Natural Language Processing

Redha Moulla

Ecole Centrale de Casablanca

Automne 2023

Plan

- Introduction au NLP
- Techniques statistiques pour le NLP
- Machine learning pour le NLP
- Deep learning pour le NLP
- Apprentissage auto-supervisé
- Apprentissage multimodal

Représentations distribuées

Introduction aux Embeddings

- **Qu'est-ce qu'un Embedding ?**
  - Un embedding est une représentation vectorielle d'un mot qui capture le contexte du mot dans un document, des relations sémantiques et syntaxiques avec d'autres mots.
  - Ils transforment des mots en vecteurs de nombres pour que les algorithmes de machine learning puissent les traiter efficacement.
- **Pourquoi sont-ils importants ?**
  - Les embeddings permettent de capturer non seulement l'identité d'un mot mais aussi ses aspects sémantiques et contextuels.
  - Ils facilitent des tâches telles que la classification de texte, la traduction automatique, et la détection des sentiments.
- **Évolution des embeddings**
  - Historiquement, les mots étaient représentés comme des indices ou des vecteurs one-hot, où chaque mot est indépendant des autres.
  - Les embeddings modernes, tels que Word2Vec et GloVe, représentent les mots dans des espaces vectoriels continus où les mots de sens similaires sont proches les uns des autres.

Différence entre One-hot Encoding et Embeddings

- **One-hot Encoding**
  - Chaque mot est représenté par un vecteur avec un '1' dans la position qui lui est propre et des '0' partout ailleurs. Ce type de représentation est simple mais très inefficace en termes d'espace et ne capture pas les relations entre les mots.
  - Exemple : pour un vocabulaire de dimension 100 000 :

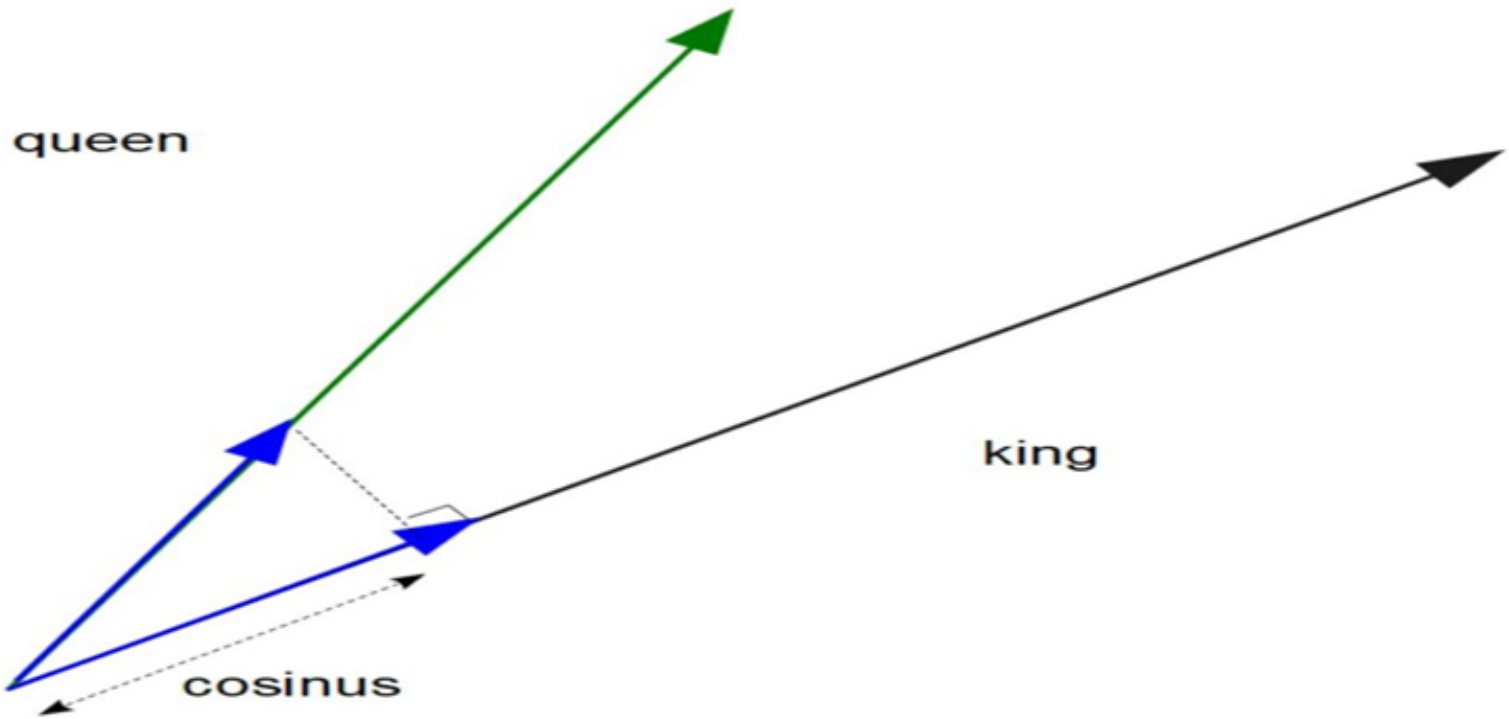
    Véhicule = [0, 0, 1, 0, 0, 0, 0, 0, 0,. . . ,0 ,0]
    Voiture = [0, 0, 0, 0, 0, 1, 0, 0, 0,. . . ,0 ,0]

    Cette representation ne permet pas de prendre en compte la dimension sémantique

- **Embeddings**
  - Les embeddings représentent les mots comme des vecteurs denses de nombres flottants (généralement entre 50 et 300 dimensions).
  - Cette représentation est beaucoup plus riche et peut capturer des relations complexes entre les mots, comme la similarité sémantique.

Similarité sémantique

Nous sommes intéressés par des représentations de mots qui capturent la distance sémantique, sous forme de produit scalaire par exemple (ou distance cosine).



Représentations distribuées : Principes

"You shall know a word by the company it keeps"

— J.R. Firth (1957)

Les mots sont similaires s'ils apparaissent fréquemment dans le même contexte.

- Il conduit son *véhicule* pour rentrer à la maison.
- Il conduit sa *voiture* pour rentrer chez lui.

Construction d'une représentation distribuée

Considérons le corpus suivant à titre d'exemple : **cnn in crop analysis**

cnn and svm are widely used.
linear_regression performed along with svm
linear_regression for crop and farm
svm being used for farm monitoring
Do cnn, svm and linear_regression appear in the same context?

Le vocabulaire est alors :

`[cnn, in, crop, analysis, and, svm, are, widely, used, linear_regression, performed, along, with, for, farm, being, monitoring, do, appear, the, same, context]`

$$\dim(\text{vocabulaire}) = 22$$

Similarité sémantique

La matrice de co-occurrence représente la fréquence à laquelle les mots apparaissent ensemble deux à deux.

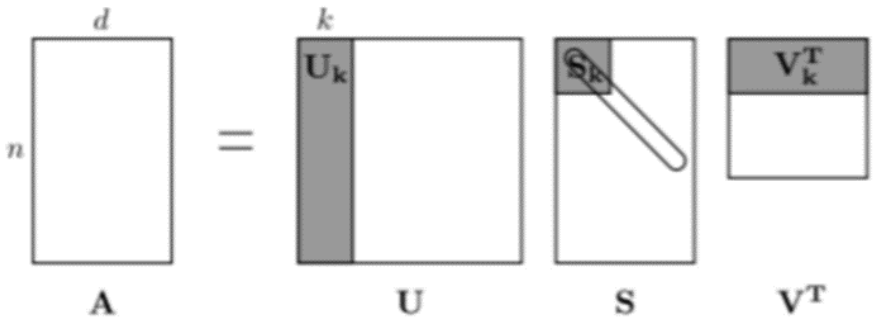
	cnn	in	crop	...
cnn	0	2	1	1	1	2	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1
in	2	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1
crop	1	1	0	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0
...	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	1	0	0	1	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0
	2	1	0	0	1	0	1	1	2	2	1	1	1	1	1	1	1	1	1	1	1	1
	1	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	1	2	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0
	1	1	1	0	1	2	0	0	0	0	1	1	1	1	1	0	0	1	1	1	1	1
	0	0	0	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	1	1	0	0	1	1	0	0	0	0	2	1	1	0	0	0	0	0
	0	0	1	0	1	1	0	0	1	1	0	0	0	2	0	1	1	0	0	0	0	0
	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0
	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0
	1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1
	1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1	1
	1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1
	1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	1
	1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	0

Réduction de la dimension

La décomposition en valeurs singulières est une technique permettant de réduire la dimension des données (similaire à l'ACP).

Étant donné une matrice $A \in \mathbb{R}^{n \times d}$

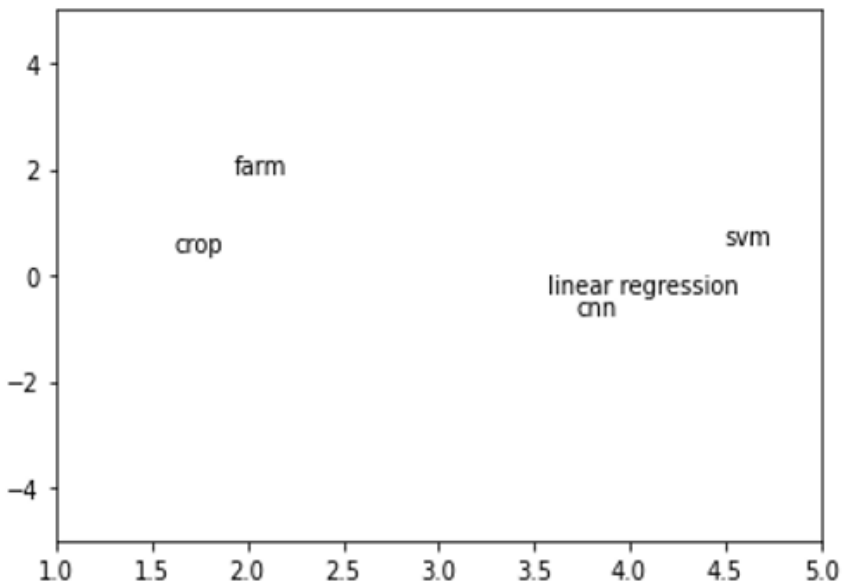
$$A = UDV^T \quad \text{où} \quad U \in \mathbb{R}^{n \times r}, D \in \mathbb{R}^{r \times r}, V \in \mathbb{R}^{d \times r}.$$



U est la matrice contenant les représentations (vecteurs de mots)

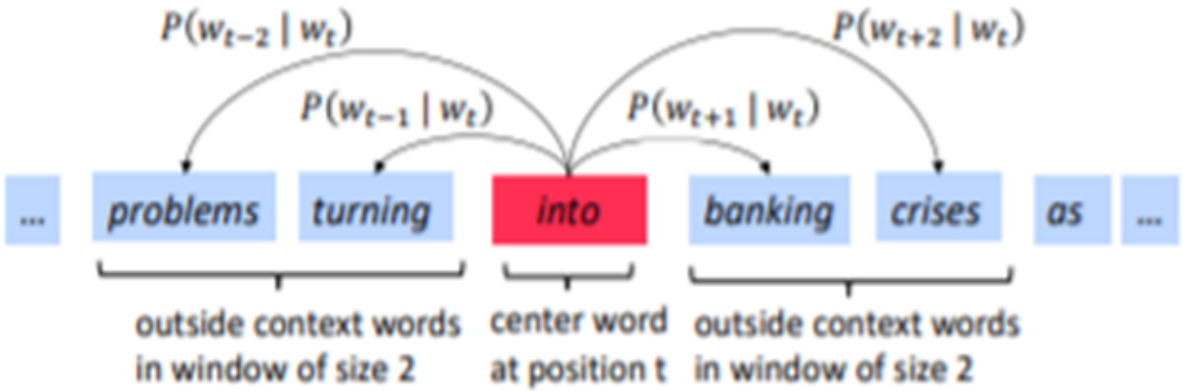
Vsiualisation des représentations distribuées

```
cnn [[ 3.72113518, -0.73233585],
ln   [ 2.7940387 , -1.40864784],
crop [ 1.61178082,  0.44786021],
...  [ 0.77784994, -0.31230389],
     [ 2.12245048,  1.29571556],
     [ 4.49293777,  0.58090417],
     [ 1.33312748,  0.66758743],
     [ 1.33312748,  0.66758743],
     [ 2.25882809,  1.80738544],
     [ 3.56593605, -0.31006976],
     [ 0.95394179,  0.079159  ],
     [ 0.95394179,  0.079159  ],
     [ 0.95394179,  0.079159  ],
     [ 1.92921553,  1.94783497],
     [ 1.92921553,  1.94783497],
     [ 1.12301312,  1.42126096],
     [ 1.12301312,  1.42126096],
     [ 2.26025282, -1.31571175],
     [ 2.26025282, -1.31571175],
     [ 2.26025282, -1.31571175],
     [ 2.26025282, -1.31571175],
     [ 2.26025282, -1.31571175]]
```

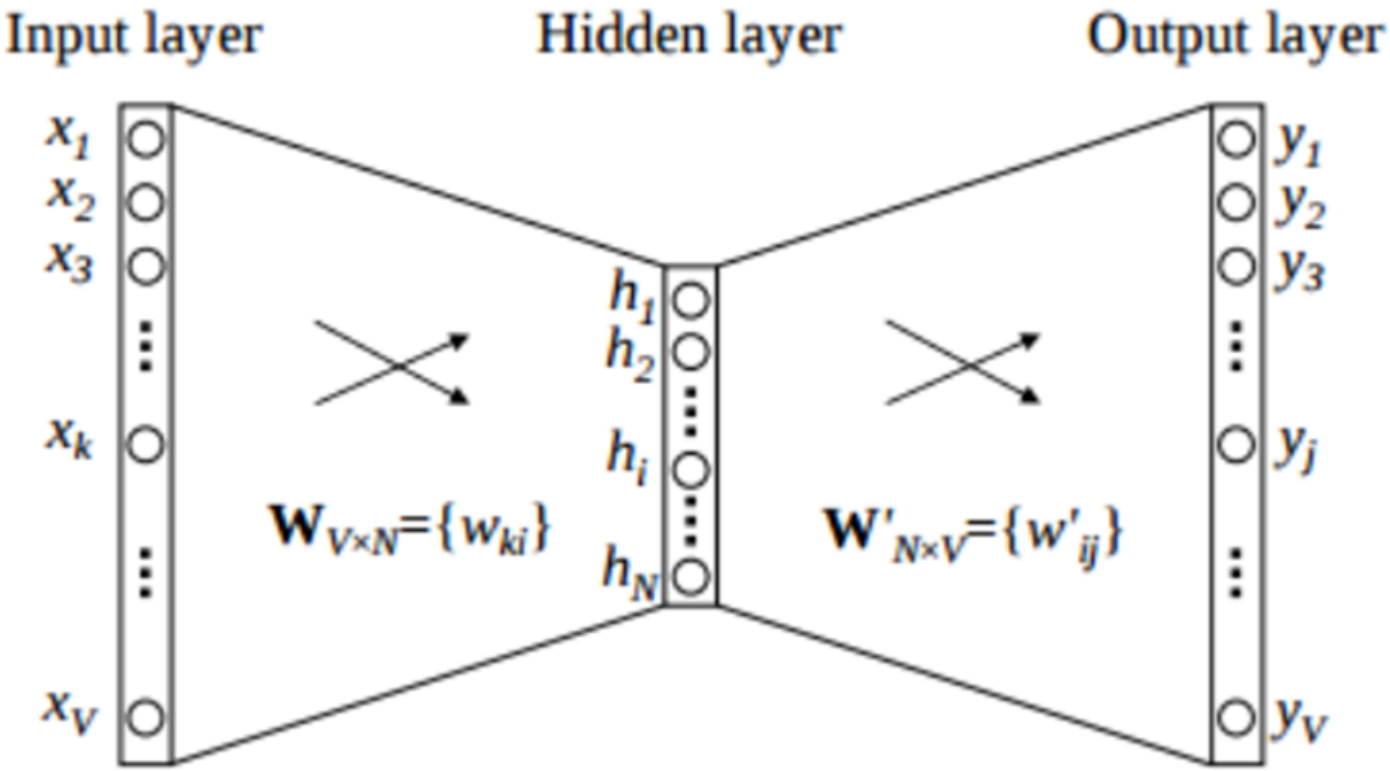


Word2Vec : Principes

- **Principes de base :**
  - Word2Vec est basé sur l'hypothèse distributionnelle : les mots qui apparaissent dans des contextes similaires ont des significations similaires.
  - Utilise un réseau de neurones peu profond pour apprendre des embeddings de mots à partir de grands corpus.
- **Deux architectures principales :**
  1. *Continuous Bag of Words (CBOW)* : Prédit le mot cible à partir du contexte.
  2. *Skip-Gram* : Prédit le contexte à partir du mot cible.



Architecture du modèle CBOW



Modèle CBOW : Formulation mathématique

- **Objectif :** Prédire le mot cible w_t en fonction du contexte C.
- **Fonction de prédiction :**

$$P(w_t|C) = \frac{e^{\mathbf{v}_{w_t}^T \cdot \mathbf{h}}}{\sum_{w \in W} e^{\mathbf{v}_w^T \cdot \mathbf{h}}} \tag{1}$$

- **Où :**
  - $\mathbf{v}_w$ est le vecteur de l'embedding du mot w.
  - $\mathbf{h}$ est le vecteur du contexte, la moyenne des embeddings des mots du contexte.
  - W est l'ensemble de tous les mots du vocabulaire.
- **Optimisation :** Minimisation de la fonction de coût, souvent une forme de cross-entropy ou log-likelihood négatif.

Modèle Skip-Gram : Formulation mathématiques

- **Objectif :** Prédire les mots de contexte à partir du mot cible w_t.
- **Fonction de prédiction :**

$$P(C|w_t) = \prod_{w_c \in C} \frac{e^{\mathbf{v}_{w_c}^T \cdot \mathbf{h}}}{\sum_{w \in W} e^{\mathbf{v}_w^T \cdot \mathbf{h}}} \tag{2}$$

- **Où :**
  - $\mathbf{v}_{w_c}$ est le vecteur de l'embedding du mot de contexte w_c.
  - $\mathbf{h}$ est le vecteur de l'embedding du mot cible w_t.
- **Optimisation :** Minimisation de la fonction de coût, souvent une forme de cross-entropy ou log-likelihood négatif.

Optimisation

- **Negative sampling :**
  - Pour chaque paire de mots (cible, contexte), des paires négatives sont générées en échantillonnant des mots aléatoires du vocabulaire.
  - Améliore l'efficacité de l'entraînement en réduisant le nombre de calculs nécessaires pour chaque étape de mise à jour.
- **Sous-échantillonnage des mots fréquents :**
  - Les mots très fréquents (par exemple, "le", "est") sont sous-échantillonnés pour améliorer la qualité des embeddings et accélérer l'entraînement.
  - Réduit la dominance des mots fréquents dans la formation des embeddings.
- **Fonction de coût :**
  - Utilisation typique de la log-likelihood négative comme fonction de coût.
  - L'objectif est de maximiser la probabilité des mots réels (positifs) tout en minimisant celle des mots échantillonnés négativement.
- **Mise à jour des poids :**
  - Les poids du réseau sont mis à jour en utilisant des méthodes de descente de gradient stochastique.
  - Les gradients sont calculés par backpropagation à travers le réseau.

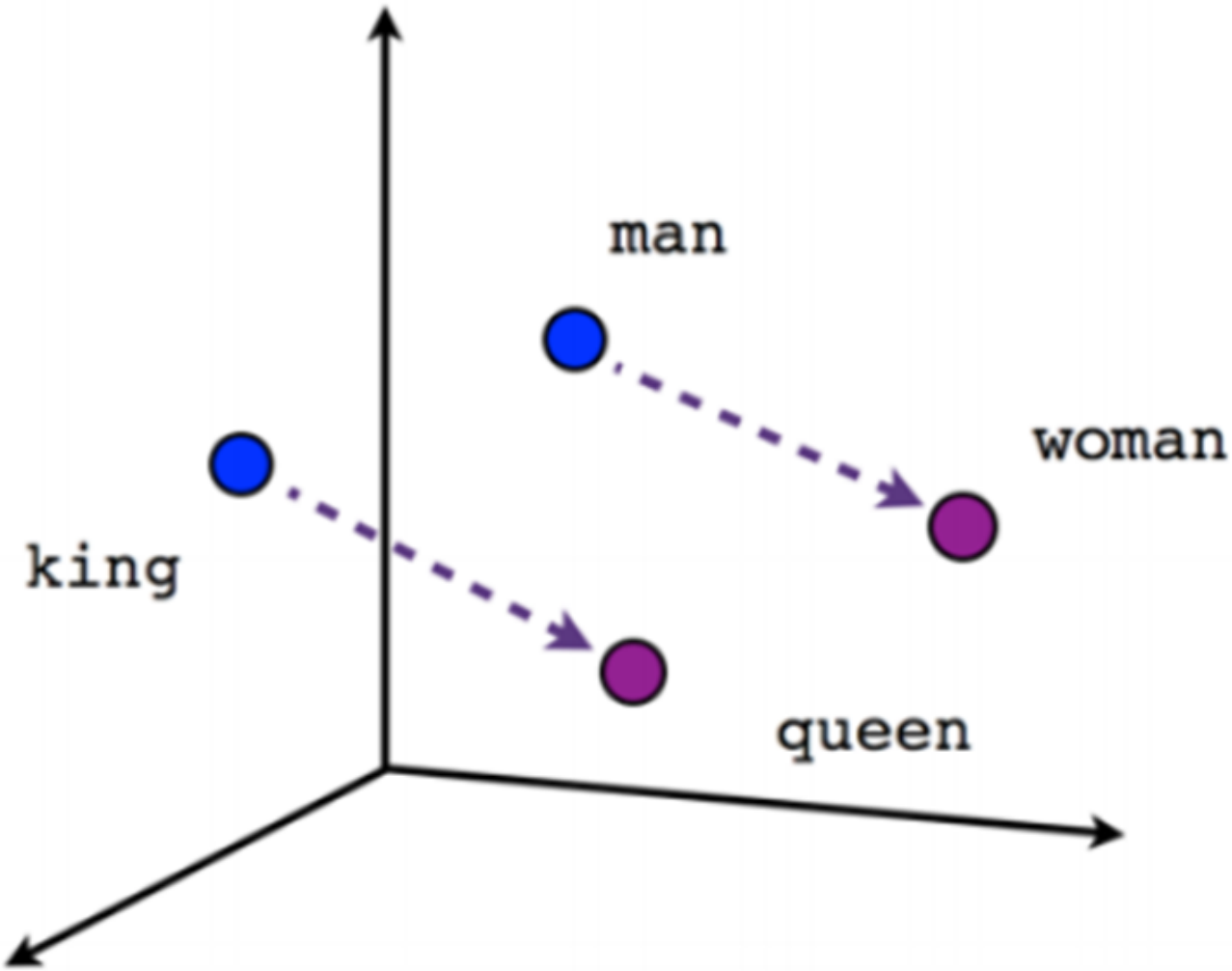
Negative sampling : Formulation mathématique

- **Objectif du Negative Sampling :**
  - Simplifier l'optimisation de la fonction de coût en remplaçant le problème de classification multinomiale par une série de classifications binaires.
- **Formule du Negative Sampling :**

$$L(\theta) = \log \sigma(\mathbf{v}_{w_O}^T \mathbf{h}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} [\log \sigma(-\mathbf{v}_{w_i}^T \mathbf{h})] \tag{3}$$

  - Où $\sigma(x) = \frac{1}{1+e^{-x}}$ est la fonction logistique.
  - $\mathbf{v}_{w_O}$ est le vecteur du mot cible et $\mathbf{h}$ est le vecteur du mot d'entrée (pour Skip-Gram) ou le vecteur du contexte (pour CBOW).
  - k est le nombre de mots négatifs à échantillonner, tirés selon une distribution de probabilité $P_n(w)$.
- **Distribution de probabilité des mots négatifs ($P_n(w)$) :**
  - Généralement, les mots négatifs sont échantillonnés selon une distribution qui favorise les mots fréquents, mais pas autant que la distribution de fréquence des mots dans le corpus.
  - Une pratique courante est d'utiliser $P_n(w) \propto (U(w))^{3/4}$, où $U(w)$ est la fréquence brute du mot w.

Représentations Word2Vec



Applications pratiques de Word2Vec

- **Analogie de mots :**
  - Word2Vec est célèbre pour capturer des relations complexes, comme "homme est à femme ce que roi est à reine".
  - Permet de résoudre des analogies en utilisant des opérations arithmétiques simples sur les vecteurs de mots.
- **Clustering sémantique :**
  - Les embeddings peuvent être utilisés pour regrouper des mots sémantiquement similaires, facilitant l'analyse de textes volumineux.
- **Amélioration des systèmes de recommandation :**
  - Les vecteurs de mots de Word2Vec peuvent être utilisés pour améliorer la précision des recommandations en comprenant mieux les préférences des utilisateurs.

Autres représentations distribuées

Il existe d'autres répsentations distribuées :

- Glove (2014) : proposé par une équipe de Stanford, il combine à la fois les techniques modernes de deep learning et les techniques statistiques (co-occurrences entre les mots). Il permet d'avoir des représentations des mots plus globales que Word2Vec.
- Fasttext (à partir de 2016) : la librairie a été créée par Facebook. Le modèle est entraîné sur des subwords (n-grams de caractères). Il est ainsi plus efficace pour traiter les mots inconnus (out of vocabulary).
- Représentations contextuelles (Transformers), etc.

Réseaux de neurones récurrents

Perspective historique des RNN

Les Réseaux de Neurones Récurrents (RNN) ont évolué au fil du temps, marquant des étapes importantes dans le domaine du machine learning et du NLP.

Développements clés :

- **1980s : Naissance des RNN** - Introduction des concepts de base des RNN par John Hopfield et David Rumelhart, posant les fondations des réseaux à mémoire.
- **1990s : Popularisation des RNN** - Jordan et Elman développent des architectures permettant une meilleure gestion des séquences temporelles et des dépendances.
- **Début des années 2000** - Identification des problèmes de disparition et d'explosion du gradient, limitant l'efficacité des RNN sur de longues séquences.
- **1997 : Avènement des LSTM** - Introduction des LSTM par Hochreiter et Schmidhuber, offrant une solution aux problèmes de mémoire à long terme.
- **2010s : GRU** - Les GRU simplifient la structure des LSTM. Les RNN sont de plus en plus utilisés dans des applications complexes comme la traduction automatique et la reconnaissance vocale.

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.

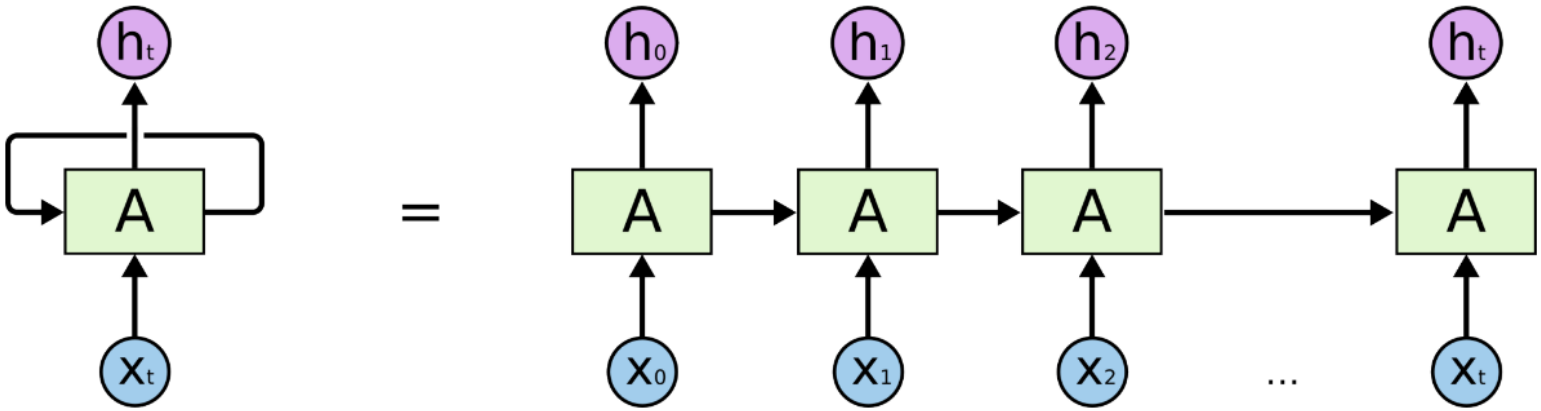
Caractéristiques:

- **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
- σ 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 ω 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.

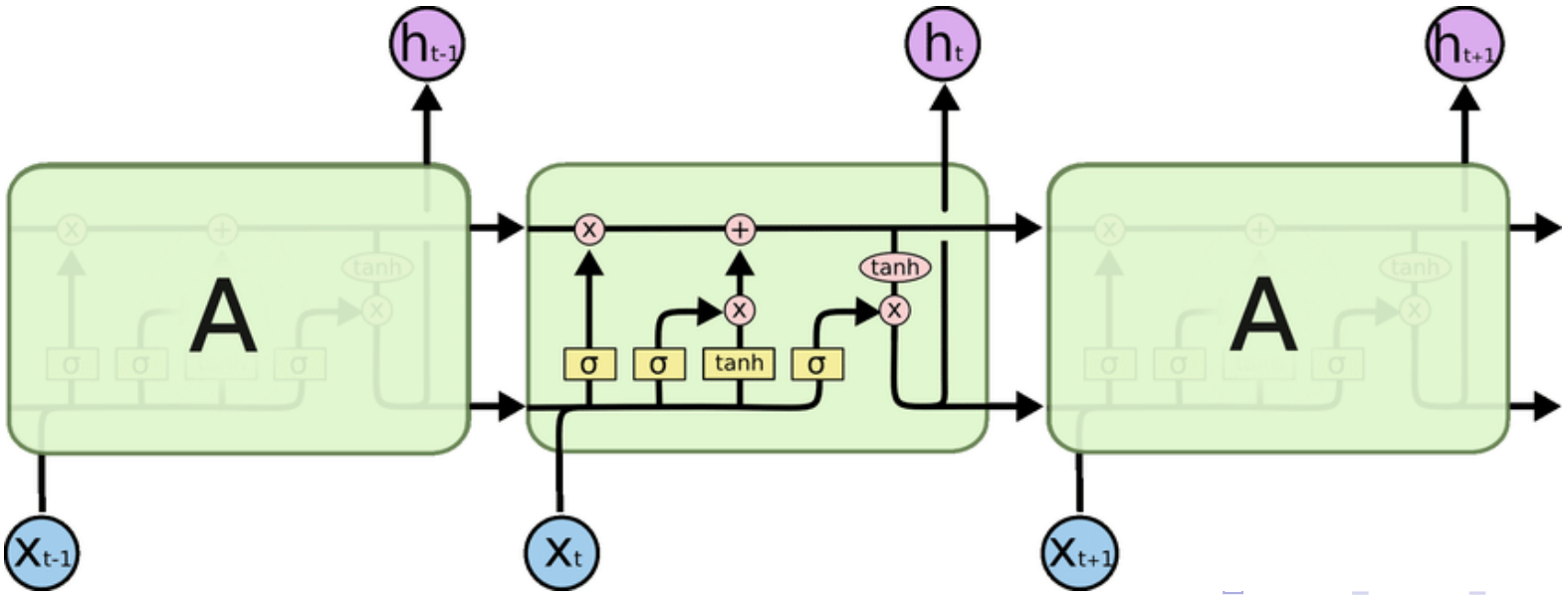
Solutions au problème de disparition du gradient :
Pour atténuer le problème de disparition du gradient, différentes solutions ont été proposées :

- 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.



Formulation mathématique des LSTM

La formulation d'un LSTM pour un instant t :

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

où :

- C_t est l'état de la cellule à l'instant t
- h_t est l'état caché à l'instant t
- f_t, i_t, o_t sont respectivement les états des portes d'oubli, d'entrée, et de sortie
- Les W et b représentent les poids et biais
- σ est la fonction sigmoïde
- $\tanh$ est la tangente hyperbolique

RNN avec Gated Recurrent Unit (GRU)

Les GRU sont une autre variante des RNN qui simplifient l'architecture des LSTM en combinant la porte d'oubli et la porte d'entrée en une seule porte de mise à jour.

Formule d'un GRU pour un instant t :

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t] + b)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

où :

- z_t est l'état de la porte de mise à jour
- r_t est l'état de la porte de réinitialisation
- $\tilde{h}_t$ est l'état candidat pour h_t
- h_t est l'état caché final à l'instant t

Applications des RNN

Les RNN sont largement utilisés dans divers domaines tels que :

- Traitement du langage naturel : traduction automatique, génération de texte.
- Reconnaissance vocale : conversion de la parole en texte.
- Prévision de séries temporelles : prévision météorologique, analyse boursière.

Modèles Seq2Seq

Introduction aux modèles Seq-to-Seq

- Les modèles de séquence à séquence (seq-to-seq) sont une classe de modèles en apprentissage profond conçus pour transformer une séquence d'entrée en une séquence de sortie.
- Ils sont essentiels dans le domaine du traitement automatique du langage naturel (NLP), avec des applications allant de la traduction automatique au résumé de texte et à la génération de dialogue.

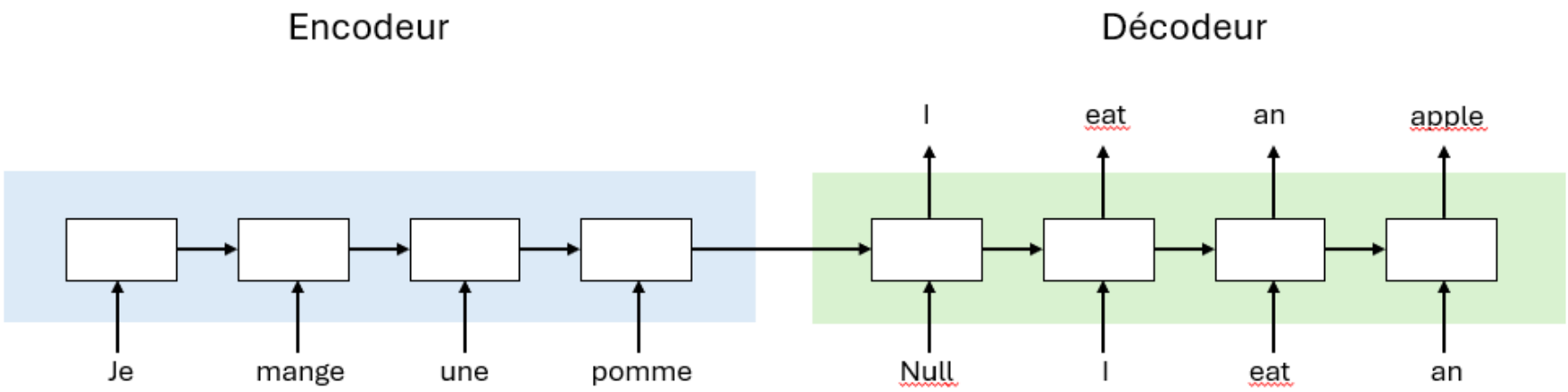
Pourquoi les modèles Seq-to-Seq?

- **Flexibilité** : Peuvent gérer des séquences d'entrée et de sortie de longueurs variables.
- **Puissance** : Capables de capturer des dépendances complexes entre les éléments de la séquence.
- **Universalité** : Applicables à une large gamme de tâches en NLP, démontrant une capacité exceptionnelle à modéliser le langage et d'autres séquences.

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.



Fonctionnement de l'Encodeur

- L'encodeur transforme la séquence d'entrée, souvent textuelle, en une série d'états cachés représentant l'information contenue dans la séquence.
- Utilise généralement des réseaux de neurones récurrents (RNN), LSTM ou GRU pour traiter les dépendances temporelles.

$$\boldsymbol{h}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

où $\boldsymbol{x}_t$ est l'entrée à l'instant t, et $\boldsymbol{h}_t$ est l'état caché.

Fonctionnement du décodeur

- Le décodeur commence par le vecteur de contexte fourni par l'encodeur pour commencer la génération de la séquence de sortie.
- À chaque étape, le décodeur est alimenté par son propre état caché précédent et parfois par une partie de la sortie précédemment générée.
- Le processus continue jusqu'à ce qu'un symbole de fin de séquence soit généré.

$$\boldsymbol{s}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{s}_{t-1}, \boldsymbol{C})$$

où $\boldsymbol{y}_{t-1}$ est la sortie générée à l'instant $t-1$, $\boldsymbol{s}_t$ est l'état caché du décodeur, et $\boldsymbol{C}$ est le vecteur de contexte.

Importance des modèles Seq-to-Seq

Les modèles seq-to-seq ont révolutionné la façon dont les machines comprennent et génèrent le langage naturel, avec des impacts significatifs dans plusieurs domaines :

- **Traduction automatique** : Ils constituent la base des systèmes de traduction automatique de pointe, permettant une traduction fluide et naturelle entre les langues.
- **Résumé automatique** : Permettent de générer des résumés concis et pertinents de longs documents ou articles.
- **Agents conversationnel** : Facilitent la création de systèmes de dialogue intelligents capables de mener des conversations naturelles avec les utilisateurs.
- **Reconnaissance et génération de la parole** : Appliqués à la conversion de la parole en texte et vice-versa, améliorant l'accessibilité et l'interaction homme-machine.

Défis des modèles Seq-to-Seq

Malgré leur succès, les modèles seq-to-seq rencontrent plusieurs défis :

- **Gestion de la Longueur** : Difficulté à maintenir la performance sur de très longues séquences due à la dilution de l'information.
- **Complexité Computationnelle** : Les opérations récurrentes sont coûteuses en termes de calcul et de mémoire.
- **Nécessité d'Attention** : Introduction de mécanismes d'attention pour améliorer la focalisation sur les parties pertinentes de l'entrée.
- **Dépendance au Contexte** : Les modèles peuvent avoir du mal à comprendre le contexte complet ou les nuances subtiles du langage.

Perspectives

Les modèles seq-to-seq continuent d'être un domaine de recherche actif et prometteur dans le NLP, offrant des opportunités d'amélioration et d'innovation :

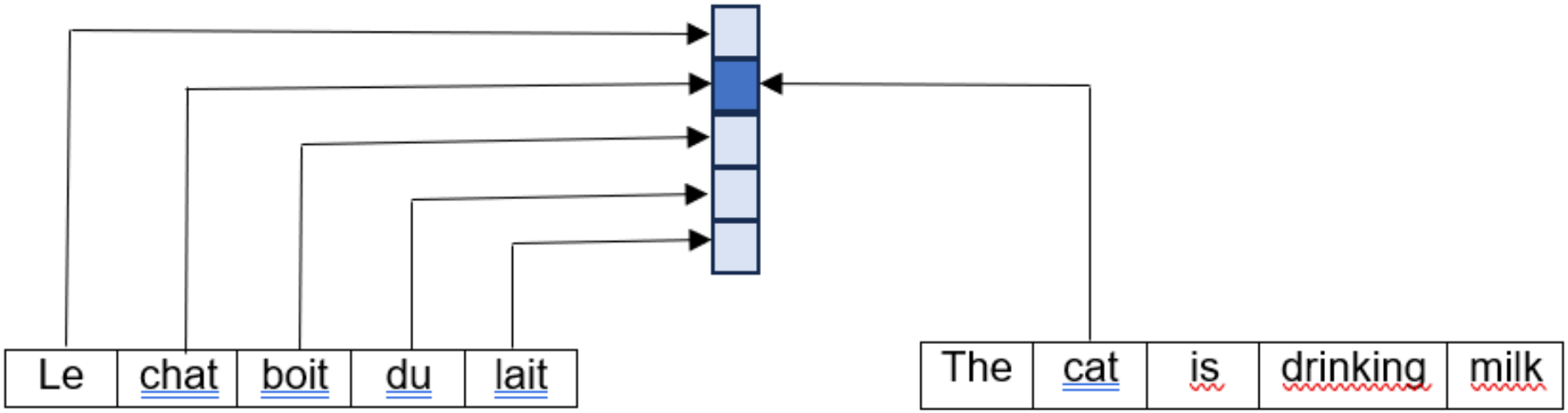
- **Optimisation avec des mécanismes d'Attention** : Améliorer l'efficacité et la précision du focus sur les éléments pertinents.
- **Gestion des longues séquences** : Développer des stratégies pour mieux gérer les informations sur de longues distances.
- **Modèles pré-entraînés** : Utiliser des modèles pré-entraînés comme base pour des tâches spécifiques, réduisant le besoin de grandes quantités de données d'entraînement.

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.

$$\alpha_{tj} = \frac{\exp(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_j))}{\sum_{k=1}^{T_x} \exp(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_k))} \tag{4}$$

$$\boldsymbol{c}_t = \sum_{j=1}^{T_x} \alpha_{tj} \bar{\boldsymbol{h}}_j \tag{5}$$

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

Types d'Attention

Il existe plusieurs variantes du mécanisme d'attention, chacune avec ses propres caractéristiques et applications :

- **Attention globale** : Le décodeur considère tous les états cachés de l'encodeur simultanément pour générer le vecteur de contexte.
- **Attention locale** : Le décodeur ne considère qu'une sous-séquence des états cachés de l'encodeur autour d'une position centrale pour chaque étape de décodage.
- **Self Attention** : Utilisée dans les transformateurs, cette forme d'attention permet à chaque position dans une séquence d'entrée de considérer toutes les positions et de s'auto-pondérer.

Chaque type d'attention est adapté à différentes tâches et configurations de modèle.

Importance de l'Attention

Le mécanisme d'attention a révolutionné la manière dont les modèles de NLP traitent et génèrent des séquences :

- **Amélioration des Performances** : Permet une meilleure gestion des dépendances à longue distance dans les données.
- **Interprétabilité** : Les poids d'attention fournissent un aperçu de la manière dont le modèle prend ses décisions, en montrant quelles parties de l'entrée influencent le plus la sortie. item **Flexibilité et Adaptabilité** : Facilite l'adaptation des modèles à différentes longueurs de séquence et types de tâches, améliorant ainsi leur applicabilité à un large éventail de domaines.

En somme, le mécanisme d'attention a non seulement amélioré la capacité des modèles à traiter des informations séquentielles complexes, mais a également ouvert la voie à des avancées significatives dans la compréhension et la génération du langage naturel par les machines.

Applications du mécanisme d'Attention

L'intégration du mécanisme d'attention dans les modèles seq-to-seq a permis des progrès significatifs dans diverses applications en NLP :

- **Traduction automatique** : Améliore la qualité de la traduction en permettant au modèle de se concentrer sur les parties pertinentes du texte source lors de la traduction.
- **Résumé automatique** : Aide à identifier les informations clés dans un texte long pour générer des résumés concis et informatifs.
- **Génération de aexte** : Permet une génération de texte plus cohérente et contextuellement riche en se concentrant sur les éléments pertinents de l'entrée tout au long du processus de génération.
- **Compréhension de texte** : Facilite la compréhension des relations complexes et des références croisées dans les textes, améliorant les performances des modèles sur les tâches de compréhension.

Conclusion sur le mécanisme d'Attention

Le mécanisme d'attention représente une avancée majeure dans le développement des modèles d'apprentissage profond pour le NLP, offrant une amélioration significative en termes de performance, d'interprétabilité, et de flexibilité des modèles seq-to-seq :

- Il a permis de surmonter certaines des limites inhérentes aux architectures seq-to-seq traditionnelles, notamment en ce qui concerne la gestion des dépendances à long terme.
- Les développements futurs et les innovations autour du mécanisme d'attention continueront sans doute à pousser les frontières de ce que les modèles d'apprentissage automatique peuvent accomplir dans le traitement et la génération de langage naturel.

Transformers

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* † University of Toronto aidan@cs.toronto.edu

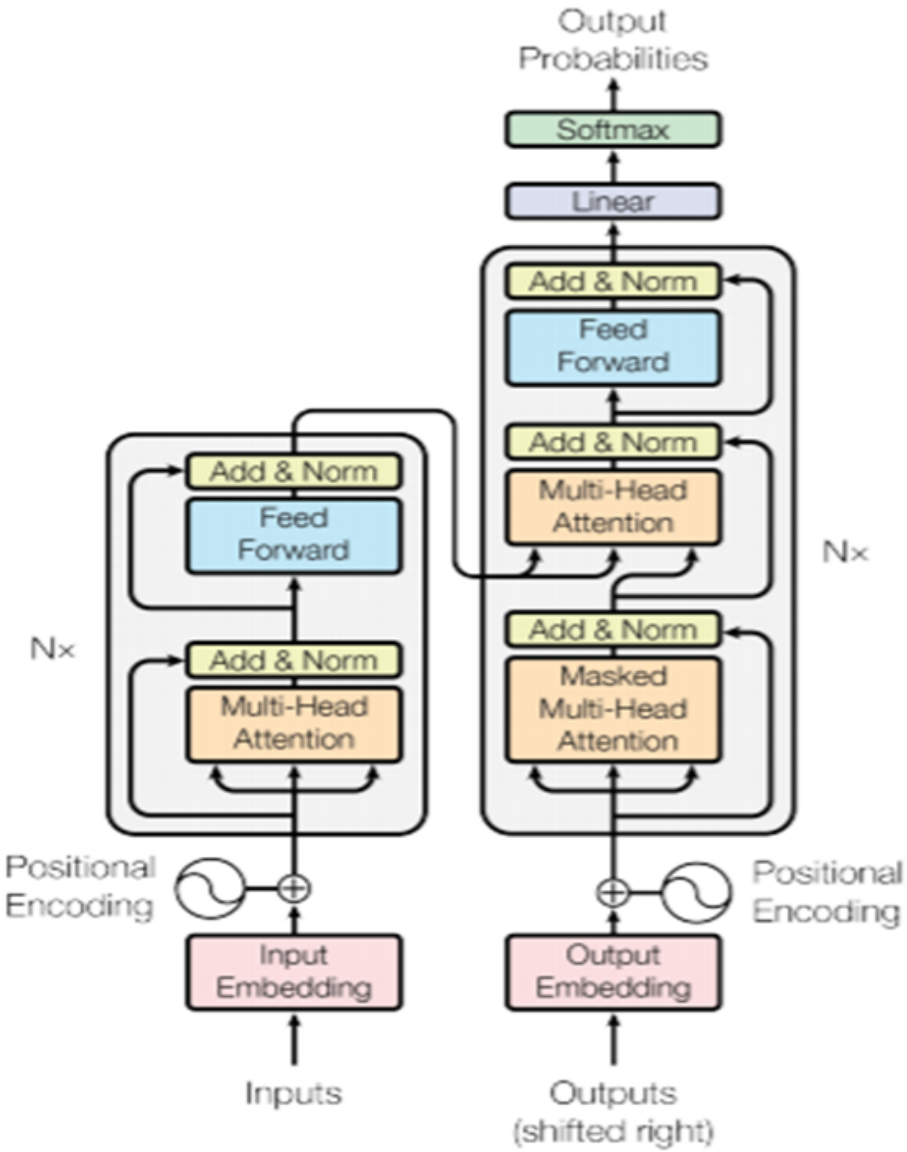
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

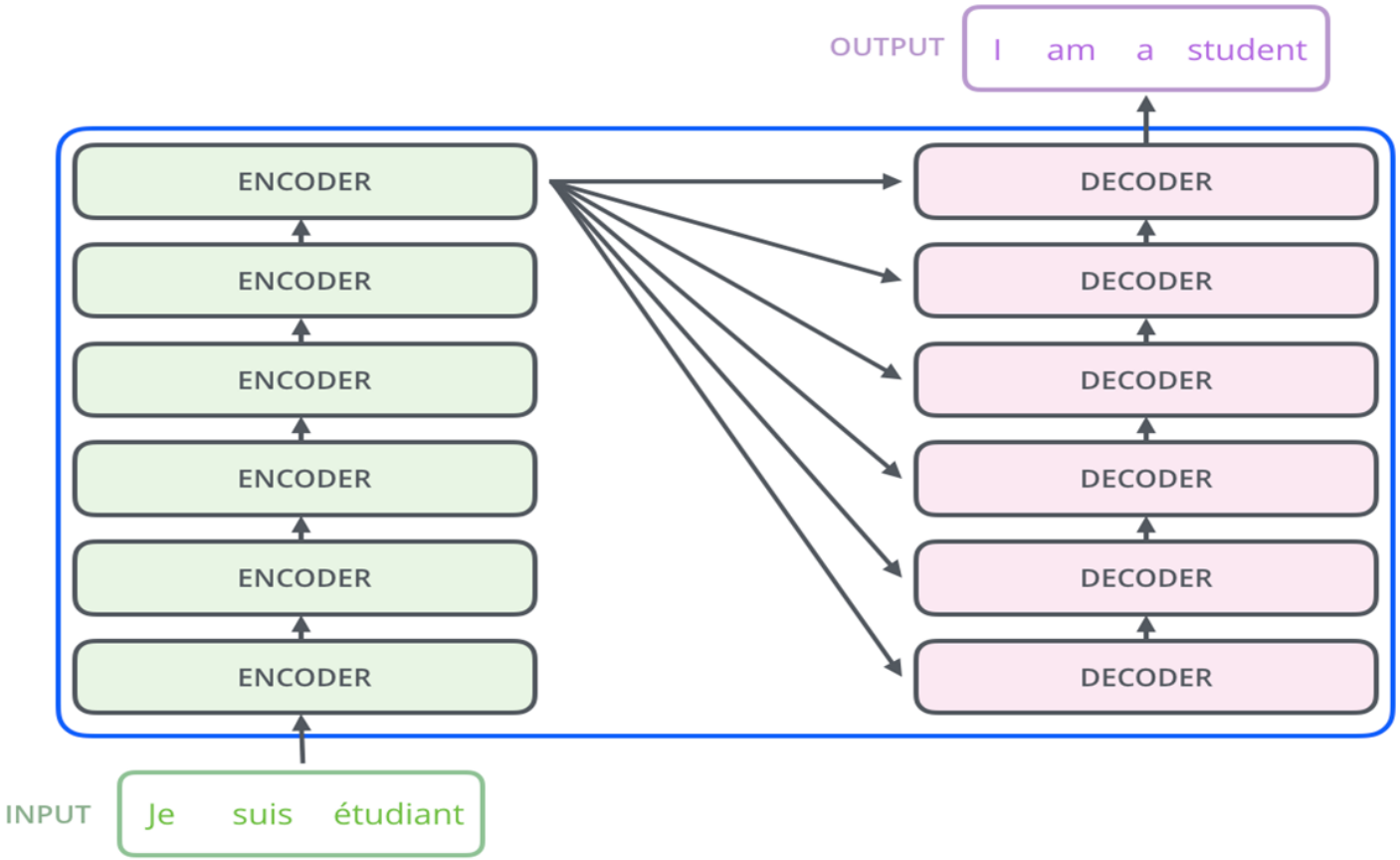
Abstract

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

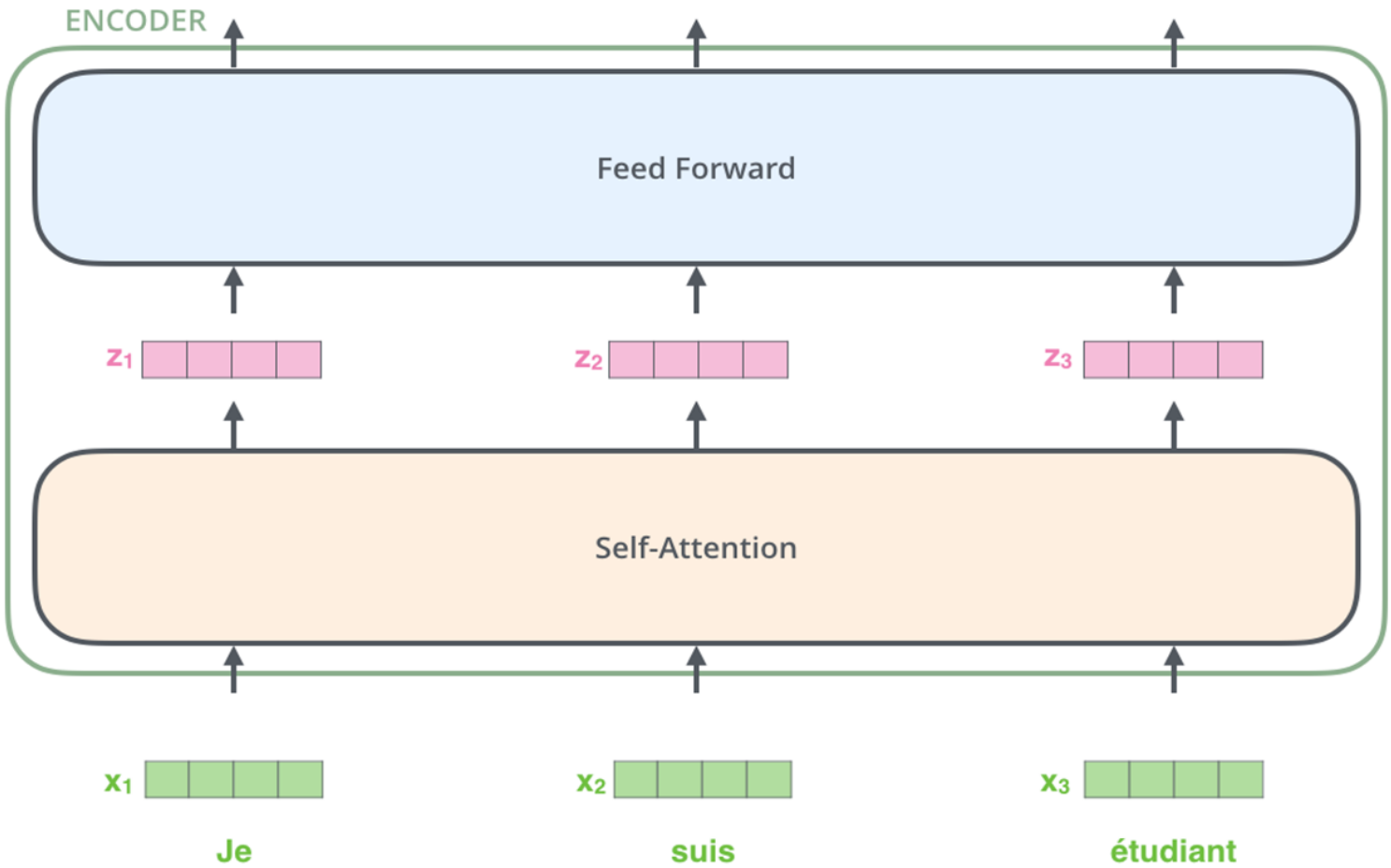
Transformer originel



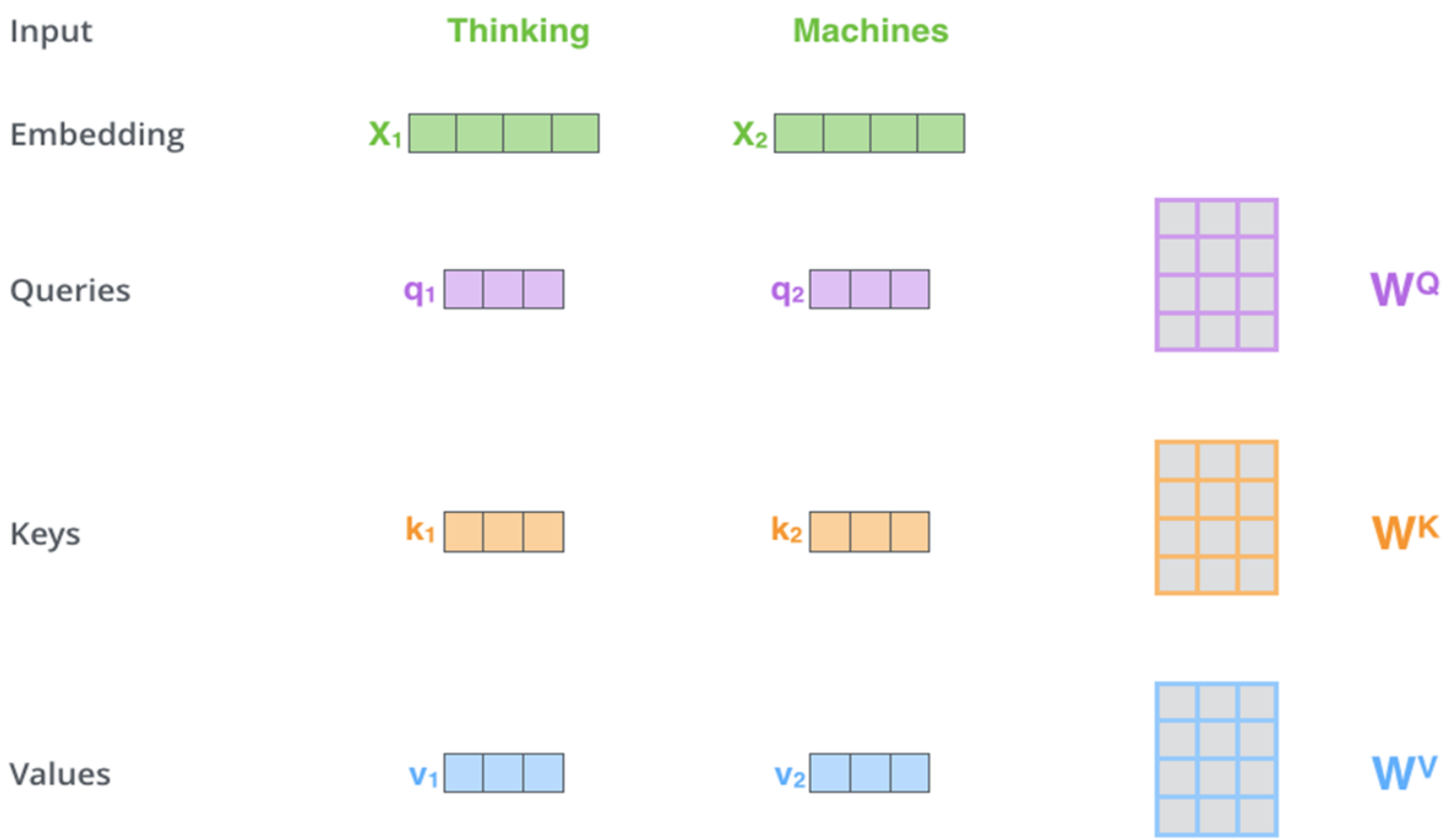
Mécanisme de cross-attention



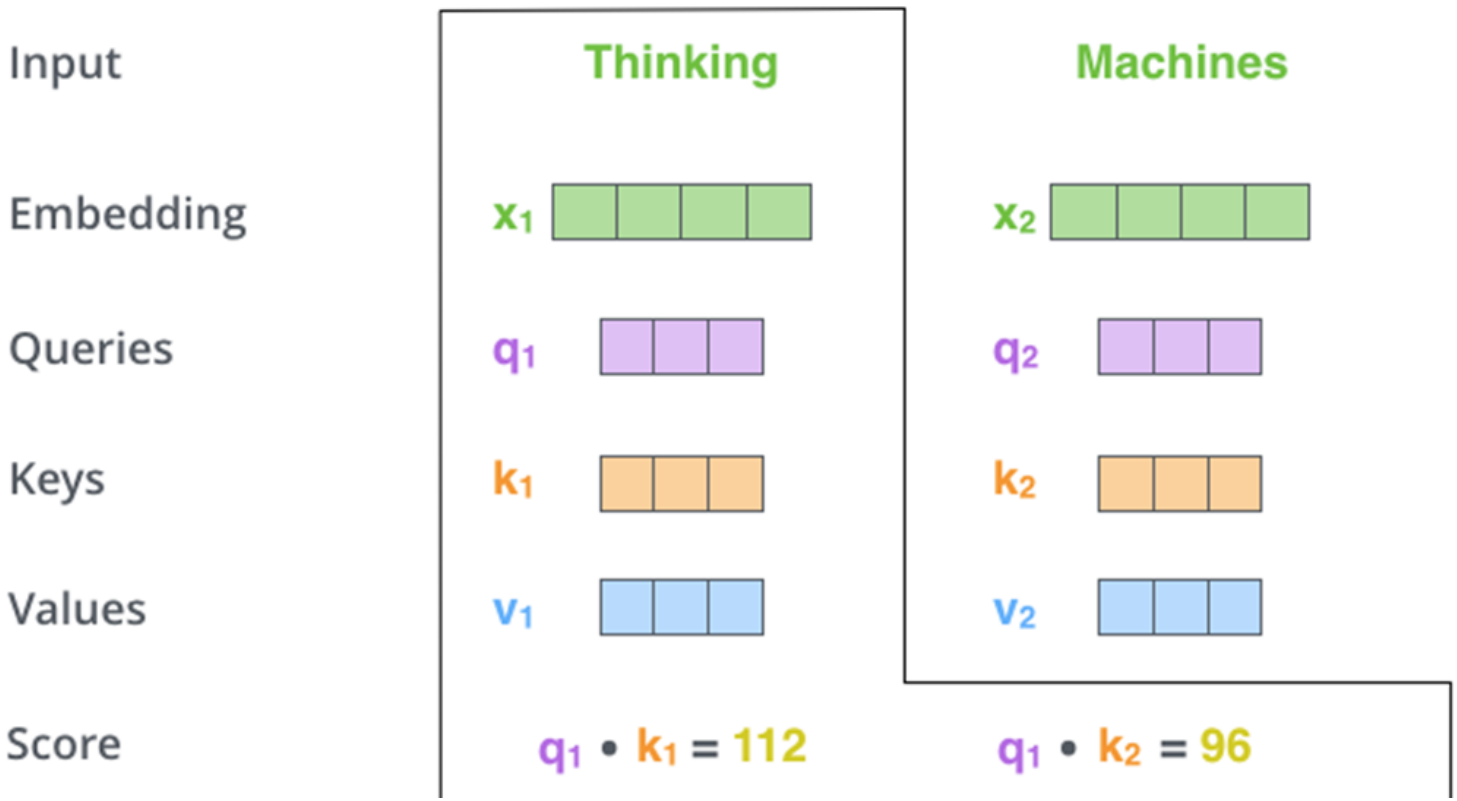
Mécanisme de self-attention



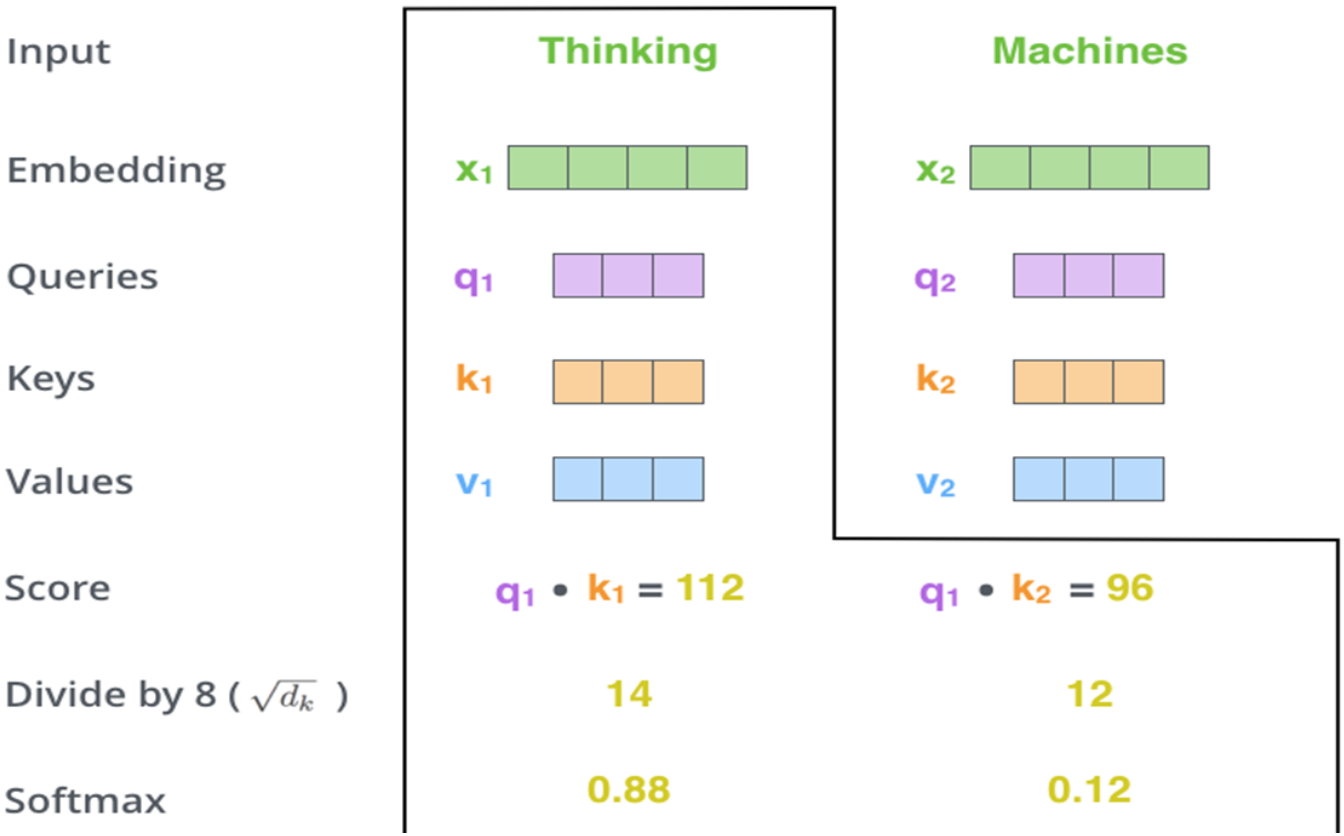
Fonctionnement du mécanisme de self-attention



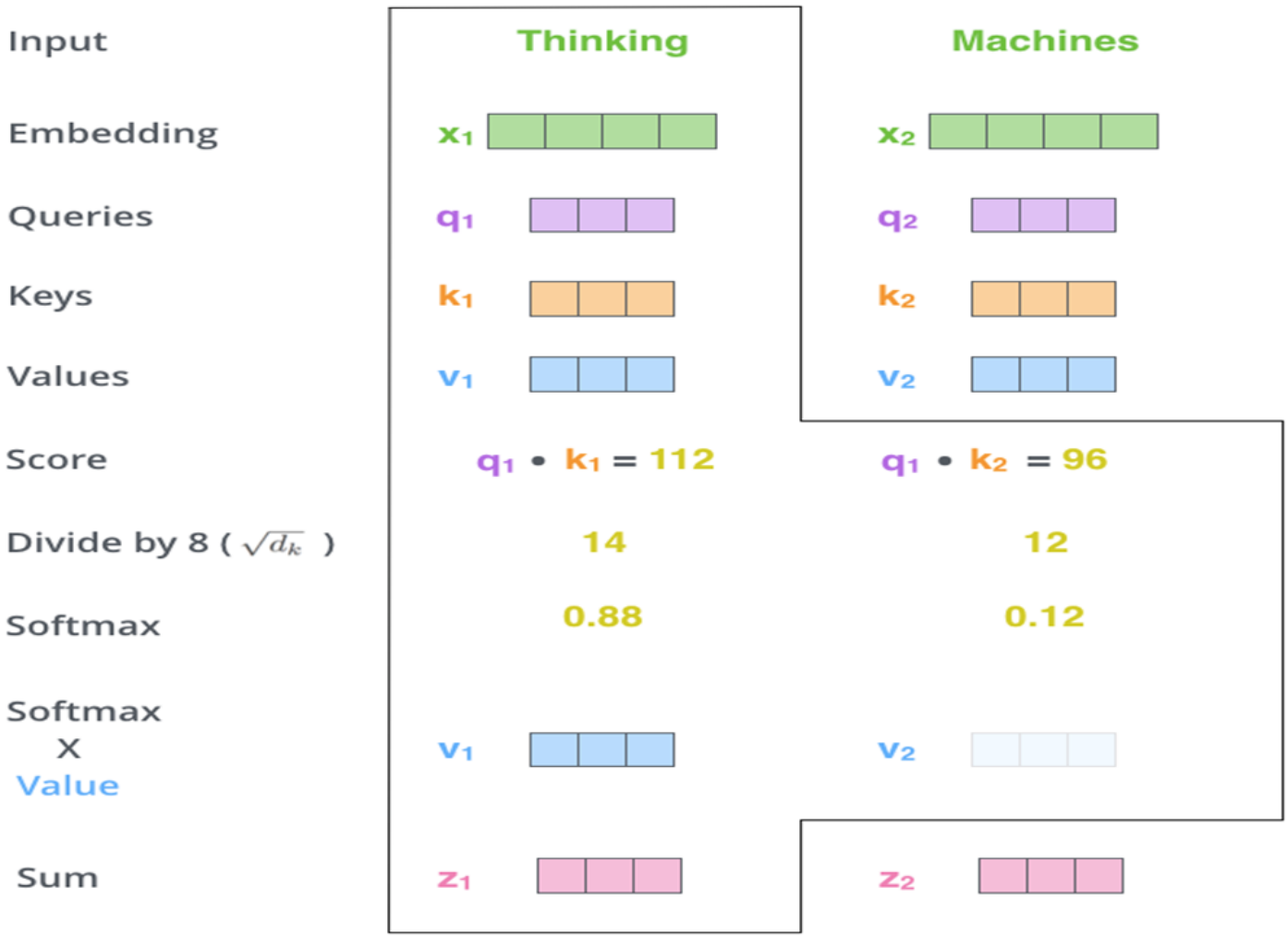
Calcul des scores de self-attention



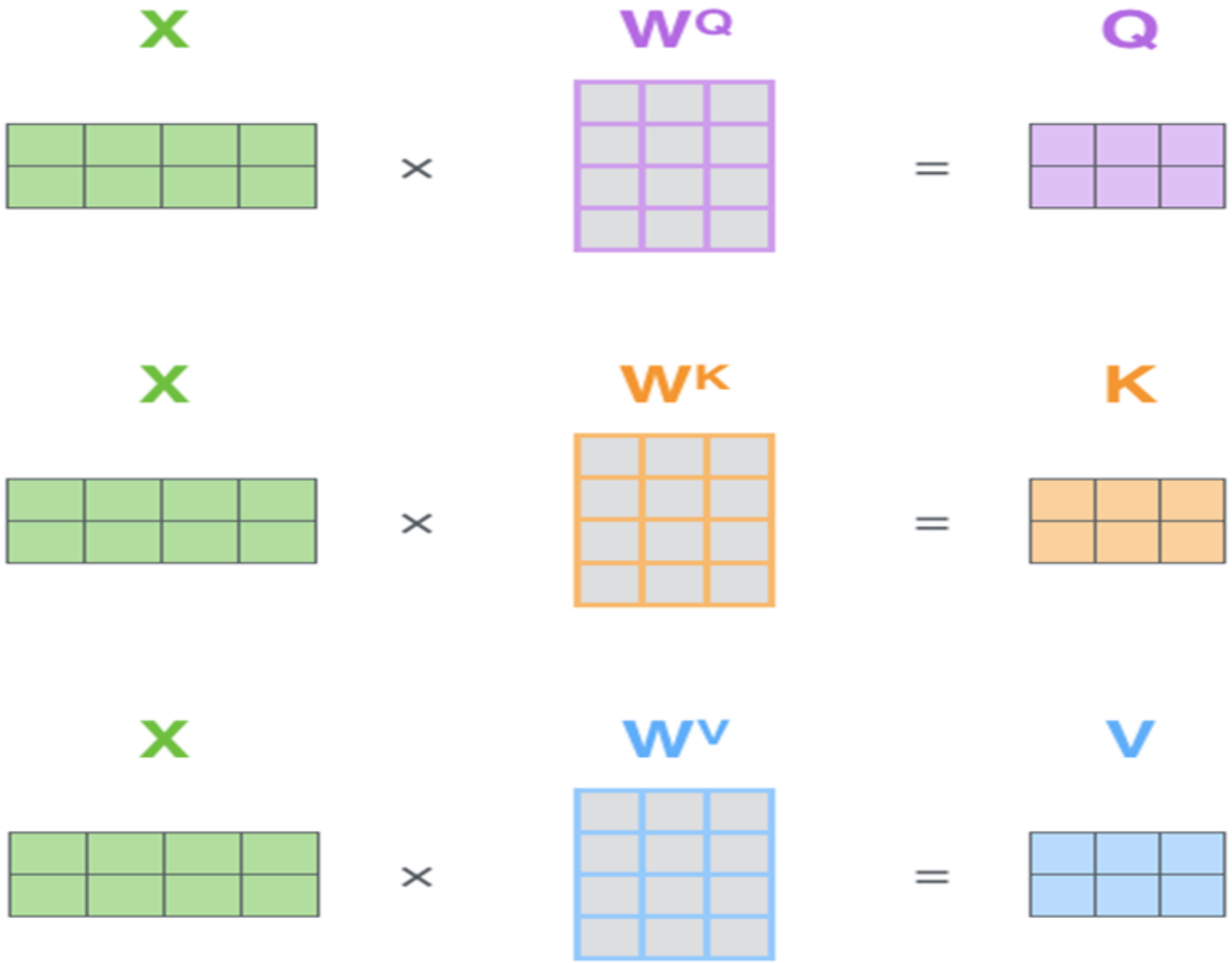
Calcul des scores de self-attention



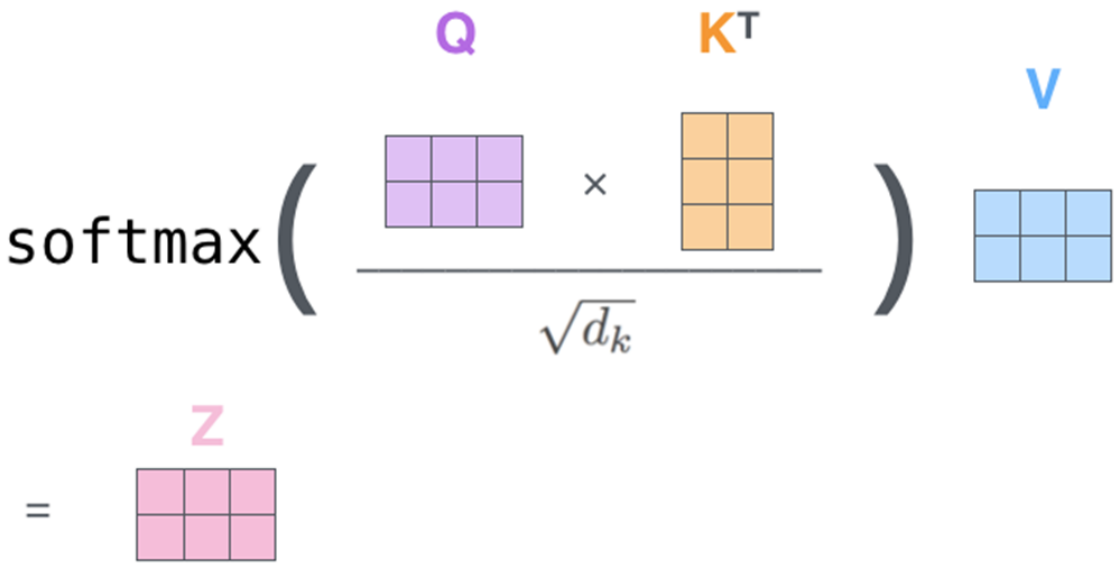
Calcul de la nouvelle représentation



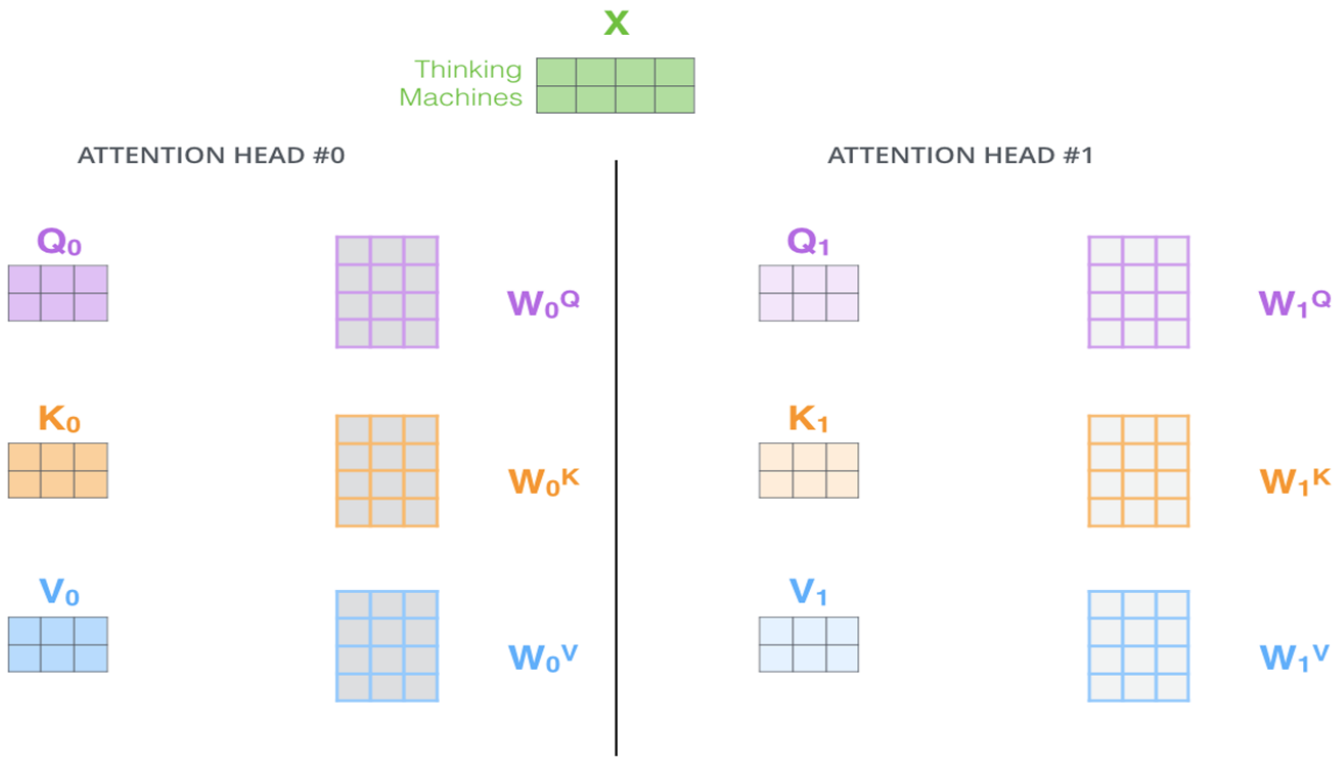
Formulation matricielle



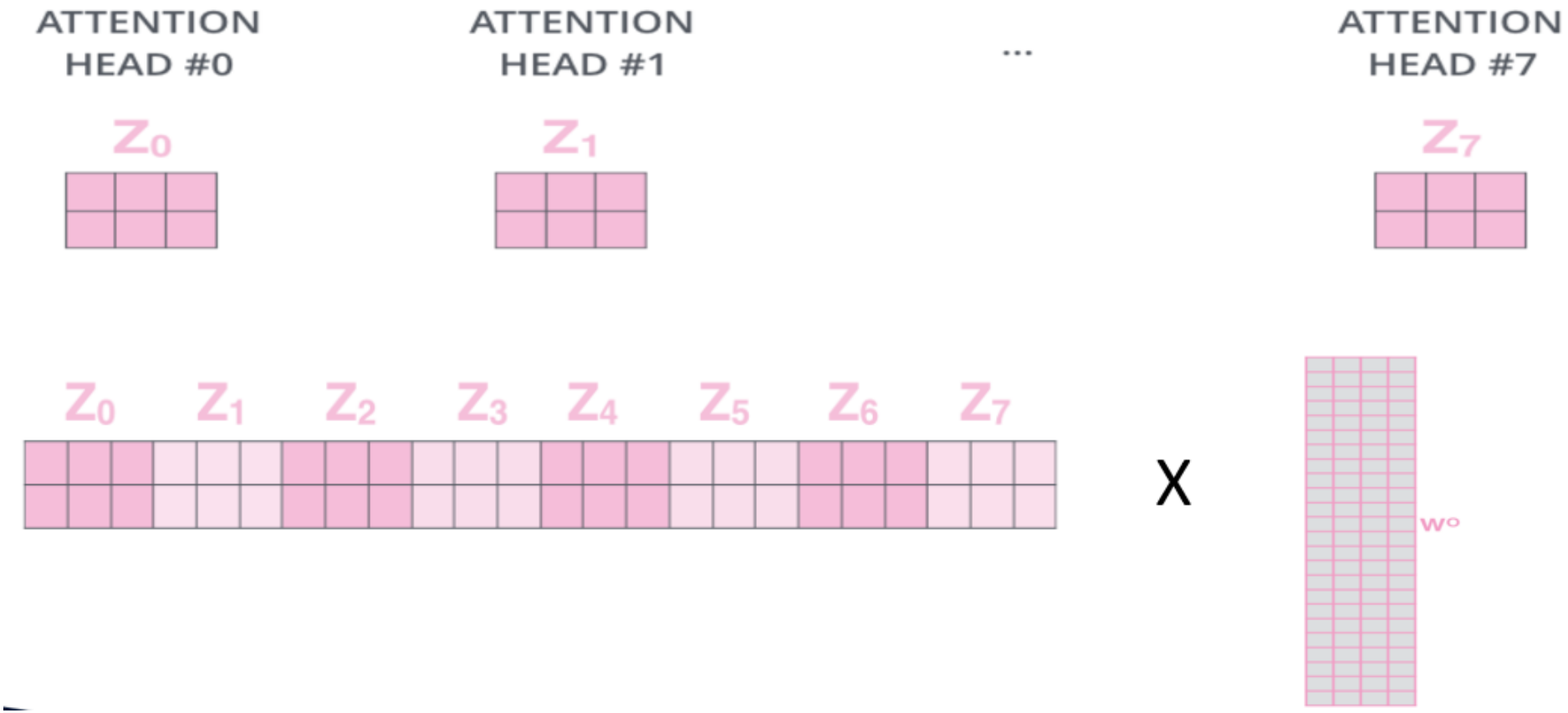
Formulation matricielle du mécanisme de self attention



Multihead attention



Multihead attention



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 heures

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	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

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 :

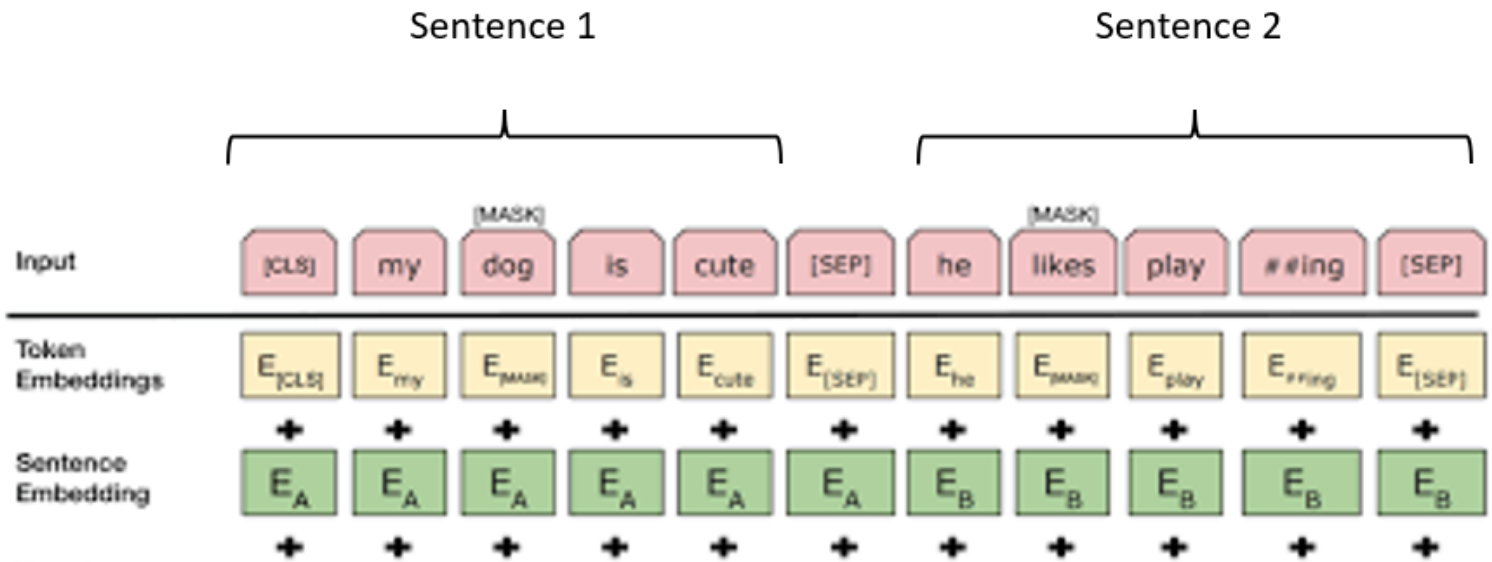
- **Masquage de token (MLM) :** 15% des tokens masqués aléatoirement, prédits par le modèle.

$$L_{\text{MLM}} = -\log P(\text{token masqué}|\text{contexte}) \tag{6}$$

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

$$L_{\text{NSP}} = -\left[y \log(\hat{y}) + (1-y)\log(1-\hat{y})\right] \tag{7}$$

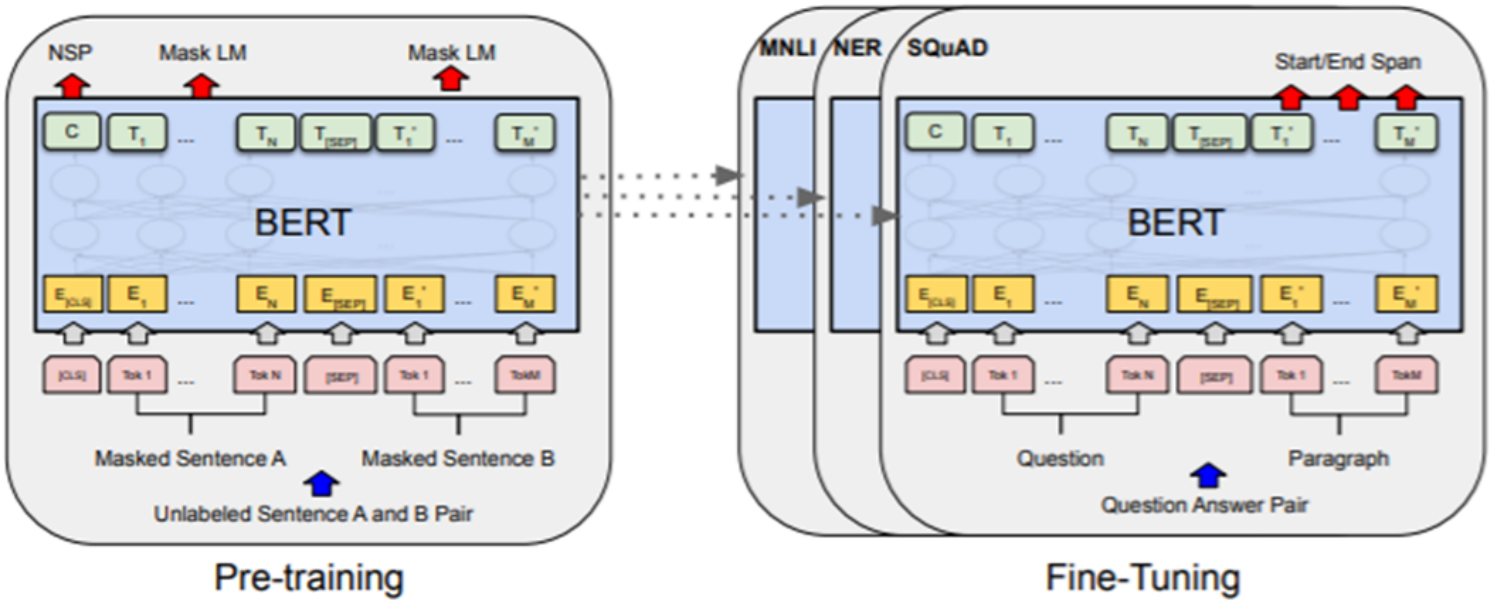
- Combinaison des pertes pour l'optimisation.



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-entrainé 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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

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 :**

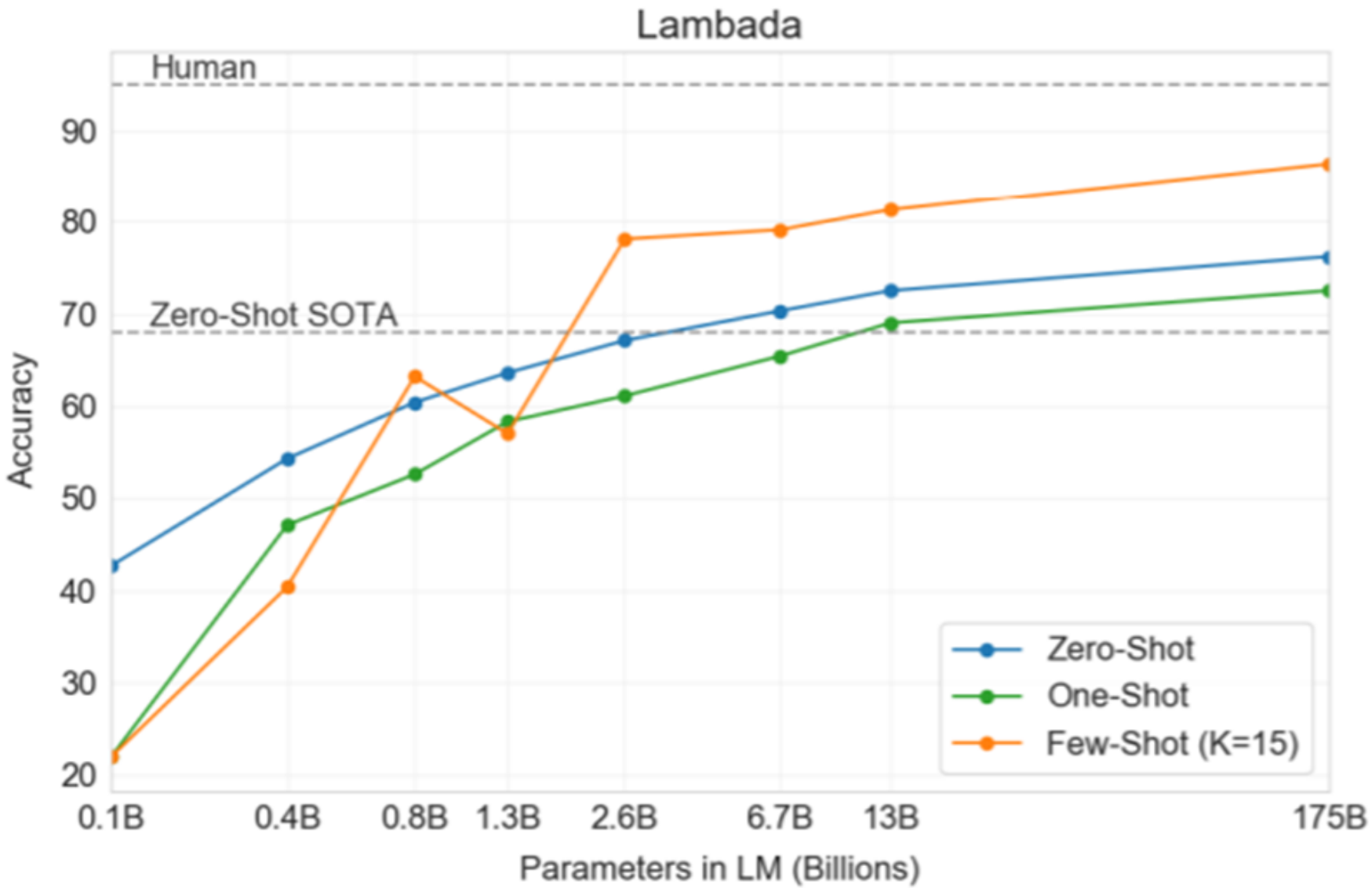
$$P(w_t|w_1, ..., w_{t-1}) = \text{softmax}(\text{Transformer}(w_1, ..., w_{t-1})). \tag{8}$$

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 langue.

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 :

1. **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.
2. **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.
3. **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 calculées par rapport aux pramètres 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 θ, 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 θ.

- **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 ϵ est un paramètre de clipping.

Direct Preference Optimization (DPO

L'optimisation directe des préférences (DPO) est une technique avancée d'apprentissage automatique conçue pour ajuster les modèles de langage aux préférences humaines de manière efficace et directe, sans les complexités de l'apprentissage par renforcement (RL).

- **Principe de fonctionnement:**
  - DPO utilise les préférences exprimées directement par les utilisateurs pour guider l'ajustement du modèle, en comparant des paires de sorties générées par le modèle et en sélectionnant celles qui correspondent le mieux aux préférences.
  - Au lieu de modeler une fonction de récompense externe et d'utiliser RL pour maximiser cette récompense, DPO optimise directement le modèle en fonction des feedbacks préférentiels, en employant une méthode d'optimisation basée sur les gradients qui ajuste les poids du modèle pour favoriser les réponses préférées.
- **Avantages de DPO :**
  - **Simplicité et efficacité :** DPO simplifie le processus d'alignement des modèles sur les préférences humaines, en évitant les défis de l'apprentissage par renforcement tels que la variance élevée et les difficultés de convergence.
  - **Stabilité:** Offre une méthode plus stable et contrôlée pour intégrer les préférences humaines dans l'entraînement des modèles, sans nécessiter un

Objectif de DPO

L'Optimisation Directe des Préférences (DPO) vise à aligner les modèles de langage avec les préférences humaines en optimisant directement une politique π par rapport à ces préférences.

Objectif DPO

$$\pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{r(x,y)}{\beta}\right) \tag{9}$$

où :

- $\pi_{\text{ref}}(y|x)$ est la politique de référence.
- $r(x,y)$ est la fonction de récompense reflétant les préférences humaines.
- $Z(x)$ est la fonction de partition, assurant la normalisation.
- β est un paramètre de température contrôlant la contrainte de divergence KL.

Modèle de Bradley-Terry

Le modèle de Bradley-Terry joue un rôle clé dans DPO en simplifiant la relation entre les préférences humaines et la politique du modèle.

Modèle de Bradley-Terry

La probabilité de préférer y_1 sur y_2 donné x, selon Bradley-Terry, est définie comme :

$$p(y_1 \succ y_2|x) = \frac{\exp(r(x, y_1))}{\exp(r(x, y_1)) + \exp(r(x, y_2))} \tag{10}$$

où $r(x, y)$ représente la récompense associée à choisir y étant donné x.

Élimination de la Fonction de Partition

En réorganisant l'expression de DPO en utilisant le modèle de Bradley-Terry, nous obtenons :

$$r(x, y) = \beta \log \left(\frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} \right) + \beta \log Z(x) \tag{11}$$

La substitution dans le modèle de préférence élimine la nécessité de calculer $Z(x)$, car elle se simplifie dans le calcul des différences de récompenses.

Expression de la politique en fonction des préférences humaines

En éliminant la fonction de partition, DPO permet d'exprimer la politique optimale π^* directement à partir des préférences humaines.

Relation entre préférences et politique

Grâce à la reparamétrisation offerte par le modèle de Bradley-Terry, on obtient une relation directe :

$$p^*(y_1 \succ y_2|x) = \sigma\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} - \beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right) \tag{12}$$

où σ est la fonction sigmoïde, reflétant la probabilité de préférer y_1 à y_2 donné x.

Déduction de la politique optimale

Cette relation permet de déduire $\pi^*(y|x)$, la politique optimale, directement en fonction des différences de préférences humaines exprimées à travers le modèle de Bradley-Terry, sans avoir besoin de modéliser explicitement ou d'estimer une fonction de récompense complexe.