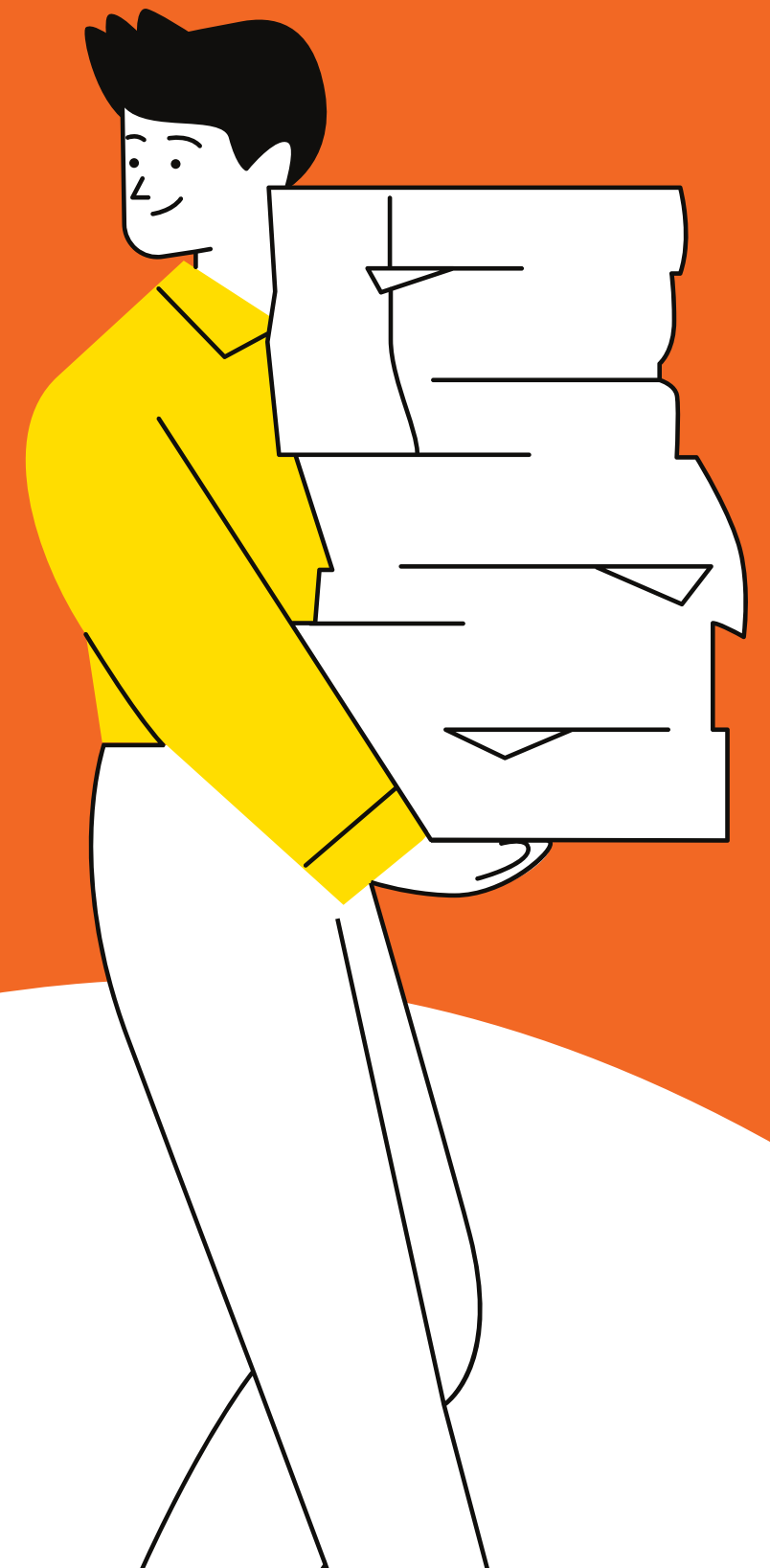


# Les transformeurs



*Prépare*  
*par:*

*Oumayma Rouahí*





# PLAN

1

Introduction

2

Les domaines d'application

3

Les différentes architectures

4

Le principe de fonctionnement

6

Exemple d'application et résultats

7

Conclusion

# Introduction

1

## Présentation du concept de transformation

Consiste à modifier une entité ou un objet d'un état initial à un état final en appliquant une série d'opérations ou de manipulations.

La transformation de données est réalisée en appliquant des transformations non linéaires telles que la convolution, la normalisation, le pooling et les fonctions d'activation.

En combinant ces transformations, les modèles de deep learning peuvent apprendre des représentations de données hautement abstraites pour résoudre des tâches complexes telles que la classification, la traduction automatique, la reconnaissance d'objets ou la synthèse de la parole.





Les transformeurs, ou "Transformers" en anglais, sont une architecture de réseau de neurones développée pour le traitement de séquences de données, telles que du texte ou de la parole. Cette architecture a été introduite en 2017 par Vaswani et al. dans un article intitulé "Attention is All You Need" et depuis lors, elle a été largement utilisée dans de nombreuses applications de traitement de langage naturel, notamment la traduction automatique, la génération de texte, la classification de texte et la réponse aux questions.



Les transformeurs se distinguent d'autres architectures de réseau de neurones, telles que les réseaux de neurones récurrents (RNN) et les réseaux de neurones à convolution (CNN), par leur utilisation d'un mécanisme d'attention pour prendre en compte les relations entre les différentes parties de la séquence d'entrée.

**Contrairement aux RNN**

les transformeurs n'ont pas besoin de traiter les données dans l'ordre séquentiel, ce qui les rend plus efficaces pour traiter des séquences longues. De plus, ils peuvent traiter plusieurs parties de la séquence simultanément, ce qui améliore les performances et réduit le temps d'entraînement.

**Les transformeurs se distinguent également des CNN**

qui sont principalement utilisés pour le traitement d'images, en ce qu'ils n'ont pas besoin de convolutions pour extraire des caractéristiques des données d'entrée. Au lieu de cela, ils utilisent une combinaison de mécanismes d'attention et de réseaux entièrement connectés pour modéliser les relations entre les différentes parties de la séquence.



le choix d'une architecture de réseau de neurones dépend des caractéristiques de la tâche à accomplir. Les transformeurs sont particulièrement bien adaptés au traitement de séquences longues et complexes.



## Les domaines d'application

1

les domaines d'application les plus courants des transformeurs.

### Traduction automatique :

Les transformeurs sont largement utilisés pour la traduction automatique, car ils sont capables de traiter des séquences de mots entiers plutôt que des mots individuels. Cela permet aux transformeurs de mieux capturer les relations complexes entre les mots et les phrases, ce qui peut améliorer la qualité de la traduction.

→ **Google Translate** : Les modèles de traduction basés sur des transformeurs tels que le modèle de traduction neurale de Google (GNMT) ont considérablement amélioré la qualité de la traduction automatique.



## Génération de texte :

Les transformeurs peuvent être entraînés à générer du texte de manière autonome en utilisant des techniques de génération de texte telles que l'apprentissage par renforcement. Ces modèles peuvent apprendre à générer des phrases cohérentes en fonction d'un contexte donné ou d'une intention spécifique.



- la rédaction de contenu pour des sites web.
- la génération de descriptions de produits pour des sites de commerce électronique .
- la création de commentaires pour des blogs ou des forums en ligne.

## Compréhension de texte :

Les transformeurs peuvent être entraînés à comprendre le sens d'un texte en utilisant des modèles d'apprentissage en profondeur, tels que les réseaux de neurones. Ces modèles peuvent identifier les entités clés dans un texte, telles que les personnes, les lieux et les dates, et extraire des informations spécifiques d'un texte.



- l'analyse de documents juridiques .
- la recherche d'informations médicales .
- la surveillance des médias sociaux pour détecter les tendances et les opinions.




## Résumé automatique de texte :

Les transformeurs peuvent également être utilisés pour résumer automatiquement un texte. Ils peuvent identifier les informations clés dans un texte et générer un résumé concis et précis.

- 
- créer des résumés pour des articles de presse, des rapports de recherche et des documents juridiques.

## Dialogue et chatbots :

Les chatbots basés sur des transformeurs peuvent offrir une expérience utilisateur plus personnalisée en comprenant les besoins et les préférences des utilisateurs. Ils peuvent également aider à améliorer l'efficacité et la productivité en réduisant le temps nécessaire pour répondre aux questions courantes ou pour effectuer des tâches simples.

- 
- les services clientèle .
  - les ventes et le marketing .
  - les soins de santé .

## 2

### Importance des transformeurs dans le traitement du langage naturel

- Les transformeurs ont révolutionné le traitement du langage naturel en permettant de créer des modèles plus performants et plus efficaces.
- Sont particulièrement efficaces pour des tâches telles que la traduction automatique, la génération de texte, l'analyse de sentiment et la recherche d'informations...
- Sa capacité à traiter des données à grande échelle et leur adaptabilité aux différents contextes et domaines.
- Réduisent la nécessité d'une ingénierie de fonctionnalités manuelle, ce qui accélère considérablement le processus de développement de modèles de traitement du langage naturel.



# Les différentes architectures

1

les différentes architectures de transformeurs les plus courantes

## Transformer de base :

une architecture de transformeur qui a été introduite en 2017 par Vaswani et al. Elle est composée de deux modules principaux : l'encodeur et le décodeur.



utilisé dans de nombreuses applications, notamment la traduction automatique, la génération de texte, la reconnaissance de la parole et la classification de texte.



## BERT (Bidirectional Encoder Representations from Transformers) :

est une architecture de transformeur introduite par Google en 2018. Elle a été conçue pour résoudre le problème de compréhension des contextes complexes des mots dans une phrase. Il utilise une approche bidirectionnelle pour le traitement de la séquence de texte, ce qui signifie qu'il prend en compte les mots qui précèdent et qui suivent un mot donné dans la phrase.



utilisé dans de nombreuses applications, notamment la classification de texte, la recherche d'informations, la compréhension de texte et la réponse automatique aux messages.

## GPT (Generative Pre-trained Transformer) :

est une architecture de transformeur introduite par OpenAI en 2018. Elle a été conçue pour générer automatiquement du texte cohérent et naturel en se basant sur un contexte donné. Il utilise une approche d'apprentissage non-supervisé en utilisant des modèles de langage pour pré-entraîner un modèle de transformer. Le modèle est ensuite affiné sur des tâches spécifiques à l'aide de données supervisées.



utilisé dans de nombreuses applications, notamment la génération de texte, la traduction automatique, la rédaction de texte, la réponse automatique aux messages, la reconnaissance de la parole et la compréhension de texte.

## XLNet :

est une architecture de transformeur introduite par Google en 2019. Elle a été conçue pour résoudre les limitations des architectures précédentes en ce qui concerne la modélisation des dépendances à longue distance dans le texte.

Il utilise une approche pré-entraînée basée sur un modèle de langage en utilisant une permutation aléatoire des mots dans le texte d'entrée pour préserver l'indépendance des prédictions pour chaque mot.



utilisé dans de nombreuses applications, notamment la compréhension de texte, la génération de texte, la traduction automatique et la réponse automatique aux messages.

## ransformer-XL :

une architecture de transformeur introduite par l'équipe de recherche de Google en 2019.

Il utilise une approche de "segment-level recurrence" pour modéliser les dépendances à longue distance dans le texte.



utilisé dans de nombreuses applications, notamment la génération de texte, la compréhension de texte, la traduction automatique et la réponse automatique aux messages.

## 2

## Comparaison des avantages et des inconvénients de chaque architecture

ARCHITECTURE	AVANTAGES	INCONVÉNIENTS
Transformer de base	<ul style="list-style-type: none"><li>• architecture simple,</li><li>• efficace pour la traduction automatique et la génération de texte</li></ul>	<ul style="list-style-type: none"><li>• nécessite une grande quantité de données d'entraînement pour obtenir des performances de pointe .</li><li>• peut être coûteux en termes de temps et de ressources de calcul.</li></ul>
BERT	<ul style="list-style-type: none"><li>• performances de pointe pour la classification de texte et la recherche d'informations .</li><li>• prend en compte le contexte des mots dans une phrase.</li></ul>	<ul style="list-style-type: none"><li>• nécessite une grande quantité de données d'entraînement .</li><li>• peut avoir des problèmes de surapprentissage sur des données d'entraînement de petite taille.</li></ul>



ARCHITECTURE	AVANTAGES	INCONVÉNIENTS
GPT	<ul style="list-style-type: none"> <li>performances de pointe pour la génération de texte, la réponse automatique aux messages et la rédaction assistée par ordinateur .</li> <li>peut être utilisé pour la création de texte créatif.</li> </ul>	<ul style="list-style-type: none"> <li>nécessite une grande quantité de données d'entraînement pour obtenir des performances de pointe .</li> <li>peut générer des réponses incohérentes ou inappropriées dans certaines situations.</li> </ul>
XLNet	<ul style="list-style-type: none"> <li>performances de pointe pour la classification de texte et la recherche d'informations.</li> <li>prend en compte toutes les relations entre les mots dans une phrase.</li> <li>moins de risque de surapprentissage.</li> </ul>	<ul style="list-style-type: none"> <li>nécessite une grande quantité de données d'entraînement pour obtenir des performances de pointe .</li> <li>peut être coûteux en termes de temps et de ressources de calcul.</li> </ul>
ransformer-XL	<ul style="list-style-type: none"> <li>performances de pointe pour la compréhension de texte et la génération de texte à long terme.</li> <li>capable de traiter des séquences de longue durée.</li> </ul>	<ul style="list-style-type: none"> <li>nécessite une grande quantité de données d'entraînement pour obtenir des performances de pointe .</li> <li>peut être coûteux en termes de temps et de ressources de calcul.</li> </ul>

## Le principe de fonctionnement

1

principe de fonctionnement des transformeurs



les transformateurs utilisent **des blocs de transformation** pour modéliser des relations non linéaires entre les différentes parties de la séquence d'entrée et **des blocs d'attention** pour calculer une pondération de l'importance de chaque partie de la séquence d'entrée pour chaque partie de la sortie.



Cette combinaison de blocs permet aux transformateurs de traiter simultanément toute la séquence d'entrée en se concentrant sur les parties les plus pertinentes.

Le principe de fonctionnement des transformeurs repose sur **le mécanisme d'attention**

## L'attention

une opération mathématique qui permet de calculer une pondération de l'importance de chaque partie de la séquence d'entrée pour chaque partie de la sortie.

→ Cette pondération permet aux transformateurs de se concentrer sur les parties les plus pertinentes de la séquence d'entrée, ce qui améliore les performances et réduit le temps de calcul.

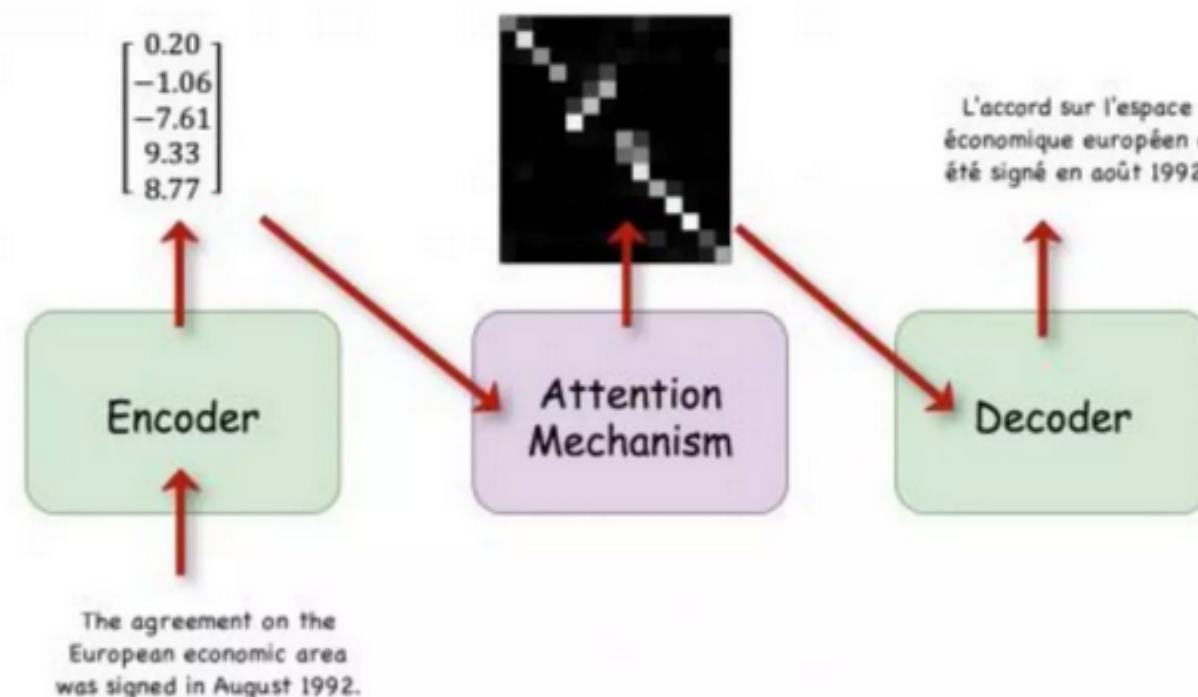
→ qui permet au réseau de se concentrer sur des parties spécifiques de la séquence d'entrée pendant la phase d'encodage. Cela permet au réseau de capturer les relations entre les éléments de la séquence d'entrée, en particulier les dépendances à long terme.

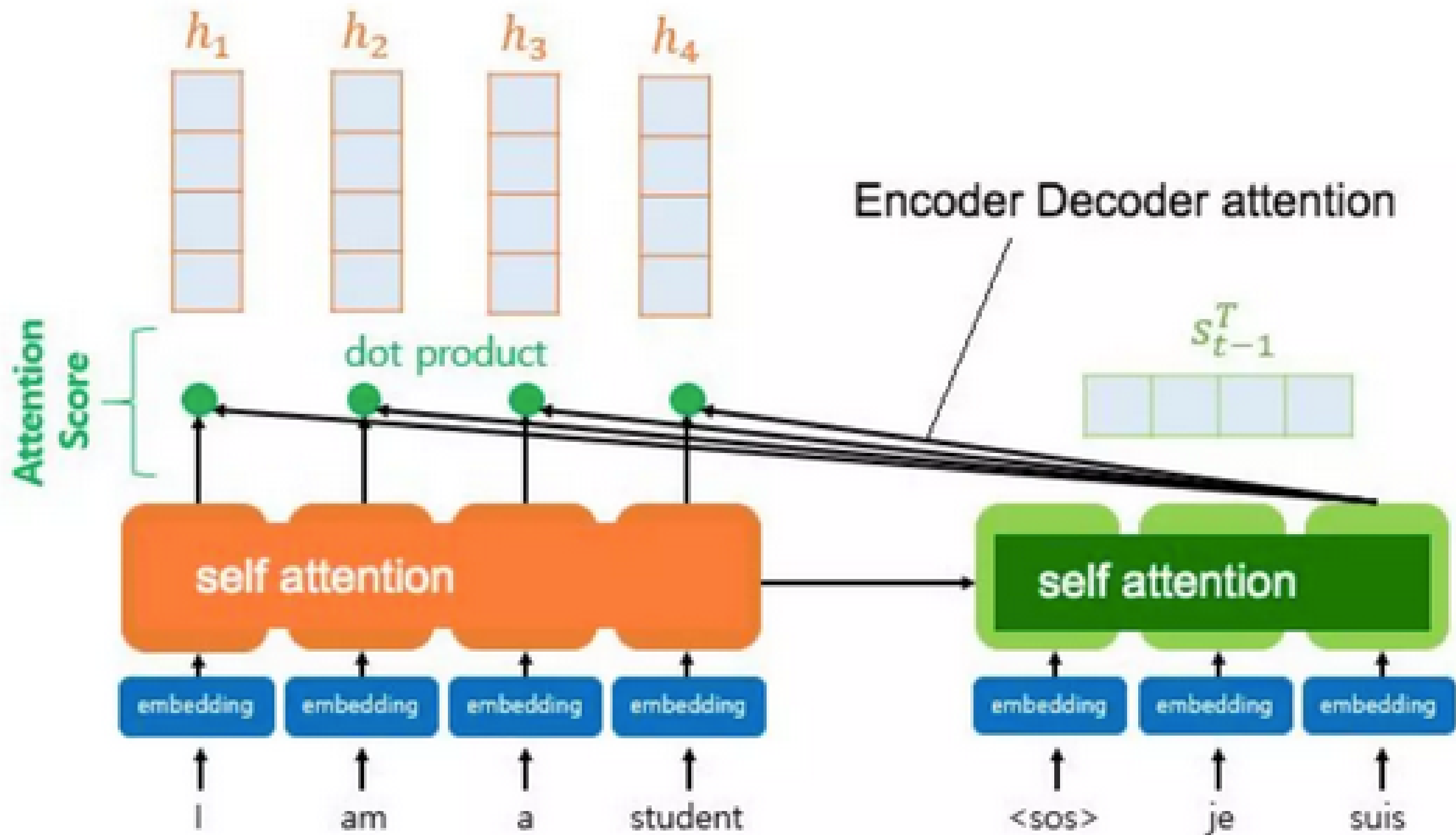


Le fonctionnement des transformeurs peut être divisé en deux parties principales :  
l'encodeur et le décodeur.

**L'encodeur** transforme une séquence d'entrée en une représentation de haute dimension appelée vecteur de contexte. Il fait cela en utilisant des couches de neurones qui calculent une attention multi-tête sur la séquence d'entrée, en prenant en compte les relations entre tous les éléments de la séquence. Cela permet à l'encodeur de capturer les informations importantes de la séquence et de les encoder dans un vecteur de contexte.

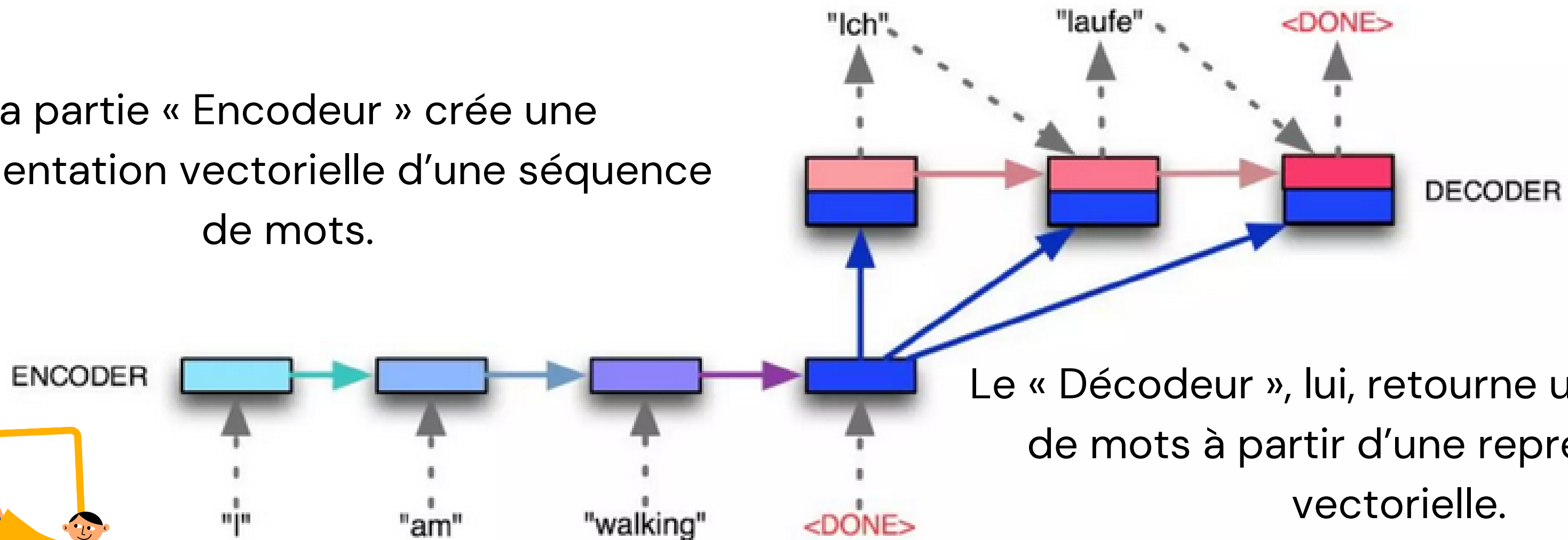
**Le décodeur** prend en entrée ce vecteur de contexte et utilise également des couches de neurones pour générer une séquence de sortie. Pour cela, il utilise également l'attention multi-tête, mais cette fois-ci pour se concentrer sur les parties de la séquence d'entrée les plus importantes pour la génération de la séquence de sortie. Le décodeur est également capable de générer de manière itérative la séquence de sortie, en utilisant les sorties de la couche précédente comme entrée pour la suivante, jusqu'à la fin de la séquence.





# Encodeur- Décodeur dans la traduction

la partie « Encodeur » crée une représentation vectorielle d'une séquence de mots.



Le « Décodeur », lui, retourne une séquence de mots à partir d'une représentation vectorielle.





## Exemple :

Prenons la phrase en français : "Le chat noir dort sur le tapis".

1) **Embedding** : chaque élément de la séquence d'entrée est représenté par un vecteur d'embedding, noté  $x_i$ , qui est obtenu en multipliant le vecteur d'entrée par une matrice de poids  $W$ , puis en appliquant une fonction d'activation non linéaire  $f(x_i)$  :

$$x_i = f(W * \text{input}_i)$$

ooo

```
tensor([[ 0.0412,  0.3475, -1.6489, -0.3344, -0.0784],
        [-1.3724,  0.9831,  0.4509,  0.6195,  0.6632],
        [-1.2835, -0.2205, -0.1645, -1.3772, -0.6476],
        [ 0.0483, -0.0383, -0.3476, -0.2257,  0.2975],
        [-0.3689, -1.0289,  0.2744, -1.1141, -0.2075],
        [ 0.0412,  0.3475, -1.6489, -0.3344, -0.0784],
        [ 0.6184, -0.4674,  1.3243, -0.5096,  1.6194]],
        grad_fn=<EmbeddingBackward>)
```

Cela devrait afficher



```
import torch.nn as nn
import torch

# définition d'un dictionnaire de correspondance de chaque mot à son index dans le vocabulaire
vocab = {"Le": 0, "chat": 1, "noir": 2, "dort": 3, "sur": 4, "le": 5, "tapis": 6}

# définition d'un embedding avec une taille de sortie de 5
embedding = nn.Embedding(len(vocab), 5)

# définition de la phrase à encoder
sentence = "Le chat noir dort sur le tapis"

# conversion de la phrase en une liste d'indices (un indice pour chaque mot)
indexed_sentence = [vocab[word] for word in sentence.split()]

# conversion de la liste en un tenseur PyTorch
indexed_sentence = torch.LongTensor(indexed_sentence)

# application de l'embedding à la phrase
embedded_sentence = embedding(indexed_sentence)

# affichage de l'embedding
print(embedded_sentence)
```



2) **Encodage** : chaque couche d'encodage prend en entrée les embeddings de la séquence d'entrée et génère une nouvelle représentation de la séquence en utilisant l'attention. La représentation de la séquence à la  $k$ -ième couche d'encodage est notée  $H^k$ .

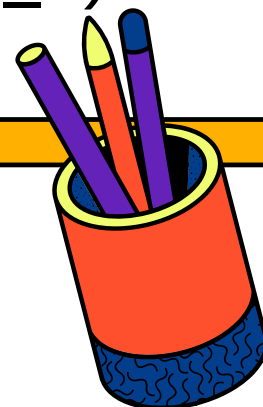
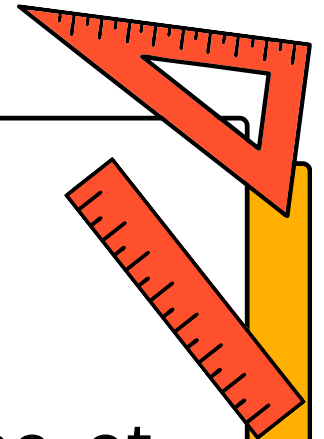
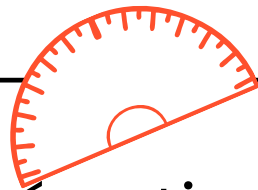
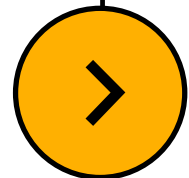
Mathématiquement, l'étape d'encodage peut être définie comme suit :

$$H^k = \text{Attention}(Q^k, K^k, V^k)$$

où  $Q^k$ ,  $K^k$  et  $V^k$  sont des matrices de poids apprises pour la  $k$ -ième couche, et Attention est une fonction d'attention qui calcule les pondérations pour chaque élément de la séquence d'entrée en utilisant la formule :

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k}) * V$$

où softmax est une fonction de normalisation,  $d_k$  est la dimension de la matrice  $K$  et la division par  $\sqrt{d_k}$  est utilisée pour normaliser les scores d'attention.

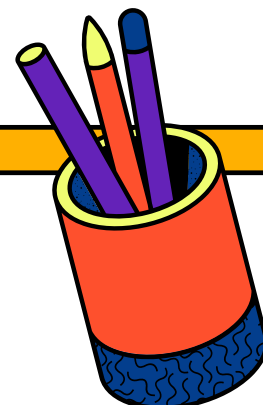
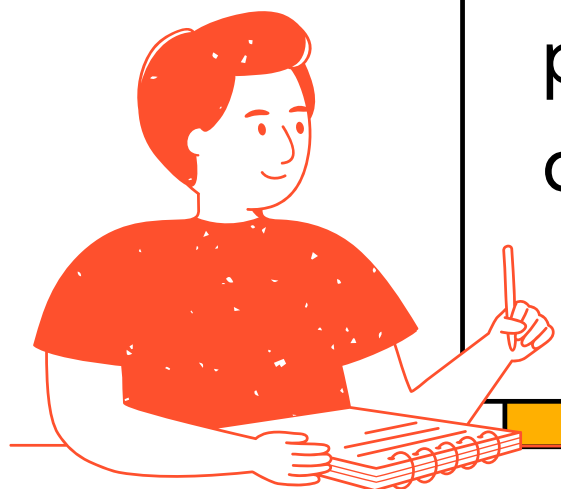
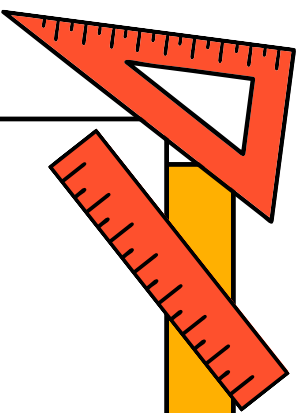
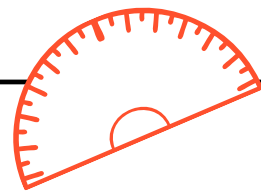
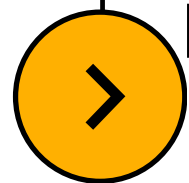


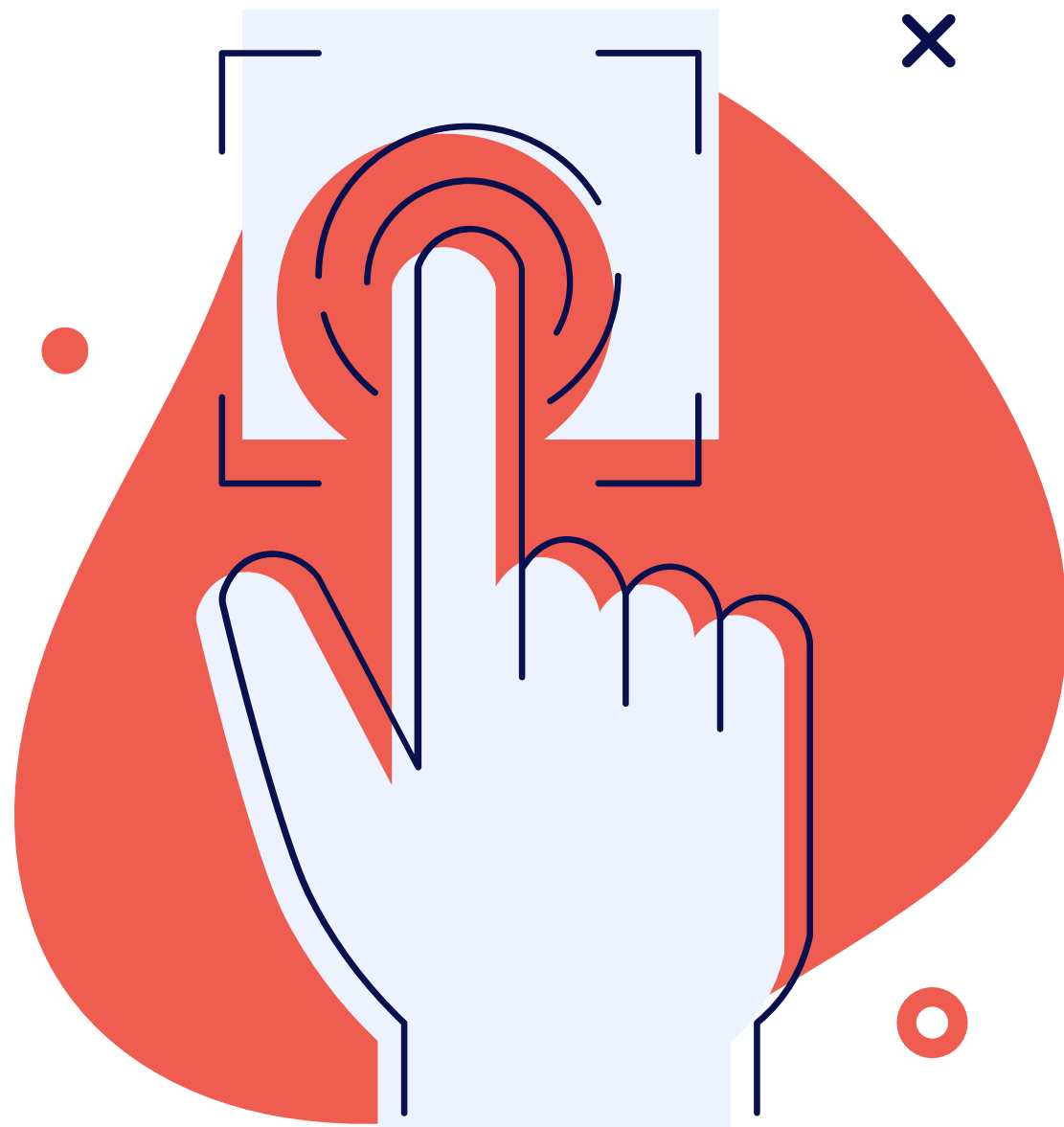
3) **Décodage** : le processus de décodage est similaire à l'encodage, mais il utilise des matrices de poids différentes pour calculer les scores d'attention. La représentation de la séquence à la  $k$ -ième couche de décodage est notée  $G^k$  .et est calculée comme suit :

Mathématiquement, l'étape de décodage peut être définie comme suit :

$$G^k = \text{Attention}(Q^k, K^k, V^k)$$

où  $Q^k$ ,  $K^k$  et  $V^k$  sont des matrices de poids apprises pour la  $k$ -ième couche, et Attention est une fonction d'attention qui utilise les scores d'attention calculés à partir de la couche précédente pour pondérer les embeddings de la séquence d'entrée.





4) **Output** : une fois que la séquence de sortie a été générée, les prédictions sont calculées en appliquant une fonction d'activation softmax sur les vecteurs de sortie, notés  $y_i$  :

$$y_i = \text{softmax}(W * G^k * v_i)$$

où  $W$  est une matrice de poids apprise,  $G^k$  est la dernière représentation de la séquence générée à la  $k$ -ième couche de décodage et  $v_i$  est un vecteur de poids pour la  $i$ -ème classe de sortie.

The black cat is sleeping on the  
carpet.

Résultat finale

Ces étapes sont répétées pour chaque phrase à traduire, et le modèle est entraîné à minimiser le score de perplexité en ajustant les poids des différentes couches du modèle.

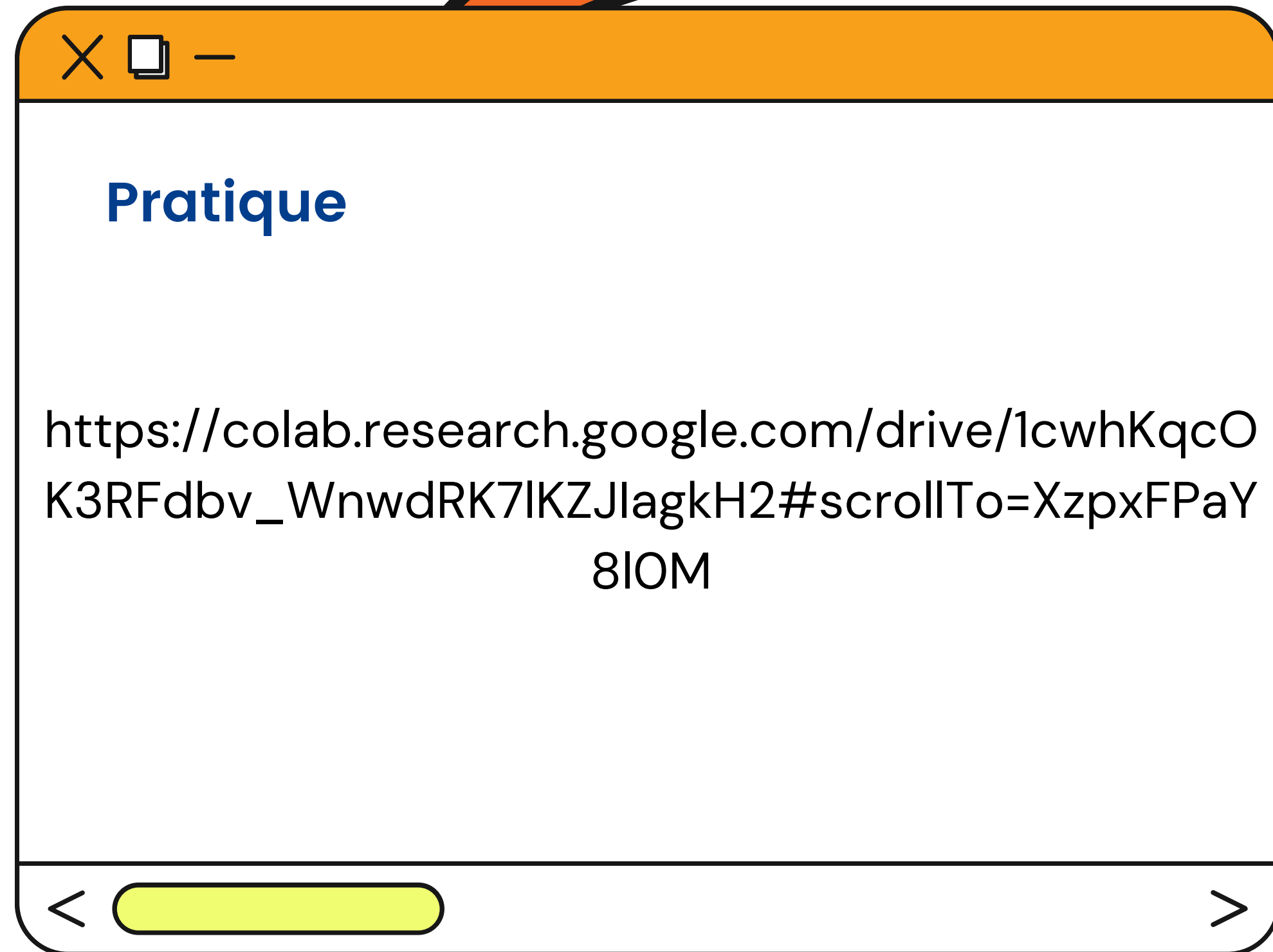
## 2

## avantages de cette architecture par rapport aux autres

- L'utilisation de l'attention multi-tête permet aux transformeurs de mieux capturer les relations à long terme entre les éléments d'une séquence, par rapport aux modèles de réseaux de neurones récurrents tels que les RNN et les LSTM. (efficacité)
- L'attention multi-tête permet également une parallélisation plus efficace de l'entraînement, améliorant ainsi la vitesse d'apprentissage des transformeurs par rapport aux modèles de RNN et LSTM. (rapidité)
- L'architecture des transformeurs est plus flexible que les modèles de RNN et LSTM car elle ne repose pas sur une séquence d'entrée linéaire. Les transformeurs peuvent donc traiter des séquences de longueur variable, sans avoir à traiter les éléments de la séquence un par un. (performances supérieures sur certaines tâches)



Exemple  
d'application et  
résultats





## Conclusion

les transformeurs sont une architecture de deep learning prometteuse pour le traitement du langage naturel, offrant une meilleure capacité à capturer les relations à long terme et une plus grande flexibilité dans la gestion des séquences de longueur variable. La recherche future dans ce domaine promet de continuer à repousser les limites de cette architecture et de conduire à des avancées significatives dans le domaine du traitement du langage naturel.



# Merci !

AVEZ-VOUS DES QUESTIONS ?

