Projet

Comparaison de performances entre un transformer encodeur-décodeur et décodeur pour la traduction machine

INF8225 - Intelligence artificielle : techniques probabilistes et d'apprentissage
Equipe 18

Alexis Lambert (2403731)
Ali Karaki (2402594)

Hiver 2025
Département de Génie Informatique et Génie Logiciel
École Polytechnique de Montréal

Comparaison de performances entre un transformer encodeur-décodeur et décodeur pour la traduction machine

Alexis Lambert , Ali Karaki
Département de Génie Informatique et Génie Logiciel, Ecole Polytechnique de Montréal
ali.karaki@polymtl.ca, alexis.lambert@polymtl.ca

Abstract

Cet article étudie la performance de deux architectures de transformer appliquées à la traduction automatique : un modèle encodeur-décodeur et un modèle décodeur uniquement. Nous présentons notre approche d'implémentation, les métriques utilisées pour évaluer la qualité des traductions, ainsi que les résultats expérimentaux obtenus. Nos analyses mettent en évidence les cas dans lesquels l'une ou l'autre architecture présente des avantages, en fonction de la complexité et de la taille des données d'entrainement.

1 Introduction

Depuis la publication de l'article *Attention is all you need* par Google Research [3], la traduction automatique a connu des avancées majeures. L'utilisation d'architectures transformer permet de contextualiser les mots dans une phrase selon leur position. L'objectif de ce projet est d'analyser les performances de deux architectures : transformer encodeur-décodeur et transformer décodeur seul, appliquées à des taches de traduction bilingue.

2 Implémentation des architectures

L'implémentation des deux architectures est disponible sur ce dépôt Github.

2.1 Transformer Encodeur-Décodeur

Description du problème

Pour une tâche de traduction automatique, pour une séquence d'entrée donnée, $x = (x_1, x_2, ..., x_n)$, où les xi sont des mots d'un langage on souhaite générer une séquence de sortie $y = (y_1, y_2, ..., y_k)$ où les yi sont des mots dans une langue différente. On peut ainsi modéliser cet objectif par la formule suivante:

$$y^* = \underset{y}{\text{argmax}}(p(y|x)) \tag{1}$$

C'est à dire qu'on souhaite obtenir la traduction la plus probable y de notre entrée x dans la langue cible.

Architecture du transformer encoder decoder

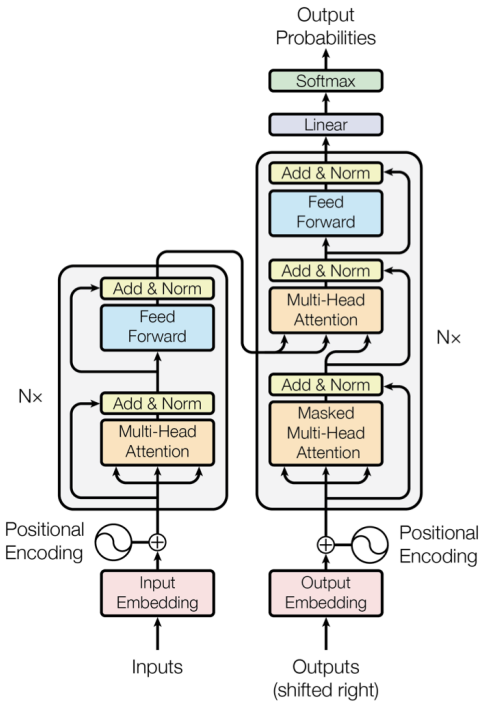

Figure 1: Architecture du modèle transformer encoder decoder [3]

L'implémentation du transformer encodeur décodeur est basé sur l'article *Attention is all you need.*

Le modèle est constitué de deux entités: une entité encodeur et une entité décodeur qui ont chacune une tâche spécifique à résoudre: L'encodeur encode l'entrée x en une séquence de représentations continues.

Le décodeur génère la sortie y de manière autorégressive en utilisant l'entrée encodée ainsi que les mots générés par le décodeur lui même.

L'encodeur est constitué de N couches identiques où chacune des couches applique un mécanisme de multihead self-attention à l'entrée suivi d'un réseau feed-forward.

Chaque mécanisme de self-attention réalise le calcul suiv-

ant:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{(d_k)}})V \quad (2)$$

où Q sont les requêtes, K les clés , V les valeurs et d_k la dimension des clés.

Le décodeur est constitué également de N couches identiques où chaque couche possède deux méchanismes d'attention: Un méchanisme de "masked self attention" et un méchanisme d'attention appliqué à la sortie de l'encodeur. A chaque étape, le décodeur réalise le calcul suivant:

$$p(y_t|y_{<t}, x) = softmax(W_o h_t) \quad (3)$$

où Wo est la matrice de poids et h_t l'état caché du décodeur à l'instant t.

Le modèle est ainsi entraîné en utilisant une technique du maximum de vraissemblance en minimisant une fonction de perte qui est l'entropie croisée.

Pour un ensemble de paires de données (x, y) la fonction de perte peut donc être écrite de la manière suivante:

$$L = -\sum_i \sum_{t=1}^{k} log(p(y_t|y_{<t}, x)) \quad (4)$$

2.2 Transformer Décodeur uniquement

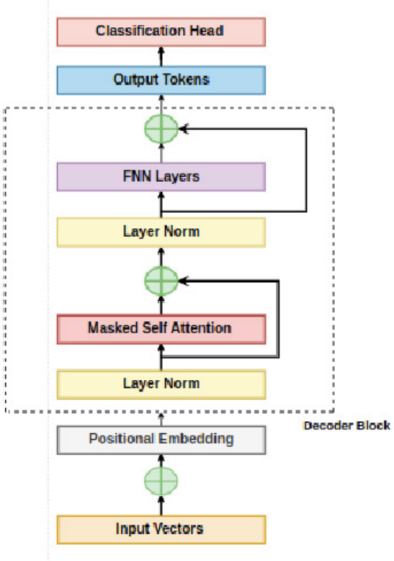

Figure 2: Architecture du modèle décodeur only provenant du blog *Mastering Decoder-Only Transformer: A Comprehensive Guide*

Architecture du transformer decoder only

Pour une architecture décodeur only, la tâche reste la même, on souhaite encore une fois pour une séquence d'entrée donnée, $x = (x_1, x_2, ..., x_n)$, générer une séquence de sortie $y = (y_1, y_2, ..., y_k)$ où les yi sont des mots dans une langue différente.

Seulement, cette architecture ne comprend pas de composante encodeur. On doit donc concaténer la séquence de sortie à la séquence d'entrée de notre modèle pour prédire la prochaine sortie de manière autorégressive:

$$x_{aug} = (x_1, x_2, ..., x_n, y_1, y_2, ..., y_{t-1}) \quad (5)$$

Avec cette séquence en entrée on prédit alors la sortie qui est y_t.

Le décodeur est donc constitué de N couches identiques qui utilisent des méchanismes de masked-self attention suivi de réseaux feed-forward.

A chaque étape, l'attention peut être écrite de la forme:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{(d_k)}} + M)V \quad (6)$$

où Q, K et V sont des projections de l'entrée z et M est une matrice contenant des masques afin d'empêcher au modèle d'utiliser les tokens futurs. Cette matrice contient $-\infty$ si le modèle à un instant t essaie d'avoir accès aux mots à un instant supérieur à t et 0 sinon. Celà empêche au modèle de "tricher" en utilisant la solution.

On note à nchaque instant la sortie des couches d'attention et des réseaux feedforward h_t et la probabilité d'obtenir le prochain mot y_t peut s'écrire de la manière suivante:

$$p(y_t|x_1, x_2, ..., x_n, y_1, y_2, ..., y_{t-1}) = softmax(W_o h_t) \quad (7)$$

et notre fonction de perte s'écrit:

$$L = -\sum_i \sum_{t=1}^{k} log(p(y_t|x_1, x_2, ..., x_n, y_1, y_2, ..., y_t - 1)) \quad (8)$$

3 Analyse des performances

Ensemble de données utilisé

Pour réaliser l'entraînement de nos modèles ainsi que pour nos expériences, nous utilisons le même ensemble de donnée que celui proposé dans le TP3. Cet ensemble est constitué de paires de phrases, une en anglais et sa traduction en français, délimitées par une tabulation. Les données proviennent du site Tatoeba et sont disponibles publiquement via le lien suivant: http://www.manythings.org/anki/fra-eng.zip. Ce dataset contient un grand nombre de paires de phrases de complexité variable, ce qui est approprié pour évaluer les performances des modèles de traduction dans divers scénarios.

Expériences

Pour nos expériences sur chaque architecture, nous faisons varier les hyperparamètres suivants : le *nombre d'époques* (de 5 à 50), la *taille maximale des séquences* (de 5 à 100) et la *taille des batchs* (de 32 à 512). Nous faisons varier chaque paramètre indépendamment, en fixant les deux autres à leurs valeurs de référence : *nombre d'époques* = 10, *taille maximale des séquences* = 10 et *taille de batch* = 128.

3.1 Modèle Encodeur-Décodeur

Nous présentons nos résultats dans les différents tableaux de mesures suivant:

Variation du nombre d'epochs

batch size	seqlen	epochs	Précision
128	10	5	0.9362
128	10	10	0.9451
128	10	20	0.9551
128	10	50	0.9590

Table 1: Précision sur l'ensemble de validation pour différentes valeurs d'**epochs**

Variation de la taille des batchs

batch size	seqlen	epochs	Précision
32	10	10	0.9437
64	10	10	0.9430
128	10	10	0.9451
512	10	10	0.9428

Table 2: Précision sur l'ensemble de validation pour différentes valeurs de **batch size**

Variation de la taille de séquence

batch size	seqlen	epochs	Précision
128	5	10	0.7984
128	10	10	0.9451
128	20	10	0.9414
128	50	10	0.9413
128	100	10	0.9459

Table 3: Précision sur l'ensemble de validation pour différentes valeurs de **taille de séquence**

On trace également la fonction de perte pour voir à quelle vitesse elle converge:

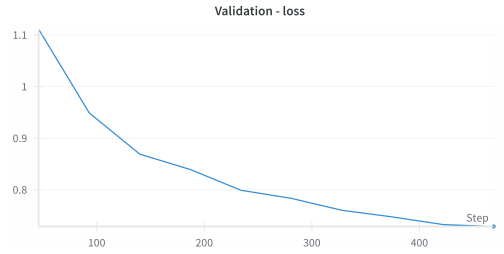

Figure 3: Perte sur l'ensemble de validation pour l'architecture encodeur décodeur

On trace également la précision sur l'ensemble d'entraînement:

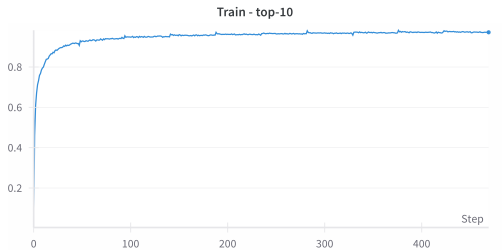

Figure 4: Précision sur l'ensemble d'entraînement

Nous avons pris pour ces exemples un batch size de 64, un nombre d'epochs de 10 et une longueur de séquence de 10 également.

3.2 Modèle Décodeur uniquement

Nous présentons nos résultats dans les différents tableaux de mesures suivant:

Variation du nombre d'epochs

batch size	seqlen	epochs	Précision
128	10	5	0.6424
128	10	10	0.6537
128	10	20	0.6607
128	10	50	0.6673

Table 4: Précision sur l'ensemble de validation pour différentes valeurs d'**epochs**

Variation de la taille des batchs

batch size	seqlen	epochs	Précision
32	10	10	0.6428
64	10	10	0.6523
128	10	10	0.6537
512	10	10	0.6471

Table 5: Précision sur l'ensemble de validation pour différentes valeurs de **batch size**

Variation de la taille de séquence

batch size	seqlen	epochs	Précision
128	5	10	0.7984
128	10	10	0.6537
128	20	10	0.8074
128	50	10	0.8068
128	100	10	0.6521

Table 6: Précision sur l'ensemble de validation pour différentes valeurs de **taille de séquence**

On trace aussi la convergence de la fonction de perte sur l'ensemble de validation pour la comparer au modèle précédent, ainsi que la précision sur l'ensemble d'entraînement. Nous avons pris pour l'exemple un batch size

de 64, un nombre d'epochs de 10 et une longueur de séquence de 10 également.

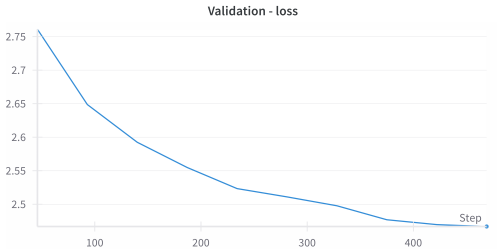

Figure 5: Perte sur l'ensemble de validation pour l'architecture encodeur décodeur

On trace également la précision sur l'ensemble d'entraînement:

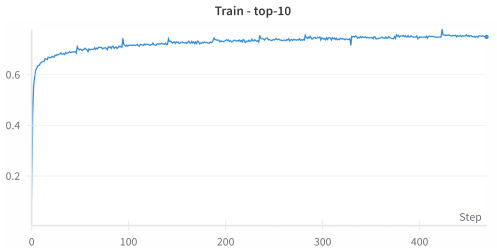

Figure 6: Précision sur l'ensemble d'entraînement

4 Comparaison

Ces résultats expérimentaux nous permettent de mettre en évidence plusieurs différences notables entre l'architecture transformer encodeur-décodeur et l'architecture transformer décodeur uniquement.

Concernant les performances de traduction, le modèle encodeur-décodeur atteint une précision plus élevée sur l'ensemble de validation que le modèle décodeur uniquement pour toutes les variations d'hyperparamètres testées.

Au niveau de la dynamique d'apprentissage, les courbes de perte (*loss*) montrent que le modèle encodeur-décodeur converge plus rapidement et atteint une valeur de perte finale plus faible que le modèle décodeur uniquement.

En observant la stabilité de l'apprentissage, les fluctuations de la perte sont plus importantes pour le modèle décodeur uniquement, alors que l'encodeur-décodeur présente une progression plus régulière.

Lors de la variation de la taille des batchs, les deux architectures montrent des performances relativement stables, mais le modèle encodeur-décodeur a toujours une meilleurs précision dans tous les cas.

En faisant varier la taille de séquence, la précision du modèle décodeur uniquement diminue de manière plus marquée que celle du modèle encodeur-décodeur.

Enfin, les résultats sur l'ensemble d'entraînement montrent également que le modèle encodeur-décodeur atteint une meilleure précision en moins d'époques.

Nous pouvons également comparer les performances temporelles des différentes architectures et l'on peut constater sur la figure 7 que l'architecture decoder-only reste plus rapide car elle nécessite moins d'opérations.

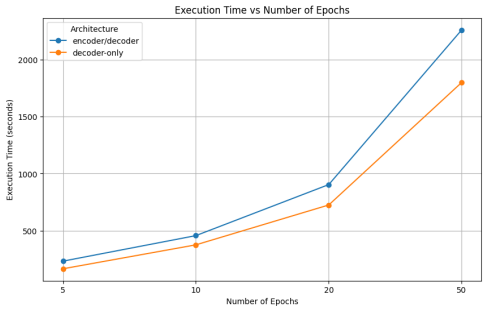

Figure 7: Comparaison du temps d'exécution entre les deux architectures

4.1 Résumé des observations

- **Précision** : supérieure pour le modèle encodeur-décodeur sur l'ensemble des expériences.
- **Convergence** : plus rapide pour le modèle encodeur-décodeur.
- **Stabilité** : meilleure pour le modèle encodeur-décodeur.
- **Sensibilité aux séquences longues** : plus forte dégradation pour le modèle décodeur uniquement.
- **Impact de la taille des batchs** : performances globalement stables pour les deux modèles, avec un avantage constant pour l'encodeur-décodeur.
- **Temps d'entraînement** : temps d'entraînements nettement moins élevés pour l'architecture encodeur décodeur.

Analyse des résultats

Nous avons pu voir avec ces expériences que l'architecture decoder only offrait des performances nettement moins bonnes que celles de l'architecture encodeur décodeur.

Les meilleures performances de l'architecture encodeur décodeur peuvent être expliquées par le fait qu'elle traite la séquence source de manière intégrale avnt de générer la réponse. L'architecture décodeur uniquement quant à elle prend la séquence source concaténée à la séquence cible et prédit le prochain mot en traitant simultanément la séquence source.

Cependant les temps d'entraînements pour l'architecture décodeur uniquement sont nettement moins élevés que pour l'architecture encodeur décodeur. Ceux ci sont expliqués par le fait que l'architecture décodeur only réalise un nombre d'opérations bien moins élevé vu qu'elle ne possède pas de composante encodeur qui traite la séquence source.

Ce type d'architecture est la plus utilisée pour les modèles génératifs du style ChatGpt pour ses avantages de rapidité et sa non nécessité de traiter toute la séquence source avec une composante encodeur. Cependant elle est moins performante pour des tâches de style traduction automatique

où la séquence source est fixée et doit être traduite le plus précisément possible.

5 Conclusion

Dans ce projet, nous avons étudié deux types d'architectures de transformers appliquées à la tâche de traduction automatique : l'architecture encodeur-décodeur classique et l'architecture décodeur uniquement.

À travers l'analyse théorique, nous avons mis en évidence les avantages spécifiques de chaque approche. Le modèle encodeur-décodeur, spécialement conçu pour les tâches de séquence-à-séquence, offre une meilleure exploitation de la séquence d'entrée grâce à une séparation claire entre l'encodage et le décodage, permettant une compréhension plus fine du contexte source.

À l'inverse, l'architecture décodeur uniquement, plus simple et flexible, permet d'aborder la traduction dans une perspective générative, en concaténant source et cible dans une seule séquence. Cependant, cette approche peut souffrir d'une compréhension moins fine du contexte source, notamment lorsque la séquence est longue.

Les expériences que l'on a pu effectuer confirment ces observations de manière empirique. Les mesures des performances des deux architectures ont bien montré que l'architecture encodeur-décodeur obtenait généralement de meilleurs résultats que l'architecture décodeur uniquement.

Ainsi, bien que le transformer décodeur uniquement offre une grande flexibilité et une simplicité d'implémentation, le transformer encodeur-décodeur reste, pour des tâches de traduction automatique ciblées, l'approche la plus performante et adaptée.

References

[1] T. Zhou, X. Zhou, Y. Wu, et al. *A Survey of Large Language Models*. arXiv preprint arXiv:2304.04052, 2023. https://arxiv.org/pdf/2304.04052

[2] Z. Wang, et al. *Scaling Laws for Large Language Model Training*. arXiv preprint arXiv:2409.13747, 2024. https://arxiv.org/pdf/2409.13747

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin. *Attention is All You Need*. In Advances in Neural Information Processing Systems, 2017. https://arxiv.org/pdf/1706.03762

[4] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. *Language Models are Unsupervised Multitask Learners*.