben' worden in elke fase bepaald door de huidige versie van het neurale net - en door de gewichten erin. Maar stel je nu voor

dat de gewichten variabelen zijn, zeg w_i. We willen uitzoeken hoe we de waarden van deze variabelen kunnen aanpassen om het verlies dat ervan afhangt te minimaliseren.

Stel je bijvoorbeeld voor (in een ongelooflijke vereenvoudiging van typische neurale netwerken die in de praktijk worden gebruikt) dat we slechts twee gewichten w_1 en w_2 hebben . Dan hebben we misschien een verlies dat er als functie van w_1 en w_2 zo uitziet:



Numerieke analyse biedt een verscheidenheid aan technieken om in dergelijke gevallen het minimum te vinden. Maar een typische benadering is om geleidelijk het pad van de steilste afdaling te volgen vanaf wat voor vorige w_1, w_2 we ook hadden:



Net als water dat van een berg naar beneden stroomt, is het enige dat gegarandeerd is dat deze procedure zal eindigen op een plaatselijk minimum van het oppervlak (“een bergmeer”); het zou wel eens niet het ultieme wereldwijde minimum kunnen bereiken.

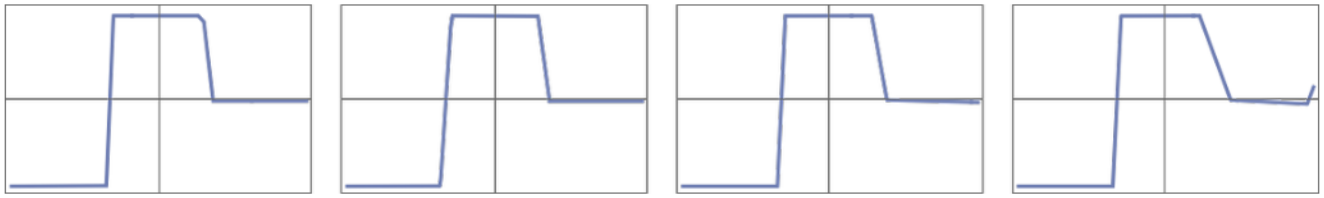
Het ligt niet voor de hand dat het haalbaar zou zijn om het pad van de steilste afdaling op het “gewichtslandschap” te vinden. Maar calculus komt te hulp. Zoals we hierboven vermeldden, kan men een neuraal net altijd beschouwen als het berekenen van een wiskundige functie - die afhangt van de invoer en de gewichten ervan. Maar overweeg nu om te differentiëren met betrekking tot deze gewichten. Het blijkt dat de kettingregel van calculus ons in feite in staat stelt de bewerkingen te "ontrafelen" die worden uitgevoerd door opeenvolgende lagen in het neurale net. En het resultaat is dat we - althans in een bepaalde lokale benadering - de werking van het neurale net kunnen 'omkeren', en progressief gewichten kunnen vinden die het verlies in verband met de output minimaliseren.

De afbeelding hierboven toont het soort minimalisatie dat we zouden moeten doen in het onrealistisch eenvoudige geval van slechts 2 gewichten. Maar het blijkt dat zelfs met veel meer gewichten (ChatGPT gebruikt 175 miljard) het nog steeds mogelijk is om te minimaliseren, althans tot op zekere hoogte. En in feite hield de grote doorbraak in "diep leren" die plaatsvond rond 2011 verband met de ontdekking dat het in zekere zin gemakkelijker kan zijn om (tenminste bij benadering) te minimaliseren wanneer er veel gewichten bij betrokken zijn dan wanneer er vrij weinig zijn.

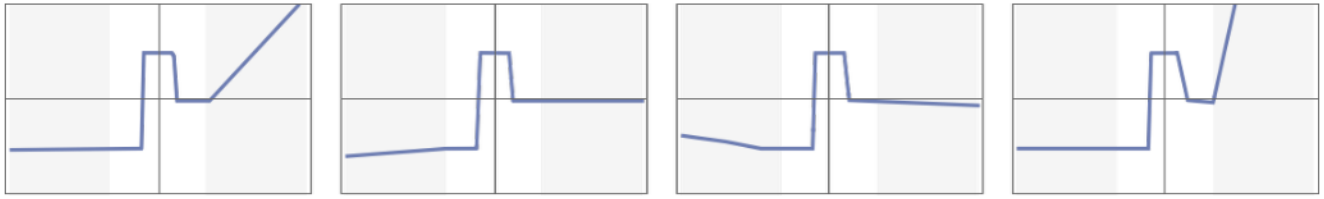
Met andere woorden - enigszins contra-intuïtief - kan het gemakkelijker zijn om meer

gecompliceerde problemen met neurale netwerken op te lossen dan eenvoudigere. En de ruwe reden hiervoor lijkt te zijn dat wanneer men veel "gewichtsvariabelen" heeft, men een hoog-dimensionale ruimte heeft met "veel verschillende richtingen" die iemand naar het minimum kunnen leiden - terwijl het met minder variabelen gemakkelijker is om te eindigen vast komen te zitten in een lokaal minimum ("bergmeer") van waaruit er geen "richting om eruit te komen" is.

Het is de moeite waard erop te wijzen dat er in typische gevallen veel verschillende verzamelingen van gewichten zijn die allemaal neurale netwerken opleveren die vrijwel dezelfde prestaties hebben. En meestal worden er bij praktische neurale nettraining veel willekeurige keuzes gemaakt, die leiden tot "verschillende maar gelijkwaardige oplossingen", zoals deze:



Maar elk van deze "andere oplossingen" zal op zijn minst iets ander gedrag vertonen. En als we bijvoorbeeld om een "extrapolatie" vragen buiten de regio waar we trainingsvoorbeelden hebben gegeven, kunnen we dramatisch andere resultaten krijgen:



Maar welke van deze is "juist"? Er is echt geen manier om te zeggen. Ze zijn allemaal "in overeenstemming met de waargenomen gegevens". Maar ze komen allemaal overeen met verschillende "aangeboren" manieren om "na te denken" over wat te doen "out of the box". En sommige lijken misschien "redelijker" voor ons mensen dan andere.

De praktijk en overlevering van neurale nettraining

Vooral in het afgelopen decennium zijn er veel vorderingen gemaakt in de kunst van het trainen van neurale netwerken. En ja, het is eigenlijk een kunst. Soms - vooral achteraf gezien - zie je op zijn minst een glimp van een 'wetenschappelijke verklaring' voor iets dat wordt gedaan. Maar de meeste dingen zijn met vallen en opstaan ontdekt, waarbij ideeën en trucs zijn toegevoegd die geleidelijk aan een belangrijke kennis hebben opgebouwd over het werken met neurale netwerken.

Er zijn verschillende belangrijke onderdelen. Ten eerste is er de kwestie van welke architectuur van het neurale net men voor een bepaalde taak moet gebruiken. Dan is er de kritieke kwestie van hoe men de gegevens gaat krijgen waarop het neurale net kan worden getraind. En in toenemende mate heeft men niet te maken met het vanaf nul trainen van een net: in plaats daarvan kan een nieuw net direct een ander reeds getraind net opnemen, of kan dat net in ieder geval gebruiken om meer trainingsvoorbeelden voor zichzelf te genereren.

Je zou kunnen denken dat je voor elk specifiek soort taak een andere architectuur van

het neurale netwerk nodig zou hebben. Maar wat is gevonden, is dat dezelfde architectuur vaak lijkt te werken, zelfs voor ogenschijnlijk heel verschillende taken. Op een bepaald niveau doet dit denken aan het idee van universele berekening (en mijn Principle of Computational Equivalence), maar, zoals ik later zal bespreken, denk ik dat het meer een weerspiegeling is van het feit dat de taken die we doorgaans proberen te krijgen neuraal zijn. netten om te doen zijn "mensachtige" - en neurale netten kunnen vrij algemene "mensachtige processen" vastleggen.

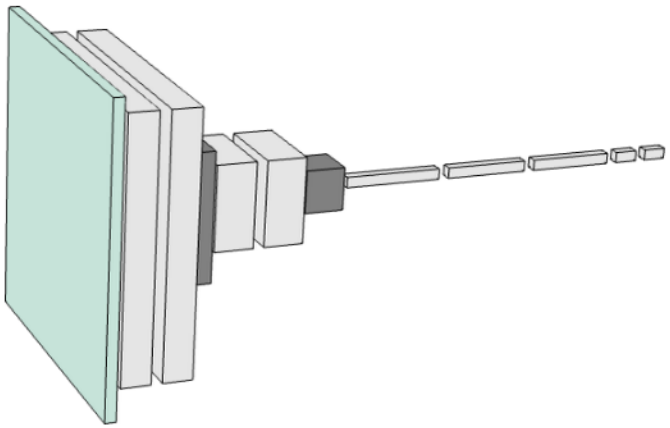
In vroegere dagen van neurale netwerken was er de neiging om "het neurale net zo min mogelijk te laten doen". Bij het omzetten van spraak naar tekst dacht men bijvoorbeeld dat men eerst de audio van de spraak moest analyseren, deze in fonemen moest opsplitsen, enz. om te proberen het neurale net te trainen op het "end-to-end-probleem", door het zelf de noodzakelijke tussenliggende functies, coderingen, enz. Te laten "ontdekken".

Er was ook het idee dat men ingewikkelde individuele componenten in het neurale net moest introduceren, om het in feite "bepaalde algoritmische ideeën expliciet te laten implementeren". Maar nogmaals, dit is meestal niet de moeite waard gebleken; in plaats daarvan is het beter om gewoon met heel eenvoudige componenten om te gaan en ze zichzelf te laten "organiseren" (zij het meestal op een manier die we niet kunnen begrijpen) om (vermoedelijk) het equivalent van die algoritmische ideeën te bereiken.

Dat wil niet zeggen dat er geen 'structurerende ideeën' zijn die relevant zijn voor neurale netwerken. Zo lijkt het hebben van 2D-arrays van neuronen met lokale verbindingen op zijn minst erg nuttig in de vroege stadia van het verwerken van

beelden. En het hebben van verbindingspatronen die zich concentreren op 'achtereenvolgens terugkijken' lijkt nuttig - zoals we later zullen zien - bij het omgaan met zaken als menselijke taal, bijvoorbeeld in ChatGPT.

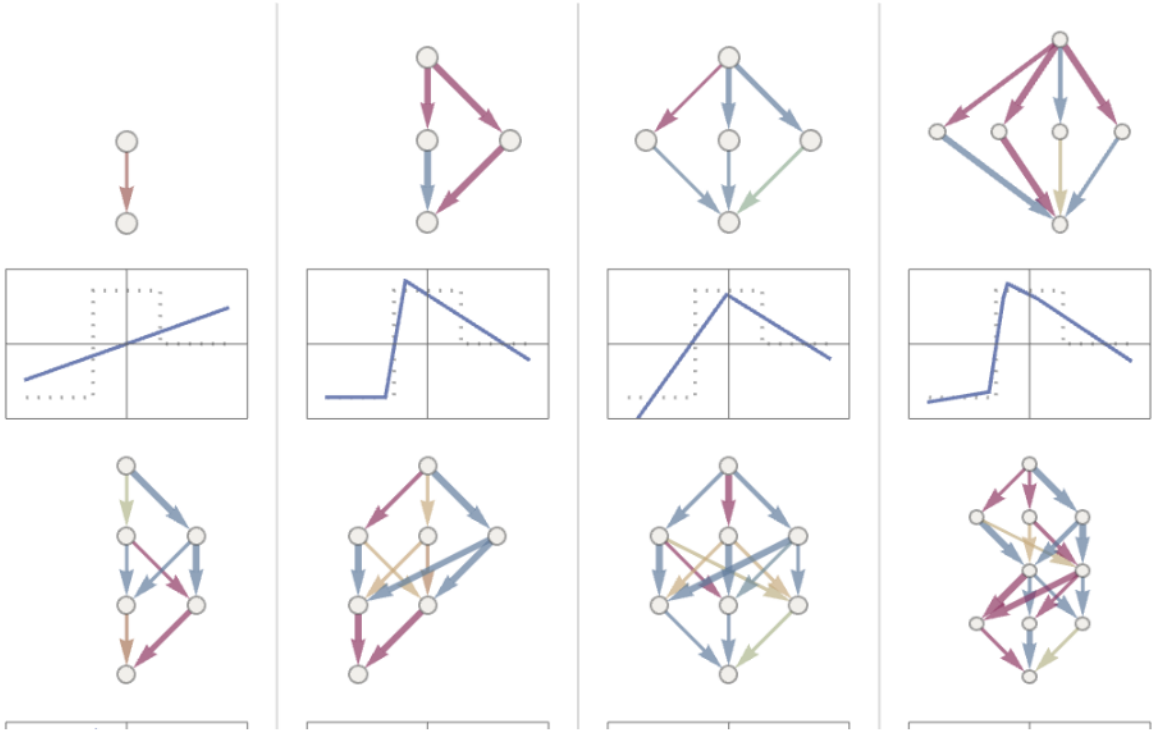
Maar een belangrijk kenmerk van neurale netwerken is dat ze, net als computers in het algemeen, uiteindelijk alleen met gegevens te maken hebben. En de huidige neurale netwerken - met de huidige benaderingen van neurale nettraining - hebben specifiek betrekking op reeksen getallen . Maar tijdens de verwerking kunnen die arrays volledig worden herschikt en hervormd. En als voorbeeld, het netwerk dat we gebruikten voor het identificeren van cijfers hierboven , begint met een 2D "beeldachtige" array, snel "verdikkend" naar veel kanalen, maar dan "concentrerend" in een 1D-array die uiteindelijk elementen zal bevatten die de verschillende mogelijke uitgangscijfers:



Maar goed, hoe weet je hoe groot een neuraal net is dat je nodig hebt voor een bepaalde taak? Het is iets van een kunst. Op een bepaald niveau is het belangrijkste om te weten "hoe moeilijk de taak is". Maar voor menselijke taken is dat meestal erg

moeilijk in te schatten. Ja, er kan een systematische manier zijn om de taak zeer "mechanisch" met de computer uit te voeren. Maar het is moeilijk om te weten of er trucs of snelkoppelingen zijn die iemand in staat stellen om de taak op zijn minst op een "menselijk niveau" veel gemakkelijker uit te voeren. Het kan nodig zijn om een gigantische spelboom op te sommen om een bepaald spel "mechanisch" te spelen; maar er zou een veel gemakkelijkere ("heuristische") manier kunnen zijn om "spelen op menselijk niveau" te bereiken.

Als je te maken hebt met kleine neurale netwerken en eenvoudige taken, kun je soms expliciet zien dat je "van hieruit niet kunt komen". Dit is bijvoorbeeld het beste dat men lijkt te kunnen doen voor de taak uit de vorige sectie met een paar kleine neurale netwerken:



En wat we zien is dat als het net te klein is, het gewoon niet de gewenste functie kan reproduceren. Maar boven een bepaalde maat heeft hij geen probleem - tenminste als men hem lang genoeg traint, met voldoende voorbeelden. En tussen twee haakjes, deze foto's illustreren een stukje neurale netwerkoverlevering: dat je vaak wegkomt met een kleiner netwerk als er een "squeeze" in het midden is die alles dwingt om door een kleiner tussenliggend aantal neuronen te gaan. (Het is ook vermeldenswaard dat "geen tussenlaag" - of zogenaamde " perceptron " -netwerken alleen in wezen lineaire functies kunnen leren - maar zodra er ook maar één tussenlaag is, is het in principe altijd mogelijkom elke functie willekeurig goed te benaderen, tenminste als men genoeg neuronen heeft, maar om het haalbaar trainbaar te maken, heeft men meestal een soort regularisatie of normalisatie .)

OK, dus laten we zeggen dat je genoegen hebt genomen met een bepaalde neurale netwerkarchitectuur. Nu is er de kwestie van het verkrijgen van gegevens om het netwerk mee te trainen. En veel van de praktische uitdagingen rond neurale netwerken - en machine learning in het algemeen - zijn gericht op het verkrijgen of voorbereiden van de benodigde trainingsgegevens. In veel gevallen ("supervised learning") wil men expliciete voorbeelden krijgen van inputs en de outputs die men daarvan verwacht. Zo zou men bijvoorbeeld afbeeldingen willen die getagd zijn door wat erin staat, of door een ander attribuut. En misschien moet men er expliciet doorheen gaan - meestal met grote moeite - en de tagging doen. Maar heel vaak blijkt

het mogelijk om mee te liften op iets dat al is gedaan, of het als een soort proxy te gebruiken. En dus zou men bijvoorbeeld alt-tags kunnen gebruiken die zijn verstrekt voor afbeeldingen op internet. Of, in een ander domein, men zou ondertitels kunnen gebruiken die zijn gemaakt voor video's. Of - voor taalvertaaltraining - kan men parallelle versies gebruiken van webpagina's of andere documenten die in verschillende talen bestaan.

Hoeveel data heb je nodig om een neuraal net te laten zien om het te trainen voor een bepaalde taak? Nogmaals, het is moeilijk in te schatten op basis van de eerste principes. De vereisten kunnen zeker drastisch worden verlaagd door "leren overdragen" te gebruiken om zaken als lijsten met belangrijke functies die al in een ander netwerk zijn geleerd, "over te dragen". Maar over het algemeen moeten neurale netwerken "veel voorbeelden zien" om goed te kunnen trainen. En voor sommige taken is het in ieder geval een belangrijk stuk neurale netwerkkennis dat de voorbeelden ongelooflijk repetitief kunnen zijn. En inderdaad, het is een standaardstrategie om een neuraal net alle voorbeelden te laten zien, keer op keer. In elk van deze "trainingsrondes" (of "epochs") zal het neurale net zich in op zijn minst een iets andere toestand bevinden, en op de een of andere manier "herinneren" aan een bepaald voorbeeld is nuttig om het "dat voorbeeld te laten onthouden". (En ja,

Maar vaak is het niet genoeg om hetzelfde voorbeeld keer op keer te herhalen. Het is ook nodig om de neurale netvariaties van het voorbeeld te tonen. En het is een kenmerk van de kennis van het neurale net dat die variaties op het gebied van 'gegevensvergroting' niet geavanceerd hoeven te zijn om bruikbaar te zijn. Door afbeeldingen slechts lichtjes aan te passen met basisbeeldverwerking, kunnen ze in

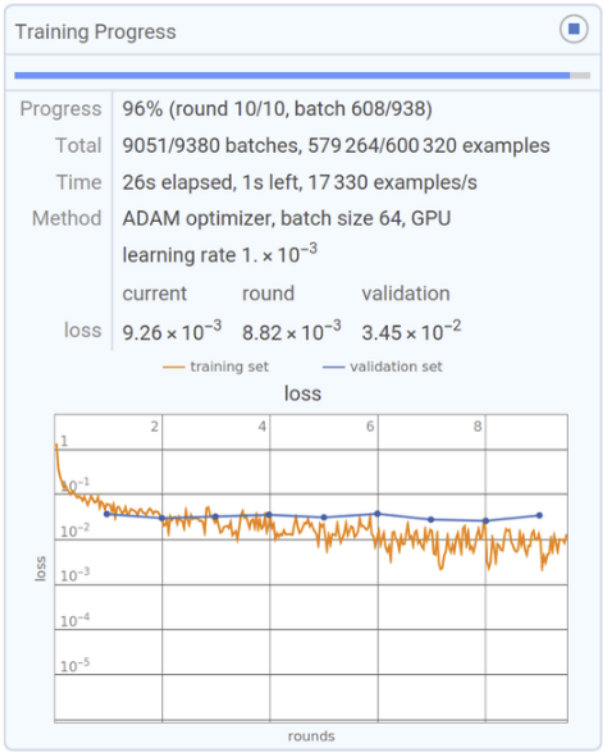
wezen "zo goed als nieuw" worden voor training van neurale netwerken. En evenzo, als je geen echte video meer hebt, enz. Voor het trainen van zelfrijdende auto's, kun je doorgaan en gewoon gegevens krijgen van het uitvoeren van simulaties in een model-videogame-achtige omgeving zonder alle details van echte scènes uit de echte wereld.

Hoe zit het met iets als ChatGPT? Nou, het heeft de leuke eigenschap dat het "unsupervised learning" kan doen, waardoor het veel gemakkelijker wordt om voorbeelden te krijgen om van te trainen. Bedenk dat de basistaak van ChatGPT is om erachter te komen hoe een gegeven tekst moet worden voortgezet. Dus om het "trainingsvoorbeelden" te krijgen, hoef je alleen maar een stuk tekst te pakken en het einde ervan te maskeren, en dit dan te gebruiken als de "invoer om van te trainen" - waarbij de "uitvoer" het volledige is, ontmaskerd stukje tekst. We zullen hier later meer over vertellen, maar het belangrijkste punt is dat - in tegenstelling tot bijvoorbeeld om te leren wat er in afbeeldingen staat - er geen "expliciete tagging" nodig is; ChatGPT kan in feite gewoon rechtstreeks leren van de voorbeelden van tekst die het krijgt.

OK, dus hoe zit het met het eigenlijke leerproces in een neuraal net? Uiteindelijk gaat het erom te bepalen welke gewichten het beste passen bij de trainingsvoorbeelden die zijn gegeven. En er zijn allerlei gedetailleerde keuzes en "hyperparameterinstellingen" (zo genoemd omdat de gewichten kunnen worden gezien als "parameters") die kunnen worden gebruikt om aan te passen hoe dit wordt gedaan. Er zijn verschillende keuzes van de verliesfunctie(som van kwadraten, som van absolute waarden, enz.). Er zijn verschillende manieren om het verlies te minimaliseren (hoe ver in de gewichtsruimte moet worden verplaatst bij elke stap, enz.). En dan zijn er nog vragen

als hoe groot een "batch" van voorbeelden moet zijn om elke opeenvolgende schatting te krijgen van het verlies dat men probeert te minimaliseren. En ja, je kunt machinaal leren toepassen (zoals we bijvoorbeeld doen in Wolfram Language) om machinaal leren te automatiseren – en om zaken als hyperparameters automatisch in te stellen.

Maar uiteindelijk kan het hele trainingsproces worden gekarakteriseerd door te zien hoe het verlies geleidelijk afneemt (zoals in deze Wolfram Language-voortgangsmonitor voor een kleine training):



En wat je meestal ziet, is dat het verlies een tijdje afneemt, maar uiteindelijk vlak wordt bij een constante waarde. Als die waarde voldoende klein is, kan de training als geslaagd worden beschouwd; anders is het waarschijnlijk een teken dat men zou

moeten proberen de netwerkarchitectuur te veranderen.

Kan iemand zeggen hoe lang het zou moeten duren voordat de "leercurve" afvlakt? Zoals voor zoveel andere dingen, lijken er bij benadering machtswet-schaalrelaties te zijn die afhangen van de grootte van het neurale net en de hoeveelheid gegevens die men gebruikt. Maar de algemene conclusie is dat het trainen van een neuraal net moeilijk is en veel rekenkracht vergt. En in de praktijk wordt het overgrote deel van die inspanning besteed aan het uitvoeren van bewerkingen op reeksen getallen, en dat is waar GPU's goed in zijn - daarom wordt neurale nettraining doorgaans beperkt door de beschikbaarheid van GPU's.

Zullen er in de toekomst fundamenteel betere manieren zijn om neurale netwerken te trainen – of in het algemeen te doen wat neurale netwerken doen? Vrijwel zeker, denk ik. Het fundamentele idee van neurale netwerken is om een flexibele "computerstructuur" te creëren uit een groot aantal eenvoudige (in wezen identieke) componenten - en om deze "structuur" er een te laten zijn die stapsgewijs kan worden aangepast om van voorbeelden te leren. In de huidige neurale netwerken gebruikt men in wezen de ideeën van calculus - toegepast op reële getallen - om die incrementele wijziging uit te voeren. Maar het wordt steeds duidelijker dat het hebben van zeer nauwkeurige getallen er niet toe doet; 8 bits of minder zou genoeg kunnen zijn, zelfs met de huidige methoden.

Met computationele systemen zoals cellulaire automaten die in principe parallel werken op veel individuele bits, is het nooit duidelijk geweest hoe dit soort incrementele modificatie moet worden uitgevoerd , maar er is geen reden om te denken dat het niet mogelijk is. En in feite kan het, net als bij de " doorbraak in diep

leren van 2012 ", zijn dat een dergelijke stapsgewijze wijziging in meer gecompliceerde gevallen gemakkelijker zal zijn dan in eenvoudige gevallen.

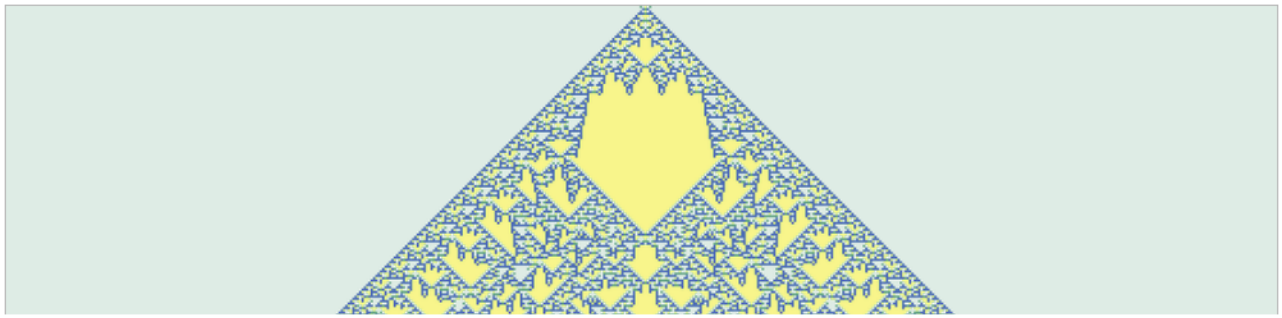
Neurale netwerken - misschien een beetje zoals hersenen - zijn opgezet om een in wezen vast netwerk van neuronen te hebben, waarbij de kracht ('gewicht') van verbindingen ertussen is gewijzigd. (Misschien kunnen er in tenminste jonge hersenen ook aanzienlijke aantallen geheel nieuwe verbindingen ontstaan.) Maar hoewel dit voor de biologie misschien een handige opzet is, is het helemaal niet duidelijk of het ook maar in de buurt komt van de beste manier om de functionaliteit te bereiken die we nodig hebben. En iets dat het equivalent van het progressief herschrijven van netwerken inhoudt (misschien doet het denken aan ons natuurkundeproject) zou uiteindelijk wel eens beter kunnen zijn.

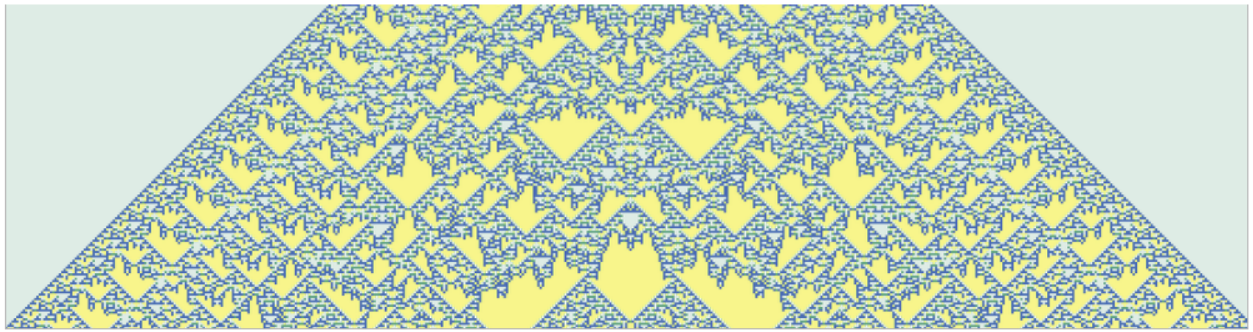
Maar zelfs binnen het kader van bestaande neurale netwerken is er momenteel een cruciale beperking: neurale nettraining zoals die nu wordt gedaan, is fundamenteel sequentieel, waarbij de effecten van elke reeks voorbeelden worden doorgegeven om de gewichten bij te werken. En inderdaad, met de huidige computerhardware - zelfs rekening houdend met GPU's - is het grootste deel van een neuraal netwerk tijdens de training meestal "inactief", waarbij slechts één onderdeel tegelijk wordt bijgewerkt. En in zekere zin komt dit omdat onze huidige computers vaak geheugen hebben dat gescheiden is van hun CPU's (of GPU's). Maar in de hersenen is het vermoedelijk anders, waarbij elk 'geheugenelement' (dwz neuron) ook een potentieel actief rekenelement is. En als we onze toekomstige computerhardware zo zouden kunnen inrichten, zou het misschien mogelijk worden om veel efficiënter te trainen.

"Een netwerk dat groot genoeg is, kan toch wel alles!"

De mogelijkheden van zoiets als ChatGPT lijken zo indrukwekkend dat je je zou kunnen voorstellen dat als je gewoon door kon gaan en steeds grotere neurale netwerken kon trainen, ze uiteindelijk in staat zouden zijn om "alles te doen". En als iemand zich bezighoudt met dingen die gemakkelijk toegankelijk zijn voor het onmiddellijke menselijke denken, is het heel goed mogelijk dat dit het geval is. Maar de les van de afgelopen honderden jaren wetenschap is dat er dingen zijn die door formele processen kunnen worden uitgezocht, maar die niet direct toegankelijk zijn voor het onmiddellijke menselijke denken.

Niet-triviale wiskunde is een groot voorbeeld. Maar het algemene geval is eigenlijk berekening. En uiteindelijk gaat het om het fenomeen van computationele onherleidbaarheid . Er zijn enkele berekeningen waarvan je zou denken dat er veel stappen nodig zijn, maar die in feite kunnen worden 'gereduceerd' tot iets heel onmiddellijks. Maar de ontdekking van computationele onherleidbaarheid impliceert dat dit niet altijd werkt. En in plaats daarvan zijn er processen - waarschijnlijk zoals die hieronder - waarbij om uit te zoeken wat er gebeurt onvermijdelijk in wezen elke rekenstap moet worden getraceerd:





Het soort dingen dat we normaal gesproken met onze hersenen doen, is vermoedelijk specifiek gekozen om computationele onherleidbaarheid te voorkomen. Het kost speciale moeite om wiskunde in de hersenen te doen. En het is in de praktijk grotendeels onmogelijk om de stappen in de werking van een niet-triviaal programma alleen maar in je hoofd te 'overdenken'.

Maar daarvoor hebben we natuurlijk computers. En met computers kunnen we gemakkelijk lange, rekenkundig onherleidbare dingen doen. En het belangrijkste punt is dat er over het algemeen geen kortere weg voor is.

Ja, we zouden veel specifieke voorbeelden kunnen onthouden van wat er in een bepaald computersysteem gebeurt. En misschien kunnen we zelfs enkele ("computationeel reduceerbare") patronen zien waarmee we een beetje kunnen generaliseren. Maar het punt is dat computationele onherleidbaarheid betekent dat we nooit kunnen garanderen dat het onverwachte niet zal gebeuren - en alleen door de berekening expliciet uit te voeren, kun je vertellen wat er werkelijk gebeurt in een bepaald geval.

En uiteindelijk is er gewoon een fundamentele spanning tussen leerbaarheid en computationele onherleidbaarheid. Leren omvat in feite het comprimeren van

gegevens door gebruik te maken van regelmatigheden . Maar computationele onherleidbaarheid houdt in dat er uiteindelijk een grens is aan de regelmatigheden die er kunnen zijn.

In de praktijk kun je je voorstellen dat je kleine computationele apparaten, zoals cellulaire automaten of Turing-machines, bouwt in trainbare systemen zoals neurale netwerken. En dergelijke apparaten kunnen inderdaad dienen als goede "tools" voor het neurale net, zoals Wolfram|Alpha een goede tool kan zijn voor ChatGPT . Maar computationele onherleidbaarheid houdt in dat je niet kunt verwachten dat je in die apparaten 'binnenkomt' en ze laat leren.

Of anders gezegd, er is een ultieme afweging tussen bekwaamheid en trainbaarheid: hoe meer je wilt dat een systeem "echt gebruik" maakt van zijn rekenmogelijkheden, hoe meer het rekenkundige onherleidbaarheid zal laten zien, en hoe minder het trainbaar zal zijn. En hoe meer het fundamenteel trainbaar is, hoe minder het in staat zal zijn om geavanceerde berekeningen uit te voeren.

(Voor ChatGPT zoals het nu is, is de situatie eigenlijk veel extremer, omdat het neurale net dat wordt gebruikt om elk uitvoertoken te genereren een puur "feed-forward"-netwerk is, zonder lussen, en daarom niet in staat is om enige vorm van berekening met niet-triviale "control flow".)

Je kunt je natuurlijk afvragen of het wel zo belangrijk is om onherleidbare berekeningen te kunnen doen. En inderdaad, voor een groot deel van de menselijke geschiedenis was het niet bijzonder belangrijk. Maar onze moderne technologische wereld is gebouwd op techniek die op zijn minst gebruik maakt van wiskundige

berekeningen - en in toenemende mate ook van meer algemene berekeningen. En als we naar de natuurlijke wereld kijken, zit die vol onherleidbare berekeningen - die we langzaamaan begrijpen hoe we die kunnen nabootsen en gebruiken voor onze technologische doeleinden.

Ja, een neuraal net kan zeker de soorten regelmatigheden in de natuurlijke wereld opmerken die we ook gemakkelijk zouden kunnen opmerken met "zonder hulp menselijk denken". Maar als we dingen willen uitwerken die binnen het bereik van wiskundige of computationele wetenschap vallen, zal het neurale net dat niet kunnen doen - tenzij het effectief een 'gewoon' computersysteem 'als hulpmiddel' gebruikt.

Maar er is iets potentieel verwarrends aan dit alles. In het verleden waren er tal van taken, waaronder het schrijven van essays, waarvan we dachten dat ze op de een of andere manier 'fundamenteel te moeilijk' waren voor computers. En nu we ze zien gedaan door bijvoorbeeld ChatGPT, hebben we de neiging om plotseling te denken dat computers enorm veel krachtiger moeten zijn geworden - in het bijzonder dingen overtreffend die ze al in principe al konden doen (zoals het geleidelijk berekenen van het gedrag van computersystemen zoals cellulaire automaten) .

Maar dit is niet de juiste conclusie om te trekken. Computationeel onherleidbare processen zijn nog steeds computationeel onherleidbaar en zijn nog steeds fundamenteel moeilijk voor computers, zelfs als computers hun individuele stappen gemakkelijk kunnen berekenen. Wat we in plaats daarvan zouden moeten concluderen, is dat taken, zoals het schrijven van essays, die wij mensen zouden kunnen doen, maar waarvan we dachten dat computers ze niet konden, in zekere zin

rekenkundig eenvoudiger zijn dan we dachten.

Met andere woorden, de reden dat een neuraal netwerk succesvol kan zijn bij het schrijven van een essay, is omdat het schrijven van een essay een "rekentechnisch ondieper" probleem blijkt te zijn dan we dachten. En in zekere zin brengt dit ons dichter bij het 'hebben van een theorie' over hoe wij mensen erin slagen dingen te doen zoals het schrijven van essays, of in het algemeen omgaan met taal.

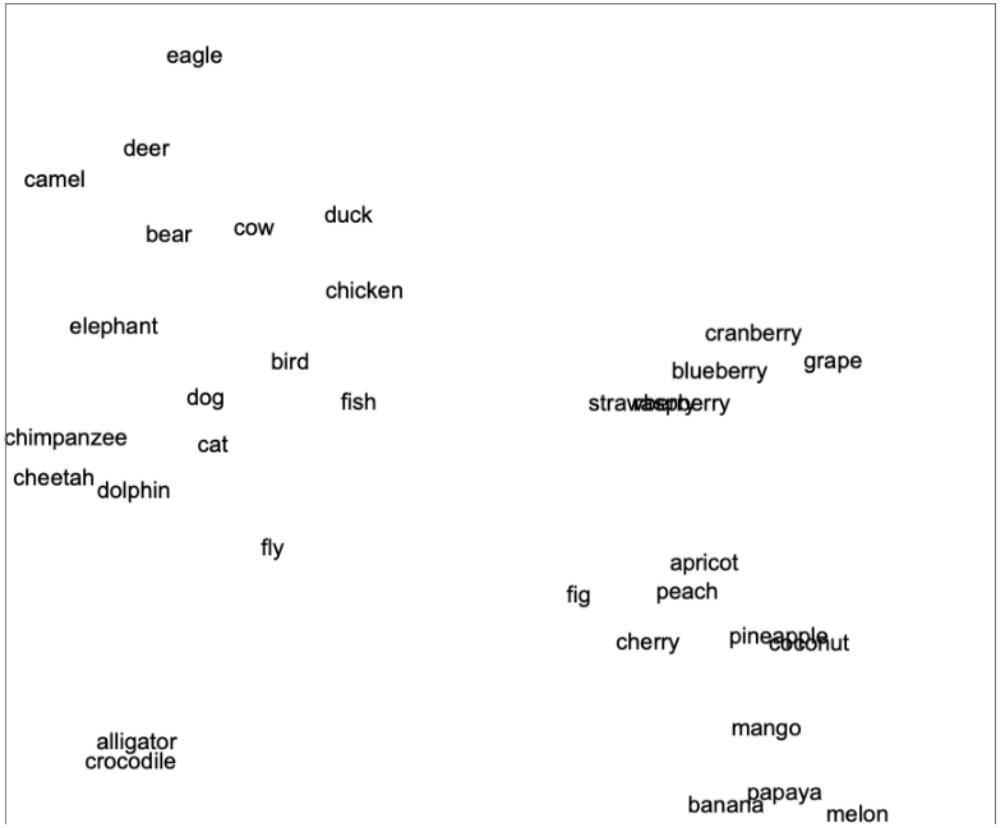
Als je een neuraal net had dat groot genoeg was, dan zou je misschien kunnen doen wat mensen gemakkelijk kunnen doen. Maar je zou niet vastleggen wat de natuurlijke wereld in het algemeen kan doen - of dat de gereedschappen die we uit de natuurlijke wereld hebben gemaakt, kunnen doen. En het is het gebruik van die tools - zowel praktische als conceptuele - die ons in de afgelopen eeuwen in staat hebben gesteld om de grenzen te overschrijden van wat toegankelijk is voor "puur menselijk denken", en voor menselijke doeleinden meer vast te leggen van wat er is in de fysieke en computationele universum.

Het concept van inbedding

Neurale netwerken, althans zoals ze nu zijn opgezet, zijn fundamenteel gebaseerd op cijfers. Dus als we ze gaan gebruiken om aan iets als tekst te werken, hebben we een manier nodig om onze tekst met getallen weer te geven . En we zouden zeker kunnen beginnen (in wezen zoals ChatGPT doet) door gewoon een nummer toe te wijzen aan elk woord in het woordenboek. Maar er is een belangrijk idee, dat bijvoorbeeld centraal staat in ChatGPT, dat verder gaat. En het is het idee van "inbedding". Men

kan een inbedding zien als een manier om te proberen de "essentie" van iets weer te geven door een reeks getallen - met de eigenschap dat "dingen in de buurt" worden weergegeven door getallen in de buurt.

En dus kunnen we bijvoorbeeld denken aan een woordinbedding als een poging om woorden in een soort 'betekenisruimte' neer te leggen waarin woorden die op de een of andere manier 'nabij in betekenis' zijn, dichtbij verschijnen in de inbedding. De daadwerkelijke inbeddingen die worden gebruikt, bijvoorbeeld in ChatGPT, hebben meestal betrekking op grote lijsten met getallen. Maar als we naar 2D projecteren, kunnen we voorbeelden laten zien van hoe woorden worden opgemaakt door de inbedding:



En ja, wat we zien doet het opmerkelijk goed in het vastleggen van typische alledaagse indrukken. Maar hoe kunnen we zo'n inbedding construeren? Het idee is grofweg om naar grote hoeveelheden tekst te kijken (hier 5 miljard woorden van het web) en dan te zien "hoe vergelijkbaar" de "omgevingen" zijn waarin verschillende woorden voorkomen. Dus, bijvoorbeeld, "alligator" en "krokodil" zullen vaak bijna uitwisselbaar voorkomen in zinnen die verder op elkaar lijken, en dat betekent dat ze dichtbij in de inbedding worden geplaatst. Maar "raap" en "adelaar" zullen niet vaak voorkomen in verder gelijkaardige zinnen, dus zullen ze ver uit elkaar geplaatst worden in de inbedding.

Maar hoe implementeer je zoiets eigenlijk met behulp van neurale netwerken? Laten we beginnen met te praten over inbedding, niet voor woorden, maar voor afbeeldingen. We willen een manier vinden om afbeeldingen te karakteriseren door middel van nummerlijsten, zodat "afbeeldingen die we als vergelijkbaar beschouwen" vergelijkbare nummerlijsten krijgen.

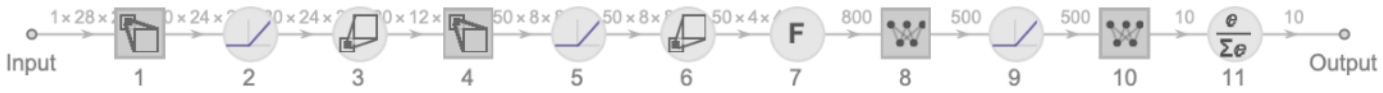
Hoe weten we of we "afbeeldingen als vergelijkbaar moeten beschouwen"? Welnu, als onze afbeeldingen bijvoorbeeld van handgeschreven cijfers zijn, zouden we "twee afbeeldingen als vergelijkbaar kunnen beschouwen" als ze van hetzelfde cijfer zijn. Eerder bespraken we een neuraal net dat was getraind om handgeschreven cijfers te herkennen. En we kunnen ons voorstellen dat dit neurale net zo is opgezet dat het in zijn uiteindelijke uitvoer afbeeldingen in 10 verschillende bakken plaatst, één voor elk

cijfer.

Maar wat als we "onderscheppen" wat er in het neurale net gebeurt voordat de definitieve "het is een '4'"-beslissing wordt genomen? We zouden kunnen verwachten dat er binnen het neurale net getallen zijn die afbeeldingen karakteriseren als "meestal 4-achtig maar een beetje 2-achtig" of iets dergelijks. En het idee is om dergelijke nummers op te pikken om als elementen in een inbedding te gebruiken.

Dus hier is het concept. In plaats van direct te proberen te karakteriseren "welk beeld is dichtbij welk ander beeld", beschouwen we in plaats daarvan een goed gedefinieerde taak (in dit geval cijferherkenning) waarvoor we expliciete trainingsgegevens kunnen krijgen - en gebruiken dan het feit dat bij het uitvoeren van deze taak de neurale net impliciet moet maken wat neerkomt op "nabijheidsbeslissingen". Dus in plaats van dat we ooit expliciet moeten praten over "nabijheid van afbeeldingen", hebben we het alleen over de concrete vraag welk cijfer een afbeelding vertegenwoordigt, en dan "laten we het over aan het neurale net" om impliciet te bepalen wat dat inhoudt over "nabijheid van beelden".

Dus hoe werkt dit meer in detail voor het cijferherkenningsnetwerk? We kunnen het netwerk zien als bestaande uit 11 opeenvolgende lagen, die we iconisch als volgt kunnen samenvatten (met activeringsfuncties weergegeven als afzonderlijke lagen):



In het begin voeren we werkelijke afbeeldingen in de eerste laag in, weergegeven door

2D-arrays van pixelwaarden. En aan het einde - van de laatste laag - krijgen we een reeks van 10 waarden, die we kunnen zeggen "hoe zeker" het netwerk is dat de afbeelding overeenkomt met elk van de cijfers 0 tot en met 9.

Voer de afbeelding in en de waarden van de neuronen in die laatste laag zijn:

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

Met andere woorden, het neurale net is op dit punt "ongelooflijk zeker" dat dit beeld een 4 is - en om daadwerkelijk de output "4" te krijgen, hoeven we alleen maar de positie van het neuron met de grootste waarde te kiezen.

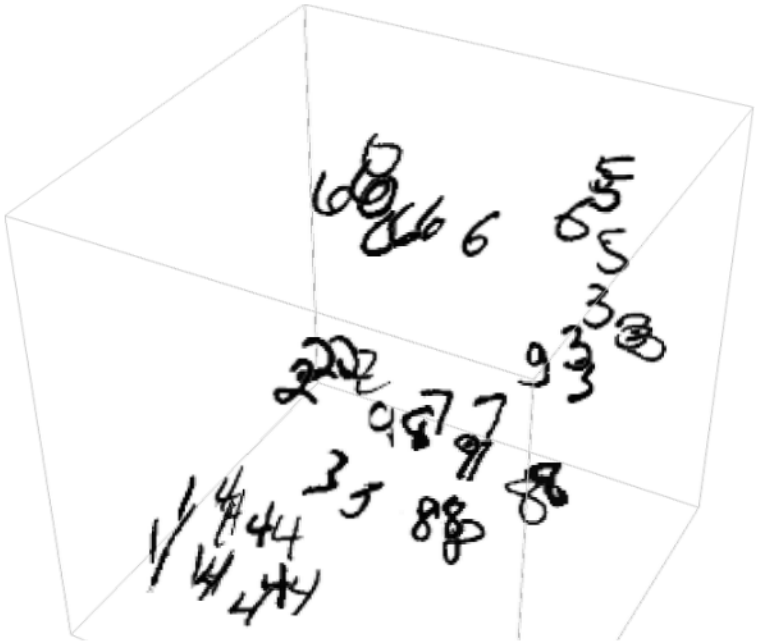
Maar wat als we een stap eerder kijken? De allerlaatste operatie in het netwerk is een zogenaamde softmax die zekerheid probeert af te dwingen. Maar voordat dat is toegepast, zijn de waarden van de neuronen:

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

Het neuron dat "4" vertegenwoordigt, heeft nog steeds de hoogste numerieke waarde. Maar er zit ook informatie in de waarden van de andere neuronen. En we kunnen verwachten dat deze lijst met getallen in zekere zin kan worden gebruikt om de 'essentie' van de afbeelding te karakteriseren - en dus iets te bieden dat we kunnen gebruiken als inbedding. En dus heeft bijvoorbeeld elk van de 4-en hier een iets andere "handtekening" (of "feature-inbedding") - allemaal heel anders dan de 8-en:

Hier gebruiken we in wezen 10 getallen om onze afbeeldingen te karakteriseren. Maar het is vaak beter om veel meer te gebruiken dan dat. En in ons cijferherkenningsnetwerk kunnen we bijvoorbeeld een reeks van 500 nummers krijgen door op de voorgaande laag te tikken. En dit is waarschijnlijk een redelijke array om te gebruiken als "image embedding".

Als we een expliciete visualisatie willen maken van "beeldruimte" voor handgeschreven cijfers, moeten we "de dimensie verkleinen", effectief door de 500-dimensionale vector die we hebben gekregen in bijvoorbeeld 3D-ruimte te projecteren:



We hebben het zojuist gehad over het creëren van een karakterisering (en dus inbedding) voor afbeeldingen, effectief gebaseerd op het identificeren van de gelijkenis van afbeeldingen door te bepalen of ze (volgens onze trainingsset) overeenkomen met hetzelfde handgeschreven cijfer. En we kunnen hetzelfde veel algemener doen voor afbeeldingen als we een trainingsset hebben die bijvoorbeeld identificeert van welke van de 5000 veelvoorkomende soorten objecten (kat, hond, stoel, ...) elke afbeelding is. En op deze manier kunnen we een beeld inbedden dat is "verankerd" door onze identificatie van gemeenschappelijke objecten, maar dan "veralgemeniseert daaromheen" volgens het gedrag van het neurale net. En het punt is dat voor zover dat gedrag overeenkomt met hoe wij mensen beelden waarnemen en interpreteren, dit uiteindelijk een inbedding zal zijn die "ons goed lijkt", en in de praktijk nuttig is bij het uitvoeren van "menselijk oordeelachtige" taken.

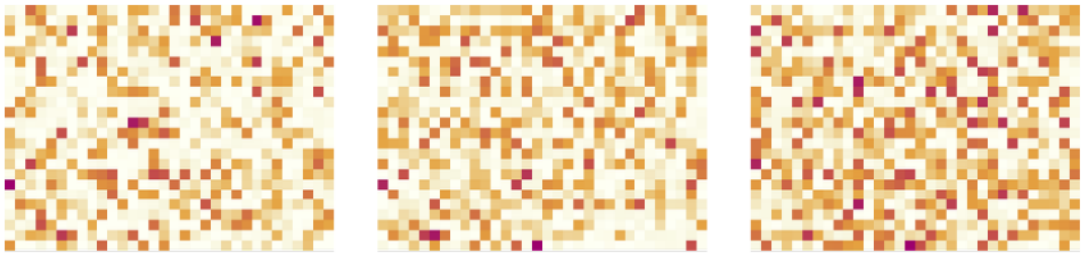
OK, dus hoe volgen we dezelfde soort benadering om inbeddingen voor woorden te vinden? De sleutel is om te beginnen met een taak over woorden waarvoor we gemakkelijk kunnen trainen. En de standaard dergelijke taak is "woordvoorspelling". Stel je voor dat we "de ____ kat" krijgen. Gebaseerd op een groot tekstcorpus (bijvoorbeeld de tekstinhoud van het web), wat zijn de kansen voor verschillende woorden die "de lege plek kunnen invullen"? Of, als alternatief, gegeven "____ zwart ____" wat zijn de kansen op verschillende "flankerende woorden"?

Hoe stellen we dit probleem op voor een neuraal net? Uiteindelijk moeten we alles in

cijfers formuleren. En een manier om dit te doen is door een uniek nummer toe te wijzen aan elk van de ongeveer 50.000 gewone woorden in het Engels. Dus, bijvoorbeeld, "de" kan 914 zijn, en "kat" (met een spatie ervoor) kan 3542 zijn. (En dit zijn de daadwerkelijke getallen die worden gebruikt door GPT-2.) Dus voor het probleem "de ___ kat" , is onze input misschien {914, 3542}. Hoe moet de output zijn? Welnu, het zou een lijst moeten zijn van ongeveer 50.000 getallen die effectief de kansen geven voor elk van de mogelijke "invulwoorden". En nogmaals, om een inbedding te vinden, willen we de "binnenkant" van het neurale net "onderscheppen" net voordat het "zijn conclusie bereikt" - en dan de lijst met getallen oppakken die daar voorkomen en die we kunnen bedenken als "kenmerk van elk woord".

OK, dus hoe zien die karakteriseringen eruit? In de afgelopen 10 jaar is er een opeenvolging van verschillende systemen ontwikkeld (word2vec , GloVe , BERT , GPT , ...), elk gebaseerd op een andere neurale netbenadering. Maar uiteindelijk nemen ze allemaal woorden en karakteriseren ze ze door lijsten van honderden tot duizenden getallen.

In hun onbewerkte vorm zijn deze "inbeddingsvectoren" vrij weinig informatief. Dit is bijvoorbeeld wat GPT-2 produceert als de onbewerkte inbeddingsvectoren voor drie specifieke woorden:



cat dog chair

Als we dingen doen zoals het meten van afstanden tussen deze vectoren, dan kunnen we dingen vinden zoals "nabijheid" van woorden. Later zullen we in meer detail bespreken wat we zouden kunnen beschouwen als de "cognitieve" betekenis van dergelijke inbeddingen. Maar voor nu is het belangrijkste punt dat we een manier hebben om woorden op een nuttige manier om te zetten in "neuraal-netvriendelijke" verzamelingen getallen.

Maar eigenlijk kunnen we verder gaan dan alleen het karakteriseren van woorden door getallenverzamelingen; we kunnen dit ook doen voor reeksen woorden, of zelfs hele tekstblokken. En binnen ChatGPT gaat het zo met dingen om. Het neemt de tekst die het tot nu toe heeft en genereert een inbeddingsvector om deze weer te geven. Vervolgens is het doel om de kansen te vinden voor verschillende woorden die daarna kunnen voorkomen. En het geeft zijn antwoord hierop weer als een lijst met getallen die in wezen de kansen geven voor elk van de ongeveer 50.000 mogelijke woorden.

(Strikt, ChatGPT werkt niet met woorden, maar met "tokens" -handige taalkundige eenheden die hele woorden kunnen zijn, of misschien alleen stukjes zoals "pre" of "ing" of "ized". Werken met tokens maakt het gemakkelijker voor ChatGPT om zeldzame, samengestelde en niet-Engelse woorden te verwerken, en soms, voor beter of slechter, om nieuwe woorden te verzinnen.)

Binnen ChatGPT

OK, dus we zijn eindelijk klaar om te bespreken wat er in ChatGPT zit. En ja,

uiteindelijk is het een gigantisch neuraal net - momenteel een versie van het zogenaamde GPT-3-netwerk met 175 miljard gewichten. In veel opzichten is dit een neuraal netwerk dat veel lijkt op de andere die we hebben besproken. Maar het is een neuraal net dat speciaal is opgezet om met taal om te gaan. En het meest opvallende kenmerk is een stuk neurale netarchitectuur dat een "transformator" wordt genoemd.

In de eerste neurale netwerken die we hierboven hebben besproken, was elk neuron op een bepaalde laag in principe verbonden (althans met enig gewicht) met elk neuron op de laag ervoor. Maar zo'n volledig verbonden netwerk is (vermoedelijk) overdreven als je werkt met gegevens die een bepaalde, bekende structuur hebben. En dus is het bijvoorbeeld in de vroege stadia van het omgaan met afbeeldingen typisch om zogenaamde convolutionele neurale netten ("convnets") te gebruiken waarin neuronen effectief op een raster worden gelegd analoog aan de pixels in de afbeelding - en verbonden alleen naar neuronen dichtbij op het rooster.

Het idee van transformatoren is om iets vergelijkbaars te doen voor reeksen tokens waaruit een stuk tekst bestaat. Maar in plaats van alleen een vast gebied in de reeks te definiëren waarover verbindingen kunnen zijn, introduceren transformatoren in plaats daarvan het begrip " aandacht " - en het idee om meer "aandacht te schenken" aan sommige delen van de reeks dan aan andere. Misschien is het op een dag logisch om gewoon een generiek neuraal netwerk te starten en alle aanpassingen door middel van training te doen. Maar vanaf nu lijkt het in de praktijk van cruciaal belang om dingen te 'modulariseren' - zoals transformatoren doen, en waarschijnlijk ook onze hersenen.

OK, dus wat doet ChatGPT (of beter gezegd het GPT-3-netwerk waarop het is gebaseerd) eigenlijk? Bedenk dat het algemene doel is om tekst op een "redelijke" manier voort te zetten, gebaseerd op wat het heeft gezien tijdens de training die het heeft gehad (die bestaat uit het bekijken van miljarden pagina's tekst van internet, enz.). Dus op een gegeven moment, het heeft een bepaalde hoeveelheid tekst - en het doel is om een geschikte keuze te maken voor het volgende toe te voegen token.

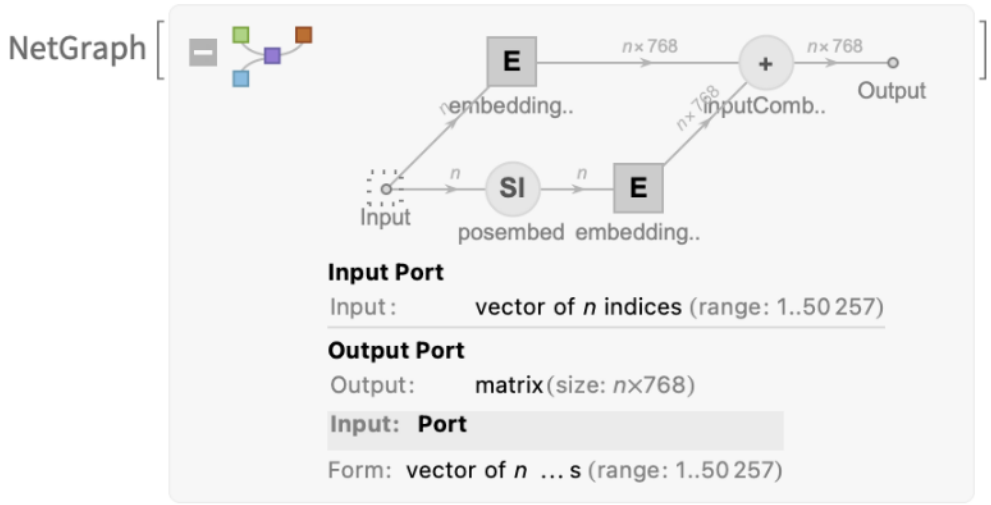
Het werkt in drie basisfasen. Eerst neemt het de reeks tokens die overeenkomt met de tekst tot nu toe, en vindt een inbedding (dwz een reeks getallen) die deze vertegenwoordigt. Vervolgens werkt het op deze inbedding - op een "standaard neurale netmanier", met waarden die door opeenvolgende lagen in een netwerk "kabbelen" - om een nieuwe inbedding te produceren (dwz een nieuwe reeks getallen). Vervolgens neemt het het laatste deel van deze array en genereert daaruit een array van ongeveer 50.000 waarden die veranderen in waarschijnlijkheden voor verschillende mogelijke volgende tokens. (En ja, het gebeurt zo dat er ongeveer evenveel tokens worden gebruikt als gewone woorden in het Engels, hoewel slechts ongeveer 3000 van de tokens hele woorden zijn en de rest fragmenten.)

Een cruciaal punt is dat elk onderdeel van deze pijplijn wordt geïmplementeerd door een neuraal netwerk, waarvan de gewichten worden bepaald door end-to-end training van het netwerk. Met andere woorden, in feite is niets anders dan de algehele architectuur "expliciet ontworpen"; alles is gewoon "geleerd" uit trainingsgegevens.

Er zijn echter tal van details in de manier waarop de architectuur is opgezet, die allerlei soorten ervaringen en neurale netwerkkennis weerspiegelen. En hoewel dit zeker in het onkruid verdwijnt, denk ik dat het nuttig is om over enkele van die details

te praten, niet in de laatste plaats om een idee te krijgen van wat er allemaal komt kijken bij het bouwen van zoiets als ChatGPT.

Eerst komt de inbeddingsmodule. Hier is een schematische Wolfram-taalweergave ervoor voor GPT-2:

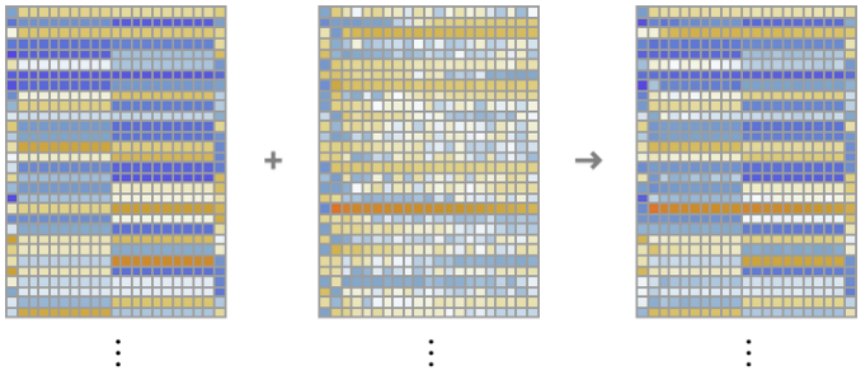


De invoer is een vector van *n* tokens (weergegeven zoals in de vorige sectie door gehele getallen van 1 tot ongeveer 50.000). Elk van deze tokens wordt omgezet (door een enkellaags neuraal net) in een inbeddingsvector (met een lengte van 768 voor GPT-2 en 12.288 voor ChatGPT's GPT-3). Ondertussen is er een "secundair pad" dat de volgorde van (gehele) posities voor de tokens neemt, en van deze gehele getallen een andere inbeddingsvector creëert. En tot slot worden de inbeddingsvectoren van de tokenwaarde en de tokenpositie bij elkaar opgeteld om de uiteindelijke reeks inbeddingsvectoren uit de inbeddingsmodule te produceren.

Waarom voegt men gewoon de inbeddingsvectoren tokenwaarde en tokenpositie bij

elkaar op? Ik denk niet dat hier enige specifieke wetenschap voor is. Het is gewoon dat er verschillende dingen zijn geprobeerd, en dit is er een die lijkt te werken. En het maakt deel uit van de overlevering van neurale netwerken dat - in zekere zin - zolang de opstelling die men heeft "ongeveer goed" is, het meestal mogelijk is om op details in te gaan door gewoon voldoende training te doen, zonder ooit echt "op een gegeven moment" te hoeven begrijpen. engineering-niveau" vrij hoe het neurale net zichzelf uiteindelijk heeft geconfigureerd.

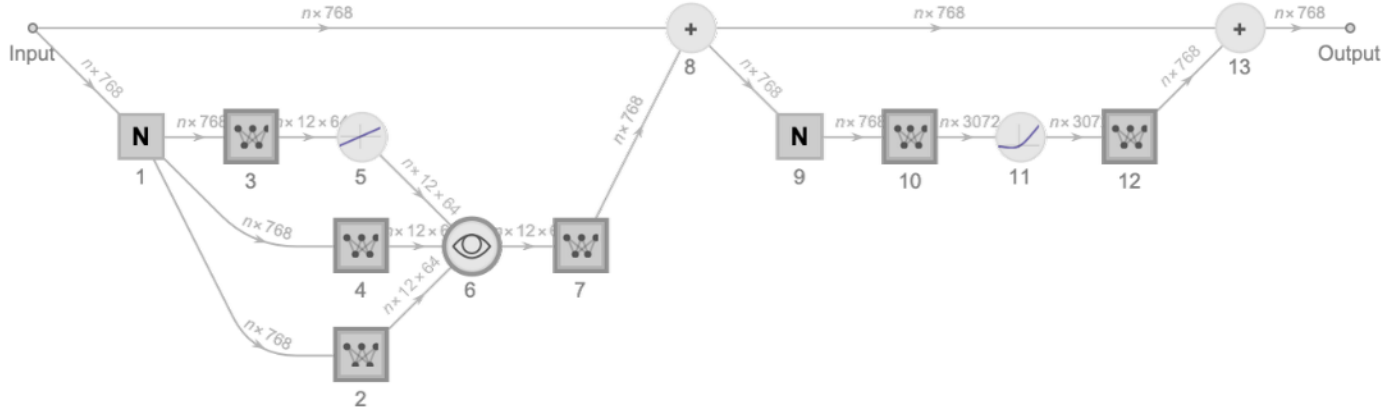
Dit is wat de inbeddingsmodule doet, werkend op de string *hallo hallo hallo hallo hallo hallo hallo hallo hallo bye bye bye bye bye bye bye bye bye bye* :



De elementen van de inbeddingsvector voor elk token worden onderaan de pagina weergegeven en over de pagina zien we eerst een reeks " *hallo* " inbeddingen, gevolgd door een reeks " *doei* ". De tweede array hierboven is de positionele inbedding, waarbij de enigszins willekeurig ogende structuur precies is wat "toevallig is geleerd" (in dit geval in GPT-2).

OK, dus na de inbeddingsmodule komt de "main event" van de transformator: een opeenvolging van zogenaamde "attention blocks" (12 voor GPT-2, 96 voor ChatGPT's

GPT-3). Het is allemaal behoorlijk gecompliceerd - en doet denken aan typische grote, moeilijk te begrijpen technische systemen, of wat dat betreft biologische systemen. Maar hoe dan ook, hier is een schematische weergave van een enkel "aandachtsblok" (voor GPT-2):

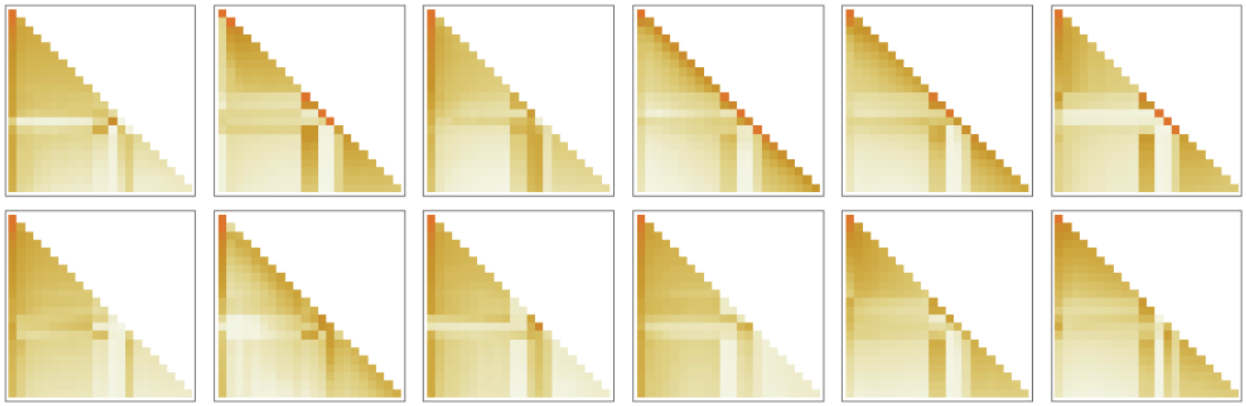


Binnen elk dergelijk aandachtsblok is er een verzameling "aandachtshoofden" (12 voor GPT-2, 96 voor ChatGPT's GPT-3) - die allemaal onafhankelijk werken op verschillende waardeblokken in de inbeddingsvector. (En ja, we weten geen specifieke reden waarom het een goed idee is om de inbeddingsvector op te splitsen, of wat de verschillende delen ervan "betekenen"; dit is slechts een van die dingen die "werken" .)

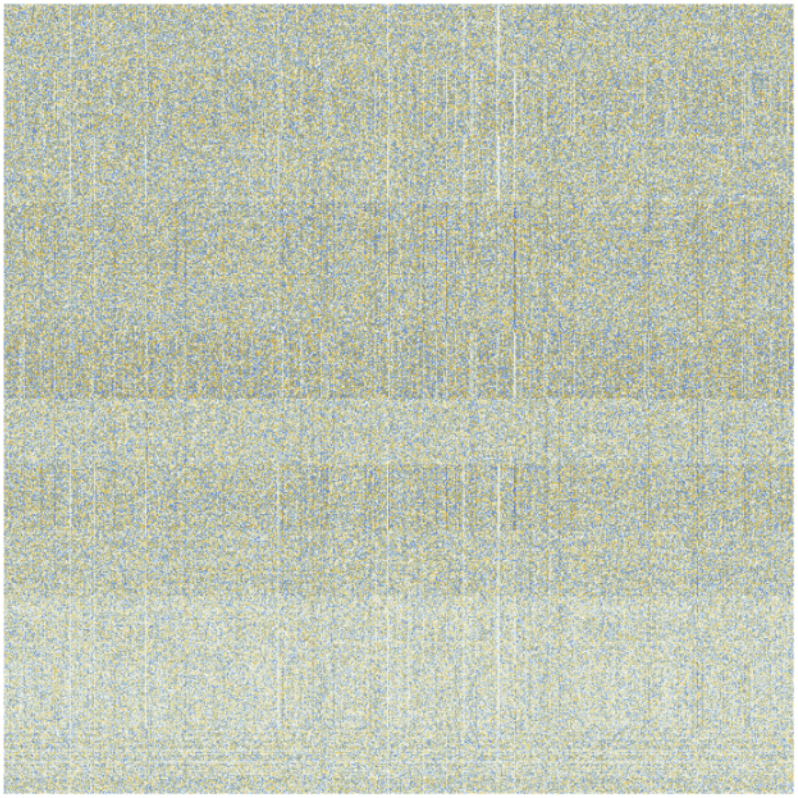
Oké, dus wat doen de aandachtshoofden? In wezen zijn ze een manier om "terug te kijken" in de volgorde van tokens (dwz in de tot nu toe geproduceerde tekst), en "het verleden te verpakken" in een vorm die handig is om het volgende token te vinden. In het eerste gedeelte hierboven hebben we het gehad over het gebruik van kansen van 2 gram om woorden te kiezen op basis van hun directe voorgangers. Wat het

"aandachtsmechanisme" in transformatoren doet, is om zelfs veel eerdere woorden "aandacht" te geven, waardoor mogelijk wordt vastgelegd hoe bijvoorbeeld werkwoorden kunnen verwijzen naar zelfstandige naamwoorden die veel woorden ervoor in een zin voorkomen.

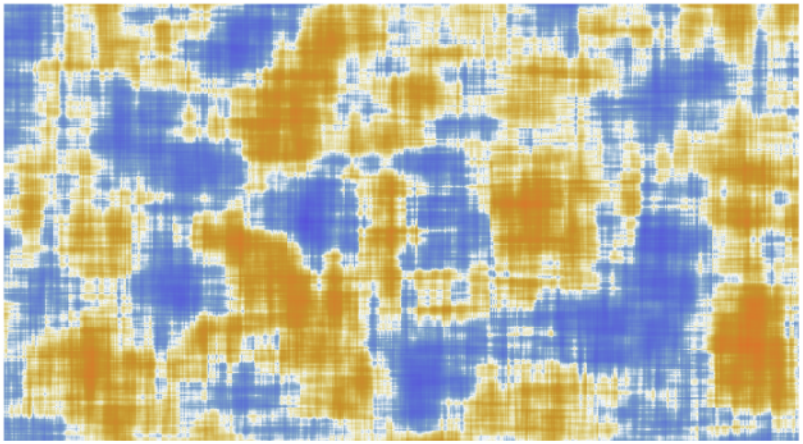
Op een meer gedetailleerd niveau, wat een aandachtshoofd doet, is het opnieuw combineren van brokken in de inbeddingsvectoren die zijn gekoppeld aan verschillende tokens, met bepaalde gewichten. En dus hebben de 12 aandachtshoofden in het eerste aandachtsblok (in GPT-2) bijvoorbeeld het volgende (“terugkijken-helemaal-naar-het-begin-van-de-reeks-van- tokens") patronen van "recombinatiegewichten" voor de bovenstaande string " *hallo , dag ":*

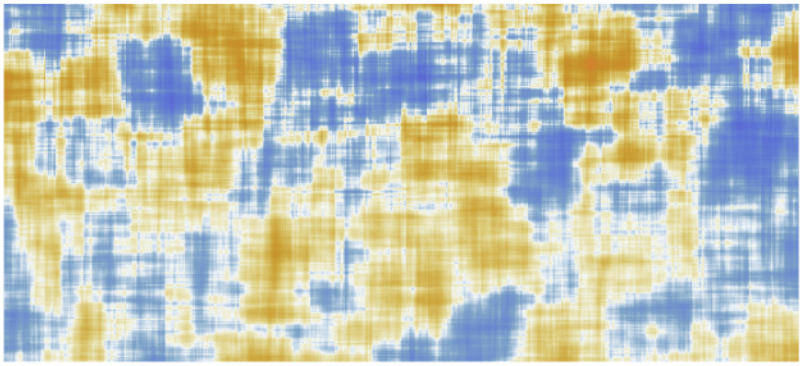


Na te zijn verwerkt door de aandachtshoofden, wordt de resulterende "opnieuw gewogen inbeddingsvector" (met een lengte van 768 voor GPT-2 en een lengte van 12.288 voor ChatGPT's GPT-3) door een standaard "volledig verbonden" neurale netlaag geleid . Het is moeilijk om grip te krijgen op wat deze laag doet. Maar hier is een plot van de 768 × 768 matrix van gewichten die het gebruikt (hier voor GPT-2):



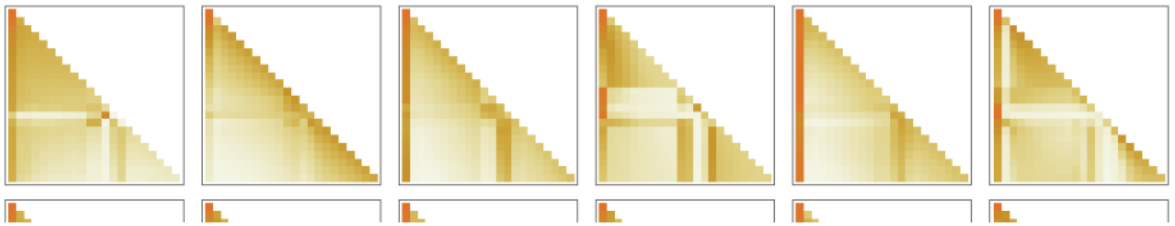
Als we voortschrijdende gemiddelden van 64 × 64 nemen, begint er een (random-walk-achtige) structuur te ontstaan:

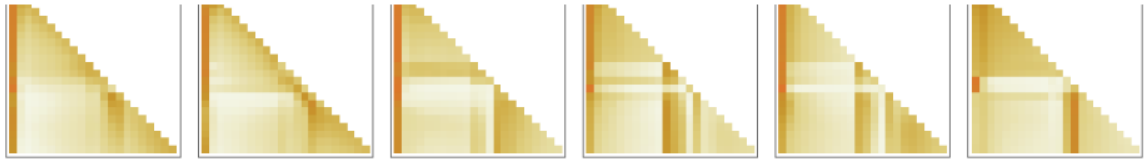




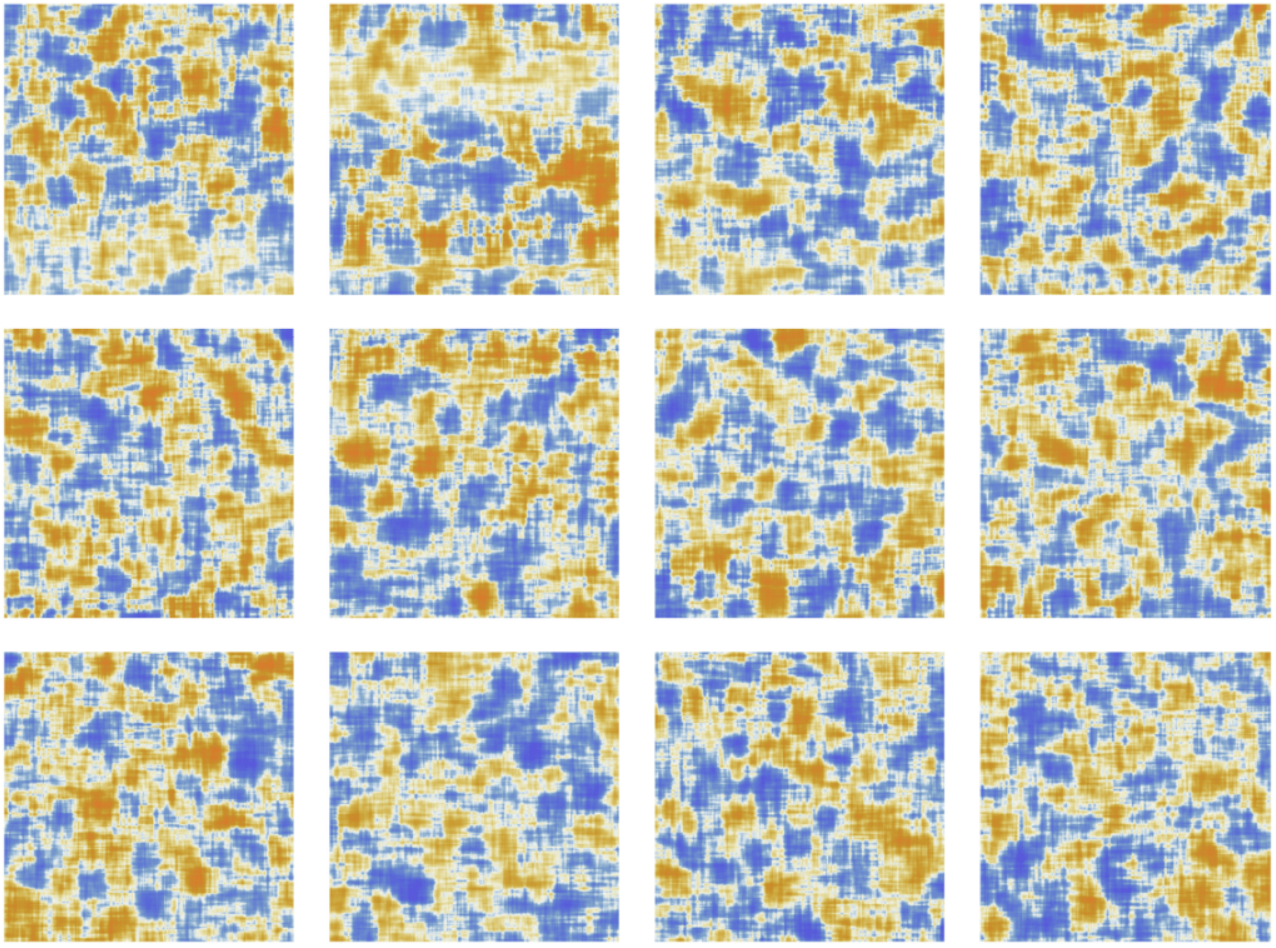
Wat bepaalt deze structuur? Uiteindelijk is het vermoedelijk een soort 'neurale netcodering' van kenmerken van de menselijke taal. Maar vanaf nu is het vrij onbekend wat die functies zouden kunnen zijn. In feite "openen we het brein van ChatGPT" (of in ieder geval GPT-2) en ontdekken we, ja, het is ingewikkeld daarbinnen, en we begrijpen het niet - ook al produceert het uiteindelijk herkenbare menselijke taal .

OK, dus nadat we één aandachtsblok hebben doorlopen, hebben we een nieuwe inbeddingsvector, die vervolgens achtereenvolgens door aanvullende aandachtsblokken wordt geleid (in totaal 12 voor GPT-2; 96 voor GPT-3). Elk aandachtsblok heeft zijn eigen specifieke patroon van "aandacht" en "volledig verbonden" gewichten. Hier voor GPT-2 is de volgorde van aandachtsgewichten voor de "hallo, dag"-invoer, voor de eerste aandachtskop:



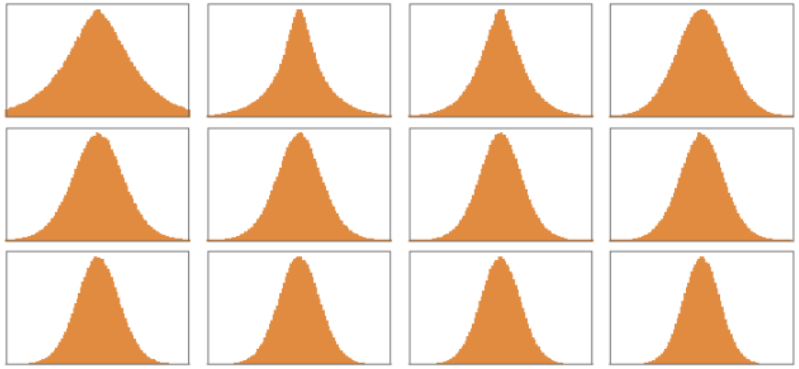


En hier zijn de (voortschrijdend gemiddelde) "matrices" voor de volledig verbonden lagen:



Vreemd genoeg, hoewel deze "matrices van gewichten" in verschillende aandachtsblokken behoorlijk op elkaar lijken, kunnen de verdelingen van de groottes

van gewichten enigszins verschillen (en zijn niet altijd Gaussiaans):



Dus wat is het netto effect van de transformator na het doorlopen van al deze aandachtsblokken? In wezen is het om de oorspronkelijke verzameling inbeddingen voor de reeks tokens om te zetten in een definitieve verzameling. En de specifieke manier waarop ChatGPT werkt, is om vervolgens de laatste inbedding in deze verzameling op te pikken en deze te "decoderen" om een lijst met waarschijnlijkheden te produceren voor welk token daarna zou moeten komen.

Dus dat is in grote lijnen wat er in ChatGPT zit. Het lijkt misschien ingewikkeld (niet in de laatste plaats vanwege de vele onvermijdelijk enigszins willekeurige 'technische keuzes'), maar in feite zijn de uiteindelijke elementen opmerkelijk eenvoudig. Want waar we uiteindelijk mee te maken hebben, is slechts een neuraal netwerk gemaakt van 'kunstmatige neuronen', die elk de simpele handeling uitvoeren van het nemen van een verzameling numerieke invoer en deze vervolgens combineren met bepaalde gewichten.

De oorspronkelijke invoer voor ChatGPT is een reeks getallen (de inbeddingsvectoren voor de tokens tot nu toe), en wat er gebeurt wanneer ChatGPT "draait" om een nieuw

token te produceren, is dat deze getallen door de lagen van het neurale net "rimpelen". waarbij elk neuron "zijn ding doet" en het resultaat doorgeeft aan neuronen op de volgende laag. Er is geen lus of "teruggaan". Alles "feedforward" gewoon door het netwerk.

Het is een heel andere opzet dan een typisch rekensysteem, zoals een Turing-machine , waarin resultaten herhaaldelijk worden 'opnieuw verwerkt' door dezelfde rekenelementen. Hier wordt - althans bij het genereren van een gegeven token van output - elk computationeel element (dwz neuron) slechts één keer gebruikt.

Maar er is in zekere zin nog steeds een "buitenste lus" die computationele elementen hergebruikt, zelfs in ChatGPT. Omdat wanneer ChatGPT een nieuw token gaat genereren, het altijd de hele reeks tokens die eraan voorafgaan "leest" (dwz als invoer neemt), inclusief tokens die ChatGPT zelf eerder heeft "geschreven". En we kunnen deze opstelling beschouwen als een betekenis dat ChatGPT - althans op het buitenste niveau - een "feedbacklus" omvat, zij het een waarin elke iteratie expliciet zichtbaar is als een token dat verschijnt in de tekst die het genereert.

Maar laten we teruggaan naar de kern van ChatGPT: het neurale netwerk dat herhaaldelijk wordt gebruikt om elk token te genereren. Op een bepaald niveau is het heel eenvoudig: een hele verzameling identieke kunstmatige neuronen. En sommige delen van het netwerk bestaan gewoon uit (" volledig verbonden ") lagen neuronen waarin elk neuron op een bepaalde laag (met enig gewicht) is verbonden met elk neuron op de laag ervoor. Maar vooral met zijn transformatorarchitectuur heeft ChatGPT delen met meer structuur, waarin alleen specifieke neuronen op

verschillende lagen met elkaar verbonden zijn. (Je zou natuurlijk nog steeds kunnen zeggen dat "alle neuronen met elkaar verbonden zijn" - maar sommige hebben gewoon geen gewicht.)

Daarnaast zijn er aspecten van het neurale net in ChatGPT waarvan niet wordt gedacht dat ze gewoon uit "homogene" lagen bestaan. En bijvoorbeeld - zoals de iconische samenvatting hierboven aangeeft - zijn er binnen een aandachtsblok plaatsen waar "meerdere kopieën worden gemaakt" van inkomende gegevens, die elk vervolgens een ander "verwerkingspad" doorlopen, mogelijk met een ander aantal lagen, en alleen later opnieuw combineren. Maar hoewel dit misschien een handige weergave is van wat er aan de hand is, is het in principe altijd mogelijk om te denken aan "dicht opvullende" lagen, maar gewoon een paar gewichten nul te hebben.

Als je kijkt naar het langste pad door ChatGPT, zijn er ongeveer 400 (kern)lagen bij betrokken - in sommige opzichten geen enorm aantal. Maar er zijn miljoenen neuronen – met in totaal 175 miljard verbindingen en dus 175 miljard gewichten. En een ding om te beseffen is dat elke keer dat ChatGPT een nieuw token genereert, het een berekening moet maken waarbij elk van deze gewichten betrokken is. In de praktijk kunnen deze berekeningen enigszins "per laag" worden georganiseerd in zeer parallelle array-bewerkingen die gemakkelijk op GPU's kunnen worden uitgevoerd. Maar voor elke geproduceerde token moeten er nog steeds 175 miljard berekeningen worden uitgevoerd (en uiteindelijk een beetje meer) - dus ja, het is niet verwonderlijk dat het even kan duren om een lang stuk tekst te genereren met ChatGPT.

Maar het opmerkelijke is uiteindelijk dat al deze bewerkingen - hoe eenvoudig ze ook zijn afzonderlijk - op de een of andere manier samen zo'n goed 'menselijk' werk

kunnen doen om tekst te genereren. Het moet nogmaals worden benadrukt dat er (althans voor zover we weten) geen "ultieme theoretische reden" is waarom zoiets zou moeten werken. En in feite, zoals we zullen bespreken, denk ik dat we dit moeten zien als een - mogelijk verrassende - wetenschappelijke ontdekking: dat het op de een of andere manier in een neuraal net als dat van ChatGPT mogelijk is om de essentie vast te leggen van wat de menselijke hersenen doen bij het genereren van taal. .

De training van ChatGPT

OK, dus we hebben nu een overzicht gegeven van hoe ChatGPT werkt als het eenmaal is ingesteld. Maar hoe is het opgezet? Hoe werden al die 175 miljard gewichten in zijn neurale net bepaald? In wezen zijn ze het resultaat van zeer grootschalige training, gebaseerd op een enorm corpus tekst - op internet, in boeken, enz. - geschreven door mensen. Zoals we al zeiden, is het, zelfs gezien al die trainingsgegevens, zeker niet vanzelfsprekend dat een neuraal netwerk in staat zou zijn om met succes "mensachtige" tekst te produceren. En nogmaals, er lijken gedetailleerde stukjes techniek nodig te zijn om dat mogelijk te maken. Maar de grote verrassing - en ontdekking - van ChatGPT is dat het überhaupt mogelijk is. En dat - in feite - een neuraal net met "slechts" 175 miljard gewichten een "redelijk model" kan maken van tekst die mensen schrijven.

In moderne tijden is er veel tekst geschreven door mensen die in digitale vorm beschikbaar is. Het openbare web heeft minstens enkele miljard door mensen geschreven pagina's, met in totaal misschien wel een biljoen woorden tekst. En als

men niet-openbare webpagina's meetelt, kunnen de aantallen minstens 100 keer groter zijn. Tot nu toe zijn er meer dan 5 miljoen gedigitaliseerde boeken beschikbaar gesteld (van de ongeveer 100 miljoen die ooit zijn gepubliceerd), goed voor nog eens 100 miljard woorden aan tekst. En dan heb ik het nog niet eens over tekst die is afgeleid van spraak in video's, enz. (Ter vergelijking: mijn totale levensproductie van gepubliceerd materiaal is iets minder dan 3 miljoen woorden, en in de afgelopen 30 jaar heb ik geschrevenongeveer 15 miljoen woorden aan e-mail, en in totaal misschien 50 miljoen woorden getypt - en in de afgelopen paar jaar heb ik meer dan 10 miljoen woorden gesproken op livestreams . En ja, van dat alles zal ik een bot trainen.)

Maar, oké, gezien al deze gegevens, hoe train je er een neuraal netwerk van? Het basisproces is grotendeels zoals we het in de eenvoudige voorbeelden hierboven hebben besproken. U presenteert een reeks voorbeelden en past vervolgens de gewichten in het netwerk aan om de fout ("verlies") die het netwerk op die voorbeelden maakt, te minimaliseren. Het belangrijkste dat duur is aan het 'terugpropaguleren' van de fout, is dat elke keer dat u dit doet, elk gewicht in het netwerk doorgaans op zijn minst een klein beetje zal veranderen, en dat er gewoon veel gewichten zijn om mee om te gaan. (De werkelijke "achterwaartse berekening" is doorgaans slechts een kleine constante factor moeilijker dan de voorwaartse.)

Met moderne GPU-hardware is het eenvoudig om de resultaten van batches van duizenden voorbeelden parallel te berekenen. Maar als het gaat om het daadwerkelijk bijwerken van de gewichten in het neurale net, vereisen de huidige methoden dat dit in principe batchgewijs wordt gedaan. (En ja, dit is waarschijnlijk waar echte hersenen - met hun gecombineerde berekenings- en geheugenelementen - voorlopig op zijn minst een architectonisch voordeel hebben.)

Zelfs in de ogenschijnlijk eenvoudige gevallen van het leren van numerieke functies die we eerder bespraken, merkten we dat we vaak miljoenen voorbeelden moesten gebruiken om een netwerk met succes te trainen, in ieder geval helemaal opnieuw. Dus hoeveel voorbeelden betekent dit dat we nodig hebben om een 'menselijk taalmodel' te trainen? Er lijkt geen fundamentele "theoretische" manier te zijn om dit te weten. Maar in de praktijk werd ChatGPT met succes getraind op een paar honderd miljard woorden tekst.

Een deel van de tekst werd meerdere keren ingevoerd, een deel slechts één keer. Maar op de een of andere manier "kreeg het wat het nodig had" uit de tekst die het zag. Maar gezien deze hoeveelheid tekst om van te leren, hoeveel netwerk zou het dan nodig hebben om het "goed te leren"? Nogmaals, we hebben nog geen fundamentele theoretische manier om te zeggen. Uiteindelijk - zoals we hieronder verder zullen bespreken - is er waarschijnlijk een bepaalde 'totale algoritmische inhoud' voor menselijke taal en wat mensen er doorgaans mee zeggen. Maar de volgende vraag is hoe efficiënt een neuraal net zal zijn bij het implementeren van een model op basis van die algoritmische inhoud. En nogmaals, we weten het niet, hoewel het succes van ChatGPT suggereert dat het redelijk efficiënt is.

En uiteindelijk kunnen we gewoon opmerken dat ChatGPT doet wat het doet met een paar honderd miljard gewichten - vergelijkbaar in aantal met het totale aantal woorden (of tokens) aan trainingsgegevens dat het heeft gekregen. In sommige opzichten is het misschien verrassend (hoewel empirisch ook waargenomen in kleinere analogen van ChatGPT) dat de "omvang van het netwerk" dat goed lijkt te

werken, zo vergelijkbaar is met de "omvang van de trainingsgegevens". Het is immers zeker niet zo dat op de een of andere manier 'binnen ChatGPT' al die tekst van internet en boeken enzovoort 'direct wordt opgeslagen'. Want wat er eigenlijk in ChatGPT zit, zijn een aantal getallen - met een nauwkeurigheid van iets minder dan 10 cijfers - die een soort gedistribueerde codering zijn van de geaggregeerde structuur van al die tekst.

Anders gezegd, we zouden ons kunnen afvragen wat de 'effectieve informatie-inhoud' is van menselijke taal en wat er doorgaans mee wordt gezegd. Er is het ruwe corpus van voorbeelden van taal. En dan is er nog de representatie in het neurale net van ChatGPT. Die representatie is zeer waarschijnlijk verre van de "algoritmisch minimale" representatie (zoals we hieronder zullen bespreken). Maar het is een representatie die gemakkelijk bruikbaar is voor het neurale net. En in deze weergave lijkt er uiteindelijk vrij weinig "compressie" van de trainingsgegevens te zijn; het lijkt gemiddeld slechts iets minder dan één neuraal netgewicht nodig te hebben om de "informatie-inhoud" van een woord aan trainingsgegevens te dragen.

Wanneer we ChatGPT gebruiken om tekst te genereren, moeten we in feite elk gewicht één keer gebruiken. Dus als er n gewichten zijn, moeten we n rekenkundige stappen uitvoeren, hoewel in de praktijk veel ervan parallel kunnen worden uitgevoerd in GPU's. Maar als we ongeveer n woorden aan trainingsgegevens nodig hebben om die gewichten in te stellen, dan kunnen we uit wat we hierboven hebben gezegd concluderen dat we ongeveer n^2 rekenstappen nodig hebben om de training van het netwerk uit te voeren - en daarom, met Met de huidige methoden moet je uiteindelijk praten over trainingsinspanningen van miljarden dollars.

Buiten de basisopleiding

Het grootste deel van de inspanning bij het trainen van ChatGPT wordt besteed aan het 'laten zien' van grote hoeveelheden bestaande tekst van internet, boeken, enz. Maar het blijkt dat er nog een ander - schijnbaar nogal belangrijk - onderdeel is.

Zodra de "onbewerkte training" van het originele tekstcorpus is voltooid, is het neurale net binnen ChatGPT klaar om zijn eigen tekst te genereren, voortgaand op prompts, enz. Maar hoewel de resultaten hiervan vaak redelijk lijken , hebben ze de neiging - vooral bij langere stukken tekst - om op vaak niet-menselijke manieren te 'afdwalen'. Het is niet iets dat men gemakkelijk kan ontdekken, bijvoorbeeld door traditionele statistieken op de tekst te doen. Maar het is iets dat echte mensen die de tekst lezen gemakkelijk opmerken.

En een belangrijk idee bij de constructie van ChatGPT was om na het "passief lezen" van dingen zoals het web nog een stap te zetten: om echte mensen actief te laten communiceren met ChatGPT, te zien wat het oplevert en in feite feedback te geven over "hoe een goede chatbot". Maar hoe kan het neurale net die feedback gebruiken? De eerste stap is om mensen de resultaten van het neurale net te laten beoordelen. Maar dan wordt er een ander neuraal netmodel gebouwd dat probeert die beoordelingen te voorspellen. Maar nu kan dit voorspellingsmodel - in wezen als een verliesfunctie - worden uitgevoerd op het oorspronkelijke netwerk, waardoor dat netwerk in feite kan worden "afgesteld" door de menselijke feedback die is gegeven. En de resultaten in de praktijk lijken een groot effect te hebben op het succes van het systeem bij het produceren van "mensachtige" output.

Over het algemeen is het interessant hoe weinig het "oorspronkelijk getrainde" netwerk nodig lijkt te hebben om het in bepaalde richtingen te laten gaan. Men zou kunnen denken dat om het netwerk zich te laten gedragen alsof het "iets nieuws heeft geleerd", men naar binnen zou moeten gaan en een trainingsalgoritme zou moeten uitvoeren, gewichten zou moeten aanpassen, enzovoort.

Maar dat is niet het geval. In plaats daarvan lijkt het voldoende om ChatGPT in feite een keer iets te vertellen - als onderdeel van de prompt die je geeft - en dan kan het met succes gebruik maken van wat je het hebt verteld wanneer het tekst genereert. En nogmaals, het feit dat dit werkt, is volgens mij een belangrijke aanwijzing om te begrijpen wat ChatGPT "echt doet" en hoe het zich verhoudt tot de structuur van de menselijke taal en denken.

Het heeft zeker iets menselijks: dat je het, als het eenmaal al die voortraining heeft gehad, tenminste één keer iets kunt vertellen en het kan "onthouden" - in ieder geval "lang genoeg" om er een stuk tekst mee te genereren . Dus wat is er aan de hand in een geval als dit? Het kan zijn dat "alles wat je zou kunnen vertellen al ergens in zit" - en je leidt het gewoon naar de juiste plek. Maar dat lijkt niet aannemelijk. Wat in plaats daarvan waarschijnlijker lijkt, is dat, ja, de elementen er al in zitten, maar de details worden bepaald door zoiets als een "traject tussen die elementen" en dat is wat je introduceert als je het iets vertelt.

En inderdaad, net als bij mensen, als je het iets bizars en onverwachts vertelt dat totaal niet past in het kader dat het kent, lijkt het er niet op dat het dit met succes zal

kunnen 'integreren'. Het kan het alleen "integreren" als het in feite op een vrij eenvoudige manier bovenop het raamwerk rijdt dat het al heeft.

Het is ook de moeite waard er nogmaals op te wijzen dat er onvermijdelijk "algoritmische limieten" zijn aan wat het neurale net kan "oppikken". Vertel het "oppervlakkige" regels van de vorm "dit gaat naar dat", enz., en het neurale net zal deze hoogstwaarschijnlijk prima kunnen weergeven en reproduceren - en inderdaad, wat het "al weet" uit de taal zal het een onmiddellijk patroon om te volgen. Maar probeer het regels te geven voor een daadwerkelijke "diepe" berekening die veel potentieel computationeel onherleidbare stappen met zich meebrengt, en het zal gewoon niet werken. (Onthoud dat het bij elke stap altijd gewoon "data forward" is in zijn netwerk, nooit in een lus, behalve door nieuwe tokens te genereren.)

Natuurlijk kan het netwerk het antwoord leren op specifieke "onherleidbare" berekeningen. Maar zodra er combinatorische getallen van mogelijkheden zijn, zal zo'n "tabel-lookup-stijl" benadering niet werken. En dus, ja, net als bij mensen, wordt het tijd dat neurale netwerken "uitreiken" en echte rekenhulpmiddelen gebruiken. (En ja, Wolfram|Alpha en Wolfram Language zijn bij uitstek geschikt , omdat ze zijn gebouwd om "over dingen in de wereld te praten", net als de neurale netwerken van het taalmodel.)

Wat laat ChatGPT echt werken?

Menselijke taal - en de denkprocessen die betrokken zijn bij het genereren ervan - leken altijd een soort toppunt van complexiteit te vertegenwoordigen. En inderdaad, het leek enigszins opmerkelijk dat de menselijke hersenen – met hun netwerk van

"slechts" 100 miljard neuronen (en misschien wel 100 biljoen verbindingen) hiervoor verantwoordelijk zouden kunnen zijn. Misschien, zou je je kunnen voorstellen, is er iets meer in de hersenen dan hun netwerken van neuronen - zoals een nieuwe laag onontdekte fysica. Maar nu met ChatGPT hebben we een belangrijk nieuw stuk informatie: we weten dat een puur, kunstmatig neuraal netwerk met ongeveer evenveel verbindingen als hersenen neuronen hebben, in staat is om verrassend goed menselijke taal te genereren.

En ja, dat is nog steeds een groot en gecompliceerd systeem - met ongeveer evenveel neurale netgewichten als er momenteel beschikbare tekst in de wereld is. Maar op een bepaald niveau lijkt het nog steeds moeilijk te geloven dat alle rijkdom van taal en de dingen waarover het kan praten, in zo'n eindig systeem kan worden ingekapseld. Een deel van wat er aan de hand is, is ongetwijfeld een weerspiegeling van het alomtegenwoordige fenomeen (dat voor het eerst duidelijk werd in het voorbeeld van regel 30) dat computationele processen de schijnbare complexiteit van systemen in feite aanzienlijk kunnen versterken, zelfs als hun onderliggende regels eenvoudig zijn. Maar, zoals we hierboven hebben besproken, zijn neurale netwerken van het soort dat in ChatGPT wordt gebruikt, in feite specifiek geconstrueerd om het effect van dit fenomeen - en de computationele onherleidbaarheid die ermee gepaard gaat - te beperken om hun training toegankelijker te maken.

Dus hoe komt het dan dat zoiets als ChatGPT zo ver kan komen als met taal? Het basisantwoord, denk ik, is dat taal op een fundamenteel niveau op de een of andere manier eenvoudiger is dan het lijkt. En dit betekent dat ChatGPT - zelfs met zijn uiteindelijk ongecompliceerde neurale netstructuur - met succes in staat is om "de

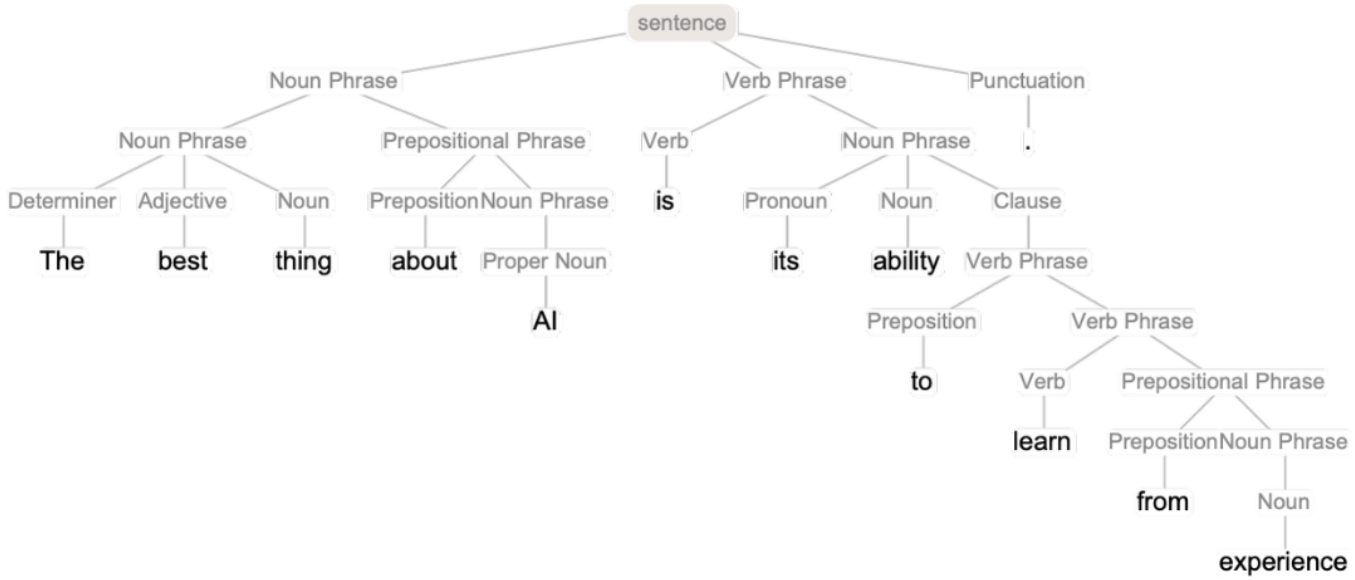
essentie vast te leggen" van menselijke taal en het denken erachter. En bovendien heeft ChatGPT tijdens zijn training op de een of andere manier "impliciet ontdekt" welke regelmaat in taal (en denken) dit mogelijk maakt.

Het succes van ChatGPT geeft ons, denk ik, het bewijs van een fundamenteel en belangrijk stuk wetenschap: het suggereert dat we kunnen verwachten dat er belangrijke nieuwe "taalwetten" - en in feite "wetten van het denken" - zijn om te ontdekken . In ChatGPT - gebouwd als een neuraal netwerk - zijn die wetten op zijn best impliciet. Maar als we de wetten op de een of andere manier expliciet zouden kunnen maken, bestaat de mogelijkheid om het soort dingen dat ChatGPT doet op veel directere, efficiëntere en transparantere manieren te doen.

Maar, OK, dus hoe zouden deze wetten eruit kunnen zien? Uiteindelijk moeten ze ons een soort recept geven voor hoe taal - en de dingen die we ermee zeggen - in elkaar zit. Later zullen we bespreken hoe "kijken in ChatGPT" ons hier enkele hints over kan geven, en hoe wat we weten uit het bouwen van computationele taal een pad voorwaarts suggereert. Maar laten we eerst twee lang bekende voorbeelden bespreken van wat neerkomt op 'taalwetten' en hoe ze verband houden met de werking van ChatGPT.

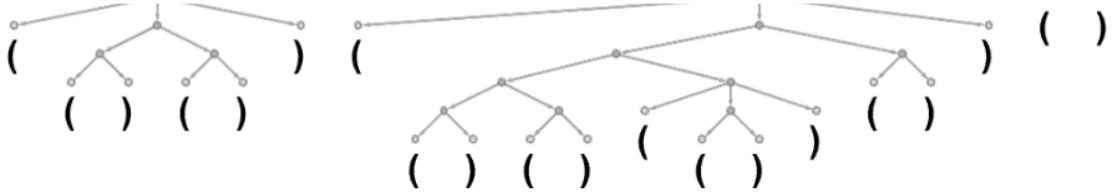
De eerste is de syntaxis van taal. Taal is niet zomaar een wirwar van woorden. In plaats daarvan zijn er (redelijk) duidelijke grammaticale regels voor hoe woorden van verschillende soorten kunnen worden samengesteld: in het Engels kunnen zelfstandige naamwoorden bijvoorbeeld worden voorafgegaan door bijvoeglijke naamwoorden en gevolgd door werkwoorden, maar meestal kunnen twee zelfstandige

naamwoorden niet direct naast elkaar staan ander. Een dergelijke grammaticale structuur kan (althans bij benadering) worden vastgelegd door een reeks regels die bepalen hoe hoeveel "bomen ontleden" kunnen worden samengesteld :



ChatGPT heeft geen expliciete "kennis" van dergelijke regels. Maar op de een of andere manier 'ontdekt' het ze tijdens zijn training impliciet - en lijkt het er dan goed in te zijn ze te volgen. Dus hoe werkt dit? Op een "big picture" niveau is het niet duidelijk. Maar om enig inzicht te krijgen is het misschien leerzaam om naar een veel eenvoudiger voorbeeld te kijken.

Beschouw een "taal" die is gevormd uit opeenvolgingen van ('s en)'s, met een grammatica die aangeeft dat haakjes altijd in evenwicht moeten zijn, zoals weergegeven door een ontleedboom zoals:



Kunnen we een neuraal net trainen om "grammaticaal correcte" haakjessequenties te produceren? Er zijn verschillende manieren om reeksen in neurale netwerken te verwerken, maar laten we transformatornetten gebruiken, zoals ChatGPT doet. En gegeven een eenvoudig transformatornet, kunnen we beginnen met het voeren van grammaticaal correcte haakjesreeksen als trainingsvoorbeelden. Een subtiliteit (die eigenlijk ook voorkomt in ChatGPT's generatie van menselijke taal) is dat we naast onze "content tokens" (hier "(" en ")") een "End" token moeten opnemen, die wordt gegenereerd om aan te geven dat de uitvoer zou niet verder moeten gaan (dwz voor ChatGPT, dat is het "einde van het verhaal").

Als we een transformatornet opzetten met slechts één aandachtsblok met 8 koppen en kenmerkvectoren met een lengte van 128 (ChatGPT gebruikt ook kenmerkvectoren met een lengte van 128, maar heeft 96 aandachtsblokken, elk met 96 koppen), dan lijkt het niet mogelijk om laat het veel leren over de taal tussen haakjes. Maar met 2 aandachtsblokken lijkt het leerproces samen te komen - in ieder geval nadat er ongeveer 10 miljoen voorbeelden zijn gegeven (en, zoals gebruikelijk bij transformatornetten, lijkt het tonen van nog meer voorbeelden de prestaties alleen maar te verslechteren).

Dus met dit netwerk kunnen we analoog doen aan wat ChatGPT doet, en vragen naar waarschijnlijkheden voor wat het volgende token zou moeten zijn - tussen haakjes:

((()()(

(	46%
)	54%
End	0.038%

((()()))

(	51%
)	15%
End	34%

En in het eerste geval is het netwerk "vrij zeker" dat de reeks hier niet kan eindigen - wat goed is, want als dat wel het geval zou zijn, zouden de haakjes onevenwichtig blijven. In het tweede geval "herkent" het echter correct dat de reeks hier kan eindigen, hoewel het ook "wijst" dat het mogelijk is om "opnieuw te beginnen", door een "(", vermoedelijk met een ")" te volgen . Maar oeps, zelfs met zijn ongeveer 400.000 moeizaam getrainde gewichten, zegt het dat er een kans van 15% is om ")" als het volgende token te hebben - wat niet juist is, want dat zou noodzakelijkerwijs leiden tot een onevenwichtig haakje.

Dit is wat we krijgen als we het netwerk vragen naar voltooiingen met de hoogste waarschijnlijkheid voor steeds langere reeksen van ('s:

```
( )
(( ))
((( )))
(((( ))))
((((( )))))
(((((( ))))))
((((((( )))))))
(((((((( ))))))))
((((((((( )))))))))
((((((((((  ))))))))))
((((((((((  ))))))))))()        (unbalanced)
((((((((((((  ))))))))))))      (unbalanced)
((((((((((((  ))))))))))))())
((((((((((((  ))))))))))))()))
```

```
((((((((((((((( ))))))))))))))
((((((((((((((( ))))))))))))))()))      (unbalanced)
((((((((((((((((( ))))))))))))))))      (unbalanced)
((((((((((((((((( ))))))))))))))))
((((((((((((((((((( ))))))))))))))))))
((((((((((((((((((( ))))))))))))))))))())
```

En ja, tot een bepaalde lengte doet het netwerk het prima. Maar dan begint het mis te gaan. Het is vrij typisch om te zien in een "precieze" situatie als deze met een neuraal net (of met machine learning in het algemeen). Gevallen die een mens "in één oogopslag kan oplossen", kan het neurale net ook oplossen. Maar in gevallen waarin iets 'meer algoritmisch' moet worden gedaan (bijvoorbeeld expliciet haakjes tellen om te zien of ze gesloten zijn), is het neurale net op de een of andere manier 'te computationeel ondiep' om betrouwbaar te doen. (Trouwens, zelfs de volledig huidige ChatGPT heeft moeite om haakjes correct te matchen in lange reeksen.)

Dus wat betekent dit voor zaken als ChatGPT en de syntaxis van een taal als Engels? De taal tussen haakjes is "sober" - en veel meer een "algoritmisch verhaal". Maar in het Engels is het veel realistischer om te kunnen 'raden' wat grammaticaal gaat passen op basis van lokale woordkeuzes en andere hints. En ja, het neurale net is hier veel beter in, ook al mist het misschien een 'formeel correcte' casus die, nou ja, mensen misschien ook missen. Maar het belangrijkste punt is dat het feit dat er een algehele syntactische structuur in de taal zit - met alle regelmaat die dat impliceert - in zekere zin beperkt "hoeveel" het neurale net moet leren.

Syntaxis biedt een soort beperking op taal. Maar er zijn er duidelijk meer. Een zin als "Nieuwsgierige elektronen eten blauwe theorieën voor vissen" is grammaticaal

correct, maar is niet iets wat je normaal zou verwachten te zeggen, en zou niet als een succes worden beschouwd als ChatGPT het zou genereren - omdat, nou ja, met de normale betekenissen voor de woorden erin, het is eigenlijk zinloos.

Maar is er een algemene manier om te zien of een zin zinvol is? Daar is geen traditionele algemene theorie voor. Maar het is iets waarvan je kunt denken dat ChatGPT impliciet "een theorie heeft ontwikkeld" nadat het is getraind met miljarden (vermoedelijk zinvolle) zinnen van internet, enz.

Hoe zou deze theorie kunnen zijn? Wel, er is een klein hoekje dat eigenlijk al twee millennia bekend is, en dat is logisch. En zeker in de syllogistische vorm waarin Aristoteles het ontdekte, is logica in wezen een manier om te zeggen dat zinnen die bepaalde patronen volgen redelijk zijn en andere niet. Zo is het bijvoorbeeld redelijk om te zeggen "Alle X zijn Y. Dit is niet Y, dus het is geen X" (zoals in "Alle vissen zijn blauw. Dit is niet blauw, dus het is geen vis."). En net zoals je je enigszins grillig kunt voorstellen dat Aristoteles de syllogistische logica ontdekte door ('machine-learning-stijl') door veel voorbeelden van retoriek te gaan, zo kan je je ook voorstellen dat hij in de training van ChatGPT in staat zal zijn geweest om 'ontdek syllogistische logica" door naar veel tekst op het web te kijken, enz. (En ja, hoewel men daarom kan verwachten dat ChatGPT tekst produceert die “juiste gevolgtrekkingen” bevat op basis van zaken als syllogistische logica,

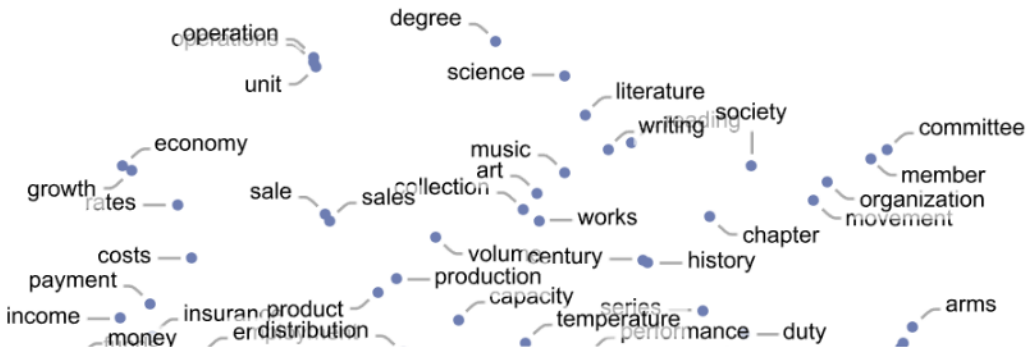
Maar afgezien van het enge voorbeeld van logica, wat kan er worden gezegd over het systematisch construeren (of herkennen) van zelfs plausibel zinvolle tekst? Ja, er zijn dingen zoals Mad Libs die zeer specifieke "frasale sjablonen" gebruiken. Maar op de een of andere manier heeft ChatGPT impliciet een veel algemenere manier om het te

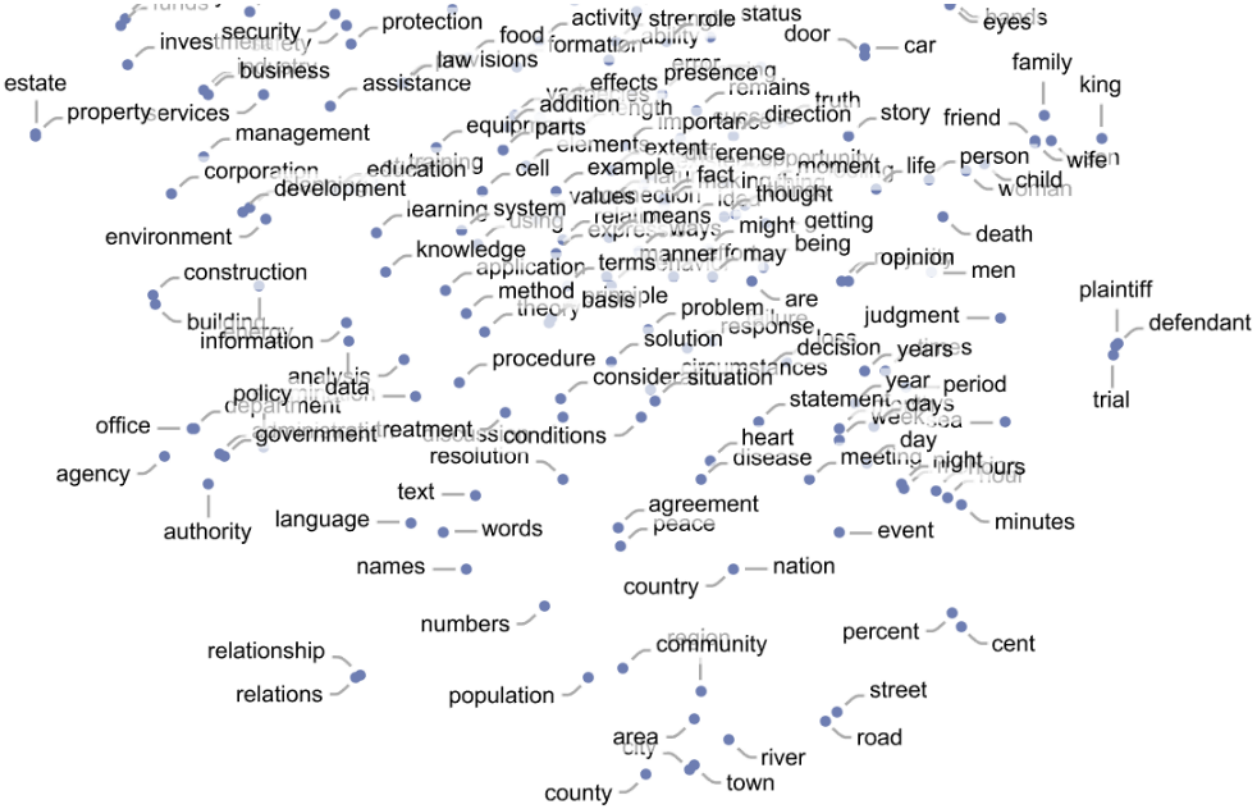
doen. En misschien valt er niets te zeggen over hoe het kan worden gedaan, behalve "op de een of andere manier gebeurt het als je 175 miljard neurale netgewichten hebt". Maar ik vermoed sterk dat er een veel eenvoudiger en sterker verhaal is.

Betekenis ruimte en semantische bewegingswetten

We hebben hierboven besproken dat elk stuk tekst in ChatGPT effectief wordt weergegeven door een reeks getallen die we kunnen zien als coördinaten van een punt in een soort "taalkundige kenmerkruimte". Dus wanneer ChatGPT een stuk tekst voortzet, komt dit overeen met het uittekenen van een traject in taalkundige kenmerkruimte. Maar nu kunnen we ons afvragen waardoor dit traject overeenkomt met tekst die we zinvol vinden. En zou er misschien een soort van "semantische bewegingswetten" zijn die definiëren - of op zijn minst beperken - hoe punten in taalkundige kenmerkruimte kunnen bewegen met behoud van "betekenis"?

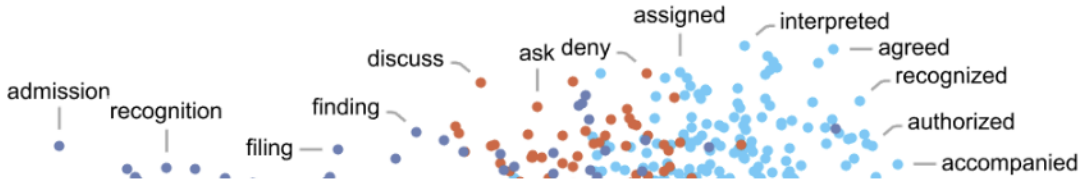
Dus hoe ziet deze taalkundige kenmerkruimte eruit? Hier is een voorbeeld van hoe afzonderlijke woorden (hier zelfstandige naamwoorden) kunnen worden opgemaakt als we zo'n kenmerkruimte naar 2D projecteren:

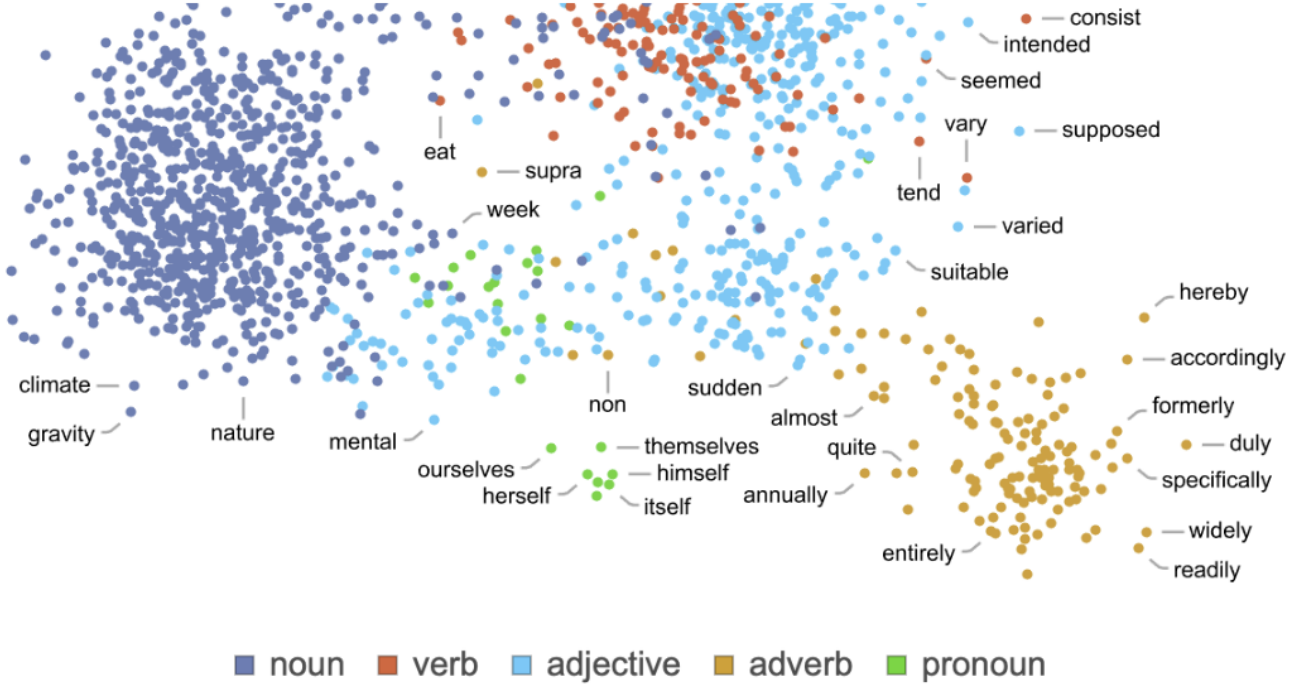




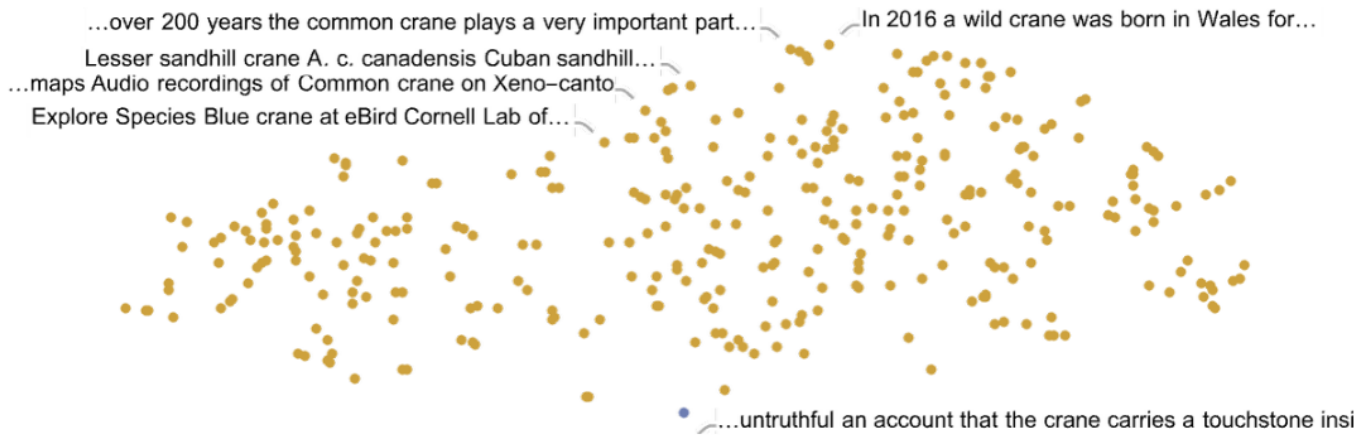
We zagen hierboven nog een voorbeeld op basis van woorden die planten en dieren vertegenwoordigen. Maar het punt in beide gevallen is dat "semantisch vergelijkbare woorden" in de buurt worden geplaatst.

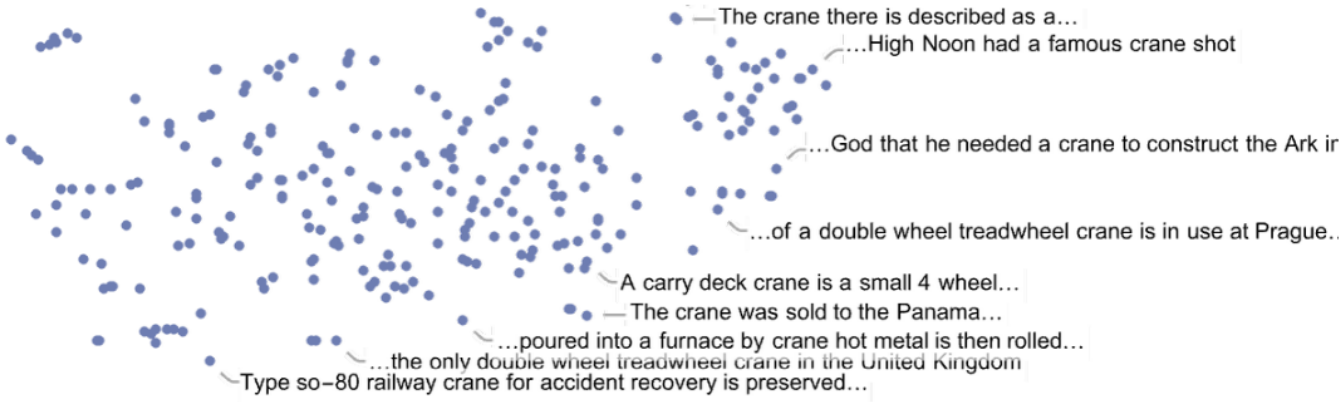
Als een ander voorbeeld, hier is hoe woorden die overeenkomen met verschillende delen van spraak worden ingedeeld:



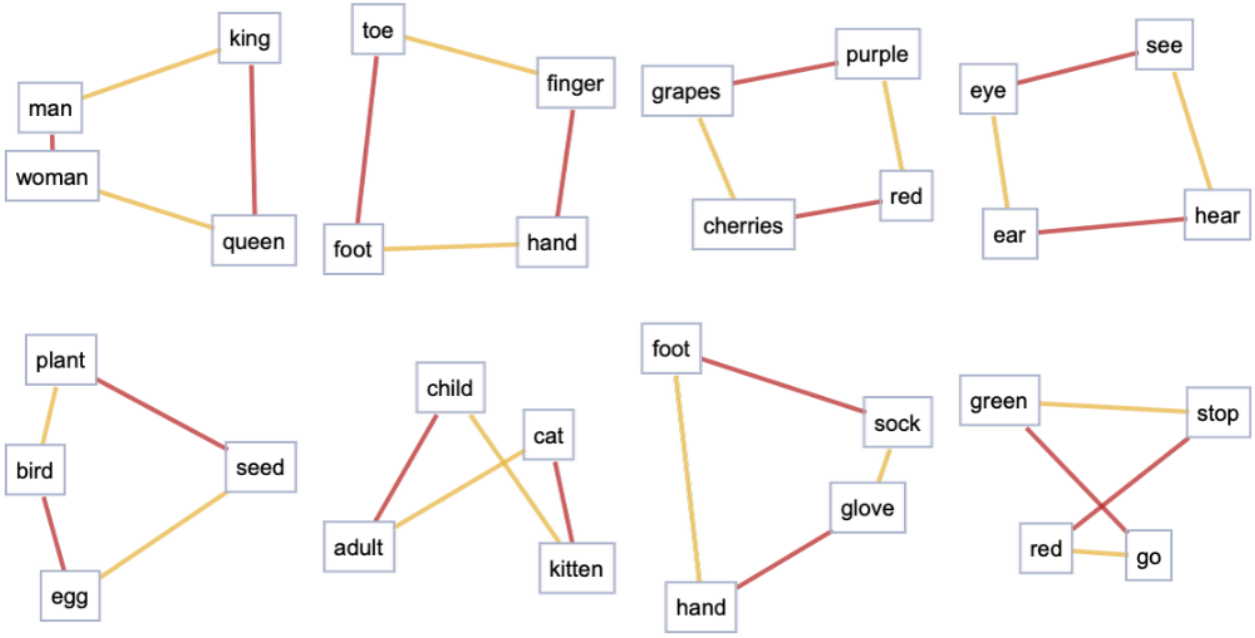


Natuurlijk heeft een bepaald woord in het algemeen niet slechts "één betekenis" (of komt het noodzakelijkerwijs overeen met slechts één woordsoort). En door te kijken hoe zinnen die een woord bevatten in de functieruimte zijn ingedeeld, kan men vaak verschillende betekenissen "uit elkaar halen" - zoals in het voorbeeld hier voor het woord "kraanvogel" (vogel of machine?):



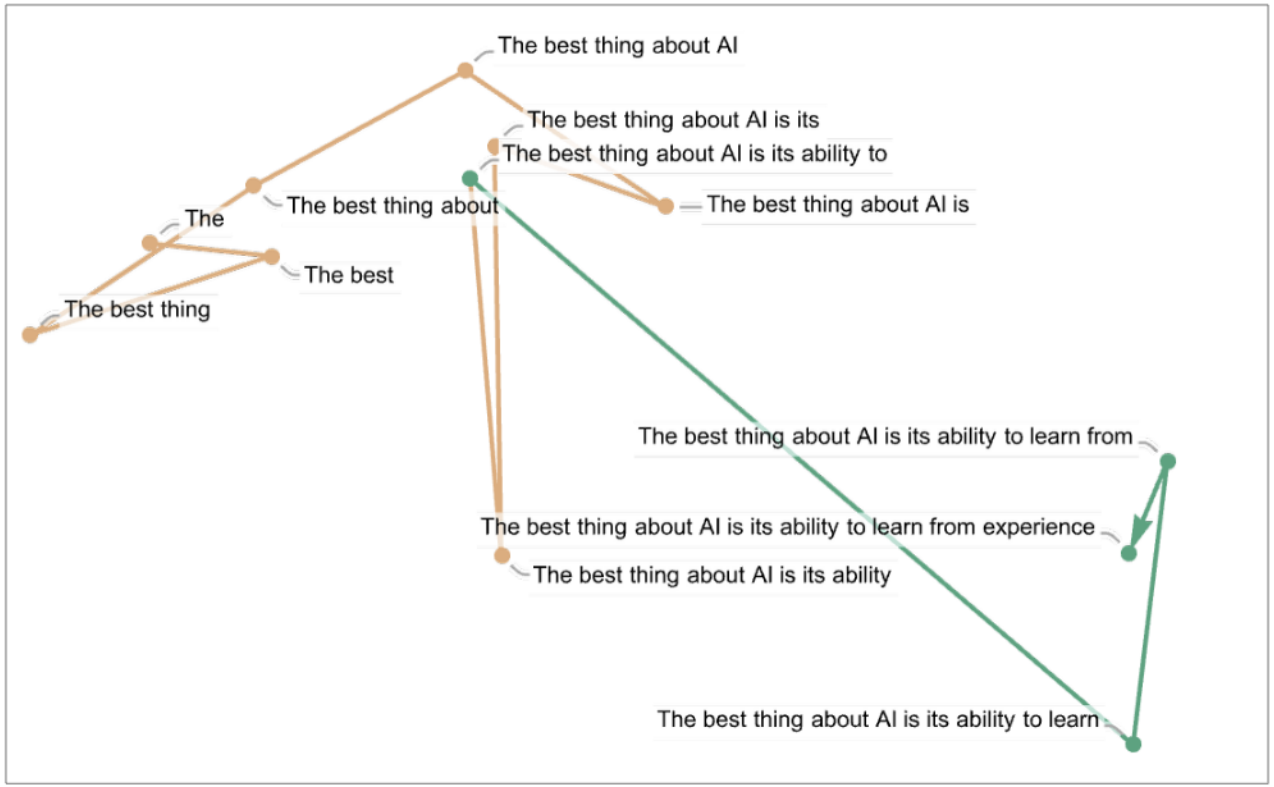


OK, dus het is op zijn minst aannemelijk dat we deze kenmerkruimte kunnen beschouwen als het plaatsen van "woorden dichtbij in betekenis" dichtbij in deze ruimte. Maar wat voor soort aanvullende structuur kunnen we in deze ruimte identificeren? Is er bijvoorbeeld een soort notie van "parallel transport" dat de "vlakheid" in de ruimte zou weerspiegelen? Een manier om daar grip op te krijgen, is door naar analogieën te kijken:



En ja, zelfs als we naar 2D projecteren, is er vaak op zijn minst een "vleugje vlakheid", hoewel het zeker niet universeel gezien wordt.

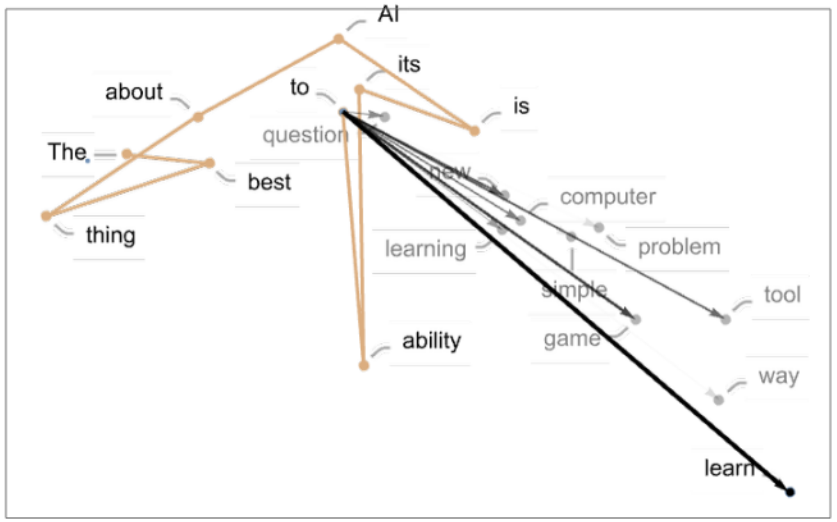
Hoe zit het dan met trajecten? We kunnen kijken naar het traject dat een prompt voor ChatGPT volgt in de functieruimte - en dan kunnen we zien hoe ChatGPT dat voortzet:



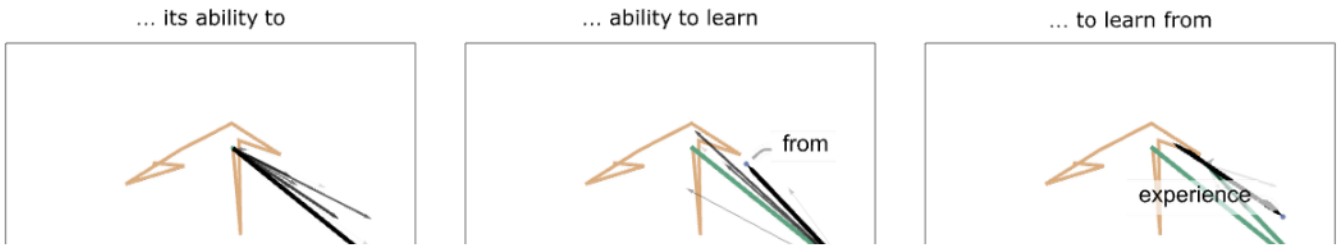
Er is hier zeker geen "geometrisch voor de hand liggende" bewegingswet. En dat is helemaal niet zo gek; we verwachten volledig dat dit een aanzienlijk ingewikkelder verhaal zal zijn . En het is bijvoorbeeld verre van duidelijk dat zelfs als er een 'semantische bewegingswet' te vinden is, in wat voor soort inbedding (of in feite welke

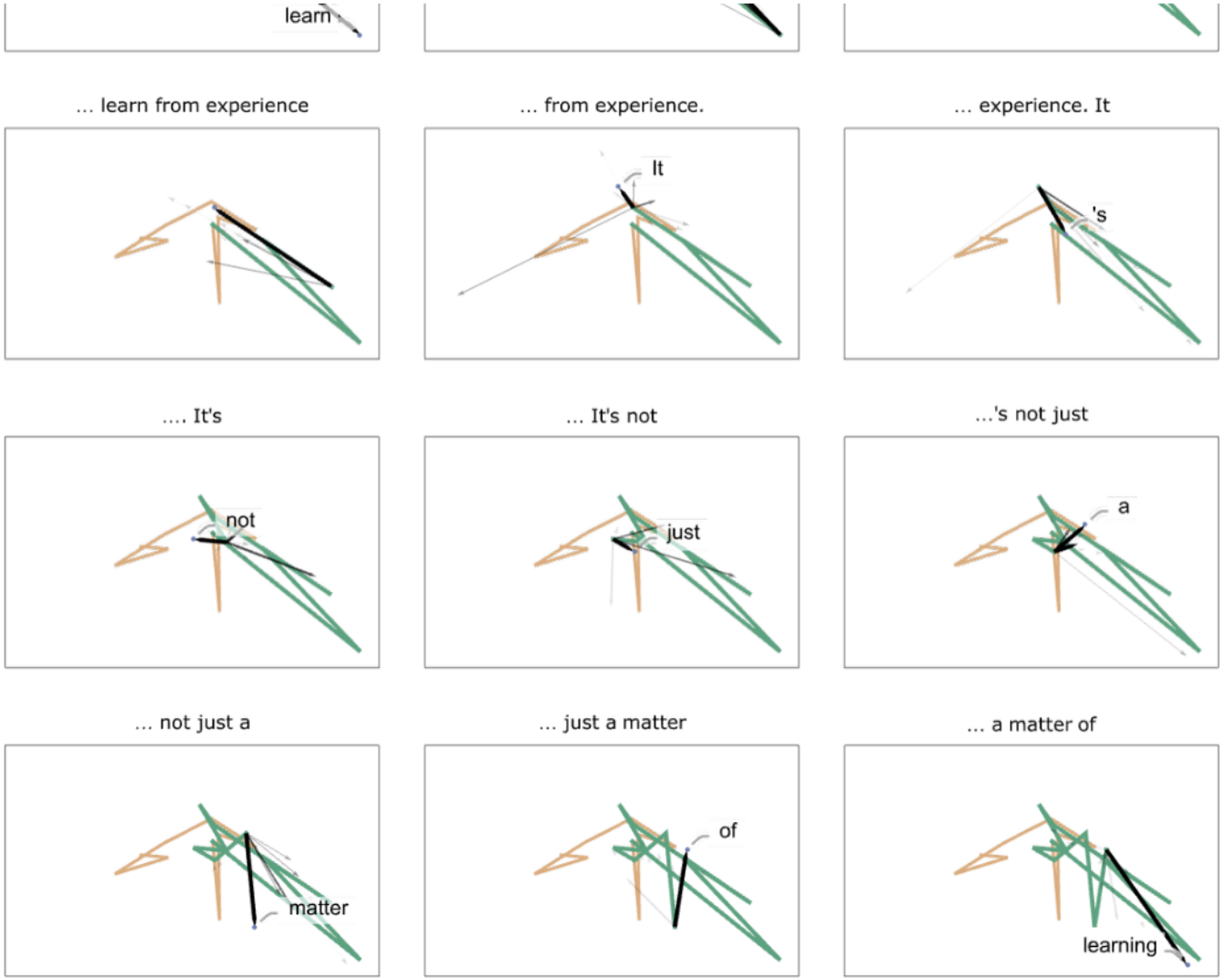
'variabelen') deze natuurlijk zal worden vermeld.

In de bovenstaande afbeelding laten we verschillende stappen in het 'traject' zien, waarbij we bij elke stap het woord kiezen dat ChatGPT het meest waarschijnlijk acht (het geval van 'temperatuur nul'). Maar we kunnen ons ook afvragen welke woorden "erna kunnen komen" met welke kansen op een bepaald punt:

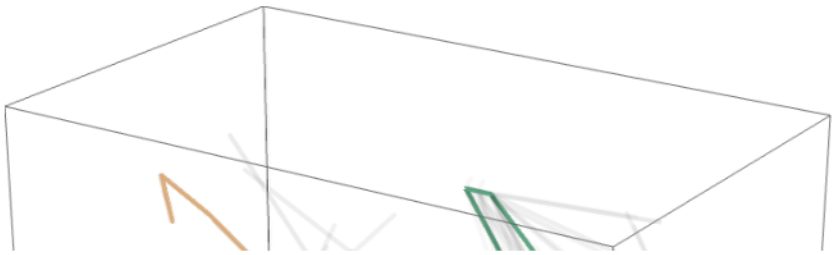


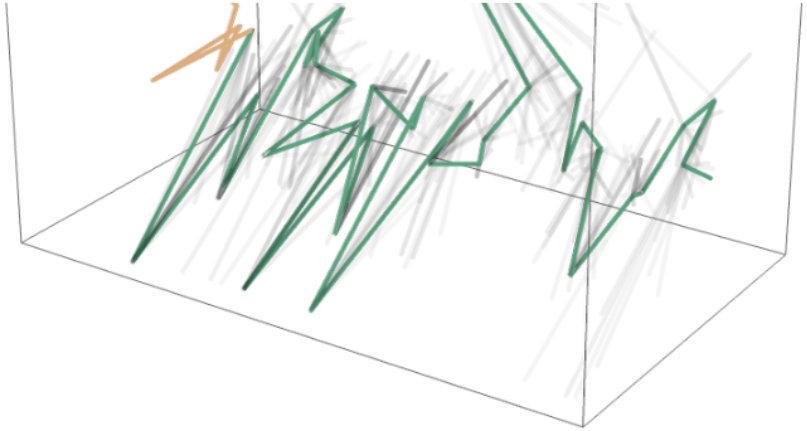
En wat we in dit geval zien, is dat er een 'waaier' is van woorden met een hoge waarschijnlijkheid die in een min of meer bepaalde richting lijkt te gaan in kenmerkruimte. Wat gebeurt er als we verder gaan? Dit zijn de opeenvolgende "fans" die verschijnen terwijl we het traject "bewegen":





Hier is een 3D-weergave, in totaal 40 stappen:





En ja, dit lijkt een puinhoop - en het doet niets om het idee aan te moedigen dat men kan verwachten "wiskundig-fysica-achtige" "semantische bewegingswetten" te identificeren door empirisch te bestuderen "wat ChatGPT van binnen doet" . Maar misschien kijken we gewoon naar de "verkeerde variabelen" (of het verkeerde coördinatensysteem) en als we maar naar de juiste kijken, zouden we meteen zien dat ChatGPT iets doet "wiskundig-natuurkunde-simpel" zoals het volgen van geodeten. Maar vanaf nu zijn we nog niet klaar om "empirisch te ontcijferen" uit zijn "intern gedrag" wat ChatGPT heeft "ontdekt" over hoe menselijke taal wordt "in elkaar gezet".

Semantische grammatica en de kracht van computertaal

Wat is er nodig om "betekenisvolle menselijke taal" te produceren? In het verleden zouden we hebben aangenomen dat het niets minder dan een menselijk brein zou kunnen zijn. Maar nu weten we dat het heel respectabel kan worden gedaan door het neurale net van ChatGPT. Maar misschien is dat wel zo ver als we kunnen gaan, en is er niets eenvoudiger - of menselijker begrijpelijker - dat zal werken. Maar mijn sterke vermoeden is dat het succes van ChatGPT impliciet een belangrijk "wetenschappelijk"

feit aan het licht brengt: dat er eigenlijk veel meer structuur en eenvoud is in betekenisvolle menselijke taal dan we ooit wisten - en dat er uiteindelijk zelfs vrij eenvoudige regels kunnen zijn die beschrijven hoe zo'n taal kan worden samengesteld.

Zoals we hierboven vermeldden, geeft syntactische grammatica regels voor hoe woorden die overeenkomen met zaken als verschillende woordsoorten, in menselijke taal kunnen worden samengesteld. Maar om met betekenis om te gaan, moeten we verder gaan. En een versie van hoe je dit doet, is door niet alleen na te denken over een syntactische grammatica voor taal, maar ook over een semantische.

Voor syntaxisdoeleinden identificeren we zaken als zelfstandige naamwoorden en werkwoorden. Maar voor doeleinden van semantiek hebben we "fijnere gradaties" nodig. We kunnen dus bijvoorbeeld het concept van "bewegen" identificeren, en het concept van een "object" dat "zijn identiteit behoudt onafhankelijk van de locatie". Er zijn eindeloos veel specifieke voorbeelden van elk van deze "semantische concepten". Maar voor de doeleinden van onze semantische grammatica hebben we gewoon een algemene regel die in feite zegt dat "objecten" kunnen "bewegen". Er is veel te zeggen over hoe dit allemaal zou kunnen werken (waarvan ik er een aantal al eerder heb gezegd). Maar ik zal mezelf hier tevreden stellen met slechts een paar opmerkingen die een deel van de mogelijke weg voorwaarts aangeven.

Het is vermeldenswaard dat zelfs als een zin volgens de semantische grammatica perfect in orde is, dit niet betekent dat deze in de praktijk is gerealiseerd (of zelfs zou kunnen worden gerealiseerd). "De olifant reisde naar de maan" zou ongetwijfeld onze semantische grammatica "passeren", maar het is zeker (nog) niet gerealiseerd in onze

werkelijke wereld - hoewel het absoluut eerlijk spel is voor een fictieve wereld.

Als we beginnen te praten over 'semantische grammatica', vragen we ons al snel af: 'Wat zit eronder?' Van welk "model van de wereld" gaat het uit? Een syntactische grammatica gaat eigenlijk alleen over de constructie van taal uit woorden. Maar een semantische grammatica houdt noodzakelijkerwijs verband met een soort 'model van de wereld' - iets dat dient als een 'skelet' waarop taal gemaakt van echte woorden kan worden gelaagd.

Tot voor kort hadden we misschien gedacht dat (menselijke) taal de enige algemene manier zou zijn om ons 'model van de wereld' te beschrijven. Reeds een paar eeuwen geleden begonnen er formaliseringen te komen van specifieke soorten dingen, met name gebaseerd op wiskunde. Maar nu is er een veel algemenere benadering van formalisering: computationele taal .

En ja, dat is mijn grote project geweest in de loop van meer dan vier decennia (zoals nu belichaamd in de Wolfram-taal): een precieze symbolische representatie ontwikkelen die zo breed mogelijk over dingen in de wereld kan praten, evenals abstracte dingen waar we om geven. En dus hebben we bijvoorbeeld symbolische representaties voor steden en moleculen en afbeeldingen en neurale netwerken , en we hebben ingebouwde kennis over hoe we over die dingen moeten rekenen.

En na tientallen jaren werk hebben we op deze manier veel gebieden behandeld. Maar in het verleden hebben we ons niet specifiek beziggehouden met het ' dagelijkse discours '. In "Ik heb twee pond appels gekocht" kunnen we gemakkelijk de "twee pond appels" weergeven (en er voeding en andere berekeningen op uitvoeren). Maar

we hebben (nog) geen symbolische voorstelling van “ik heb gekocht”.

Het is allemaal verbonden met het idee van semantische grammatica - en het doel om een generieke symbolische "constructiekit" voor concepten te hebben, die ons regels zou geven voor wat bij wat zou kunnen passen, en dus voor de "stroom" van wat we zouden kunnen veranderen. in mensentaal.

Maar laten we zeggen dat we deze "symbolische discourstaal" hadden. Wat zouden we ermee doen? We zouden bijvoorbeeld kunnen beginnen met het genereren van 'lokaal betekenisvolle tekst'. Maar uiteindelijk willen we waarschijnlijk meer 'wereldwijd betekenisvolle' resultaten - wat betekent dat we meer moeten 'berekenen' over wat werkelijk kan bestaan of gebeuren in de wereld (of misschien in een consistente fictieve wereld).

Op dit moment hebben we in Wolfram Language een enorme hoeveelheid ingebouwde computationele kennis over allerlei soorten dingen. Maar voor een complete symbolische discourstaal zouden we extra "calculi" moeten inbouwen over algemene dingen in de wereld: als een object van A naar B en van B naar C gaat, dan wordt het van A naar C verplaatst, enz.

Gegeven een symbolische discourstaal zouden we die kunnen gebruiken om 'op zichzelf staande uitspraken' te doen. Maar we kunnen het ook gebruiken om vragen over de wereld te stellen, “Wolfram|Alpha-stijl”. Of we kunnen het gebruiken om dingen te zeggen die we "zo willen maken", vermoedelijk met een extern bedieningsmechanisme. Of we kunnen het gebruiken om beweringen te doen - misschien over de echte wereld, of misschien over een specifieke wereld die we

overwegen, fictief of niet.

Menselijke taal is fundamenteel onnauwkeurig, niet in de laatste plaats omdat het niet is "vastgebonden" aan een specifieke computerimplementatie, en de betekenis ervan wordt in feite alleen bepaald door een "sociaal contract" tussen de gebruikers ervan. Maar computationele taal heeft van nature een zekere fundamentele precisie, omdat wat ze specificeert uiteindelijk altijd "ondubbelzinnig op een computer" kan worden uitgevoerd. Menselijke taal kan meestal wegkomen met een zekere vaagheid. (Als we 'planeet' zeggen, omvat dat dan exoplaneten of niet, enz.?) Maar in computertaal moeten we nauwkeurig en duidelijk zijn over alle onderscheidingen die we maken.

Het is vaak handig om gewone menselijke taal te gebruiken bij het verzinnen van namen in computertaal. Maar de betekenissen die ze hebben in computertaal zijn noodzakelijkerwijs nauwkeurig - en kunnen al dan niet een bepaalde connotatie in typisch menselijk taalgebruik dekken.

Hoe moet men de fundamentele 'ontologie' vinden die geschikt is voor een algemene symbolische discourstaal? Nou, het is niet gemakkelijk. Dat is misschien de reden waarom er zo weinig is gedaan sinds het primitieve begin dat Aristoteles meer dan twee millennia geleden maakte. Maar het helpt echt dat we nu zoveel weten over hoe we computationeel over de wereld moeten denken (en het kan geen kwaad om een "fundamentele metafysica" van ons natuurkundeproject en het idee van de ruliad te hebben).

Maar wat betekent dit allemaal in de context van ChatGPT? Van zijn training heeft

ChatGPT effectief een bepaalde (nogal indrukwekkende) hoeveelheid van wat neerkomt op semantische grammatica "in elkaar gezet". Maar juist het succes ervan geeft ons een reden om te denken dat het haalbaar zal zijn om iets completers in computationele taalvorm te construeren. En, in tegenstelling tot wat we tot nu toe hebben ontdekt over de ingewanden van ChatGPT, kunnen we verwachten dat we de computertaal zo ontwerpen dat deze gemakkelijk te begrijpen is voor mensen.

Als we het hebben over semantische grammatica, kunnen we een analogie trekken met syllogistische logica. Aanvankelijk was syllogistische logica in wezen een verzameling regels over uitspraken in menselijke taal. Maar (ja, twee millennia later) toen de formele logica werd ontwikkeld, zouden de oorspronkelijke basisconstructies van syllogistische logica nu kunnen worden gebruikt om enorme "formele torens" te bouwen die bijvoorbeeld de werking van moderne digitale schakelingen omvatten. En dus, kunnen we verwachten, zal het zijn met meer algemene semantische grammatica. In het begin kan het misschien gewoon omgaan met eenvoudige patronen, bijvoorbeeld uitgedrukt in tekst. Maar als het hele computationele taalkader eenmaal is gebouwd, kunnen we verwachten dat het gebruikt zal kunnen worden om hoge torens van "gegeneraliseerde semantische logica" op te richten, die ons in staat stellen om op een precieze en formele manier te werken met allerlei dingen die nooit eerder voor ons toegankelijk geweest, behalve alleen op een "begane grond" door menselijke taal, met al zijn vaagheid.

We kunnen de constructie van computationele taal - en semantische grammatica - beschouwen als een soort ultieme compressie bij het weergeven van dingen. Omdat het ons in staat stelt om te praten over de essentie van wat mogelijk is, zonder

bijvoorbeeld al de "uitdrukkingen" die in de gewone menselijke taal bestaan, aan te pakken. En we kunnen de grote kracht van ChatGPT zien als iets vergelijkbaars: omdat het ook in zekere zin is "doorgeboord" tot het punt waarop het "taal op een semantisch zinvolle manier kan samenstellen" zonder zich zorgen te maken over verschillende mogelijke wendingen zin.

Dus wat zou er gebeuren als we ChatGPT zouden toepassen op de onderliggende computertaal? De computertaal kan beschrijven wat mogelijk is. Maar wat nog kan worden toegevoegd, is een gevoel van 'wat populair is', bijvoorbeeld gebaseerd op het lezen van al die inhoud op internet. Maar daaronder betekent het werken met computertaal dat zoiets als ChatGPT onmiddellijke en fundamentele toegang heeft tot de ultieme tools om gebruik te maken van potentieel onherleidbare berekeningen. En dat maakt het tot een systeem dat niet alleen "redelijke tekst kan genereren", maar ook kan verwachten uit te werken wat er ook maar kan worden uitgewerkt of die tekst daadwerkelijk "juiste" uitspraken doet over de wereld - of waar het ook over zou moeten gaan.

Dus ... Wat doet ChatGPT en waarom werkt het?

Het basisconcept van ChatGPT is op een bepaald niveau vrij eenvoudig. Begin met een enorme steekproef van door mensen gemaakte tekst van internet, boeken, enz. Train vervolgens een neuraal netwerk om tekst te genereren die "zo" is. En zorg er vooral voor dat het kan beginnen met een "prompt" en dan verder kan gaan met tekst die "lijkt op waarmee het is getraind".

Zoals we hebben gezien, bestaat het werkelijke neurale netwerk in ChatGPT uit zeer eenvoudige elementen, hoewel het er miljarden zijn. En de basiswerking van het neurale net is ook heel eenvoudig en bestaat in wezen uit het doorgeven van invoer die is afgeleid van de tekst die het tot nu toe heeft gegenereerd "een keer door de elementen" (zonder lussen, enz.) voor elk nieuw woord (of deel van een woord).) die het genereert.

Maar het opmerkelijke - en onverwachte - is dat dit proces tekst kan produceren die met succes "zoals" wat er op het web, in boeken, enz. staat. En niet alleen is het coherente menselijke taal, het "zegt ook dingen" die "volgen". het is prompt" door gebruik te maken van de inhoud die het "gelezen" heeft. Het zegt niet altijd dingen die "algemeen logisch zijn" (of overeenkomen met correcte berekeningen) - omdat (zonder bijvoorbeeld toegang te krijgen tot de "computationele superkrachten" van Wolfram | Alpha) het gewoon dingen zegt die "goed klinken" op basis van hoe dingen "klonken" in het trainingsmateriaal.

De specifieke techniek van ChatGPT heeft het behoorlijk aantrekkelijk gemaakt. Maar uiteindelijk (tenminste totdat het externe tools kan gebruiken) haalt ChatGPT "slechts" een "coherente draad tekst" uit de "statistieken van conventionele wijsheid" die het heeft verzameld. Maar het is verbazingwekkend hoe menselijk de resultaten zijn. En zoals ik heb besproken, suggereert dit iets dat op zijn minst wetenschappelijk erg belangrijk is: dat menselijke taal (en de denkpatronen erachter) op de een of andere manier eenvoudiger en meer "wetachtig" in hun structuur zijn dan we dachten. ChatGPT heeft het impliciet ontdekt. Maar we kunnen het potentieel expliciet blootleggen, met semantische grammatica, computertaal, enz.

Wat ChatGPT doet bij het genereren van tekst is zeer indrukwekkend - en de resultaten lijken meestal erg op wat wij mensen zouden produceren. Betekent dit dat ChatGPT werkt als een brein? De onderliggende kunstmatige neurale netwerkstructuur werd uiteindelijk gemodelleerd naar een idealisering van de hersenen. En het lijkt vrij waarschijnlijk dat wanneer wij mensen taal genereren, veel aspecten van wat er gaande is vrij gelijkaardig zijn.

Als het gaat om training (AKA-leren) dwingt de verschillende "hardware" van de hersenen en van de huidige computers (evenals misschien enkele onontwikkelde algoritmische ideeën) ChatGPT ertoe een strategie te gebruiken die waarschijnlijk nogal anders is (en in sommige opzichten veel minder efficiënt) dan de hersenen. En er is nog iets anders: in tegenstelling tot zelfs bij typische algoritmische berekeningen, heeft ChatGPT intern geen "lussen" of "herberekenen op basis van gegevens". En dat beperkt onvermijdelijk zijn rekenvermogen - zelfs met betrekking tot de huidige computers, maar zeker met betrekking tot de hersenen.

Het is niet duidelijk hoe dit te "repareren" en toch de mogelijkheid te behouden om het systeem met redelijke efficiëntie te trainen. Maar om dit te doen, zal een toekomstige ChatGPT vermoedelijk nog meer "hersenachtige dingen" kunnen doen. Natuurlijk zijn er tal van dingen waar hersenen niet zo goed in zijn , vooral als het gaat om onherleidbare berekeningen. En hiervoor moeten zowel de hersenen als dingen zoals ChatGPT "externe tools" zoeken, zoals Wolfram Language .

Maar voor nu is het spannend om te zien wat ChatGPT al heeft kunnen doen. Op een bepaald niveau is het een geweldig voorbeeld van het fundamentele

wetenschappelijke feit dat grote aantallen eenvoudige rekenelementen opmerkelijke en onverwachte dingen kunnen doen. Maar het biedt misschien ook wel de beste impuls die we in tweeduizend jaar hebben gehad om beter te begrijpen wat het fundamentele karakter en de principes zouden kunnen zijn van dat centrale kenmerk van de menselijke conditie dat de menselijke taal is en de denkprocessen die erachter zitten.

Bedankt

Ik volg de ontwikkeling van neurale netwerken nu al zo'n 43 jaar, en in die tijd heb ik met veel mensen erover gesproken. Onder hen - sommige van lang geleden, sommige van recentelijk en sommige van vele jaren - waren: Giulio Alessandrini, Dario Amodei, Etienne Bernard, Taliesin Beynon, Sebastian Bodenstein, Greg Brockman, Jack Cowan, Pedro Domingos, Jesse Galef, Roger Germundsson , Robert Hecht-Nielsen, Geoff Hinton, John Hopfield, Yann LeCun, Jerry Lettvin, Jerome Louradour, Marvin Minsky, Eric Mjolsness, Cayden Pierce, Tomaso Poggio, Matteo Salvarezza, Terry Sejnowski, Oliver Selfridge, Gordon Shaw, Jonas Sjöberg, Ilya Sutskever , Gerry Tesauro en Timothee Verdier. Voor hulp bij dit stuk wil ik in het bijzonder Giulio Alessandrini en Brad Klee bedanken.

Aanvullende bronnen

"Machine Learning voor middelbare scholieren" »

Inleiding tot machinaal leren »

Wolfram U Machine Learning-cursussen »

Noem dit als >	Stephen Wolfram (2023), "Wat doet ChatGPT ... en waarom werkt het?", Geschriften van Stephen Wolfram. geschriften.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work.

Geplaatst in: Kunstmatige intelligentie , Taal en communicatie , Nieuwe technologie

Doe mee aan de discussie

+53 opmerkingen

Gerelateerde geschriften

Introductie van Chat Notebooks: LLM's integreren in het Notebook Paradigma

8 juni 2023

Prompts voor werk en ontspanning: de Wolfram Prompt Repository starten

7 juni 2023

De nieuwe wereld van LLM-functies: LLM-technologie integreren in de Wolfram-taal

23 mei 2023

Directe plug-ins voor ChatGPT: introductie van de Wolfram ChatGPT-plug-inkit

27 april 2023

Deze website maakt gebruik van cookies om uw ervaring met onze diensten op de site te optimaliseren, zoals beschreven in ons privacybeleid
Zoeken
13:39
21-6-2023
2