Du Réseau de Neurones à GPT

Vidéos YouTube

Andrej Karpathy - Construisons GPT : de zéro, en code, expliqué pas à pas.

Umar Jamil - Attention is all you need (Transformer) - Explication du modèle (y compris les mathématiques), Inférence et Entraînement

StatQuest avec Josh Starmer - Les réseaux de neurones Transformer, fondation de ChatGPT, expliqués clairement !!!

Pascal Poupart - Cours CS480/680, Leçon 19 : Attention et réseaux Transformer

Le Pirate de l'IA - Michael Phi - Guide Illustré des Réseaux de Neurones Transformers : Une explication étape par étape

Comment j'apprends

Après avoir lu la moitié du livre "Neural Networks and Deep Learning", j'ai commencé à reproduire l'exemple de réseau neuronal pour la reconnaissance de chiffres manuscrits. J'ai créé un dépôt sur GitHub, https://github.com/lzwjava/neural-networks-and-zhiwei-learning.

C'est là que réside la véritable difficulté. Si quelqu'un peut l'écrire de zéro sans copier aucun code, cela signifie qu'il comprend très bien.

Mon code de réplication manque encore l'implémentation de `update_mini_batch` et `backprop`. Cependant, en observant attentivement les variables lors de la phase de chargement des données, de la propagation avant et de l'évaluation, j'ai acquis une bien meilleure compréhension des vecteurs, de la dimensionnalité, des matrices et de la forme des objets.

Et j'ai commencé à apprendre l'implémentation du GPT et du transformer. Grâce à l'incorporation de mots et au codage positionnel, le texte se transforme en nombres. Ensuite, en essence, cela ne diffère pas d'un simple réseau de neurones pour reconnaître des chiffres manuscrits.

La conférence d'Andrej Karpathy intitulée "Let's build GPT" est très instructive. Il explique les choses de manière claire.

La première raison est que c'est vraiment à partir de zéro. Nous voyons d'abord comment générer le texte. C'est un peu flou et aléatoire. La deuxième raison est qu'Andrej sait expliquer les choses de manière très intuitive. Andrej a travaillé sur le projet nanoGPT pendant plusieurs mois.

Je viens d'avoir une nouvelle idée pour juger de la qualité d'une conférence. L'auteur peut-il vraiment écrire ces codes ? Pourquoi est-ce que je ne comprends pas et quel sujet l'auteur a-t-il manqué ? En plus de ces diagrammes et animations élégants, quels sont leurs défauts et imperfections ?

Revenons au sujet du machine learning lui-même. Comme le mentionne Andrej, le dropout, la connexion résiduelle, le Self-Attention, le Multi-Head Attention, et le Masked Attention.

En regardant davantage les vidéos ci-dessus, j'ai commencé à comprendre un peu.

En utilisant l'encodage positionnel avec les fonctions sinus et cosinus, nous obtenons certains poids. Par le biais de l'incorporation de mots (word embedding), nous transformons les mots en nombres.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

La pizza est sortie du four et elle était délicieuse.

Dans cette phrase, comment l'algorithme sait-il s'il fait référence à la pizza ou au four ? Comment calculons-nous les similarités pour chaque mot de la phrase ?

Nous voulons un ensemble de poids. Si nous utilisons le réseau de transformateurs pour effectuer la tâche de traduction, chaque fois que nous entrons une phrase, il peut produire la phrase correspondante dans une autre langue.

À propos du produit scalaire ici. Une raison pour laquelle nous utilisons le produit scalaire ici est que le produit scalaire prend en compte chaque nombre dans le vecteur. Et si nous utilisions le produit scalaire au carré ? Nous calculons d'abord le carré des nombres, puis nous les faisons participer au produit scalaire. Et si nous faisions un produit scalaire inversé ?

À propos du masquage ici, nous changeons les nombres de la moitié de la matrice en moins l'infini. Ensuite, nous utilisons softmax pour que les valeurs varient entre 0 et 1. Et si nous changions les nombres en bas à gauche en moins l'infini ?

Plan

Continuer à lire du code et des articles, et à regarder des vidéos. Juste m'amuser et suivre ma curiosité.

https://github.com/karpathy/nanoGPT

https://github.com/jadore801120/attention-is-all-you-need-pytorch