



Traduction automatique

Transformers

Enzo Doyen

enzo.doyen@unistra.fr

2025-07-18

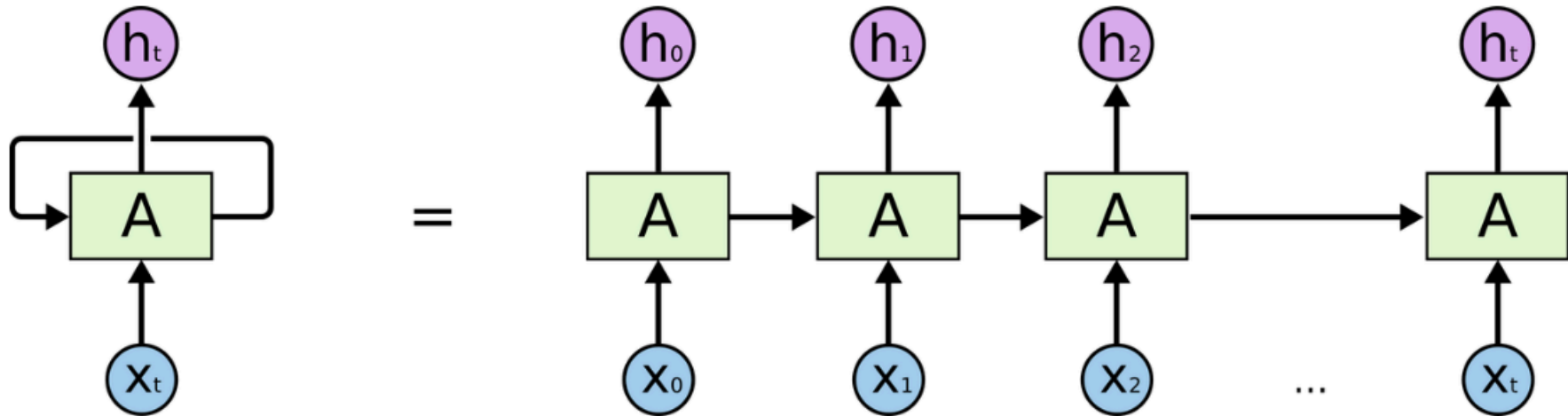
Plan

- I. Détails sur l'architecture Transformer
- II. Transformers pour la traduction automatique
- III. Mise en pratique



I. Détails sur l'architecture Transformer

Réseaux de neurones récurrents



Source : <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

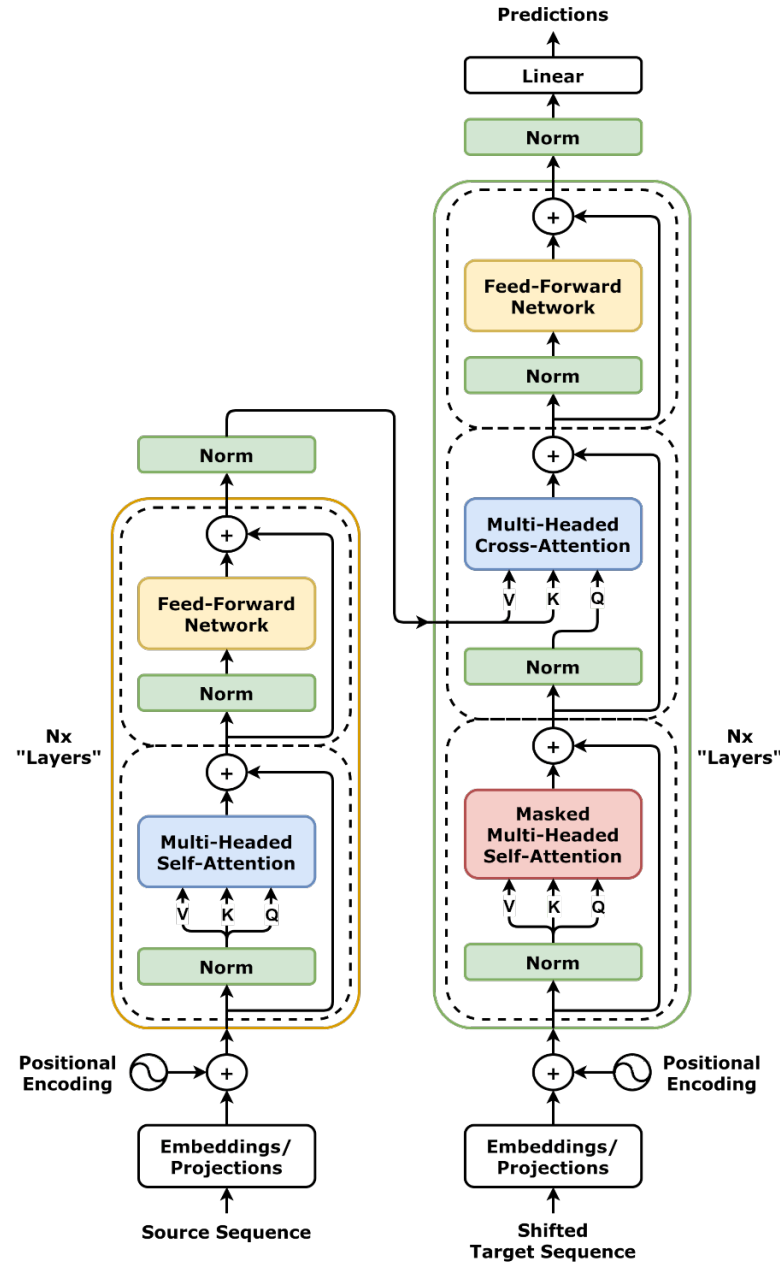
Limites des RNN

- ♦ **Traitement séquentiel** : les RNN traitent les entrées une par une, ce qui rend l'entraînement long.
 - Ce faisant, parallélisation impossible.
- ♦ **Problème du *vanishing gradient*** : sur les séquences longues, le modèle ne parvient plus à capturer les informations pertinentes vues en début de séquence.

Architecture Transformer

Transformer : type d'architecture introduit par Vaswani et al. (2017).

Abandonne complètement les RNN/CNN et utilise uniquement des **mécanismes d'attention** pour encoder les séquences.



Architecture Transformer

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

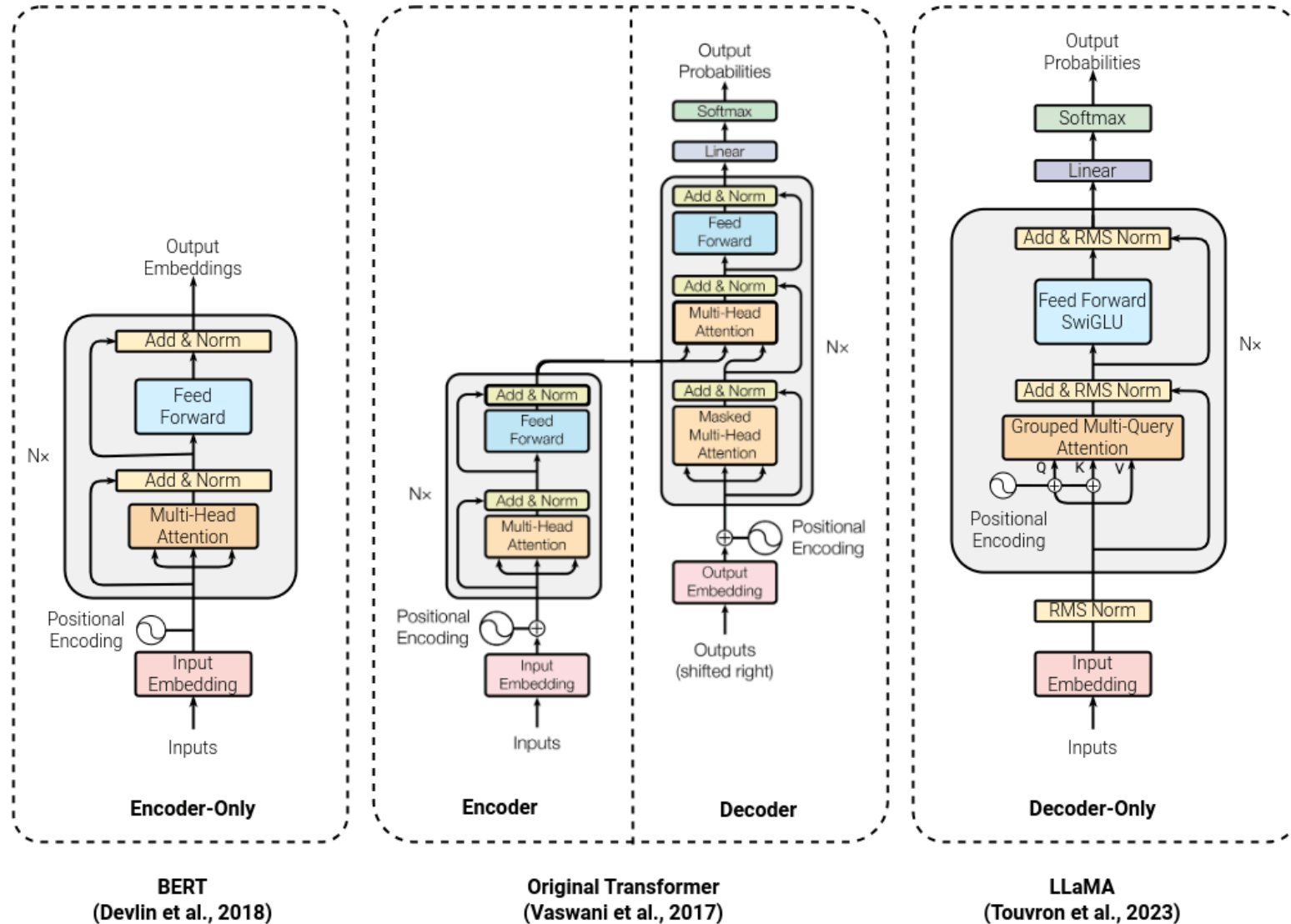
Łukasz Kaiser*
Google Brain
lukaszkaizer@google.com

Illia Polosukhin*
illia.polosukhin@gmail.com

Architecture Transformer : types de modèles

- ♦ **Modèles encodeurs** : encodent la séquence d'entrée et produisent une représentation de celle-ci.
 - Utilisés pour des tâches de classification, d'analyse de sentiments, etc. (BERT, RoBERTa, DistilBERT...)
- ♦ **Modèles décodeurs (auto-régressifs)** : génèrent une séquence de sortie à partir d'une représentation d'entrée.
 - Utilisés pour des tâches de génération de texte (GPT, Llama...)
- ♦ **Modèles encodeurs-décodeurs (séquence-à-séquence)** : combinent les deux précédents pour des tâches de traduction, de résumé, etc. (T5, BART...)

Source : Luke Ditria



Architecture Transformer : performances en traduction (encodeur-décodeur)

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	$2.3 \cdot 10^{19}$	

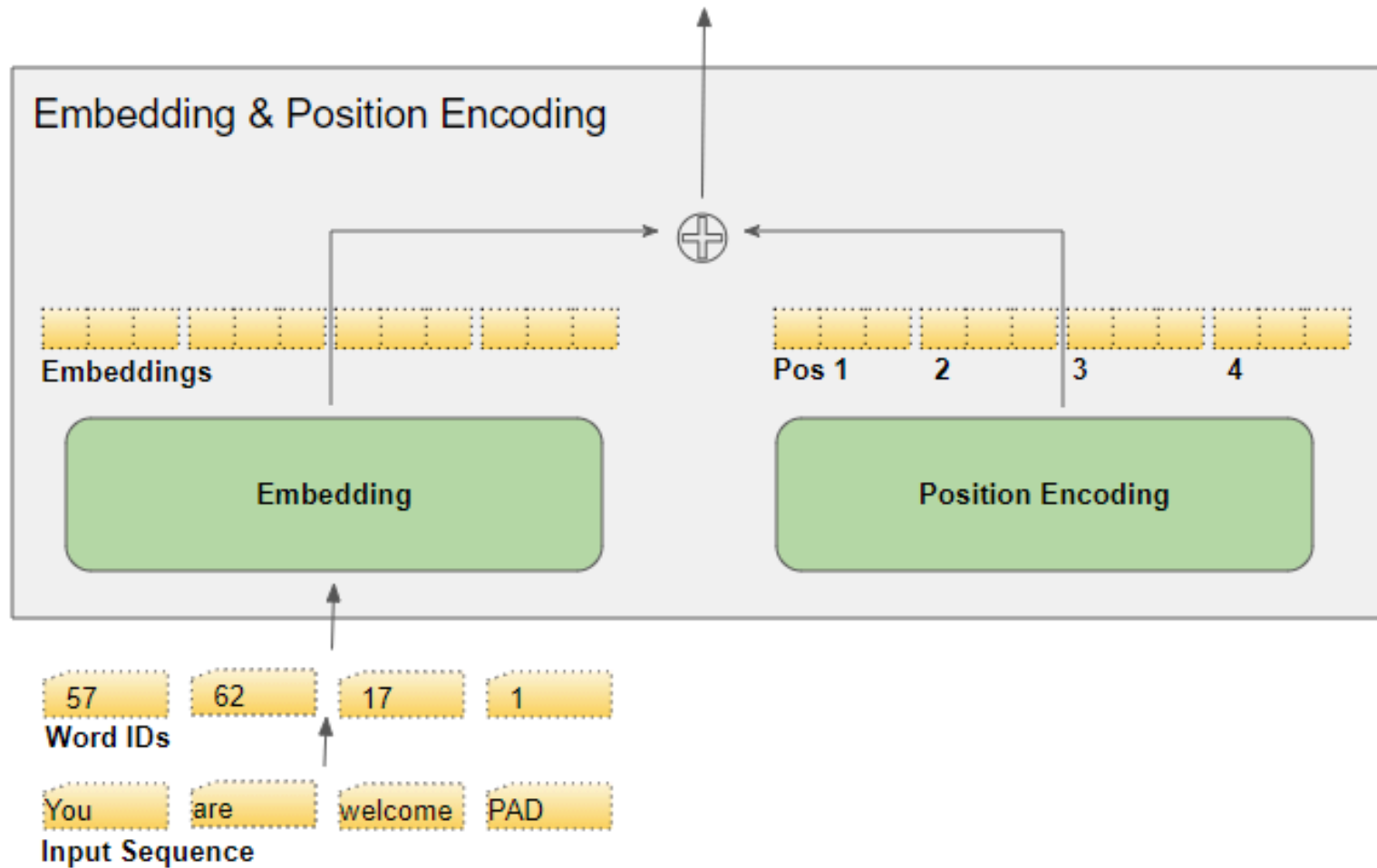
Source : **Vaswani et al. (2017)**

Architecture Transformer : couches d'encodage et de décodage

- ♦ L'architecture Transformer se base sur le cumul de N couches (généralement, $N = 6$).
- ♦ Ce cumul de couches permet au modèle de revoir plusieurs fois la séquence d'entrée et de la transformer en une représentation plus riche.
- ♦ Cela implique aussi une puissance de calcul plus importante.

Architecture Transformer : positionnement des tokens

- ♦ Vu que l'architecture Transformer ne traite pas les entrées de manière séquentielle (contrairement aux RNN), il est nécessaire d'ajouter des informations de position aux tokens pour savoir où ils se situent dans la séquence.
- ♦ Utilisation de **couches d'encodage de position** (*position encoding layers*) pour ajouter ces informations.

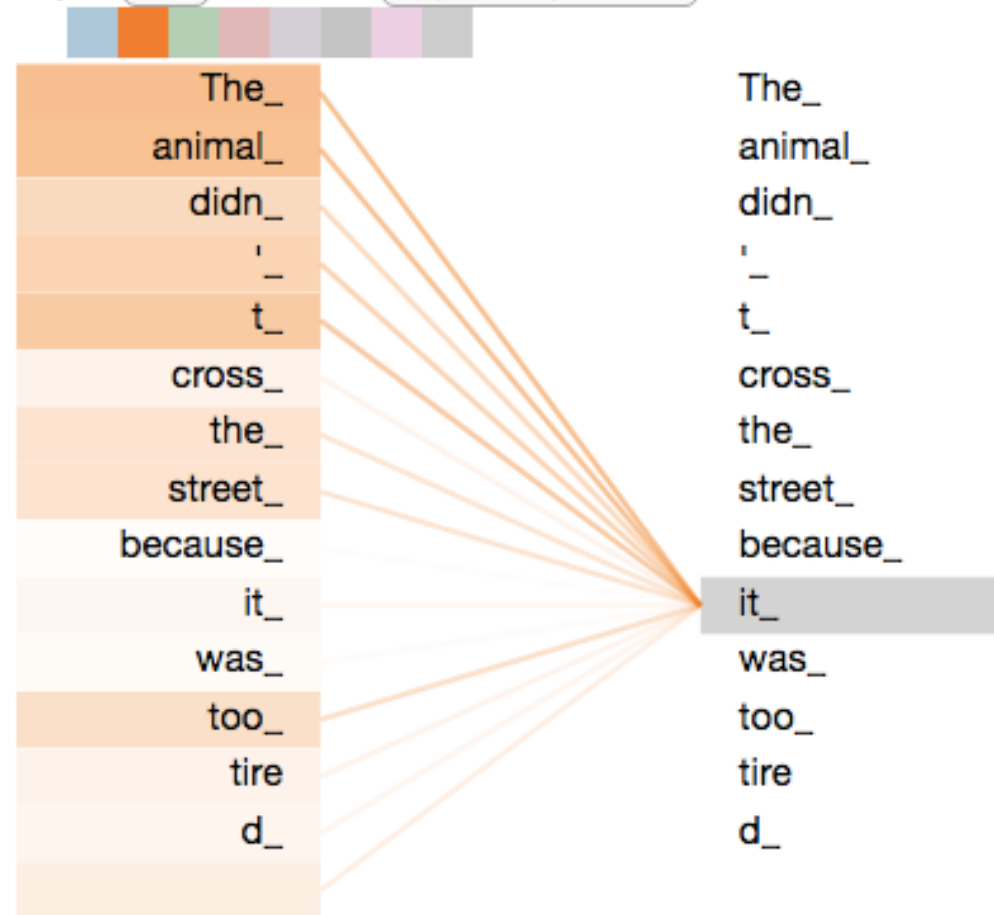


Source : <https://towardsdatascience.com/transformers-explained-visually-part-2-how-it-works-step-by-step-b49fa4a64f34/>

Architecture Transformer : mécanisme d'autoattention (*self-attention*)

- ♦ Mécanisme qui permet au modèle de donner des poids à chacune des différentes parties de la séquence d'entrée.
- ♦ Permet de capturer les relations entre les mots, même s'ils sont éloignés dans la séquence.

Layer: 5 ▾ Attention: Input - Input ▾



Architecture Transformer : mécanisme d'autoattention (*self-attention*)

- ♦ Repose sur trois matrices pour chaque token : *queries* (Q), *keys* (K) et *values* (V).
- ♦ Ces matrices sont utilisées pour calculer combien d'attention un token doit porter à un autre dans la séquence, et quelles valeurs il doit en extraire.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

T : matrice transposée

d_k : dimension des matrices Q et K

Architecture Transformer : mécanisme d'autoattention (*self-attention*)

Reprenons notre exemple précédent :

« The animal didn't cross the street because it was too tired. »

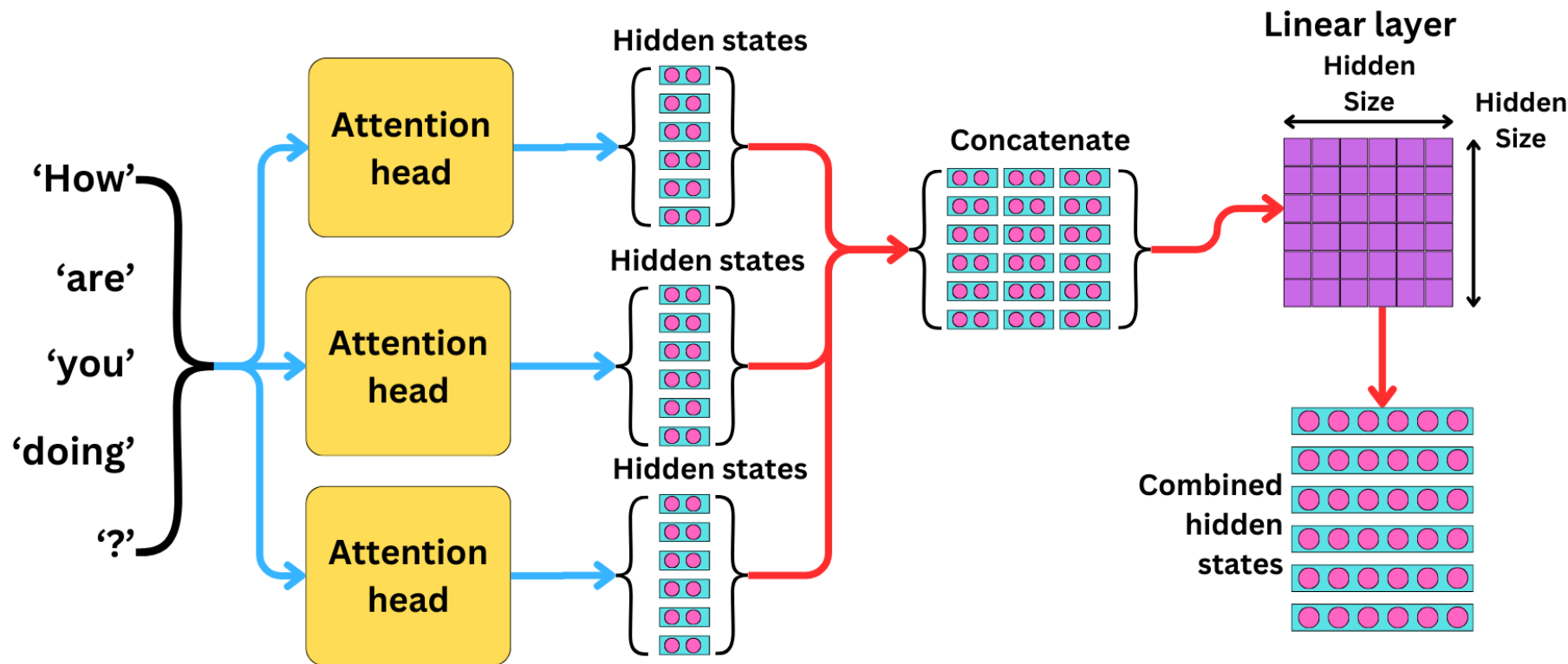
Si nous voulons savoir à quoi le mot « it » fait référence :

- ♦ le vecteur Q du mot « it » est comparé aux vecteurs K de tous les autres mots de la phrase.
- ♦ le token avec le vecteur K le plus similaire au vecteur Q de « it » est « animal ».
- ♦ la valeur V associée à « animal » est extraite et utilisée pour enrichir la représentation de « it ».

Architecture Transformer : *multi-head attention*

- ♦ L'architecture Transformer n'utilise pas une seule couche d'autoattention, mais plusieurs couches en parallèle (*multi-head attention*).
- ♦ Le mécanisme d'autoattention est appliqué plusieurs fois en parallèle, avec des poids différents pour chaque « tête » (*head*), ce qui permet au modèle de capturer différentes relations entre les tokens.

Architecture Transformer : *multi-head attention*




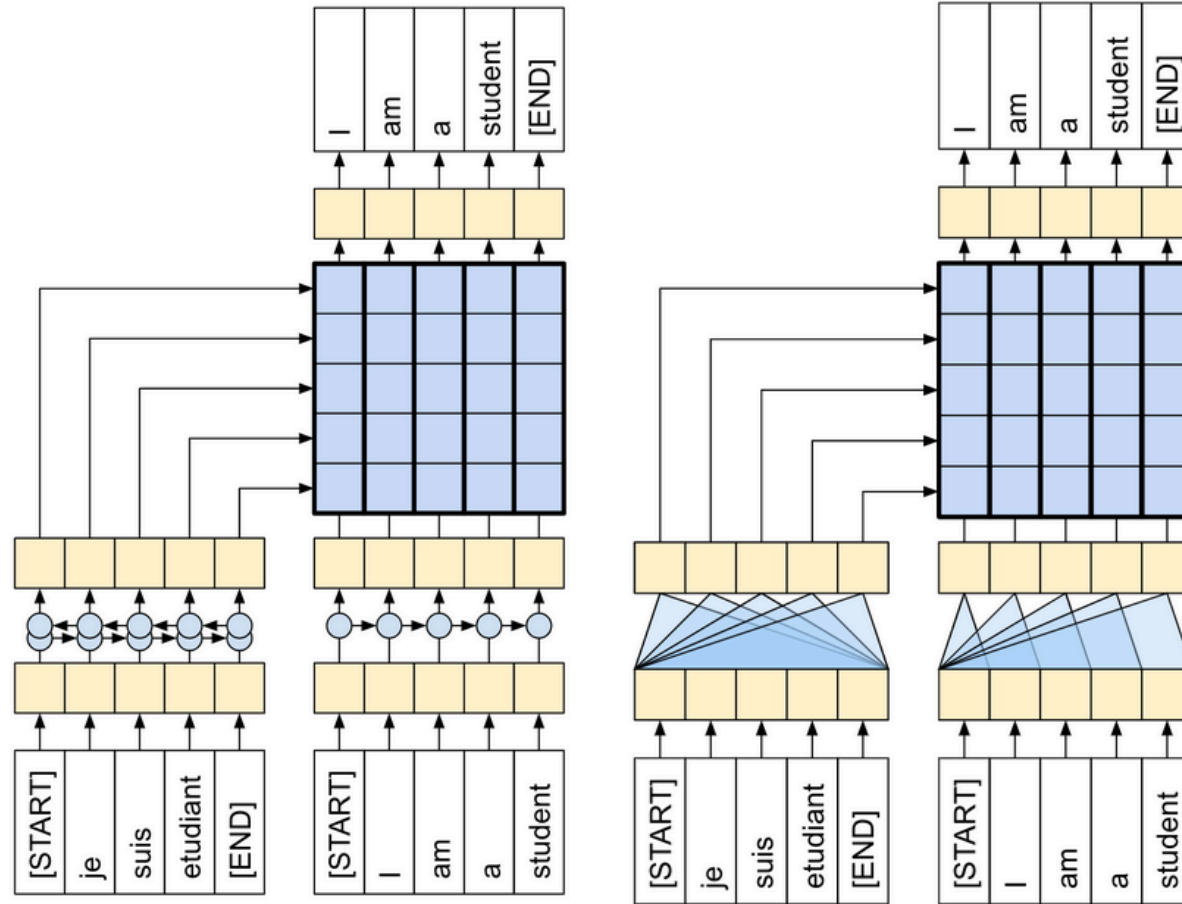
Source : <https://newsletter.theaiedge.io/p/the-multi-head-attention-mechanism>



II. Transformers pour la traduction automatique

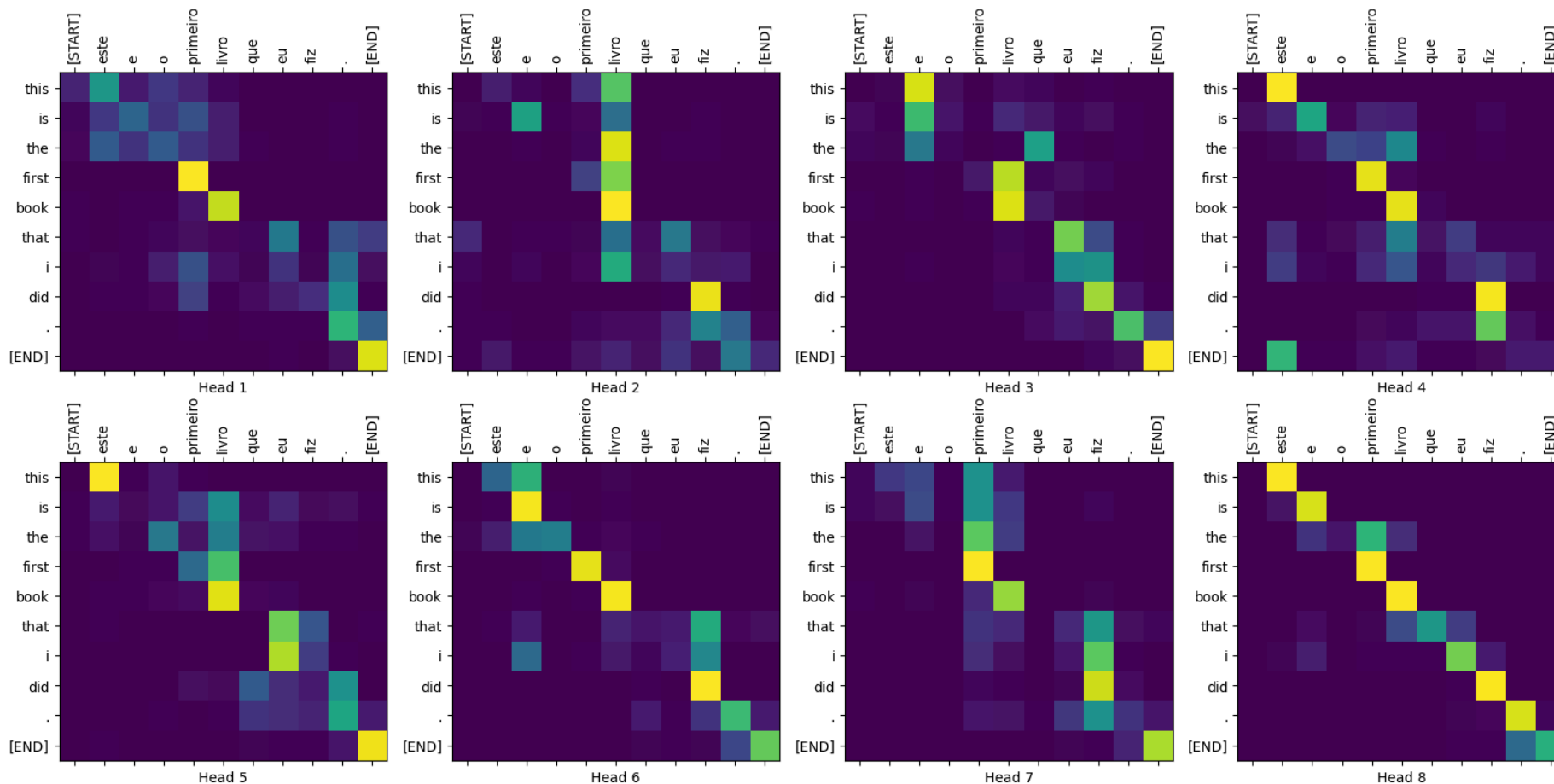
Architecture Transformer : traduction automatique

- Pour une tâche de traduction automatique, l'architecture Transformer est utilisée comme un modèle encodeur-décodeur.
- Le modèle encodeur transforme la séquence d'entrée (phrase source) en une représentation riche ; le modèle décodeur utilise cette représentation pour générer la séquence de sortie (phrase cible).
- En particulier, le **mécanisme d'attention** vu précédemment permet d'utiliser le contexte de la phrase source pour enrichir sa représentation, et ainsi produire une meilleure traduction.
- **Animation d'exemple pour l'encodage/décodage pour une tâche de traduction**  (Google AI Blog).



Différence RNN/Transformer

Source : <https://www.tensorflow.org/text/tutorials/transformer>



Représentation des *attention heads* pour une tâche de traduction

Source : <https://www.tensorflow.org/text/tutorials/transformer>

Préentraînement de modèles (*pretraining*)

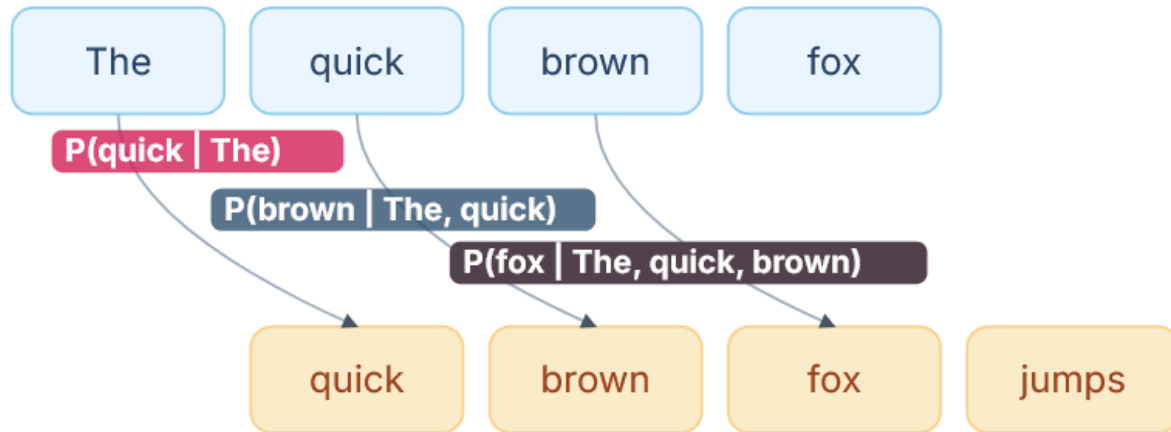
- ♦ Les modèles Transformer sont souvent **préentraînés** sur de grandes quantités de données non étiquetées avant d'être ajustés (*fine-tuned*) pour des tâches spécifiques.
- ♦ Le préentraînement permet au modèle d'apprendre des représentations générales du langage, ce qui améliore ses performances sur des tâches spécifiques, y compris la traduction automatique.
- ♦ Idée initialement développée par **Dai et Le (2015)** sous le nom de *semi-supervised sequence learning*, mais véritablement mise en place par l'arrivée de l'architecture Transformer et le développement de BERT (*Bidirectional Encoder Representations from Transformers*) par **Devlin et al. (2018)**, et de GPT-2 (*Generative Pre-trained Transformer 2*) par **Radford et al. (2019)**.

Préentraînement de modèles (*pretraining*)

- ♦ Utilisation de grandes quantités de données non étiquetées pour préentraîner le modèle (livres, articles, sites Web, etc.) : tâche non supervisée.
- ♦ Différents objectifs de préentraînement pour que le modèle apprenne des représentations du langage :
 - **Causal Language Modeling** (CLM) : prédit le prochain mot dans une séquence, en utilisant les mots précédents comme contexte (GPT-2).
 - **Masked Language Modeling** (MLM) : masque certains tokens dans la séquence et demande au modèle de prédire ces tokens masqués (BERT).
 - **Denoising** : masque ou modifie certains tokens dans la séquence et demande au modèle de prédire la séquence originale (T5, BART).

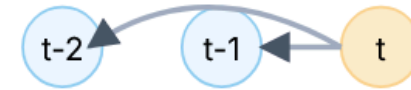
Préentraînement de modèles : CLM

Séquence d'entrée



Mécanisme d'attention

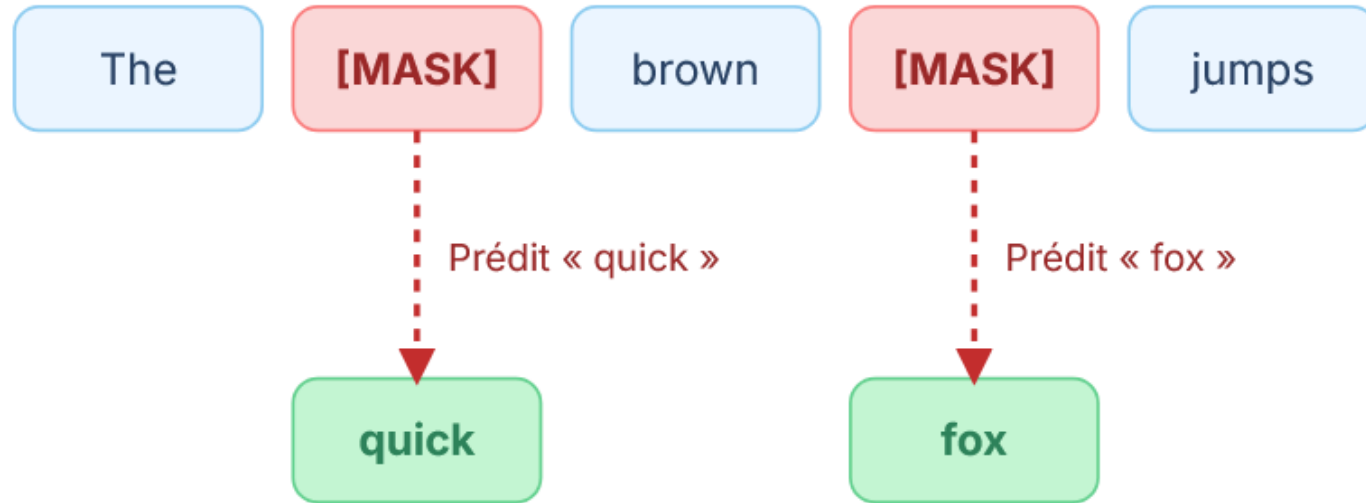
(Peut uniquement voir les tokens précédents)



$$= \prod_{t=1}^n P(w_t | w_{<t})$$

Préentraînement de modèles : MLM

Séquence corrompue



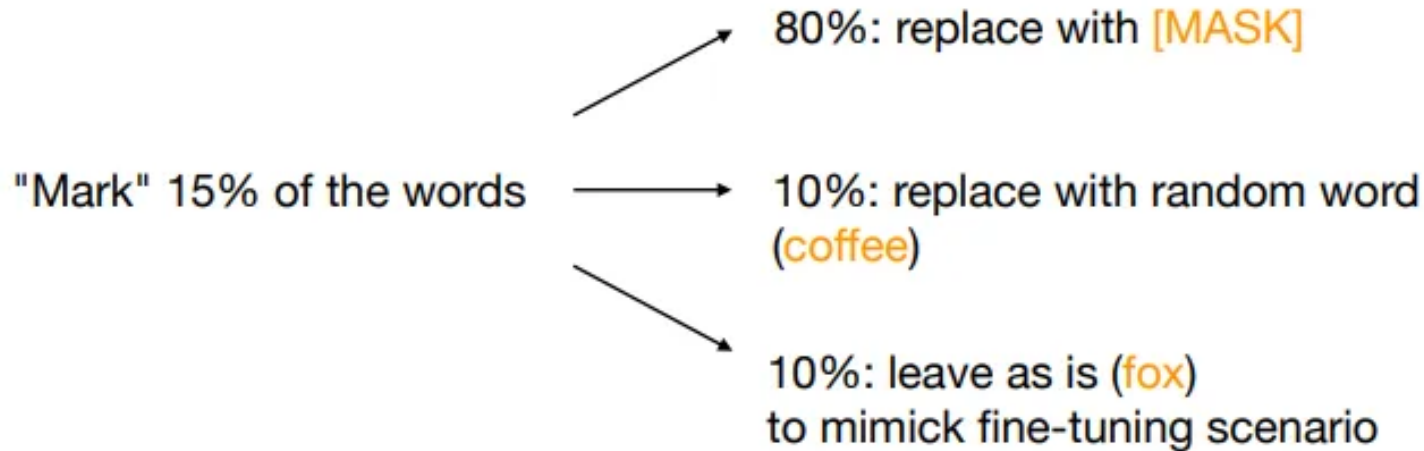
Mécanisme d'attention

(Peut voir tous les autres tokens)



Préentraînement de modèles : MLM

Input sentence: A quick brown fox jumps over the lazy dog



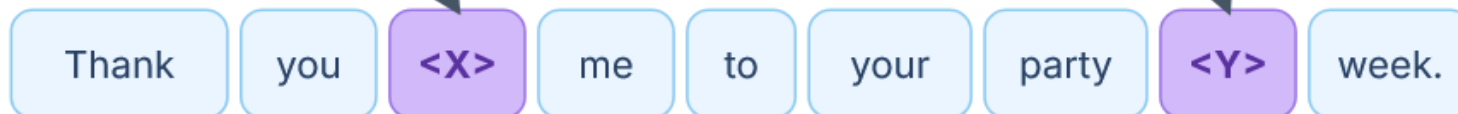
Source : <https://medium.com/@zaiinn440/a-comparative-analysis-of-lms-like-bert-bart-and-t5>

Préentraînement de modèles : denoising (T5)

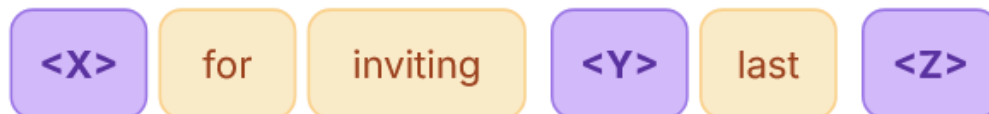
Séquence originale

Thank you for inviting me to your party last week.

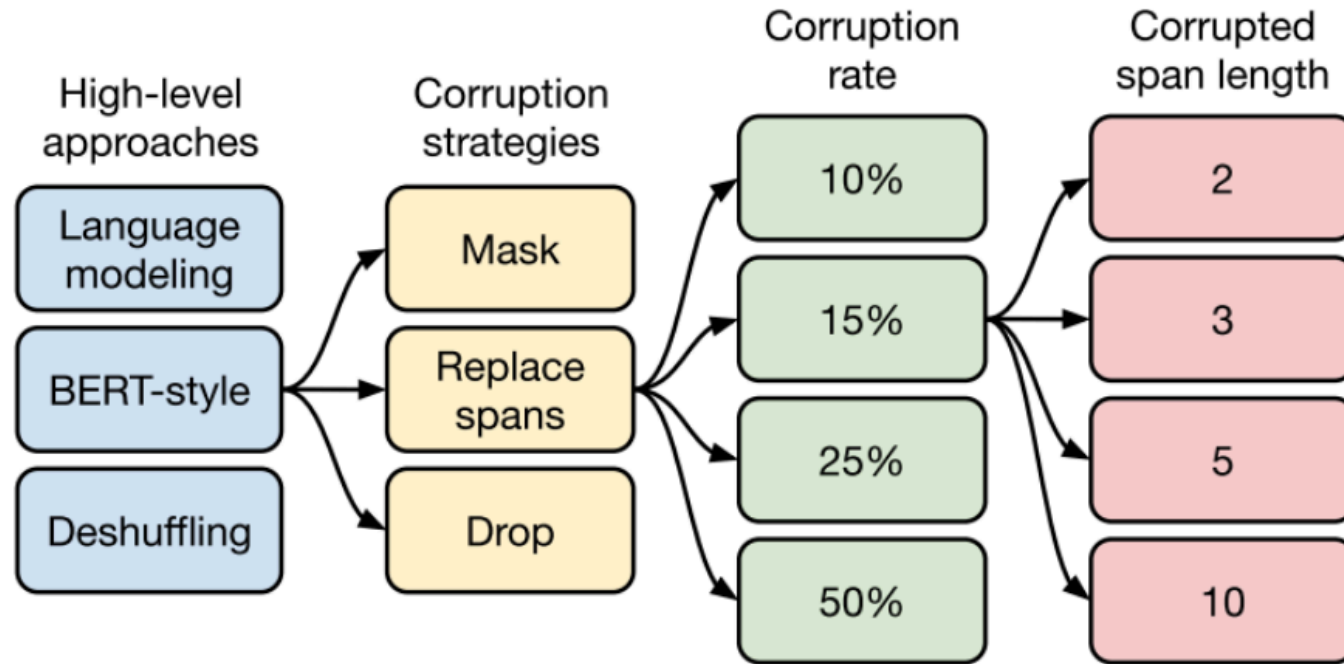
Entrée corrompue



Cibles

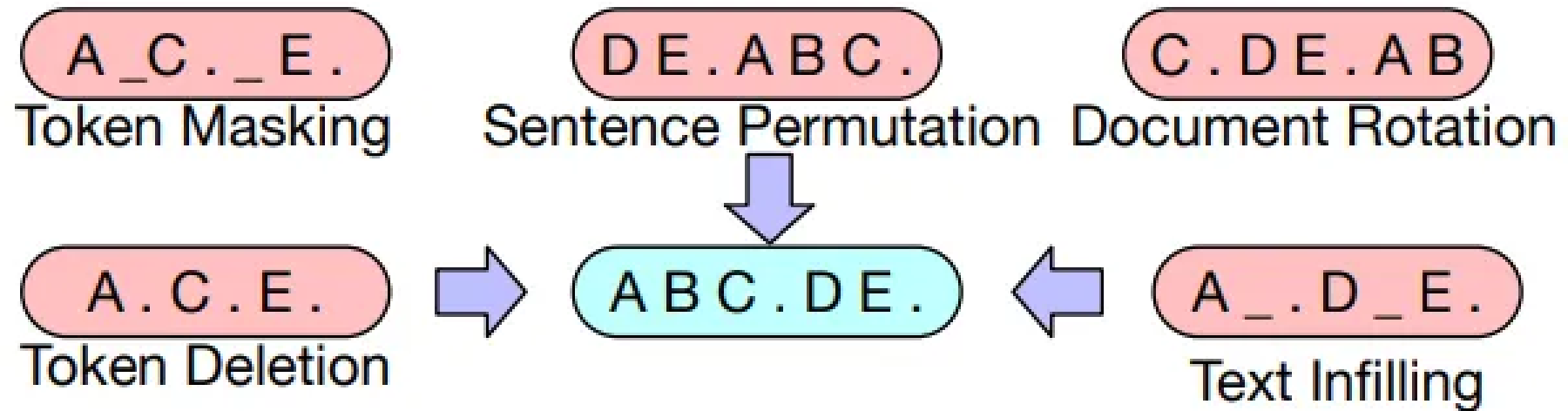


Préentraînement de modèles : denoising (T5)



Source : **Raffel et al. (2020)**

Préentraînement de modèles : denoising

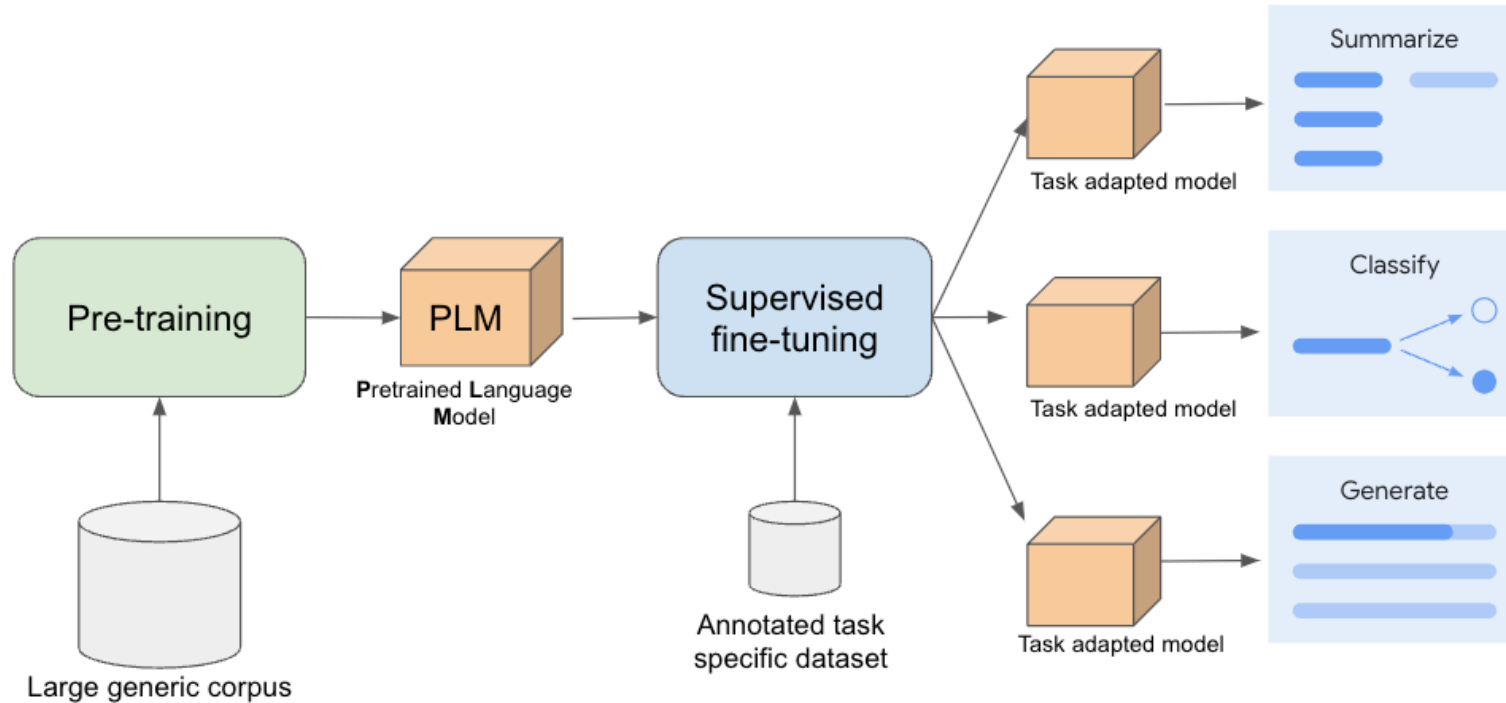


Source : <https://medium.com/@zaiinn440/a-comparative-analysis-of-llms-like-bert-bart-and-t5>

Préentraînement de modèles : affinage

- ♦ Modèles théoriquement déjà utilisables pour de la traduction, mais on fait généralement de l'affinage (*fine-tuning*) sur des données de traduction parallèles pour améliorer les performances.

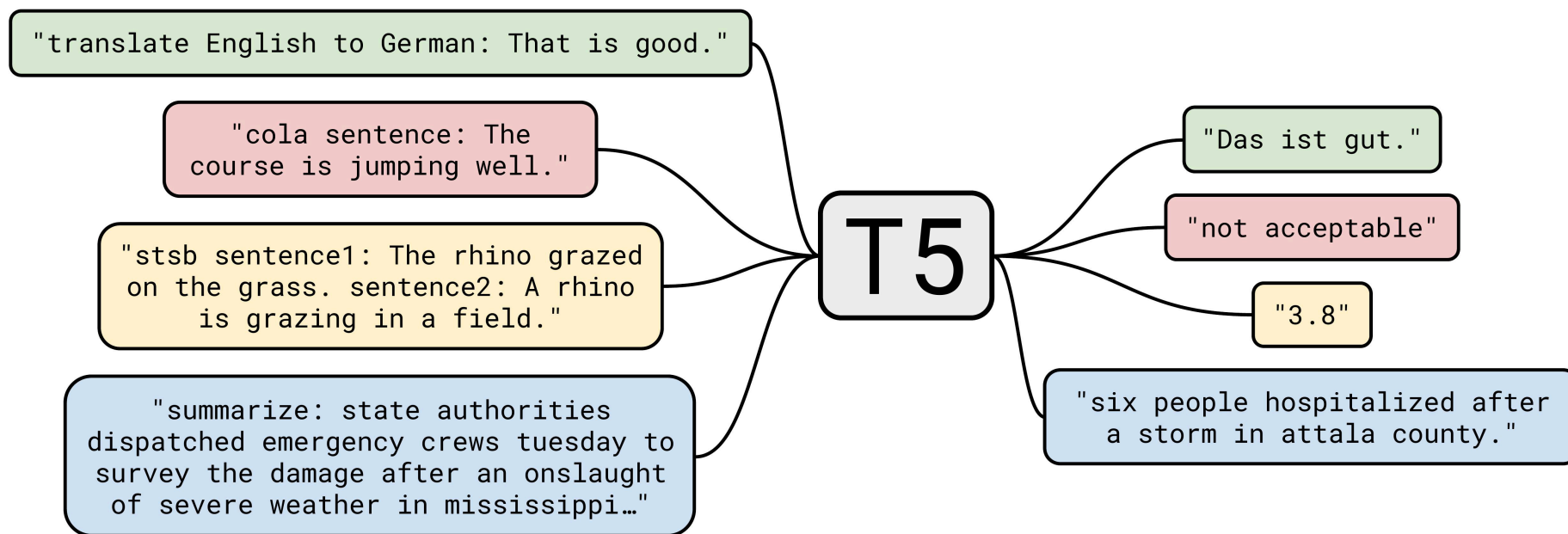
Préentraînement de modèles : affinage



Source : <https://cloud.google.com/blog/products/ai-machine-learning/supervised-fine-tuning-for-gemini-1.1>

T5 (Text-to-Text Transfer Transformer)

- ♦ Modèle Transformer séquence-à-séquence préentraîné sur plusieurs tâches.

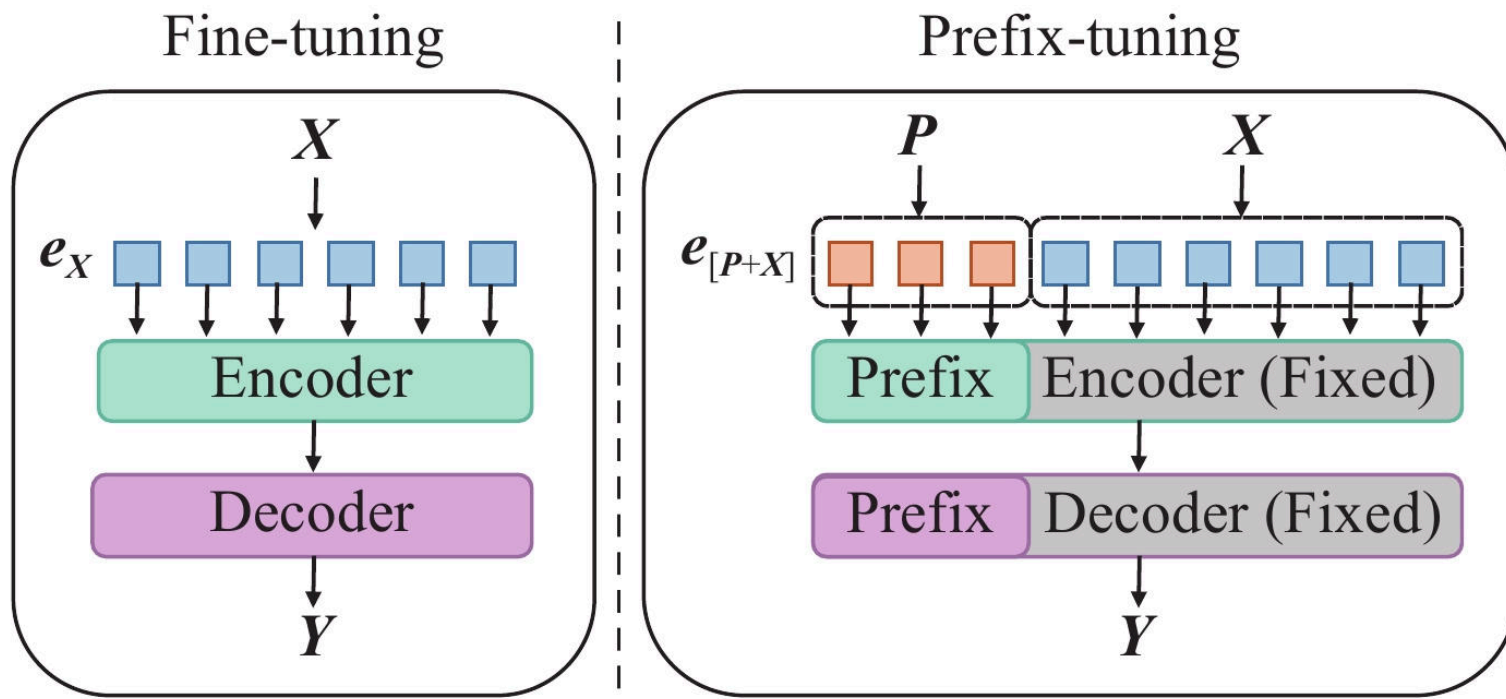


Source : **Raffel et al. (2020)**

T5 (Text-to-Text Transfer Transformer)

- ♦ Modèle Transformer séquence-à-séquence préentraîné sur plusieurs tâches.
- ♦ Très bonnes performances de base sur de nombreuses tâches (pour un modèle préentraîné), dont la traduction automatique.
- ♦ Repose sur du *prefix-tuning* : tokens « virtuels » qui conditionnent les réponses du modèle : ce sont en fait des vecteurs appris pendant l'entraînement et utilisés comme paires clé/valeur dans le mécanisme d'attention.
- ♦ Les poids du modèle restent inchangés (contrairement à de l'affinage).

Prefix-tuning



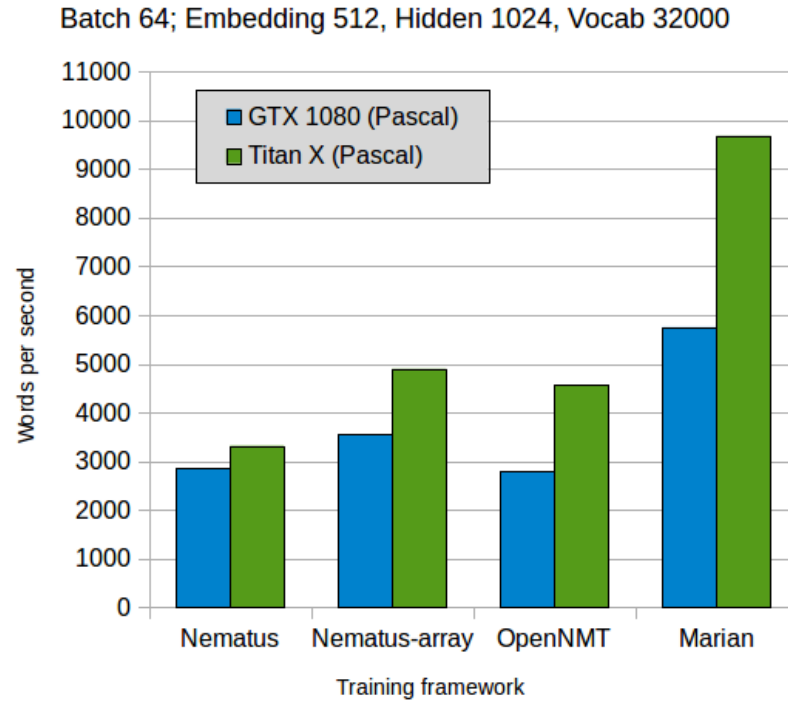
Source : **Chen et al. (2023)**

MarianMT

- ♦ Modèle de traduction automatique basé sur l'architecture Transformer, conçu par **Junczys-Dowmunt et al. (2018)**.
- ♦ Développé pour être léger et rapide (implémentation en C++), tout en offrant de bonnes performances de traduction.
- ♦ Site web : <https://marian-nmt.github.io/>

MarianMT : vitesse d'entraînement

Training speed on NVIDIA GTX 1080 and Titan X Pascal



Modèles autorégressifs

- ♦ Recours de plus en plus fréquent à des modèles autorégressifs (*Transformer decoder-only*) pour la traduction automatique (GPT-4, Llama 3/4, Qwen, Claude...) en raison de leurs performances de plus en plus élevées pour la tâche.
- ♦ Nous aborderons plus en détail ce type de modèle au prochain cours.



III. Mise en pratique

Mise en pratique

- ♦ Notebook disponible sur Moodle pour l'entraînement d'un modèle Seq2Seq T5 pour de la traduction automatique de l'anglais vers le français.

Bibliographie

- Chen, R., Li, F., et Wang, Z. (2023). Prefix-LSDPM: A Few-shot Oriented Online Learning Session Dropout Prediction Model. *Journal of East China University of Science and Technology*, 49(5), 754-763. [10.14135/j.cnki.1006-3080.20230206003](#)
- Dai, A. M., et Le, Q. V. (novembre 2015). *Semi-Supervised Sequence Learning* (Numéro arXiv:1511.01432). arXiv. [10.48550/arXiv.1511.01432](#)
- Devlin, J., Chang, M.-W., Lee, K., et Toutanova, K. (octobre 2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (Numéro arXiv:1810.04805). arXiv. [10.48550/arXiv.1810.04805](#)
- Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Hermann, U., Aji, A. F., Bogoychev, N., Martins, A. F. T., et Birch, A. (avril 2018). *Marian: Fast Neural Machine Translation in C++* (Numéro arXiv:1804.00344). arXiv. [10.48550/arXiv.1804.00344](#)
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., et Sutskever, I. (2019). *Language Models Are Unsupervised Multitask Learners*.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., et Liu, P. J. (juillet 2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., et Polosukhin, I. (juin 2017). *Attention Is All You Need* (Numéro arXiv:1706.03762). arXiv. [10.48550/arXiv.1706.03762](#)