

# Machine learning : méthodes et solutions

Redha Moulla

27 février - 1 mars 2024

# Réseaux de neurones récurrents

# Introduction aux RNN

Les Réseaux de Neurones Récurrents (RNN) sont une classe de réseaux de neurones artificiels spécialement conçus pour traiter des séquences de données, tels que des séries temporelles ou des séquences textuelles.

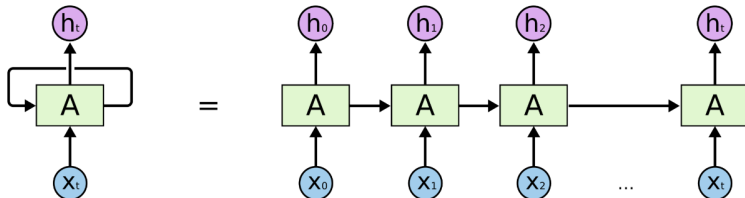
## Caractéristiques:

- **Mémoire à court terme** : Les RNN ont la capacité de se "souvenir" d'informations passées grâce à leurs connexions récurrentes.
- **Traitement de séquences** : Ils sont particulièrement adaptés pour des tâches où les données sont séquentielles et où le contexte est important (ex: langage naturel, musique).
- **Modélisation de dépendances temporelles** : Les RNN peuvent capturer des dépendances temporelles et contextuelles dans les données.

# Architecture des RNN

Cette architecture représente un RNN à la fois dans sa forme condensée et dépliée à travers le temps.

- À gauche, le RNN est présenté de manière condensée avec:
  - Une entrée  $x_t$ .
  - Une sortie d'état caché  $h_t$ .
  - Un bloc  $A$  représentant la cellule RNN qui effectue les calculs à partir de l'état caché précédent et de l'entrée courante pour produire le nouvel état caché.
- À droite, l'architecture est déroulée pour montrer la séquence complète:
  - Chaque bloc  $A$  est le même RNN à différents instants temporels.
  - Le bloc  $A$  prend une entrée  $x_t$  et produit un état caché  $h_t$ , qui est alors passé à la même cellule au pas de temps suivant.
  - L'état initial  $h_0$  est souvent initialisé à zéro ou une petite valeur aléatoire.



# Formulation mathématique des RNN

Un RNN est un réseau de neurones conçu pour traiter des séquences de données, caractérisé par sa capacité à maintenir une 'mémoire' des entrées antérieures dans ses connexions récurrentes.

Formulation mathématique d'un RNN simple pour une séquence de temps  $t$  :

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

$$y_t = W_{hy}h_t + b_y$$

où :

- $x_t$  est l'entrée à l'instant  $t$
- $h_t$  est l'état caché à l'instant  $t$
- $y_t$  est la sortie à l'instant  $t$
- $W$  et  $b$  sont les paramètres du réseau
- $\sigma$  est une fonction d'activation non-linéaire

# Aspects Mathématiques de la Rétropropagation dans les RNN

La rétropropagation à travers le temps (BPTT) est la méthode clé pour l'entraînement des RNN, impliquant des calculs détaillés pour ajuster les poids en fonction des gradients.

**Formule de Base de la Rétropropagation :** Pour un état caché  $h_t$  et une sortie  $y_t$  à l'instant  $t$ , avec une fonction de perte  $L$ , les gradients sont calculés comme suit :

$$\frac{\partial L}{\partial \omega} = \sum_{t=1}^T \frac{\partial L}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial \omega}$$

où  $\omega$  représente les paramètres du modèle (poids).

**Calcul du Gradient à Travers les Étapes Temporelles :** Le gradient d'un paramètre à un instant  $t$  dépend non seulement de l'erreur à cet instant mais aussi des gradients des étapes futures :

$$\frac{\partial L}{\partial \omega} = \sum_{t=1}^T \sum_{k=t}^T \frac{\partial L}{\partial y_k} \frac{\partial y_k}{\partial h_k} \frac{\partial h_k}{\partial h_t} \frac{\partial h_t}{\partial \omega}$$

# Problème de disparition du gradient

Le problème de disparition du gradient survient lors de la rétropropagation dans les RNN traditionnels. Les gradients des paramètres peuvent devenir très petits, rendant l'apprentissage des dépendances à long terme difficile.

Mathématiquement, pendant la rétropropagation, si les valeurs de  $\frac{\partial h_t}{\partial W}$  sont petites, le gradient total peut tendre vers zéro à mesure que  $T$  augmente.

## Solutions au problème de disparition du gradient :

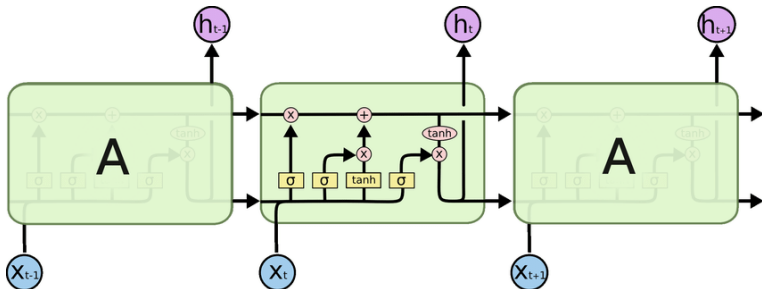
Pour atténuer le problème de disparition du gradient, différentes solutions ont été proposées :

- Utilisation de fonctions d'activation comme ReLU au lieu de sigmoïde ou tanh.
- Introduction d'architectures RNN avancées comme LSTM et GRU.
- Techniques de clipping de gradient pour éviter l'explosion des gradients.

# RNN avec Long Short-Term Memory (LSTM)

Les LSTM sont une variante des RNN conçus pour mieux capturer les dépendances à long terme. Ils introduisent des 'cellules mémoire' avec des portes de régulation :

- Porte d'oubli : contrôle la quantité d'informations à retenir de l'état précédent.
- Porte d'entrée : contrôle la quantité d'informations à ajouter de l'entrée actuelle.
- Porte de sortie : détermine la quantité d'informations à transmettre à l'état suivant.



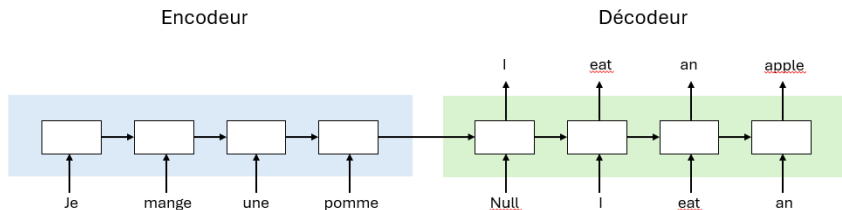


# Modèles Seq2Seq

# Architecture de base

L'architecture seq-to-seq comprend deux composants principaux :

- **Encodeur** : Un réseau de neurones qui lit et encode la séquence d'entrée en un vecteur de contexte (une représentation dense de la séquence).
- **Décodeur** : Un autre réseau de neurones qui lit le vecteur de contexte pour générer la séquence de sortie, un élément à la fois.

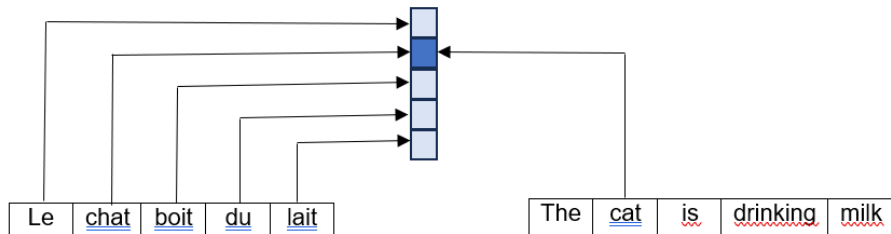


# Introduction au mécanisme d'Attention

Le mécanisme d'attention est une amélioration clé apportée aux modèles seq-to-seq traditionnels, permettant au modèle de se "concentrer" sur différentes parties de la séquence d'entrée lors de la génération de la séquence de sortie.

- **Objectif** : Surmonter les limitations des encodeurs seq-to-seq qui compressent toute l'information d'une séquence d'entrée dans un vecteur de contexte fixe.
- **Avantage** : Améliore la capacité du modèle à gérer de longues séquences d'entrée, en rendant le processus de génération de la séquence de sortie plus dynamique et contextuellement informé.

# Illustration du mécanisme d'Attention



# Comment fonctionne l'Attention ?

L'attention fonctionne en calculant un ensemble de poids qui détermine à quel point chaque partie de la séquence d'entrée est importante pour chaque étape de la génération de la séquence de sortie.

- À chaque étape de décodage, le modèle calcule un score d'attention entre l'état caché du décodeur et chaque état caché de l'encodeur.
- Ces scores sont utilisés pour créer un vecteur de contexte pondéré, qui est ensuite passé au décodeur pour générer l'élément suivant de la séquence de sortie.

$$\alpha_{tj} = \frac{\exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_j))}{\sum_{k=1}^{T_x} \exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_k))} \quad (1)$$

$$\mathbf{c}_t = \sum_{j=1}^{T_x} \alpha_{tj} \bar{\mathbf{h}}_j \quad (2)$$

où  $\mathbf{h}_t$  est l'état caché du décodeur,  $\bar{\mathbf{h}}_j$  sont les états cachés de l'encodeur, et  $\mathbf{c}_t$  est le vecteur de contexte pour l'étape  $t$ .

---

## Attention Is All You Need

---

**Ashish Vaswani\***  
Google Brain  
avaswani@google.com

**Noam Shazeer\***  
Google Brain  
noam@google.com

**Niki Parmar\***  
Google Research  
nikip@google.com

**Jakob Uszkoreit\***  
Google Research  
usz@google.com

**Llion Jones\***  
Google Research  
llion@google.com

**Aidan N. Gomez\*<sup>†</sup>**  
University of Toronto  
aidan@cs.toronto.edu

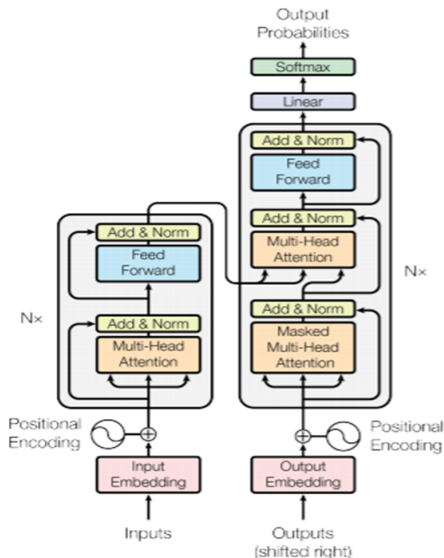
**Lukasz Kaiser\***  
Google Brain  
lukaszkaier@google.com

**Illia Polosukhin\*<sup>‡</sup>**  
illia.polosukhin@gmail.com

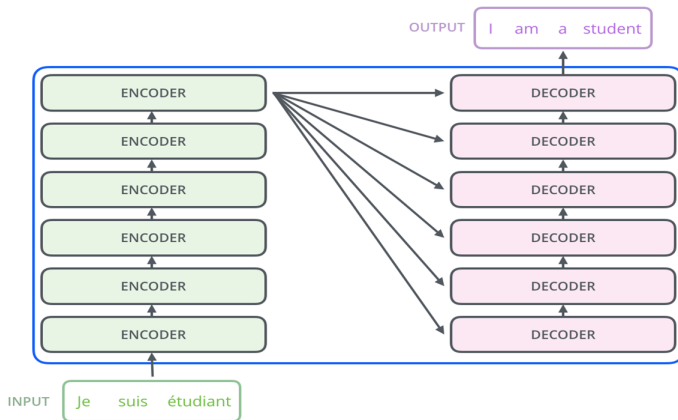
### Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to

# Transformer originel

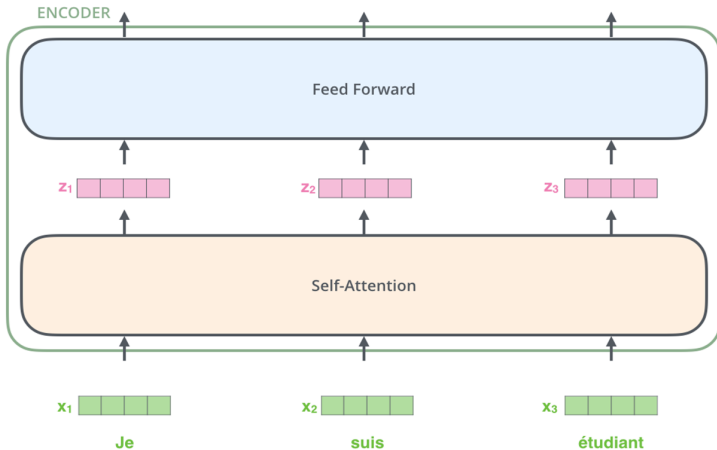


# Mécanisme de cross-attention

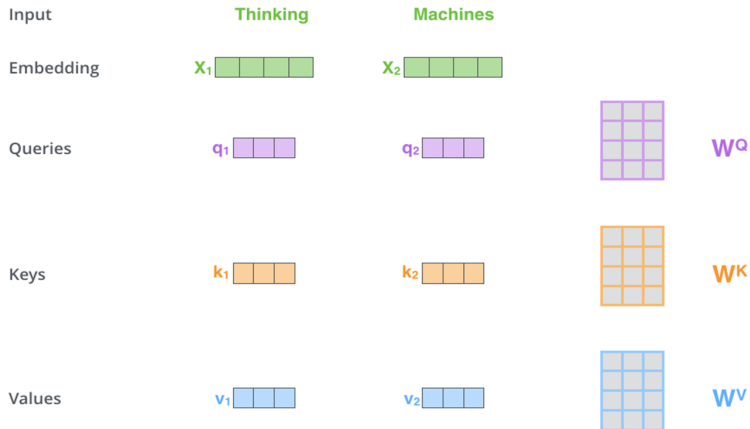




# Mécanisme de self-attention



# Fonctionnement du mécanisme de self-attention



# Calcul des scores de self-attention

Input

Embedding

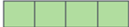
Queries

Keys

Values

Score

Thinking

$x_1$  

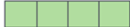
$q_1$  

$k_1$  

$v_1$  


$$q_1 \cdot k_1 = 112$$

Machines

$x_2$  

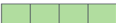
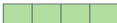






$q_2$  

$k_2$  

$v_2$  

$$q_1 \cdot k_2 = 96$$

# Calcul des scores de self-attention

Input	Thinking	Machines
Embedding	$x_1$ 	$x_2$ 
Queries	$q_1$ 	$q_2$ 
Keys	$k_1$ 	$k_2$ 
Values	$v_1$ 	$v_2$ 
Score	$q_1 \cdot k_1 = 112$	$q_2 \cdot k_2 = 96$
Divide by $8 (\sqrt{d_k})$	14	12
Softmax	0.88	0.12

# Calcul de la nouvelle représentation

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 (  $\sqrt{d_k}$  )

Softmax


Softmax  
X  
Value

Sum

Thinking

$x_1$  

$q_1$  


$k_1$  

$v_1$  

$q_1 \cdot k_1 = 112$

14

0.88

$v_1$  

$z_1$  

Machines

$x_2$  

$q_2$  


$k_2$  

$v_2$  

$q_1 \cdot k_2 = 96$

12

0.12

$v_2$  

$z_2$  

# Formulation matricielle

$$\mathbf{X} \times \mathbf{W}^Q = \mathbf{Q}$$


A diagram illustrating matrix multiplication. On the left, a green 2x4 matrix  $\mathbf{X}$  is multiplied by a purple 4x4 matrix  $\mathbf{W}^Q$ . The result is a purple 2x3 matrix  $\mathbf{Q}$ .

$$\mathbf{X} \times \mathbf{W}^K = \mathbf{K}$$


A diagram illustrating matrix multiplication. On the left, a green 2x4 matrix  $\mathbf{X}$  is multiplied by an orange 4x4 matrix  $\mathbf{W}^K$ . The result is an orange 2x3 matrix  $\mathbf{K}$ .

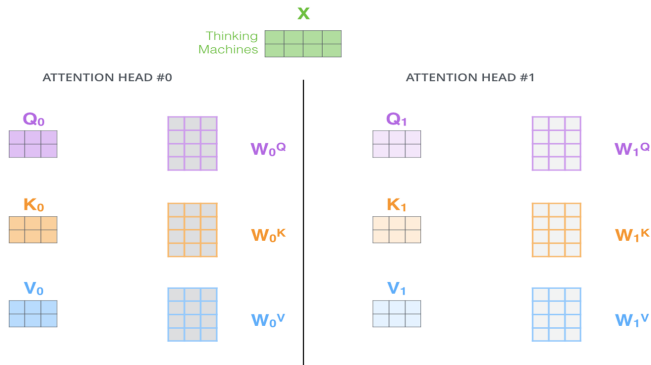
$$\mathbf{X} \times \mathbf{W}^V = \mathbf{V}$$


A diagram illustrating matrix multiplication. On the left, a green 2x4 matrix  $\mathbf{X}$  is multiplied by a blue 4x4 matrix  $\mathbf{W}^V$ . The result is a blue 2x3 matrix  $\mathbf{V}$ .

# Formulation matricielle du mécanisme de self attention

$$\text{softmax}\left(\frac{\begin{matrix} \text{Q} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \text{K}^T \\ \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline & \\ \hline \end{array} \end{matrix}\right) \begin{matrix} \text{V} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$
$$= \begin{matrix} \text{Z} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

# Multihead attention





# Multihead attention

ATTENTION  
HEAD #0

$Z_0$



ATTENTION  
HEAD #1

$Z_1$



...

ATTENTION  
HEAD #7

$Z_7$



$Z_0$

$Z_1$

$Z_2$

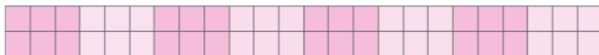
$Z_3$

$Z_4$

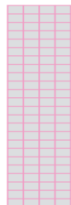
$Z_5$

$Z_6$

$Z_7$



X



$W_0$

# Performances du tranformer originel

Entraîné sur WMT 2014 English- German dataset, comprenant près de 4.5 millions de phrases sentence, le WMT 2014 English-French dataset, comprenant près de 36 millions de phrases.

- 8 NVIDIA P10 GPUs
- **Base** : 12 heures
- **Large** : 3.5 jours

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	<b><math>3.3 \cdot 10^{18}</math></b>	
Transformer (big)	<b>28.4</b>	<b>41.8</b>	$2.3 \cdot 10^{19}$	

# BERT

BERT (Bidirectional Encoder Representations from Transformers) représente une avancée majeure dans le NLP, introduisant une approche novatrice pour la modélisation de langage.

- **Contextualisation profonde** : Première architecture à utiliser pleinement la bidirectionnalité pour comprendre le contexte des mots, permettant une compréhension plus fine du langage.
- **Pré-entraînement et fine-tuning** : Méthodologie en deux étapes offrant une flexibilité pour l'adaptation à diverses tâches de TALN sans architecture spécifique à la tâche.
- **Impact sur le NLP** : Avant BERT, les approches de modélisation de langage étaient limitées par la compréhension unidirectionnelle ou des représentations statiques des mots. BERT a changé la donne en permettant des représentations contextuelles dynamiques.
- **Performances révolutionnaires** : À son introduction, BERT a établi de nouveaux standards de performance sur une gamme de benchmarks de NLP, y compris GLUE, SQuAD, etc.

# Pré-entraînement de BERT

Objectifs de pré-entraînement :

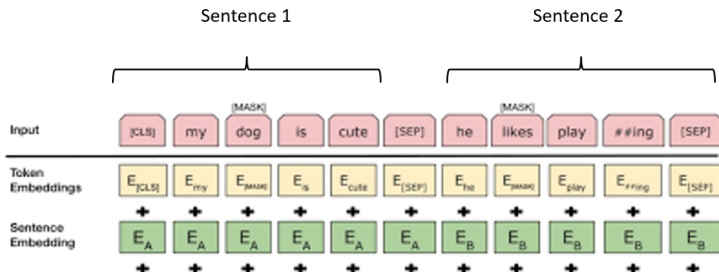
- **Masquage de token (MLM)** : 15% des tokens masqués aléatoirement, prédits par le modèle.

$$L_{\text{MLM}} = -\log P(\text{token masqué}|\text{contexte}) \quad (3)$$

- **Prédiction de la prochaine phrase (NSP)** : Déterminer si une phrase B suit logiquement une phrase A.

$$L_{\text{NSP}} = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})] \quad (4)$$

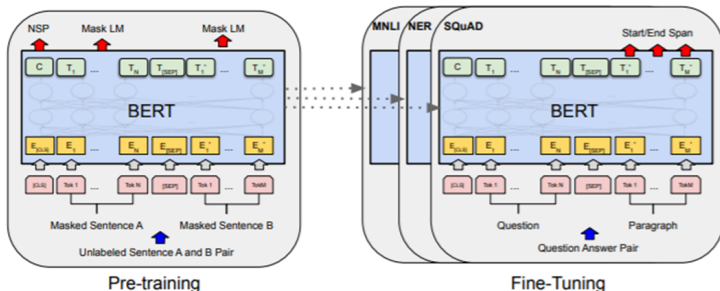
- Combinaison des pertes pour l'optimisation.



# Fine-tuning de BERT pour des tâches spécifiques

Après pré-entraînement, BERT est affiné pour des tâches spécifiques:

- Ajout d'une couche de sortie spécifique à la tâche (classification, NER, QA).
- Fine-tuning de tous les paramètres du modèle pré-entraîné sur le corpus de la tâche.



# Performances de BERT

BERT a été pré-entraîné sur le BookCorpus (800 millions de mots) et English Wikipedia (2,500 millions de mots).

## Deux modèles :

- **BERT Base** : 12 couches avec 110 millions de paramètres.
- **BERT Large** : 24 couches avec 340 millions de paramètres.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>92.7</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>82.1</b>

# Introduction aux modèles autorégressifs - GPT-3

GPT-3 (Generative Pre-trained Transformer 3) est le modèle de langue développé par OpenAI, représentant la troisième génération de la série GPT. Avec 175 milliards de paramètres, GPT-3 pousse les limites de la génération de texte et de la compréhension du langage naturel.

- **Capacités générales** : GPT-3 excelle dans une variété de tâches de TALN sans fine-tuning spécifique, grâce à sa puissance de modélisation du langage.
- **Architecture autorégressive** : Utilise un modèle Transformer pour prédire le mot suivant dans un texte, apprenant des patterns complexes sur des données massives.
- **Approche few-shot learning** : Capable d'adapter ses réponses à des tâches spécifiques avec peu ou pas d'exemples d'entraînement.
- **Impact sur IA et le NLP** : GPT-3 a significativement avancé les capacités des systèmes d'IA dans la compréhension et la génération de langage naturel, ouvrant de nouvelles voies pour l'application de l'intelligence artificielle.

# Architecture de GPT-3

GPT-3 repose sur une architecture Transformer améliorée, optimisée pour traiter efficacement de grandes quantités d'informations et générer des réponses cohérentes et contextuellement pertinentes.

- **Taille du modèle** : Avec 175 milliards de paramètres, GPT-3 est l'un des modèles de langue les plus grands et les plus complexes à ce jour.
- **Mécanisme d'attention** : L'attention multi-têtes permet au modèle de pondérer différemment les parties d'un input, améliorant la compréhension du contexte.
- **Optimisation et entraînement** : Techniques d'optimisation avancées pour gérer la taille du modèle et l'efficacité de l'entraînement.
- **Formulation** :

$$P(w_t | w_1, \dots, w_{t-1}) = \text{softmax}(\text{Transformer}(w_1, \dots, w_{t-1})). \quad (5)$$

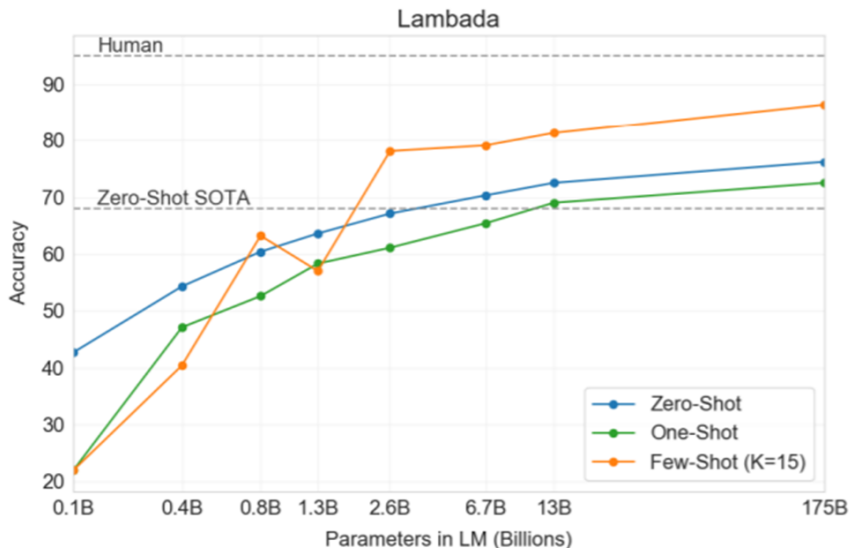


# Few-Shot Learning avec GPT-3

L'une des innovations les plus remarquables de GPT-3 réside dans sa capacité à effectuer du few-shot learning, permettant au modèle de comprendre et d'exécuter des tâches spécifiques avec très peu d'exemples.

- **Définition** : Le few-shot learning désigne la capacité d'un modèle à apprendre une nouvelle tâche à partir d'un très petit nombre d'exemples d'entraînement, souvent seulement quelques-uns.
- **Mécanisme dans GPT-3** :
  - GPT-3 utilise des prompts contenant quelques exemples de la tâche désirée pour guider le modèle sur ce qui est attendu, avant de présenter la question ou la tâche à résoudre.
  - Le modèle généralise ensuite à partir de ces exemples pour générer des réponses ou des solutions aux problèmes posés, montrant une compréhension étonnante de la tâche avec un minimum de guidance.
- **Exemples d'application** :
  - Classification de texte, génération de résumés, réponse à des questions spécifiques, et plus, avec seulement quelques exemples pour chaque tâche.
- **Impact** : Cette capacité émergente réduit considérablement le besoin de vastes ensembles de données d'entraînement spécifiques à la tâche, ouvrant la voie à des applications plus flexibles et accessibles de modèles de langage.

# Performances de GPT-3 en few-shot learning



# ChatGPT

ChatGPT, développé par OpenAI, est un modèle de langage basé sur l'architecture GPT (Generative Pre-trained Transformer) optimisé pour comprendre et générer des dialogues naturels. L'entraînement de ChatGPT se décline selon les étapes

suivantes :

- ➊ **Pré-entraînement sur un corpus volumineux** : Comme GPT-3, ChatGPT est d'abord pré-entraîné sur un vaste ensemble de données textuelles, englobant un large éventail de la littérature disponible sur Internet, pour apprendre une compréhension générale du langage.
- ➋ **Fine-tuning supervisé** : Ensuite, ChatGPT est affiné sur des dialogues spécifiques pour améliorer ses compétences conversationnelles. Cette étape utilise des paires question-réponse et des conversations pour enseigner au modèle des structures de dialogue et des réponses contextuellement appropriées.
- ➌ **Reinforcement Learning from Human Feedback (RLHF)** : Utilisation de techniques de renforcement pour ajuster les réponses du modèle basées sur les préférences et les corrections fournies par des évaluateurs humains, raffinant davantage la pertinence et la naturalité des réponses.

# Fine-Tuning supervisé (FST)

Le fine-tuning supervisé joue un rôle essentiel dans l'adaptation de ChatGPT à des tâches conversationnelles spécifiques, améliorant sa capacité à fournir des réponses pertinentes et contextuellement adaptées. **Méthodologie:**

- 1 Collecte d'un ensemble de données de dialogues de haute qualité, comprenant des échanges humains et des interactions annotées pour capturer divers contextes conversationnels.
- 2 Transformation de ces dialogues en paires de prompts et réponses pour créer des exemples d'entraînement qui guident le modèle dans l'apprentissage de structures conversationnelles et de réponses appropriées.
- 3 Utilisation d'un processus d'entraînement supervisé où le modèle apprend à prédire la réponse la plus probable à un prompt donné, en ajustant ses paramètres internes pour minimiser la divergence entre les réponses générées et les réponses attendues (annotations).

# Alignement des LLM avec RLHF

- La génération de texte par les LLM est essentielle dans de nombreux domaines tels que la traduction automatique, la résumé automatique, et la création de contenu.
- Cependant, la qualité du texte généré peut être variable et dépend souvent des données d'entraînement disponibles.
- L'ajout du feedback humain permet d'améliorer la qualité du texte généré en fournissant une supervision supplémentaire.
- RLHF combine l'apprentissage par renforcement avec le feedback humain pour guider l'apprentissage des LLM de manière plus précise et efficace.
- L'objectif est d'entraîner les LLM à générer du texte de meilleure qualité en s'appuyant sur le savoir-faire humain pour corriger les erreurs et améliorer les performances.

# Approche RLHF pour l'entraînement des LLM

- Collecte des feedbacks :
  - Les LLM génèrent du texte qui est évalué par des humains.
  - Les humains fournissent des annotations telles que des corrections, des scores de qualité ou des préférences.
- Calcul de la récompense :
  - Le feedback humain est utilisé pour calculer une récompense.
  - Différentes métriques peuvent être utilisées pour quantifier la qualité du texte généré.
  - La récompense peut être une fonction de ces métriques et des préférences humaines.
- Entraînement des LLM :
  - L'objectif est d'entraîner les LLM à maximiser la récompense définie par le feedback humain.
  - Les LLM sont entraînés à générer du texte de meilleure qualité en fonction de cette récompense.

# Construction des feedbacks humains

- Les feedbacks humains sont construits sur la base des réponses générées par les LLM.
- Ces réponses sont ensuite évaluées par les humains, qui fournissent des annotations telles que des corrections, des scores de qualité, ou des préférences.
- Les feedbacks humains sont utilisés comme données d'entraînement pour apprendre au modèle de récompense à évaluer la qualité du texte généré.
- Les caractéristiques du texte généré, telles que la similarité avec des textes de référence ou la pertinence du contenu, peuvent être utilisées comme des caractéristiques pour l'entraînement du modèle.
- L'objectif est de créer un modèle de récompense capable d'évaluer de manière précise et fiable la qualité du texte généré par les LLM, selon les critères humains.

# Entraînement des LLM avec PPO

- **Initialisation** : Les paramètres du LLM sont initialisés à partir d'un modèle pré-entraîné sur de vastes ensembles de données textuelles.
- **Génération de données** : Le LLM génère une séquence de texte en fonction de son état actuel et de la politique actuelle. Les séquences de texte générées sont évaluées à l'aide du modèle de récompense pour calculer les récompenses associées à chaque état.
- **Mise à jour de la politique** : Une fonction de loss est calculée à l'aide des séquences de texte générées. Les gradients sont calculés par rapport aux paramètres du LLM. Les paramètres sont ensuite mis à jour pour maximiser les récompenses futures.
- **Itération** : Les étapes précédentes sont répétées de manière itérative jusqu'à ce que la politique converge vers une performance satisfaisante.



# Entraînement des LLM avec PPO

- **Initialisation** : Les paramètres du LLM sont représentés par  $\theta$ , initialisés à partir d'un modèle pré-entraîné.
- **Génération de données** : Le LLM génère une séquence de texte  $s_t$  en fonction de son état actuel et de la politique actuelle, paramétré par  $\theta$ .
- **Mise à jour de la politique** : La probabilité d'action pour chaque état  $s_t$  est donnée par  $\pi_\theta(s_t)$ . La fonction de perte utilisée dans PPO est définie comme suit :

$$\mathcal{L}(\theta) = \mathbb{E}_t \left[ \min \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t, \text{clip} \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1 - \epsilon, 1 + \epsilon \right) \right) \right]$$

- où  $A_t$  est l'avantage avancé estimé pour l'action  $a_t$  et  $\epsilon$  est un paramètre de clipping.

# Merci de votre attention

redha\_moulla@yahoo.fr