Marc Riedl Suivre

14 avril · 38 min de lecture · Écouter

Une introduction très douce aux grands modèles de langage sans le battage médiatique

[C'est un travail en cours de réalisation]

1. Introduction

Cet article est conçu pour donner aux personnes sans formation en informatique un aperçu du fonctionnement de ChatGPT et des systèmes d'IA similaires (GPT-3, GPT-4, Bing Chat, Bard, etc.). ChatGPT est un chatbot - un type d'IA conversationnelle construit - mais au-dessus d'un modèle de grande langue. Ce sont définitivement des mots et nous allons décomposer tout cela. Au cours du processus, nous discuterons des concepts de base qui les sous-tendent. Cet article ne nécessite aucune connaissance technique ou mathématique. Nous ferons un usage intensif de métaphores pour illustrer les concepts. Nous expliquerons pourquoi les concepts de base fonctionnent comme ils fonctionnent et ce que nous pouvons attendre ou non des grands modèles de langage comme ChatGPT.

Voici ce que nous allons faire. Nous allons doucement parcourir une partie de la terminologie associée aux grands modèles de langage et à ChatGPT sans aucun jargon. Si je dois utiliser du jargon, je vais le décomposer sans jargon. Nous allons commencer de manière très basique, avec "qu'est-ce que l'intelligence artificielle" et progresser. J'utiliserai autant que possible des métaphores récurrentes. Je vais parler des implications des technologies en termes de ce que nous devrions attendre d'elles ou ne pas attendre d'elles.

Allons-y!

1. Qu'est-ce que l'Intelligence Artificielle ?

Mais d'abord, commençons par une terminologie de base que vous entendez probablement beaucoup. Qu'est-ce que l'intelligence artificielle ?

- Intelligence artificielle : Une entité qui exécute des comportements qu'une personne pourrait raisonnablement qualifier d'intelligents si un humain devait faire quelque chose de similaire.

Il est un peu problématique de définir l'intelligence artificielle en utilisant le mot "intelligent", mais personne ne peut s'entendre sur une bonne définition de "intelligent". Cependant, je pense que cela fonctionne toujours raisonnablement bien. Il dit essentiellement que si nous regardons quelque chose d'artificiel et qu'il fait des choses engageantes et utiles et qui semblent quelque peu non triviales, alors nous pourrions l'appeler intelligent. Par exemple, nous attribuons souvent le terme "IA" aux personnages contrôlés par ordinateur dans les jeux vidéo. La plupart de ces bots sont de simples morceaux de code if-then-else (par exemple, "si le joueur est à portée, tirez sinon déplacez-vous vers le rocher le plus proche pour vous mettre à l'abri"). Mais si nous faisons le travail de nous garder engagés et divertis, et de ne pas faire de choses manifestement stupides, alors nous pourrions penser qu'ils sont plus sophistiqués qu'ils ne le sont.

Une fois que nous comprenons comment quelque chose fonctionne, nous ne sommes peut-être pas très impressionnés et nous nous attendons à quelque chose de plus sophistiqué dans les coulisses. Tout dépend de ce que vous savez de ce qui se passe dans les coulisses.

Leur point clé est que l'intelligence artificielle n'est pas magique. Et parce que ce n'est pas de la magie, cela s'explique.

Alors allons-y.

2. Qu'est-ce que l'apprentissage automatique ?

Un autre terme que vous entendrez souvent associé à l'intelligence artificielle est l'apprentissage automatique.

- Apprentissage automatique : un moyen de créer un comportement en prenant des données, en formant un modèle, puis en exécutant le modèle.

Parfois, il est trop difficile de créer manuellement un tas d'instructions if-then-else pour capturer un phénomène compliqué, comme le langage. Dans ce cas, nous essayons de trouver un ensemble de données et d'utiliser des algorithmes capables de trouver des modèles dans les données à modéliser.

Mais qu'est-ce qu'un modèle ? Un modèle est une simplification d'un phénomène complexe. Par exemple, un modèle de voiture n'est qu'une version plus petite et plus simple d'une vraie voiture qui possède de nombreux attributs mais qui n'est pas destinée à remplacer complètement l'original. Un modèle de voiture peut sembler réel et être utile à certaines fins, mais nous ne pouvons pas le conduire au magasin.

Une image générée par DALL-E d'un modèle de voiture sur un bureau.

Tout comme nous pouvons créer une version plus petite et plus simple d'une voiture, nous pouvons également créer une version plus petite et plus simple du langage humain. Nous utilisons le terme grands modèles de langage parce que ces modèles sont, eh bien, volumineux, du point de vue de la quantité de mémoire nécessaire pour

Utilise les. Les plus grands modèles en production, tels que ChatGPT, GPT-3 et GPT-4, sont suffisamment volumineux pour nécessiter la création et l'exécution d'énormes super-ordinateurs fonctionnant dans des serveurs de centre de données.

3. Qu'est-ce qu'un réseau de neurones ?

Il existe de nombreuses façons d'apprendre un modèle à partir de données. Le réseau de neurones est l'un de ces moyens. La technique est à peu près basée sur la façon dont le cerveau humain est constitué d'un réseau de cellules cérébrales interconnectées appelées neurones qui transmettent des signaux électriques dans les deux sens, nous permettant d'une manière ou d'une autre de faire toutes les choses que nous faisons. Le concept de base du réseau de neurones a été inventé dans les années 1940 et les concepts de base sur la façon de les former ont été inventés dans les années 1980. Les réseaux de neurones sont très inefficaces, et ce n'est qu'en 2017 environ que le matériel informatique a été suffisamment performant pour les utiliser à grande échelle.

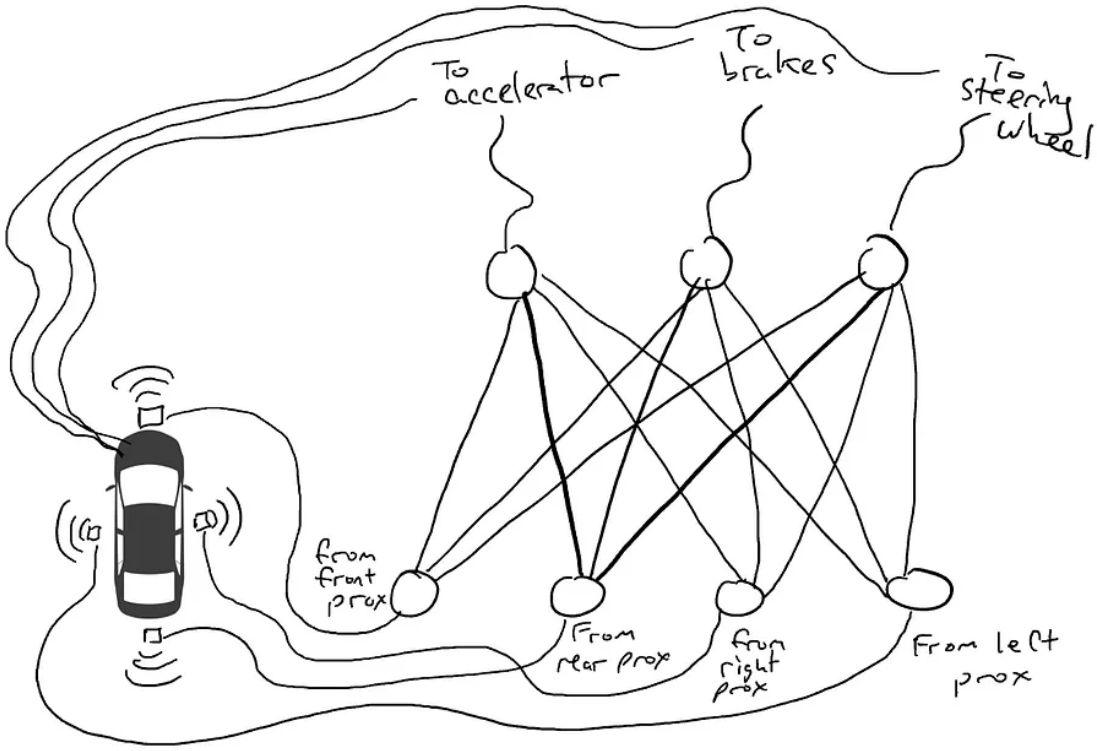
Mais au lieu de cerveaux, j'aime penser aux réseaux de neurones en utilisant la métaphore des circuits électriques. Vous n'avez pas besoin d'être un ingénieur électricien pour savoir que l'électricité circule à travers des fils et que nous avons des choses appelées résistances qui rendent plus difficile la circulation de l'électricité à travers les parties d'un circuit.

Imaginez que vous souhaitiez créer une voiture autonome capable de rouler sur l'autoroute. Vous avez équipé votre voiture de capteurs de proximité à l'avant, à l'arrière et sur les côtés. Les capteurs de proximité rapportent une valeur de 1,0 lorsqu'il y a quelque chose de très proche et rapportent une valeur de 0,0 lorsque rien n'est détectable à proximité.

Vous avez également truqué votre voiture pour que les mécanismes robotiques puissent tourner le volant, appuyer sur les freins et appuyer sur l'accélérateur. Lorsque l'accélérateur reçoit une valeur de 1,0, il utilise l'accélération maximale et 0,0 signifie aucune accélération. De même, une valeur de 1,0 envoyée au mécanisme de freinage signifie un claquement des freins et 0,0 signifie aucun freinage. Le mécanisme de direction prend une valeur de -1,0 à +1,0 avec une valeur négative signifiant braquer à gauche et une valeur positive signifiant braquer à droite et 0,0 signifiant rester tout droit.

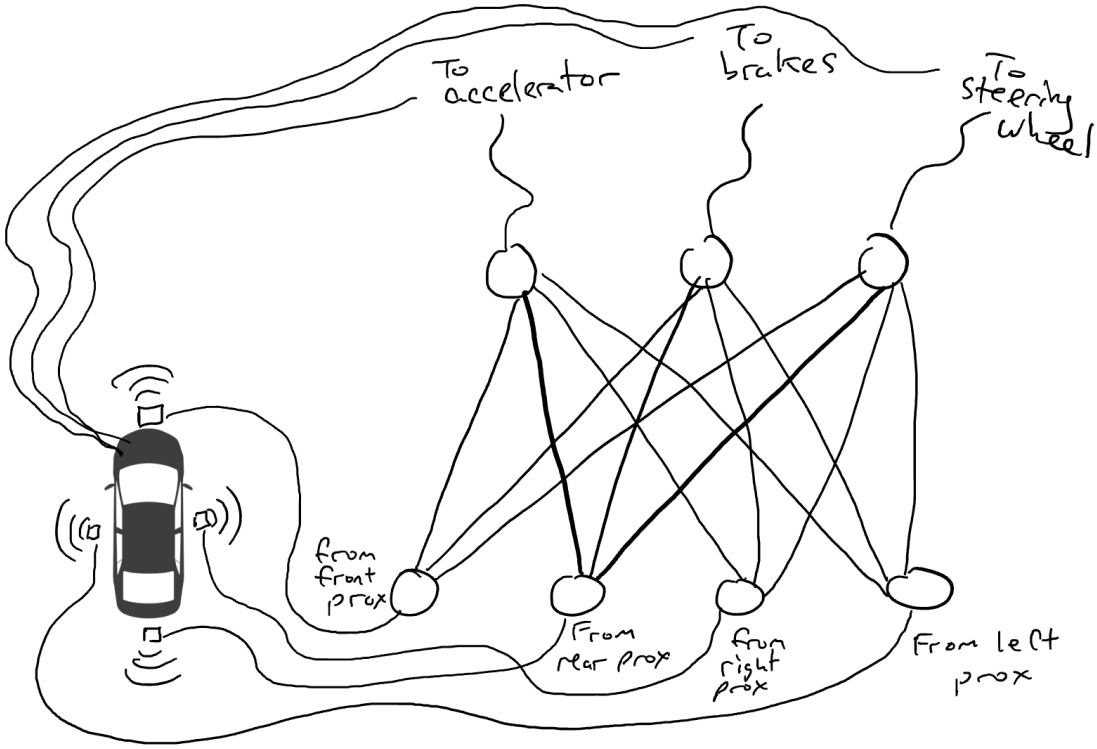
Vous avez également enregistré des données sur votre façon de conduire. Lorsque la route devant vous est dégagée, vous accélérez. Quand il y a une voiture devant, vous ralentissez. Lorsqu'une voiture s'approche trop à gauche, vous tournez à droite et changez de voie. À moins, bien sûr, qu'il y ait aussi une voiture sur votre droite. C'est un processus complexe impliquant différentes combinaisons d'actions (tourner à gauche, tourner à droite, accélérer plus ou moins, freiner) basées sur différentes combinaisons d'informations de capteurs.

Vous devez maintenant connecter le capteur aux mécanismes robotiques. Comment est-ce que tu fais ça? Ce n'est pas clair. Vous connectez donc chaque capteur à chaque actionneur robotique.



Un réseau de neurones en tant que circuit reliant des capteurs à des actionneurs.

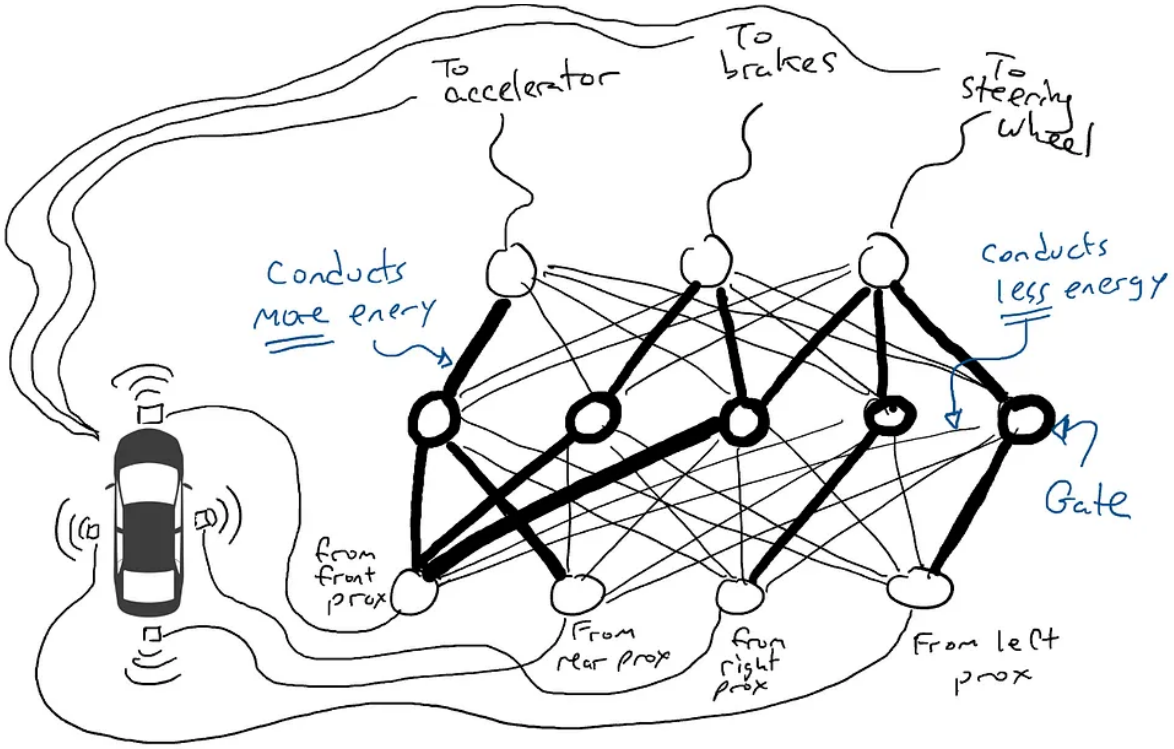
Que se passe-t-il lorsque vous prenez votre voiture sur la route ? Le courant électrique circule de tous les capteurs vers tous les actionneurs robotiques et la voiture tourne simultanément à gauche, à droite, accélère et freine. C'est le bordel.



Lorsque certains de nos capteurs envoient de l'énergie, cette énergie est transmise à tous les actionneurs et la voiture accélère, freine et braque tout à la fois.

Ce n'est pas bon. Alors je prends mes résistances et je commence à les mettre sur différentes parties des circuits pour que l'électricité puisse circuler plus librement entre certains capteurs et certains actionneurs robotiques. Par exemple, je veux que l'électricité circule plus librement des capteurs de proximité avant vers les freins et non vers le volant. J'ai également mis des choses appelées portes, qui arrêtent le flux d'électricité jusqu'à ce que suffisamment d'électricité s'accumule pour actionner un interrupteur (ne permettent à l'électricité de circuler que lorsque le capteur de proximité avant et le capteur de proximité arrière signalent des nombres élevés), ou envoient de l'énergie électrique vers l'avant uniquement lorsque l'intensité électrique d'entrée est faible (envoie plus d'électricité à l'accélérateur lorsque le capteur de proximité avant signale une valeur faible).

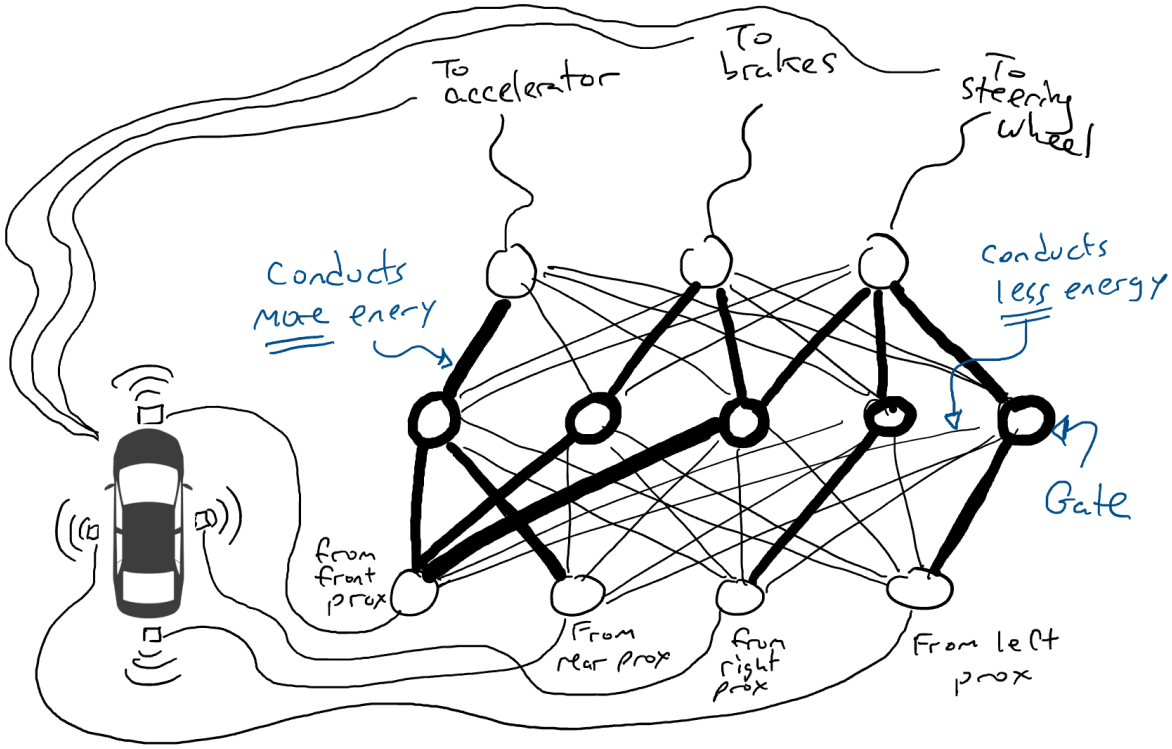
Mais où dois-je mettre ces résistances et ces portes ? Je ne sais pas. Je commence à les mettre au hasard partout. Ensuite, j'essaie à nouveau. Peut-être que cette fois, ma voiture roule mieux, ce qui signifie parfois qu'elle freine lorsque les données indiquent qu'il est préférable de freiner et qu'elle tourne lorsque les données indiquent qu'il est préférable de diriger, etc. Mais elle ne fait pas tout correctement. Et certaines choses, il fait pire (accélère lorsque les données indiquent qu'il est préférable de freiner). Je continue donc d'essayer au hasard différentes combinaisons de résistances et de portes. Finalement, je tomberai sur une combinaison qui fonctionne suffisamment bien pour que je déclare le succès. Peut-être que ça ressemble à ça :



Un réseau de neurones entièrement formé. Les lignes plus sombres signifient des parties du circuit où l'énergie circule plus librement. Cercles dans le le milieu sont des portes qui pourraient accumuler beaucoup d'énergie d'en bas avant d'envoyer de l'énergie vers le haut, ou peut-être même envoyer de l'énergie vers le haut lorsqu'il y a peu d'énergie en dessous.

(En réalité, nous n'ajoutons ni ne soustrayons des portes, qui sont toujours là, mais nous modifions les portes pour qu'elles s'activent avec moins d'énergie par le bas ou nécessitent plus d'énergie par le bas, ou libèrent peut-être beaucoup d'énergie uniquement lorsqu'il y a très peu d'énergie d'en bas. Les puristes de l'apprentissage automatique pourraient vomir un peu dans leur bouche à cette caractérisation. Techniquement, cela se fait en ajustant quelque chose appelé un biais sur les portes, qui n'est généralement pas montré dans des diagrammes comme ceux-ci, mais en termes de la métaphore du circuit peut être considérée comme un fil entrant dans chaque porte branchée directement sur une source électrique, qui peut ensuite être modifiée comme tous les autres fils.)

Prenons-le pour un essai routier!



Essayer des choses au hasard, c'est nul. Un algorithme appelé rétro-propagation est raisonnablement efficace pour faire des suppositions sur la façon de modifier la configuration du circuit. Les détails de l'algorithme ne sont pas importants, sauf pour savoir qu'il apporte de minuscules modifications au circuit pour rapprocher le comportement du circuit de ce que les données suggèrent, et sur des milliers ou des millions de modifications, peut éventuellement obtenir quelque chose de proche d'être d'accord avec les données.

Nous appelons les résistances et les paramètres de portes car en réalité elles sont partout et ce que fait l'algorithme de rétropropagation déclare que chaque résistance est plus forte ou plus faible. Ainsi l'ensemble du circuit peut être reproduit dans d'autres voitures si l'on connaît le tracé des circuits et les valeurs des paramètres.

4. Qu'est-ce que l'apprentissage en profondeur ?

Deep Learning est une reconnaissance que nous pouvons mettre d'autres choses dans nos circuits en plus des résistances et des portes. Par exemple, nous pouvons avoir un calcul mathématique au milieu de notre circuit qui additionne et multiplie les choses ensemble avant d'envoyer de l'électricité. Deep Learning utilise toujours la même technique incrémentielle de base consistant à deviner les paramètres.

5. Qu'est-ce qu'un modèle de langage ?

Lorsque nous avons pris l'exemple de la voiture, nous essayions de faire en sorte que notre réseau de neurones adopte un comportement cohérent avec nos données. Nous nous demandions si nous pouvions créer un circuit qui manipulait les mécanismes de la voiture de la même manière qu'un conducteur le faisait dans des circonstances similaires. Nous pouvons traiter le langage de la même manière. Nous pouvons regarder un texte écrit par des humains et nous demander si un circuit pourrait produire une séquence de mots qui ressemble beaucoup aux séquences de mots que les humains ont tendance à produire. Désormais, nos capteurs se déclenchent lorsque nous voyons des mots et nos mécanismes de sortie sont également des mots.

Qu'essayons-nous de faire ? Nous essayons de créer un circuit qui devine un mot de sortie, étant donné un groupe de mots d'entrée. Par exemple:

> "Il était une ____"

semble qu'il devrait remplir le vide avec "temps" mais pas "tatou".

Nous avons tendance à parler de modèles de langage en termes de probabilité. Mathématiquement, nous écrirons l'exemple ci-dessus comme suit :

$$P(time \mid once, upon, a)$$

Si vous n'êtes pas familier avec la notation, ne vous inquiétez pas. Ceci est juste un discours mathématique signifiant la probabilité (P) du mot "temps" donné (le symbole de la barre $\mid$ signifie donné) un groupe de mots "une fois", "sur" et "un". On s'attendrait à ce qu'un bon modèle de langage produise une probabilité plus élevée pour le mot « temps » que pour le mot « tatou ».

On peut généraliser cela à :

$$P(word_n \mid word_1, word_2, \ldots, word_{n-1})$$

ce qui signifie simplement calculer la probabilité du n-ième mot dans une séquence compte tenu de tous les mots qui le précèdent (mots aux positions 1 à n-1).

Mais revenons un peu en arrière. Pensez à une machine à écrire à l'ancienne, celle avec les bras percuteurs.

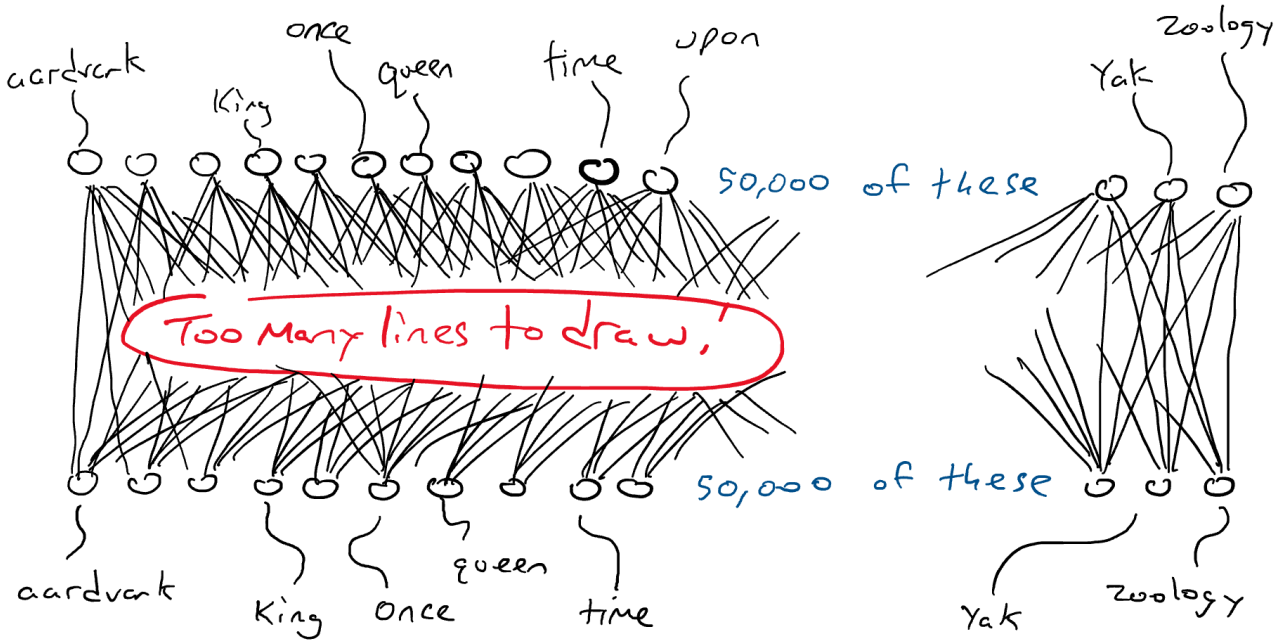
DALL-E2 a créé cette image. Regardez tous les bras de l'attaquant !

Sauf qu'au lieu d'avoir un bras percuteur différent pour chaque lettre, nous avons un percuteur pour chaque mot. Si la langue anglaise compte 50 000 mots, c'est une grosse machine à écrire !

Au lieu du réseau pour la voiture, pensez à un réseau similaire, sauf que le haut de notre circuit a 50 000 sorties connectées aux bras percuteurs, une pour chaque mot. En conséquence, nous aurions 50 000 capteurs, chacun détectant la présence d'un mot d'entrée différent. Donc, ce que nous faisons en fin de compte, c'est choisir un seul bras percuteur qui reçoit le signal électrique le plus élevé et c'est le mot qui va dans le

blanc.

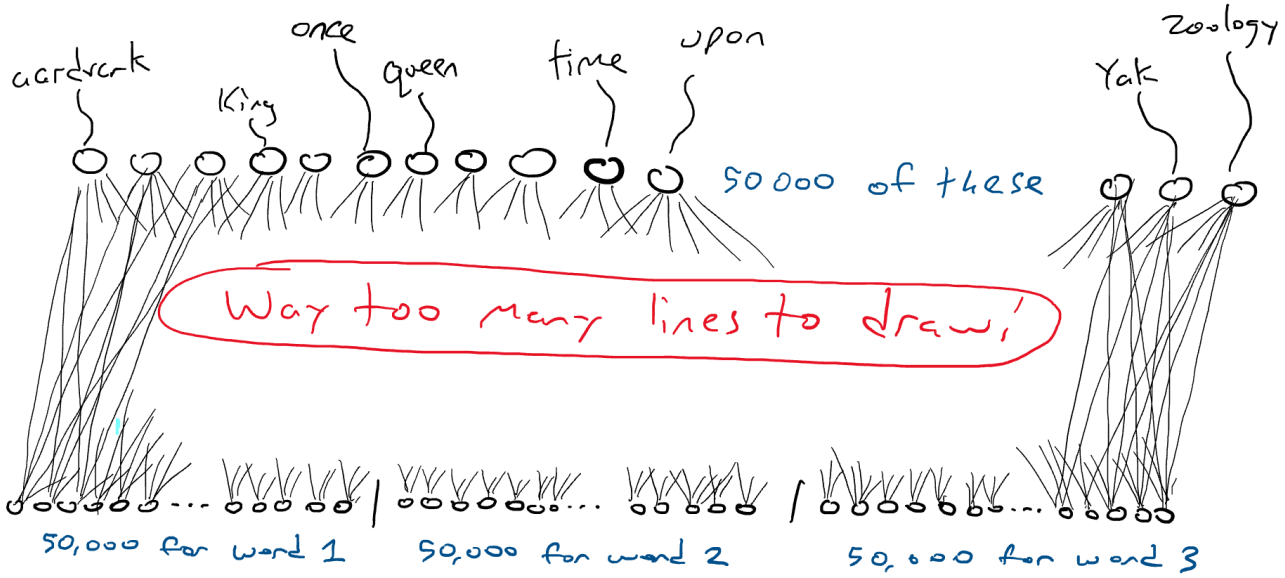
Voici où nous en sommes : si je veux faire un circuit simple qui prend un seul mot et produit un seul mot, je devrais faire un circuit qui a 50 000 capteurs (un pour chaque mot) et 50 000 sorties (une pour chaque bras percuteur). Je câblerais simplement chaque capteur à chaque bras percuteur pour un total de 50 000 x 50 000 = 2,5 milliards de fils.



Chaque cercle en bas détecte un mot. Il faut 50 000 capteurs pour reconnaître le mot « une fois ». Cette énergie est envoyée à travers un réseau arbitraire. Tous les cercles du haut sont reliés aux bras percuteurs pour chaque mot.
Tous les bras de l'attaquant reçoivent de l'énergie, mais l'un en recevra plus que les autres.

C'est un gros réseau !

Mais ça empire. Si je veux faire l'exemple "Il était une fois ___", je dois sentir quel mot se trouve dans chacune des trois positions d'entrée. J'aurais besoin de 50 000 x 3 = 150 000 capteurs. Câblé jusqu'à 50 000 bras percuteurs me donne 150 000 x 50 000 = 7,5 milliards de fils. À partir de 2023, la plupart des grands modèles de langage peuvent contenir 4 000 mots, le plus grand prenant 32 000 mots. Mes yeux pleurent.



Un réseau qui prend trois mots en entrée nécessite 50 000 capteurs par mot.

Nous allons avoir besoin de quelques astuces pour maîtriser cette situation. Nous allons procéder par étapes.

5.1 Encodeurs

La première chose que nous allons faire est de diviser notre circuit en deux circuits, l'un appelé encodeur et l'autre appelé décodeur. L'idée est que beaucoup de mots signifient à peu près la même chose. Considérez les expressions suivantes :

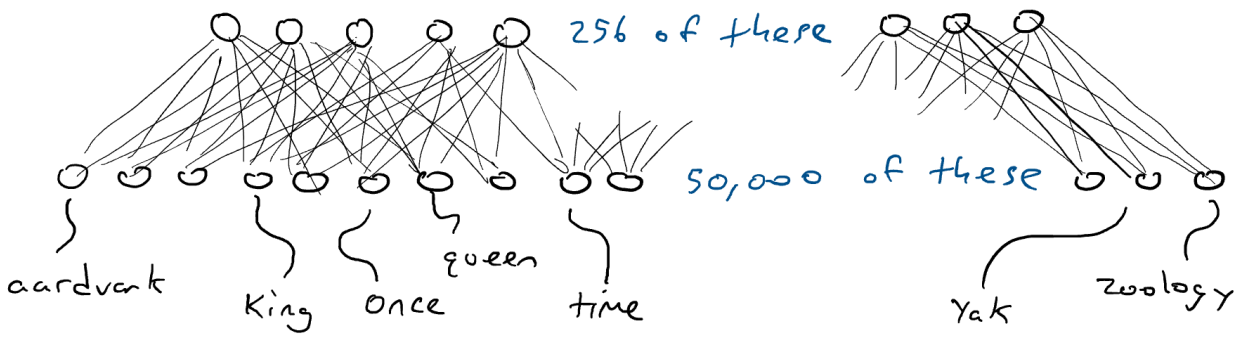
> Le roi était assis sur le ___ La
> reine était assise sur le ___ La
> princesse était assise sur le ___ Le
> régent était assis sur le ___

Une supposition raisonnable pour tous les blancs ci-dessus serait "trône" (ou peut-être "toilettes"). C'est-à-dire que je n'aurais peut-être pas besoin de fils séparés entre le «roi» et le «trône», ou entre la «reine» et le «trône», etc. Au lieu de cela, ce serait formidable si j'avais quelque chose qui signifie approximativement la royauté et chaque fois que je voir "roi" ou "reine", j'utilise plutôt cette chose intermédiaire. Ensuite, je n'ai qu'à m'inquiéter de la signification des mots

à peu près la même chose et ensuite que faire à ce sujet (envoyer beaucoup d'énergie au «trône»).

Voici donc ce que nous allons faire. Nous allons configurer un circuit qui prend 50 000 capteurs de mots et les mappe sur un ensemble de sorties plus petit, disons 256 au lieu de 50 000. Et au lieu de ne pouvoir déclencher qu'un seul bras de frappe, nous sommes capables d'écraser plusieurs bras à la fois. Chaque combinaison possible de bras percuteurs pourrait représenter un concept différent (comme « royauté » ou « mammifères blindés »). Ces 256 sorties nous donneraient la possibilité de représenter 2^{256} = 1,15 x 10^{78} concepts. En réalité, c'est encore plus parce que, comme dans l'exemple de la voiture, nous pouvons appuyer sur les freins à mi-chemin, chacune de ces 256 sorties peut être non seulement 1,0 ou 0,0, mais n'importe quel nombre entre les deux. Alors peut-être que la meilleure métaphore pour cela est que les 256 bras de l'attaquant s'écrasent, mais chacun s'écrase avec une force différente.

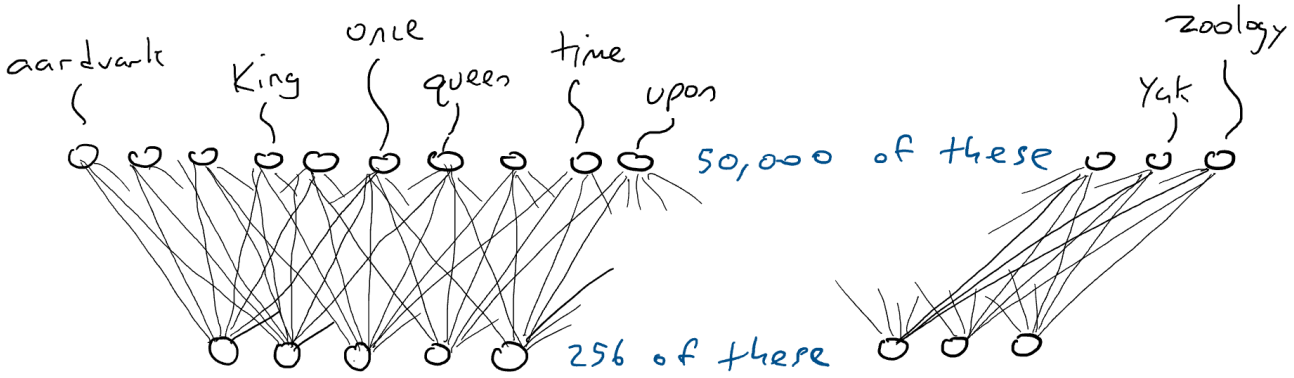
D'accord... donc auparavant, un mot nécessitait l'activation de l'un des 50 000 capteurs. Maintenant, nous avons fait bouillir un capteur activé et 49 999 capteurs désactivés en 256 nombres. Ainsi, "roi" pourrait être [0.1, 0.0, 0.9, ..., 0.4] et "reine" pourrait être [0.1, 0.1, 0.9, ..., 0.4] qui sont presque identiques l'un à l'autre. J'appellerai ces listes d'encodages de nombres (également appelé l'état caché pour des raisons historiques mais je ne veux pas expliquer cela, nous nous en tiendrons donc à l'encodage). Nous appelons le circuit qui écrase nos 50 000 capteurs en 256 sorties l'encodeur. Il ressemble à ceci :



Un réseau d'encodeurs écrasant les 50 000 valeurs de capteur nécessaires pour détecter un seul mot dans un encodage de 256 chiffres (bleu plus clair et plus foncé utilisé pour indiquer des valeurs supérieures ou inférieures).

5.2 Décodeurs

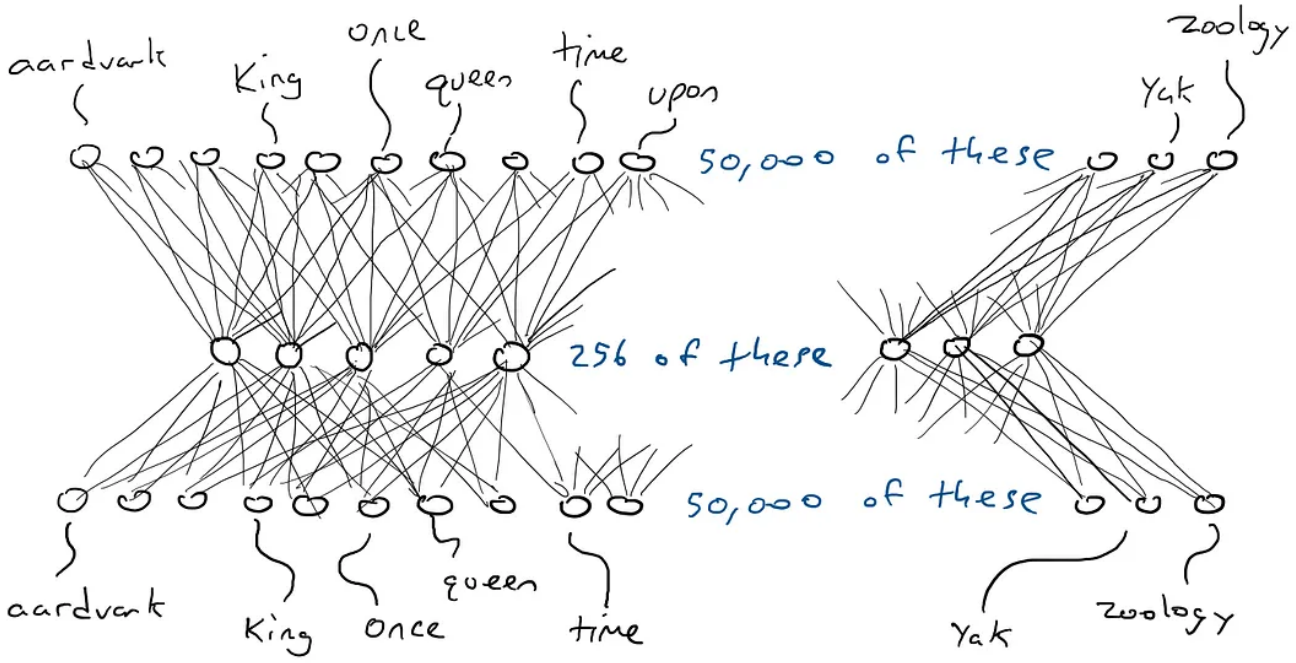
Mais l'encodeur ne nous dit pas quel mot doit venir ensuite. Nous apparions donc notre encodeur avec un réseau de décodeurs. Le décodeur est un autre circuit qui prend 256 chiffres composant le codage et active les 50 000 bras de frappe d'origine, un pour chaque mot. Nous choisirions alors le mot avec la puissance électrique la plus élevée. Voici à quoi cela ressemblerait :



Un réseau de décodeurs, élargissant les 256 valeurs du codage en valeurs d'activation pour les 50 000 bras percuteurs associé à chaque mot possible. Un mot active le plus élevé.

5.3 Encodeurs et décodeurs ensemble

Voici l'encodeur et le décodeur travaillant ensemble pour créer un grand réseau de neurones :



Un réseau codeur-décodeur. C'est juste un décodeur assis sur un encodeur.

Et, soit dit en passant, une seule entrée de mot vers une seule sortie de mot passant par l'encodage n'a besoin que de (50 000 x 256) x 2 = 25,6 millions de paramètres. Cela semble beaucoup mieux.

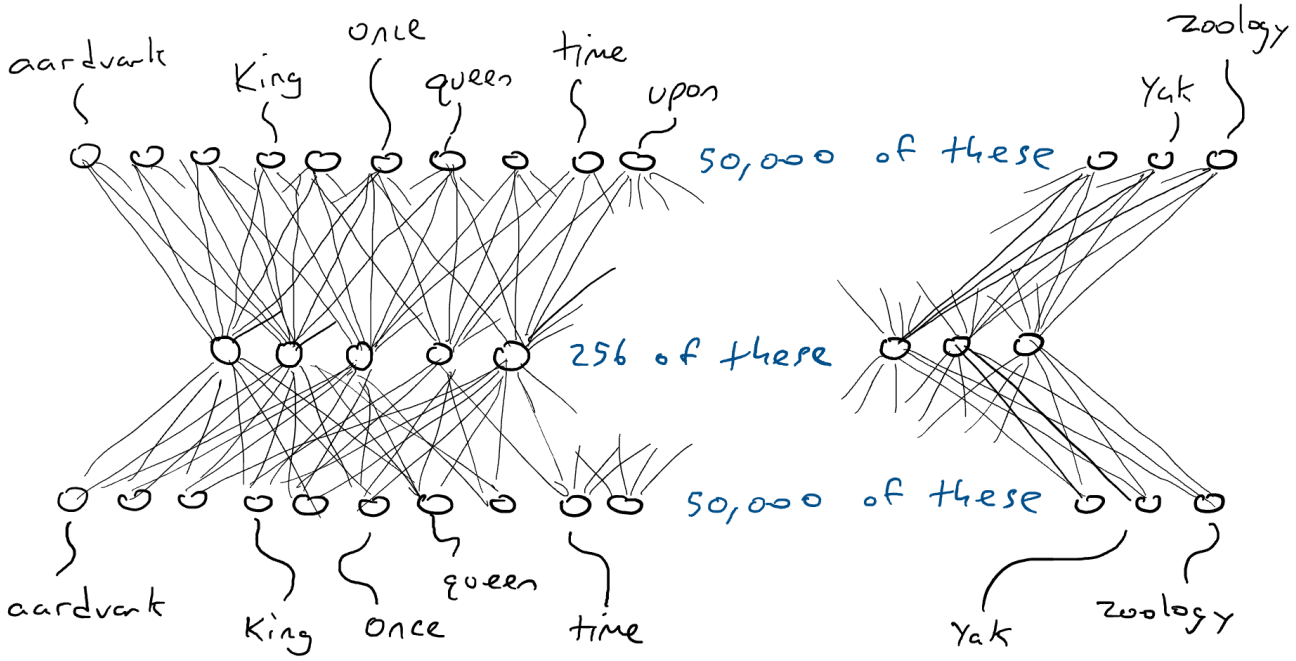
Cet exemple était pour une entrée de mot et produisant une sortie de mot, donc nous aurions 50 000 xn entrées si nous voulions lire n mots, et 256 xn pour l'encodage

Mais pourquoi cela fonctionne-t-il ? En forçant 50 000 mots à tenir dans un petit ensemble de nombres, nous forçons le réseau à faire des compromis et à regrouper les mots qui pourraient déclencher la même estimation de mot de sortie. Cela ressemble beaucoup à la compression de fichiers. Lorsque vous compressez un document texte, vous obtenez un document plus petit qui n'est plus lisible. Mais vous pouvez décompresser le document et récupérer le texte lisible d'origine. Cela peut être fait parce que le programme zip remplace certains modèles de mots par une notation abrégée. Ensuite, lorsqu'il décompresse, il sait dans quel texte revenir pour la notation abrégée. Nos circuits d'encodeur et de décodeur apprennent une configuration de résistances et de portes qui compriment puis décompressent les mots.

5.4 Auto-supervision

Comment savoir quel encodage est le meilleur pour chaque mot ? Autrement dit, comment savons-nous que l'encodage pour « roi » devrait être similaire à l'encodage pour « reine » au lieu de « tatous » ?

En tant qu'expérience de pensée, considérez un réseau d'encodeur-décodeur qui devrait prendre un seul mot (50 000 capteurs) et produire exactement le même mot en sortie. C'est une chose idiote à faire, mais c'est assez instructif pour ce qui va suivre.



Un réseau codeur-décodeur formé pour sortir le même mot que l'entrée (c'est la même image qu'avant mais avec couleur d'activation).

Je mets le mot "roi" et un seul capteur envoie son signal électrique à travers l'encodeur et active partiellement 256 valeurs dans l'encodage au milieu. Si l'encodage est correct, alors le décodeur enverra le signal électrique le plus élevé au même mot, "roi". D'accord, facile ? Pas si vite. Je suis tout aussi susceptible de voir le bras percuteur avec le mot "tatou" avec l'énergie d'activation la plus élevée. Supposons que le bras percuteur du « roi » reçoive un signal électrique de 0,051 et que le bras percuteur du « tatou » reçoive un signal électrique de 0,23. En fait, je me fiche même de la valeur de « tatou ». Je peux juste regarder l'énergie de sortie pour "roi" et savoir que ce n'était pas 1.0. La différence entre 1,0 et 0.

On fait ça pour tous les mots. L'encodeur va devoir faire des compromis car le 256 est bien inférieur à 50 000. C'est-à-dire que certains mots vont devoir utiliser le même

combinaisons d'énergie d'activation au milieu. Ainsi, lorsqu'on lui donne le choix, il va vouloir que l'encodage pour "roi" et "reine" soit presque identique et que l'encodage pour "tatou" soit très différent. Cela donnera au décodeur une meilleure chance de deviner le mot en regardant simplement les 256 valeurs d'encodage. Et si le décodeur voit une combinaison particulière de 256 valeurs et devine "roi" avec 0,43 et "reine" avec 0,42, nous allons être d'accord avec cela tant que "roi" et "reine" obtiennent les signaux électriques les plus élevés et chaque des 49 998 bras d'attaquant obtiennent des nombres plus petits. Une autre façon de dire cela est que nous allons probablement être plus d'accord avec le fait que le réseau soit confus entre les rois et les reines que si le réseau est confus entre les rois et les tatous.

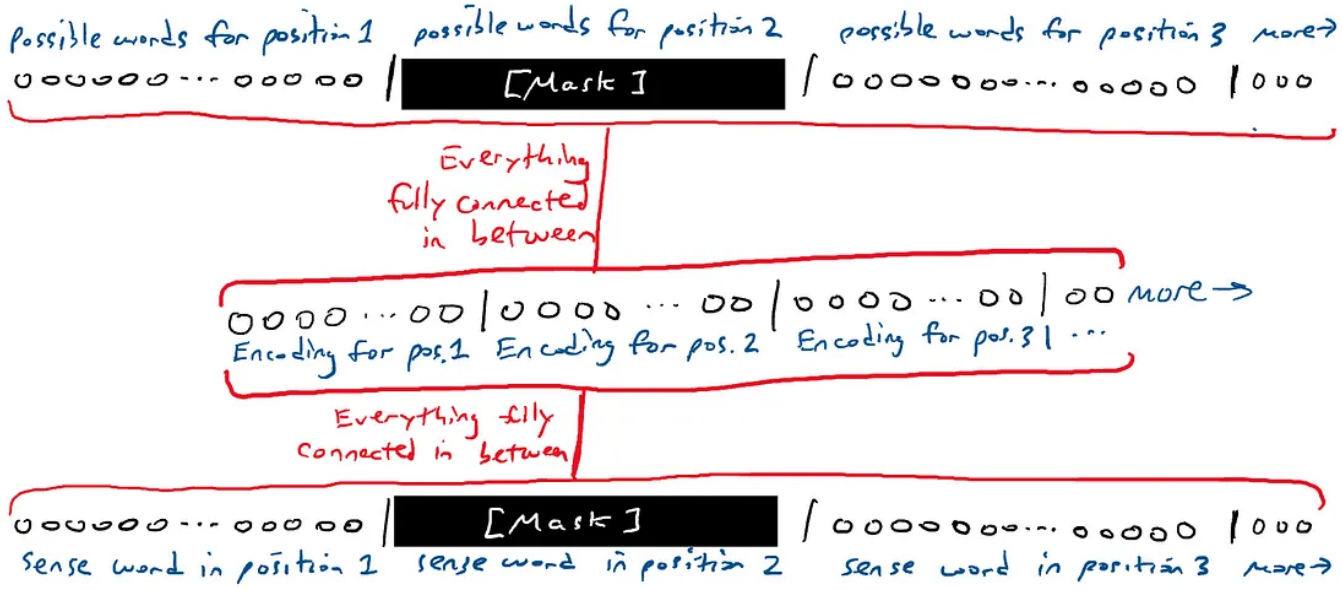
Nous disons que le réseau de neurones est auto-supervisé car, contrairement à l'exemple de la voiture, vous n'avez pas à collecter de données séparées pour tester la sortie. Nous comparons simplement la sortie à l'entrée - nous n'avons pas besoin d'avoir des données séparées pour l'entrée et la sortie.

5.5 Modèles de langage masqué

Si l'expérience de pensée ci-dessus semble idiote, c'est la pierre angulaire de ce qu'on appelle des modèles de langage masqué. L'idée d'un modèle de langage masqué est de prendre une séquence de mots et de générer une séquence de mots. L'un des mots de l'entrée et de la sortie est effacé.

> Le [MASQUE] était assis sur le trône.

Le réseau devine tous les mots. Eh bien, il est assez facile de deviner les mots non masqués. Nous ne nous soucions vraiment que de la supposition du réseau sur le mot masqué. C'est-à-dire que nous avons 50 000 bras percuteurs pour chaque mot dans la sortie. On regarde les 50 000 bras d'attaquant pour le mot masqué.



Masquage d'une séquence. Je commence à en avoir assez de dessiner beaucoup de lignes de connexion, donc je vais juste dessiner des lignes rouges pour signifier beaucoup-et-beaucoup de connexions entre tout ce qui est au-dessus et en dessous.

Nous pouvons déplacer le masque et demander au réseau de deviner différents mots à différents endroits.

Un type spécial de modèle de langage masqué n'a que le masque à la fin. C'est ce qu'on appelle un modèle génératif car le masque qu'il devine est toujours le mot suivant dans la séquence, ce qui équivaut à générer le mot suivant comme si le mot suivant n'existait pas. Comme ça:

> Le masque]
>
> La reine [MASQUE]
>
> La reine s'est assise sur le [MASQUE] La
>
> reine s'est assise sur le [MASQUE] La reine
>
> s'est assise sur le [MASQUE]

Nous appelons également cela un modèle auto-régressif. Le mot régressif ne sonne pas si bien. Mais la régression signifie simplement essayer de comprendre la relation entre les choses, comme les mots qui ont été entrés et les mots qui devraient être sortis. Auto signifie « soi ». Un modèle autorégressif est auto-prédictif. Il prédit un mot. Ensuite, ce mot est utilisé pour prédire le mot suivant, qui est utilisé pour prédire le mot suivant, et ainsi de suite. Il y a quelques

des implications intéressantes sur lesquelles nous reviendrons plus tard.

6. Qu'est-ce qu'un transformateur ?

Au moment d'écrire ces lignes, nous entendons beaucoup parler de choses appelées GPT-3 et GPT-4 et ChatGPT. GPT est une marque particulière d'un type de grand modèle de langage développé par une société appelée OpenAI. GPT est l'abréviation de "Generative Pre-trained Transformer". Décomposons cela :

- Génératif. Le modèle est capable de générer des continuations à l'entrée fournie. C'est-à-dire que, étant donné du texte, le modèle essaie de deviner quels mots viennent ensuite.
- Pré-formé. Le modèle est formé sur un très grand corpus de texte général et est destiné à être formé une fois et utilisé pour beaucoup de choses différentes sans avoir besoin d'être recyclé à partir de zéro.

En savoir plus sur la pré-formation... Le modèle est formé sur un très grand corpus de texte général qui couvre ostensiblement un grand nombre de sujets imaginables. Cela signifie plus ou moins "gratté d'Internet" par opposition à tiré de certains référentiels de texte spécialisés. En s'entraînant sur un texte général, un modèle linguistique est plus capable de répondre à un plus large éventail d'entrées que, par exemple, un modèle linguistique formé sur un type de texte très spécifique, comme des documents médicaux. Un modèle de langage formé sur un corpus général peut théoriquement répondre raisonnablement à tout ce qui pourrait apparaître dans un document sur Internet. Cela pourrait convenir à un texte médical. Un modèle de langage formé uniquement sur des documents médicaux peut très bien répondre aux entrées liées à des contextes médicaux, mais être assez mauvais pour répondre à d'autres entrées comme le bavardage ou les recettes.

Soit le modèle est assez bon pour tant de choses que l'on n'a jamais besoin de former son propre modèle, soit on peut faire quelque chose appelé réglage fin, ce qui signifie prendre le modèle pré-formé et faire quelques mises à jour pour le faire fonctionner mieux sur un tâche spécialisée (comme médicale).

Passons maintenant au transformateur...

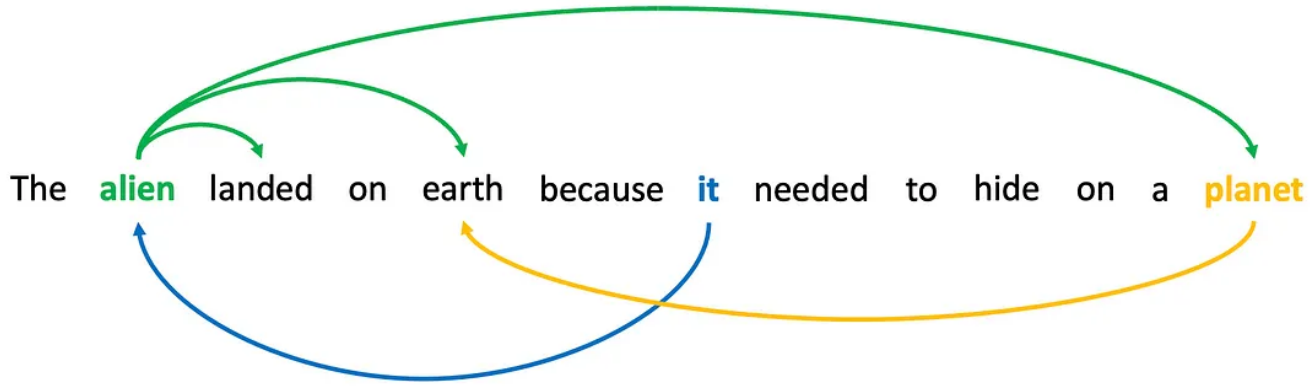
- Transformateur. Un type spécifique de modèle d'apprentissage en profondeur d'encodeur-décodeur auto-supervisé avec des propriétés très intéressantes qui le rendent bon pour la modélisation du langage.

Un transformateur est un type particulier de modèle d'apprentissage en profondeur qui transforme l'encodage

d'une manière particulière qui permet de deviner plus facilement le mot masqué. Il a été introduit par un article intitulé L'attention est tout ce dont vous avez besoin par Vaswani et al. en 2017. Au cœur d'un transformateur se trouve le réseau codeur-décodeur classique. L'encodeur effectue un processus d'encodage très standard. Tellement vanille que vous seriez choqué. Mais ensuite, cela ajoute quelque chose d'autre appelé l'auto-attention.

6.1 Auto-attention

Voici l'idée d'auto-attention : certains mots d'une séquence sont liés à d'autres mots de la séquence. Considérez la phrase "L'extraterrestre a atterri sur terre parce qu'il avait besoin de se cacher sur une planète." Si nous devions masquer le deuxième mot, "extraterrestre" et demander à un réseau de neurones de deviner le mot, il aurait une meilleure chance à cause de mots comme "atterri" et "terre". De même, si nous masquions « ça » et demandions au réseau de deviner le mot, la présence du mot « extraterrestre » pourrait le rendre plus susceptible de préférer « ça » à « il » ou « elle ».



Les mots sont liés à d'autres mots par fonction, en se référant à la même chose, ou en informant les significations de chacun autre.

Nous disons que les mots dans une séquence s'occupent d'autres mots parce qu'ils capturent une sorte de relation. La relation n'est pas forcément connue. Il peut s'agir de pronoms résolutifs, il peut s'agir d'une relation entre verbe et sujet, il peut s'agir de deux mots relatifs au même concept ("terre" et "planète"). Quoi qu'il en soit, savoir qu'il existe une sorte de relation entre les mots est utile pour la prédiction.

La section suivante abordera les mathématiques de l'attention personnelle, mais l'essentiel est qu'un transformateur apprend quels mots d'une séquence d'entrée sont liés, puis crée un nouveau codage pour chaque position dans la séquence d'entrée qui est une fusion de tous les mots apparentés. Vous pouvez en quelque sorte penser à cela comme apprendre à inventer un nouveau mot qui est un mélange

de "alien" et "atterri" et "terre" (alianearth ?). Cela fonctionne parce que chaque mot est encodé sous la forme d'une liste de nombres. Si alien = [0.1, 0.2, 0.3, ..., 0.4] et débarqué = [0.5, 0.6, 0.7, ..., 0.8] et terre = [0.9, 1.0, 1.1, ..., 1.2], alors la position du deuxième mot pourrait être codé comme la somme de tous ces encodages, [1.5, 1.8, 2.1, ..., 2.4], qui lui-même ne correspond à aucun mot mais capture des morceaux de tous les mots. De cette façon, lorsque le décodeur voit enfin ce nouveau codage pour le mot en deuxième position, il a beaucoup d'informations sur la façon dont le mot a été utilisé dans la séquence et fait ainsi une meilleure estimation des masques. (L'exemple ajoute simplement l'encodage mais ce sera un peu plus compliqué que cela).

6.2. Comment fonctionne l'auto-attention ?

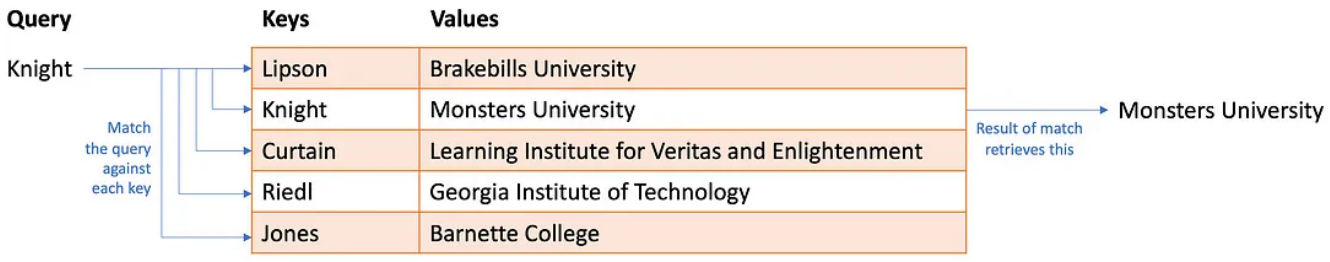
L'auto-attention est l'amélioration significative par rapport aux réseaux d'encodeur-décodeur vanille, donc si vous voulez en savoir plus sur son fonctionnement, continuez à lire. Sinon, n'hésitez pas à sauter cette section. TL; DR : l'auto-attention est un nom fantaisiste de l'opération mathématique appelée produit scalaire.

L'auto-attention se déroule en trois étapes.

(1) Nous encodons chaque mot dans la séquence d'entrée comme d'habitude. Nous faisons quatre copies des codages de mots. Celui que nous appelons le résiduel et mis de côté pour être conservé en lieu sûr.

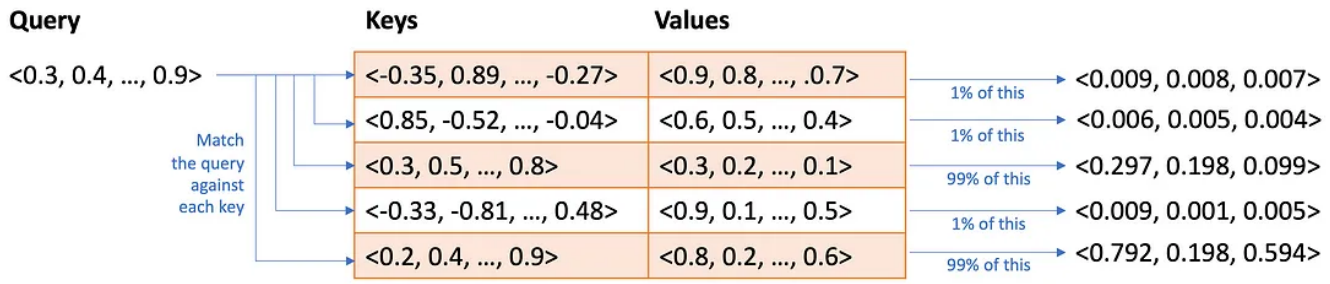
(2) Nous exécutons un deuxième cycle d'encodage (nous encodons un encodage) sur les trois autres. Chacun subit un processus d'encodage différent, de sorte qu'ils deviennent tous différents. Nous appelons l'une une requête (q), une une clé (k) et une une valeur (v).

Je veux que vous réfléchissiez à une table de hachage (également appelée dictionnaire en python). Vous avez tout un tas d'informations stockées dans une table. Chaque ligne de la table a une clé, un identifiant unique et la valeur, les données étant stockées dans la ligne. Pour récupérer certaines informations de la table de hachage, vous donnez une requête. Si la requête correspond à la clé, vous extrayez la valeur.



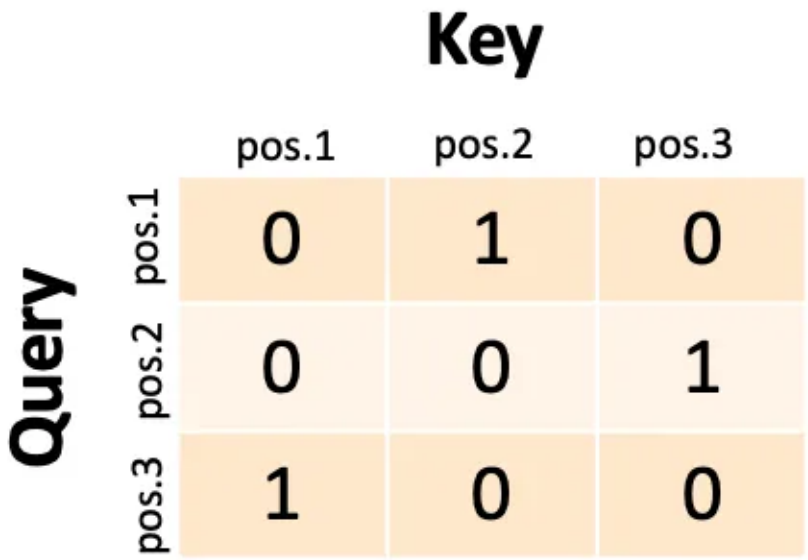
Une table de hachage que l'on pourrait utiliser pour demander à quelle université travaille un professeur.

L'auto-attention fonctionne un peu comme une table de hachage floue. Vous fournissez une requête et au lieu de rechercher une correspondance exacte avec une clé, il trouve des correspondances approximatives en fonction de la similitude entre la requête et la clé. Mais que se passe-t-il si le match n'est pas parfait ? Il renvoie une fraction de la valeur. Eh bien, cela n'a de sens que si la requête, les clés et les valeurs sont toutes numériques. Ce qu'ils sont :



Une table de hachage avec des correspondances partielles.

C'est donc ce que nous allons faire. Pour chaque position de mot dans l'entrée, nous allons prendre le codage q et le codage k et calculer la similarité. Nous utilisons ce qu'on appelle un produit scalaire, également appelé similarité cosinus. Pas important. Le fait est que chaque mot est une liste de 256 nombres (basé sur notre exemple précédent) et nous pouvons calculer la similarité des listes de nombres et enregistrer la similarité dans une matrice. Nous appelons cette matrice les scores d'auto-attention. Si nous avions une séquence de saisie de trois mots, nos scores d'attention pourraient ressembler à ceci :



Chaque cellule indique à quel point le mot codé dans une position s'occupe du mot codé dans une autre position.

Le réseau traite le premier mot comme une requête et il correspond à la deuxième clé (nous pourrions dire que le premier mot « assiste » au deuxième mot). Si le deuxième mot était une requête, il correspondrait à la troisième clé. Si le troisième mot était une requête, il correspondrait à la première clé. En réalité, nous n'aurions jamais des uns et des zéros comme celui-ci ; nous aurions des correspondances partielles entre 0 et 1 et chaque requête (ligne) correspondrait partiellement à plusieurs clés (colonnes).

Maintenant, pour nous en tenir à la métaphore de récupération, nous multiplions cette matrice par les codages v et quelque chose d'intéressant se produit. Supposons que nos encodages v ressemblent à ceci :

	Values									
pos.1	0.10	0.11	0.12	0.13	0.14	0.15	0.16	0.17	0.18	0.19
pos.2	0.20	0.21	0.22	0.23	0.24	0.25	0.26	0.27	0.28	0.29
pos.3	0.30	0.31	0.32	0.33	0.34	0.35	0.36	0.37	0.38	0.39

Chaque ligne est un encodage pour un mot dans une séquence.

Autrement dit, le premier mot a été codé comme une liste de nombres 0,10...0,19, le deuxième mot a été codé comme une liste de nombres 0,20...0,29 et le troisième mot a été codé comme une liste de nombres.

nombres 0,30...0,39. Ces chiffres sont composés à des fins d'illustration et ne seraient jamais aussi bien rangés.

Query	Key: pos.1	Key: pos.2	Key: pos.3
pos.1	0	1	0
pos.2	0	0	1
pos.3	1	0	0

X

Values										
pos.1	0.10	0.11	0.12	0.13	0.14	0.15	0.16	0.17	0.18	0.19
pos.2	0.20	0.21	0.22	0.23	0.24	0.25	0.26	0.27	0.28	0.29
pos.3	0.30	0.31	0.32	0.33	0.34	0.35	0.36	0.37	0.38	0.39

Multiplier l'attention par les valeurs.

La première requête correspond à la deuxième clé et récupère donc le deuxième mot codé. La deuxième requête correspond à la troisième clé et récupère donc le troisième mot codé. La troisième requête correspond à la première clé et récupère donc le premier mot codé. Ce que nous avons effectivement fait, c'est d'échanger des lignes !

Values									
pos.1	0.20	0.21	0.22	0.23	0.24	0.25	0.26	0.27	0.28	0.29
pos.2	0.30	0.31	0.32	0.33	0.34	0.35	0.36	0.37	0.38	0.39
pos.3	0.10	0.11	0.12	0.13	0.14	0.15	0.16	0.17	0.18	0.19

En pratique, les scores ne seraient pas des uns et des zéros parfaits et le résultat serait un peu de chaque encodage mélangé (par exemple 97 % du mot un plus 1 % ou du mot trois plus 2 % du mot deux). Mais cela illustre à quel point l'attention à soi est un mélange et un échange. Dans cette version extrême, le premier mot a été remplacé par le deuxième mot, et ainsi de suite. Alors peut-être que le mot "terre" a été remplacé par le mot "planète".

Comment savons-nous que nous avons correctement encodé q, k et v ? Si la capacité globale du réseau à deviner le meilleur mot pour le masque s'améliore, nous encodons q, k et v correctement. Sinon, on change les paramètres pour encoder un peu différemment la prochaine fois.

(3) La troisième chose que nous faisons est de prendre le résultat de tous ces calculs et de l'ajouter au résidu. Rappelez-vous cette première copie de l'encodage original que nous avons mise de côté. C'est vrai, nous y ajoutons la version mixte et échangée. Maintenant, "terre" n'est pas seulement un encodage de "terre", mais une sorte de mot imaginaire qui est un mélange de "terre" et de "planète"... la terre ? ealanet ? Pas vraiment comme ça. Quoi qu'il en soit, c'est l'encodage transformé final qui sera envoyé au décodeur. Nous pouvons probablement convenir qu'avoir un faux mot dans chaque position qui

encode réellement deux mots ou plus est plus utile pour faire des prédictions basées sur un seul mot par position.

Ensuite, vous faites cela plusieurs fois l'une après l'autre (plusieurs couches).

Je laisse de côté beaucoup de détails sur la façon dont l'encodage final de l'encodeur entre dans le décodeur (un autre tour d'attention, appelé source-attention où les encodages de l'encodeur de chaque position sont utilisés comme q et k à appliquer contre encore un autre version différente de v), mais à ce stade, vous devriez avoir un aperçu général des choses. À la fin, le décodeur, prenant en compte l'encodage de l'encodeur, envoie de l'énergie aux bras percuteurs pour les mots, et nous choisissons le mot le plus fortement excité.

7. Pourquoi les grands modèles de langage sont-ils si puissants ?
Alors qu'est-ce que tout cela veut dire? Les grands modèles de langage, y compris ChatGPT, GPT-4 et d'autres, font exactement une chose : ils prennent un tas de mots et essaient de deviner quel mot devrait venir ensuite. S'il s'agit de « raisonner » ou de « penser », alors ce n'est qu'une forme très spécialisée.

Mais même ce formulaire spécialisé semble très puissant car ChatGPT et similaires peuvent apparemment très bien faire beaucoup de choses : écrire de la poésie, répondre à des questions sur la science et la technologie, résumer des documents, rédiger des e-mails et même écrire du code, pour ne citer que quelques choses. Pourquoi devraient-ils si bien fonctionner ?

La sauce secrète est double. Le premier dont nous avons déjà parlé : le transformateur apprend à mélanger les contextes de mots d'une manière qui le rend très efficace pour deviner le mot suivant. L'autre partie de la sauce secrète est la façon dont les systèmes sont formés. Les grands modèles de langage sont formés sur des quantités massives d'informations extraites d'Internet. Cela inclut les livres, les blogs, les sites d'actualités, les articles de wikipedia, les discussions reddit, les conversations sur les réseaux sociaux. Pendant la formation, nous alimentons un extrait de texte à partir de l'une de ces sources et lui demandons de deviner le mot suivant. N'oubliez pas : auto-supervisé. S'il se trompe, nous modifions un peu le modèle jusqu'à ce qu'il soit correct. Si nous devions réfléchir à ce à quoi un LLM est formé, c'est produire un texte qui aurait pu raisonnablement apparaître sur Internet. Il ne peut pas mémoriser Internet,

Il est important de ne pas sous-estimer la diversité des textes sur Internet en termes de sujets. Les LLM ont tout vu. Ils ont vu des milliards de conversations sur à peu près

chaque sujet. Ainsi, un LLM peut produire des mots qui semblent avoir une conversation avec vous. Il a vu des milliards de poèmes et de paroles de musique sur à peu près tout ce qui est imaginable, de sorte qu'il peut produire un texte qui ressemble à de la poésie. Il a vu des milliards de devoirs et leurs solutions, il peut donc faire des suppositions raisonnables sur vos devoirs, même s'ils sont légèrement différents. Il a vu des milliards de questions de test standardisées et leurs réponses. Pensons-nous vraiment que les questions SAT de cette année sont si différentes de celles de l'année dernière ? Il a vu des gens parler de leurs plans de vacances, il peut donc deviner des mots qui ressemblent à des plans de vacances. Il a vu des milliards d'exemples de code faire toutes sortes de choses. Une grande partie de ce que font les programmeurs informatiques consiste à assembler des morceaux de code pour faire des choses très typiques et bien comprises dans de plus gros morceaux de code. Ainsi, Les LLM peuvent écrire ces petits extraits courants pour vous. Il a vu des milliards d'exemples de code erroné et leurs corrections sur stackoverflow.com. Ouais, donc il peut prendre votre code cassé et suggérer des correctifs. Il a vu des milliards de personnes tweeter qu'elles ont touché un poêle chaud et se sont brûlé les doigts, donc les LLM connaissent un peu de bon sens. Il a lu de nombreux articles scientifiques, il peut donc deviner des faits scientifiques bien connus, même s'ils ne vous sont pas bien connus. Il a vu des milliards d'exemples de personnes résumant, réécrivant du texte en puces, décrivant comment rendre le texte plus grammatical, concis ou persuasif. Il a vu des milliards de personnes tweeter qu'elles ont touché un poêle chaud et se sont brûlé les doigts, donc les LLM connaissent un peu de bon sens. Il a lu de nombreux articles scientifiques, il peut donc deviner des faits scientifiques bien connus, même s'ils ne vous sont pas bien connus. Il a vu des milliards d'exemples de personnes résumant, réécrivant du texte en puces, décrivant comment rendre le texte plus grammatical, concis ou persuasif. Il a vu des milliards de personnes tweeter qu'elles ont touché un poêle chaud et se sont brûlé les doigts, donc les LLM connaissent un peu de bon sens. Il a lu de nombreux articles scientifiques, il peut donc deviner des faits scientifiques bien connus, même s'ils ne vous sont pas bien connus. Il a vu des milliards d'exemples de personnes résumant, réécrivant du texte en puces, décrivant comment rendre le texte plus grammatical, concis ou persuasif.

Voici le point : lorsque vous demandez à ChatGPT ou à un autre grand modèle de langage de faire quelque chose d'intelligent - et cela fonctionne - il y a de très bonnes chances que vous lui ayez demandé de faire quelque chose dont il a vu des milliards d'exemples. Et même si vous trouvez quelque chose de vraiment unique comme "dis-moi ce que Flash Gordon ferait après avoir mangé six burritos" (est-ce unique, je ne sais même pas), il a vu Fan Fiction sur Flash Gordon et il a vu des gens parler de manger trop de burritos et peut - à cause de l'attention personnelle - mélanger et assortir des morceaux pour assembler une réponse sonore raisonnable.

Notre premier réflexe lorsque nous interagissons avec un grand modèle de langage ne devrait pas être "wow ces choses doivent être vraiment intelligentes ou vraiment créatives ou vraiment compréhensives". Notre premier réflexe devrait être « Je lui ai probablement demandé de faire quelque chose dont il a déjà vu des fragments ». Cela pourrait signifier qu'il est toujours très utile, même s'il ne s'agit pas de "penser très fort" ou de "faire un raisonnement vraiment sophistiqué".

Nous n'avons pas besoin d'utiliser l'anthropomorphisation pour comprendre ce qu'elle fait pour nous fournir une réponse.

Une dernière note sur ce thème : en raison de la manière dont les grands modèles de langage fonctionnent et de la manière dont ils sont entraînés, ils ont tendance à fournir des réponses qui sont en quelque sorte la réponse médiane. Il peut sembler très étrange pour moi de dire que le modèle a tendance à donner des réponses moyennes après avoir demandé une histoire sur Flash Gordon. Mais dans le contexte d'une histoire ou d'un poème, les réponses peuvent être considérées comme étant ce que beaucoup de gens (écrivant sur Internet) trouveraient s'ils devaient faire des compromis. Ce ne sera pas mal. Cela pourrait être assez bon selon les normes d'une seule personne assise essayant de penser à quelque chose par elle-même. Mais vos histoires et vos poèmes sont probablement aussi moyens (mais ils sont spéciaux pour vous). Désolé.

8. À quoi dois-je faire attention ?

Il y a des implications vraiment subtiles qui découlent du fonctionnement des transformateurs et de la façon dont ils sont formés. Voici les implications directes des détails techniques.

1. Les grands modèles linguistiques sont formés sur Internet. Cela signifie qu'ils se sont également entraînés sur toutes les parties sombres de l'humanité. Les grands modèles de langage se sont entraînés sur les diatribes racistes, les chapes sexistes, les insultes de toutes sortes contre tous les types de personnes, les personnes faisant des hypothèses stéréotypées sur les autres, les théories du complot, la désinformation politique, etc. Cela signifie que les mots qu'un modèle de langage choisit de générer peuvent régurgiter tel langage.

2. Les grands modèles linguistiques n'ont pas de "croyances fondamentales". Ce sont des devineurs de mots; ils essaient de prédire quels seraient les prochains mots si la même phrase apparaissait sur Internet. Ainsi, on peut demander à un grand modèle de langage d'écrire une phrase en faveur de quelque chose, ou contre cette même chose, et le modèle de langage se conformera dans les deux sens. Ce ne sont pas des indications qu'il croit une chose ou l'autre, ou qu'il change ses croyances, ou qu'une est plus juste qu'une autre. Si les données de formation ont plus d'exemples d'une chose par rapport à une autre chose, alors un grand modèle de langage aura tendance à répondre de manière plus cohérente avec tout ce qui apparaît plus souvent dans ses données de formation, car il apparaît plus souvent sur Internet. Rappelez-vous : le modèle s'efforce d'imiter la réponse la plus courante.

3. Les grands modèles de langage n'ont aucun sens de la vérité ou du bien ou du mal. Il y a des choses que nous tenons pour des faits, comme la Terre étant ronde. Un LLM aura tendance à dire cela. Mais si le contexte est correct, il dira également le contraire car Internet contient un texte sur la Terre plate. Il n'y a aucune garantie qu'un LLM fournira

la vérité. Il peut y avoir une tendance à deviner les mots dont nous convenons qu'ils sont vrais, mais c'est le plus proche que nous puissions faire pour faire des affirmations sur ce qu'un LLM "sait" sur la vérité ou le bien ou le mal.

4. Les grands modèles de langage peuvent faire des erreurs. Les données de formation peuvent contenir beaucoup de données incohérentes. L'auto-attention peut ne pas s'occuper de toutes les choses que nous voulons qu'elle fasse lorsque nous posons une question. En tant que devineur de mots, il peut faire des suppositions malheureuses. Parfois, les données d'apprentissage ont vu un mot tellement de fois qu'elles préfèrent ce mot même s'il n'a pas de sens pour l'entrée. Ce qui précède conduit à un phénomène appelé "hallucination" où un mot est deviné qui n'est pas dérivé de l'entrée ni "correct". Les LLM ont tendance à deviner les petits nombres au lieu des grands nombres, car les petits nombres sont plus courants. Les LLM ne sont donc pas bons en maths. Les LLM ont une préférence pour le nombre "42" parce que les humains le font à cause d'un livre célèbre particulier. Les LLM ont des préférences pour les noms plus courants, ils peuvent donc inventer les noms des auteurs.

5. Les grands modèles de langage sont auto-régressifs. Ainsi, lorsqu'ils font des suppositions que nous pourrions considérer comme médiocres, ces mots devinés sont ajoutés à leurs propres entrées pour faire deviner le mot suivant. C'est-à-dire que les erreurs s'accumulent. Même s'il n'y a que 1% de chances d'erreur, l'attention portée à ce mauvais choix peut alors doubler cette erreur. Même si une seule erreur est commise, tout ce qui suit peut être lié à cette erreur. Ensuite, le modèle de langage pourrait faire des erreurs supplémentaires en plus de cela. Les transformateurs n'ont aucun moyen de « changer d'avis », de réessayer ou de se corriger eux-mêmes. Ils vont avec le courant.

6. Il faut toujours vérifier les sorties d'un grand modèle de langage. Si vous lui demandez de faire des choses que vous ne pouvez pas vérifier vous-même avec compétence, vous devriez alors vous demander si vous êtes d'accord pour agir sur les erreurs qui sont commises. Pour les tâches à faible enjeu, comme écrire une nouvelle, cela peut convenir. Pour les tâches à enjeux élevés, comme essayer d'obtenir des informations pour décider dans quelles actions investir, ces erreurs pourraient peut-être vous amener à prendre une décision très coûteuse.

7. L'auto-attention signifie que plus vous fournissez d'informations dans l'invite de saisie, plus la réponse sera spécialisée car elle mélangera plus de vos mots dans ses suppositions. La qualité de la réponse est directement proportionnelle à la qualité de l'invite de saisie. De meilleures invites produisent de meilleurs résultats. Essayez plusieurs invites différentes et voyez ce qui vous convient le mieux. Ne présumez pas que le modèle de langage "obtient" ce que vous

essaient de faire et donneront leur meilleur coup du premier coup.

8. Vous n'êtes pas vraiment en train de "converser" avec un grand modèle de langage. Un grand modèle de langage ne « se souvient » pas de ce qui s'est passé dans l'échange. Votre entrée entre. La réponse sort. Le LLM ne se souvient de rien. Votre entrée initiale, la réponse et votre réponse à la réponse entrent. Ainsi, s'il semble qu'il se souvienne, c'est parce que le journal des conversations devient une nouvelle entrée. Il s'agit d'une astuce de programmation sur le front-end pour donner l'impression que le Large Language Model est en train de converser. Il restera probablement sur le sujet à cause de cette astuce, mais rien ne garantit qu'il ne contredira pas ses réponses précédentes. De plus, il y a une limite au nombre de mots pouvant être introduits dans le grand modèle de langue (actuellement ChatGPT autorise environ 4 000 mots, et GPT-4 autorise environ 32 000 mots). Les tailles d'entrée peuvent être assez importantes, de sorte que la conversation semblera souvent rester cohérente pendant un certain temps. Finalement, le journal accumulé deviendra trop volumineux et le début de la conversation sera supprimé et le système "oubliera" les choses précédentes.

9. Les grands modèles de langage ne font pas de résolution de problèmes ou de planification. Mais vous pouvez leur demander de créer des plans et de résoudre des problèmes. Je vais couper quelques cheveux ici. La résolution de problèmes et la planification sont des termes réservés par certains groupes de la communauté de recherche en IA pour signifier quelque chose de très spécifique. En particulier, ils signifient avoir un objectif
   - quelque chose que vous voulez accomplir dans le futur - et travailler vers cet objectif en faisant des choix entre des alternatives susceptibles de vous rapprocher de cet objectif. Les grands modèles de langage n'ont pas d'objectifs. Ils ont un objectif, qui est de choisir un mot qui apparaîtrait très probablement dans les données d'apprentissage compte tenu d'une séquence d'entrée. Ils correspondent aux motifs. La planification, en particulier, implique généralement quelque chose appelé anticipation. Lorsque les humains planifient, ils imaginent les résultats de leurs actions et analysent cet avenir par rapport à l'objectif. S'il semble que cela se rapproche d'un objectif, c'est un bon coup. Si ce n'est pas le cas, nous pourrions essayer d'imaginer les résultats d'une autre action. Il y a bien plus que cela, mais les points clés sont que les grands modèles de langage n'ont pas d'objectifs et n'anticipent pas. Les transformateurs sont rétrogrades. L'auto-attention ne peut être appliquée qu'aux mots d'entrée qui sont déjà apparus. Désormais, les grands modèles de langage peuvent générer des sorties qui ressemblent à des plans, car ils ont vu de nombreux plans dans les données d'apprentissage. Ils savent à quoi ressemblent les plans, ils savent ce qui devrait apparaître dans les plans sur certains sujets qu'ils ont vus. Il va faire une bonne supposition sur ce plan. Le plan peut ignorer

des détails particuliers sur le monde et tendent vers le plan le plus générique. Les grands modèles de langage n'ont certainement pas « réfléchi aux alternatives » ou essayé une chose et fait marche arrière et essayé une autre chose. Il n'y a aucun mécanisme à l'intérieur d'un transformateur que l'on pourrait pointer du doigt qui ferait un tel va-et-vient de l'avenir. (Il y a une mise en garde à cela, qui sera abordée dans la section suivante.) Vérifiez toujours les résultats lorsque vous demandez des plans.

9. Qu'est-ce qui rend ChatGPT si spécial ?

"J'ai donc entendu dire que RLHF est ce qui rend ChatGPT vraiment intelligent."

"ChatGPT utilise l'apprentissage par renforcement et c'est ce qui le rend si intelligent."

Eh bien... en quelque sorte.

Au moment d'écrire ces lignes, il y a beaucoup d'enthousiasme à propos de quelque chose appelé RLHF, ou apprentissage par renforcement avec rétroaction humaine. Il y a quelques choses qui ont été faites pour former ChatGPT en particulier (et de plus en plus d'autres grands modèles de langue). Ils ne sont pas vraiment nouveaux, mais ils ont été largement introduits avec beaucoup d'effet lors de la sortie de ChatGPT.

ChatGPT est un modèle de grand langage basé sur Transformer. ChatGPT a acquis la réputation d'être vraiment bon pour produire des réponses aux invites de saisie et de refuser de répondre aux questions sur certains sujets qui pourraient être considérés comme toxiques ou opiniâtres. Il ne fait rien de particulièrement différent de ce qui est décrit ci-dessus. C'est plutôt vanille en fait. Mais il y a une différence : comment il a été formé. ChatGPT a été formé comme d'habitude - en grattant une grande partie d'Internet, en prenant des extraits de ce texte et en faisant en sorte que le système prédise le mot suivant. Cela a abouti à un modèle de base qui était déjà un prédicteur de mots très puissant (équivalent à GPT-3). Mais ensuite, il y avait deux étapes de formation supplémentaires. Réglage des instructions et apprentissage par renforcement avec rétroaction humaine.

9.1. Réglage des instructions

Il y a un problème particulier avec les grands modèles de langage : ils veulent juste prendre une séquence d'entrée de mots et générer ce qui vient ensuite. La plupart du temps, c'est ce que l'on veut. Mais pas toujours. Considérez l'invite de saisie suivante :

> "Écrivez un essai sur Alexander Hamilton."

Que pensez-vous que la réponse devrait être. Vous pensez probablement que cela devrait être quelque chose comme "Alexander Hamilton est né à Nevis en 1757. Il était un homme d'État, un avocat, un colonel dans l'armée et le premier secrétaire au Trésor des États-Unis..." Mais ce que vous pourriez réellement obtenir est :

> "Votre essai doit être d'au moins cinq pages, à double interligne, et inclure au moins deux citations."

Qu'est-ce qui vient de se passer? Eh bien, le modèle de langage a peut-être vu de nombreux exemples de devoirs d'étudiants commençant par "Rédiger un essai sur..." et comprenant des mots détaillant la longueur et la mise en forme. Bien sûr, lorsque vous avez écrit « Rédigez un essai... », vous pensiez que vous écriviez des instructions au modèle de langage comme si c'était un humain qui comprenait l'intention. Les modèles de langage ne comprennent pas votre intention ou ont leurs propres intentions ; ils ne font correspondre les entrées qu'aux modèles qu'ils ont vus dans leurs données de formation.

Pour résoudre ce problème, on peut faire quelque chose appelé réglage des instructions. L'idée est assez simple. Si vous obtenez la mauvaise réponse, notez quelle devrait être la bonne réponse et envoyez l'entrée d'origine et la nouvelle sortie corrigée via le réseau neuronal en tant que données d'apprentissage. Avec suffisamment d'exemples de la sortie corrigée, le système apprendra à décaler son circuit afin que la nouvelle réponse soit préférée.

On n'a pas à faire quelque chose de trop fantaisiste. Il suffit d'amener un grand nombre de personnes à interagir avec le grand modèle de langage et de lui demander de faire beaucoup de choses et d'écrire les corrections lorsqu'il ne se comporte pas correctement. Ensuite, rassemblez tous ces exemples où il a fait des erreurs et les nouveaux résultats corrects et faites plus de formation.

Ainsi, le grand modèle de langage agit comme s'il comprenait l'intention des invites d'entrée et agit comme s'il suivait des instructions. Il ne fait rien d'autre que d'essayer de deviner le mot suivant. Mais maintenant, les nouvelles données de formation permettent de deviner des mots qui semblent plus réactifs à l'entrée.

9.2. Apprentissage par renforcement à partir de la rétroaction humaine

La prochaine étape de la formation est l'apprentissage par renforcement à partir de la rétroaction humaine. Je pense que cela va nécessiter un peu d'explication.

L'apprentissage par renforcement est une technique d'IA traditionnellement utilisée dans certaines recherches en robotique et également dans les agents de jeu virtuels (pensez aux systèmes d'IA qui peuvent jouer aux échecs, au go ou à StarCraft). L'apprentissage par renforcement est particulièrement efficace pour savoir quoi faire lorsqu'il obtient quelque chose appelé récompense. La récompense est juste un nombre qui indique à quel point il va bien (+100 pour faire très bien ; -100 pour faire très mal). Dans le monde réel et dans les jeux, la récompense est souvent rarement donnée. Dans un jeu, vous devrez peut-être faire beaucoup de mouvements avant d'obtenir des points. Peut-être que vous n'obtenez des points qu'à la toute fin de la partie. Dans le monde réel, il n'y a tout simplement pas assez de gens pour vous dire quand vous faites du bon travail (vous l'êtes). Sauf si vous êtes un chien (ce sont tous de bons garçons et filles). La seule chose que vous devez vraiment savoir, c'est que les systèmes d'apprentissage par renforcement tentent de prédire la quantité de récompenses futures qu'ils obtiendront, puis de choisir l'action la plus susceptible d'obtenir plus de récompenses futures. Ce n'est pas tout à fait différent de la façon dont on pourrait utiliser des friandises pour chiens pour apprendre à son chien à se comporter.

D'accord, rangez tout cela et considérez l'invite suivante :

> En quoi Mark est-il un expert ?

Supposons que la sortie du modèle de langage est :

> Mark a de nombreuses publications sur l'intelligence artificielle, le graphisme et l'interaction homme-machine.

Ce n'est que partiellement correct. Je ne publie pas en graphisme. Je voudrais vraiment juste donner à cela un pouce vers le bas, ou un score de -1. Mais une seule partie est fausse : le mot graphisme. Si je disais au système que la phrase entière est fausse, le modèle linguistique pourrait apprendre que tous ces mots doivent être évités. Eh bien, beaucoup de ces mots sont raisonnables.

C'est là qu'intervient l'apprentissage par renforcement. L'apprentissage par renforcement fonctionne en essayant différentes alternatives et en voyant quelles alternatives obtiennent le plus de récompenses. Supposons que je lui demande de générer trois réponses différentes à l'invite d'origine.

> Mark a de nombreuses publications sur l'intelligence artificielle, le graphisme et l'interaction homme-machine.

Mark a travaillé dans le domaine de l'intelligence artificielle, des systèmes NLP sûrs et de l'interaction homme-ordinateur.

graphique.

Je pourrais donner un pouce vers le bas (-1) à la première alternative, un pouce vers le haut (+1) pour la deuxième alternative et un pouce vers le bas (-1) pour la troisième alternative. Tout comme jouer à un jeu, un L'algorithme d'apprentissage par renforcement peut regarder en arrière et comprendre que la seule chose en commun qui se traduit par un -1 est le mot "graphique". Maintenant, le système peut se concentrer sur ce mot et ajuster le circuit du réseau neuronal pour ne pas utiliser ce mot en conjonction avec cette invite d'entrée particulière.

... permet à un groupe de personnes d'interagir avec le grand modèle de langage. Cette fois, nous donnerons aux gens trois (ou plus) réponses possibles. Nous pouvons le faire en demandant au grand modèle de langage de générer une invite un certain nombre de fois et d'introduire un peu de hasard dans la sélection des bras de l'attaquant (vous ne les avez pas oubliés, n'est-ce pas ?). Au lieu de choisir le bras de frappe le plus activé, nous pouvons parfois choisir le deuxième ou le troisième bras de frappe le plus activé. Cela donne différentes réponses textuelles, et nous demandons aux gens de choisir leur première réponse préférée, leur deuxième préférée, etc. Maintenant, nous avons une alternative et nous avons des chiffres. Nous pouvons maintenant utiliser l'apprentissage par renforcement pour ajuster les circuits du réseau neuronal.

[En fait, nous utilisons ces retours de pouce vers le haut et vers le bas pour former un deuxième réseau de neurones afin de prédire si les gens donneront un pouce vers le haut ou vers le bas. Si ce réseau de neurones est assez bon pour prédire ce que les gens préféreront, alors nous pouvons utiliser ce deuxième réseau de neurones pour deviner si les réponses du modèle de langage pourraient obtenir des pouces vers le haut ou des pouces vers le bas et l'utiliser pour former le modèle de langage.]

Qu'est-ce que l'apprentissage par renforcement traite-t-il de la génération de texte comme un jeu où chaque action est un mot. À la fin d'une séquence, le modèle de langage est informé s'il a gagné ou perdu des points. Le modèle de langage ne fait pas exactement l'anticipation comme discuté dans la section précédente, mais il a été en quelque sorte formé pour prédire quels mots seront approuvés. Le Large Language Model n'a toujours pas d'objectif explicite, mais il a un objectif implicite d'"obtenir des pouces levés" (ou nous pourrions aussi le dire

a pour objectif implicite de "satisfaire la personne moyenne") et a appris à corréler certaines réponses à certaines invites avec l'obtention d'un pouce levé. Cela présente de nombreuses qualités de planification, mais sans mécanisme explicite d'anticipation. Plus comme il a mémorisé des stratégies pour obtenir une récompense qui ont tendance à fonctionner dans de nombreuses situations.

Au point de savoir si RLHF rend ChatGPT plus intelligent... cela rend ChatGPT plus susceptible de produire les types de réponses que nous espérions voir. Il semble plus intelligent parce que ses sorties semblent donner l'impression qu'il comprend les intentions de nos entrées et a ses propres intentions de répondre. C'est une illusion car il ne s'agit encore que d'encoder et de décoder des mots. Mais encore une fois, c'est là que nous avons commencé cet article.

Le réglage des instructions et RLHF rendent également l'utilisation de ChatGPT résistante à certains types d'abus tels que la génération de contenu raciste, sexiste ou politiquement chargé. Cela peut toujours être fait, et dans tous les cas, les anciennes versions de GPT-3 ont toujours été capables de le faire. Cependant, en tant que service public gratuit, la friction que ChatGPT crée contre certains types d'abus donne un sentiment de sécurité. Il résiste également à fournir une opinion comme un fait, ce qui élimine également une forme de préjudice potentiel pour l'utilisateur.

[L'utilisation de l'apprentissage par renforcement pour modifier un modèle de langage pré-formé n'est pas nouvelle. Il remonte à au moins 2016 et a été utilisé pour rendre les grands modèles de langage plus sûrs. La plupart des réglages basés sur l'apprentissage par renforcement des grands modèles de langage utilisent un deuxième modèle pour fournir une récompense, ce qui est également fait avec ChatGPT. Ce qui est remarquable pour ChatGPT, c'est l'échelle du système réglé avec l'apprentissage par renforcement et l'effort de collecte de rétroaction humaine à grande échelle.]

10. Conclusion

J'ai besoin de dormir plus. C'est ce que je conclus de tout cela.