POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

INF8225

Intelligence artificielle : techniques probabilistes et d'apprentissage
Rapport

TP3 - Machine Translation

Élèves :
Marc Zhang 2312403

Enseignant :
Christopher Pal

6 avril 2024

Table des matières

1 Comparaison des modèles

Nous allons tout d'abord comparer nos différents modèles (RNN, GRU, transformer) avec les mêmes paramètres.

Paramètres d'entrainement :

— epochs = 5
— batch size = 128
— lr = 1e-3
— betas = (0.9, 0.99)

Paramètres des modèles :

— n heads = 4,
— dim embedding = 196
— dim hidden = 256
— n layers = 3
— dropout = 0.1

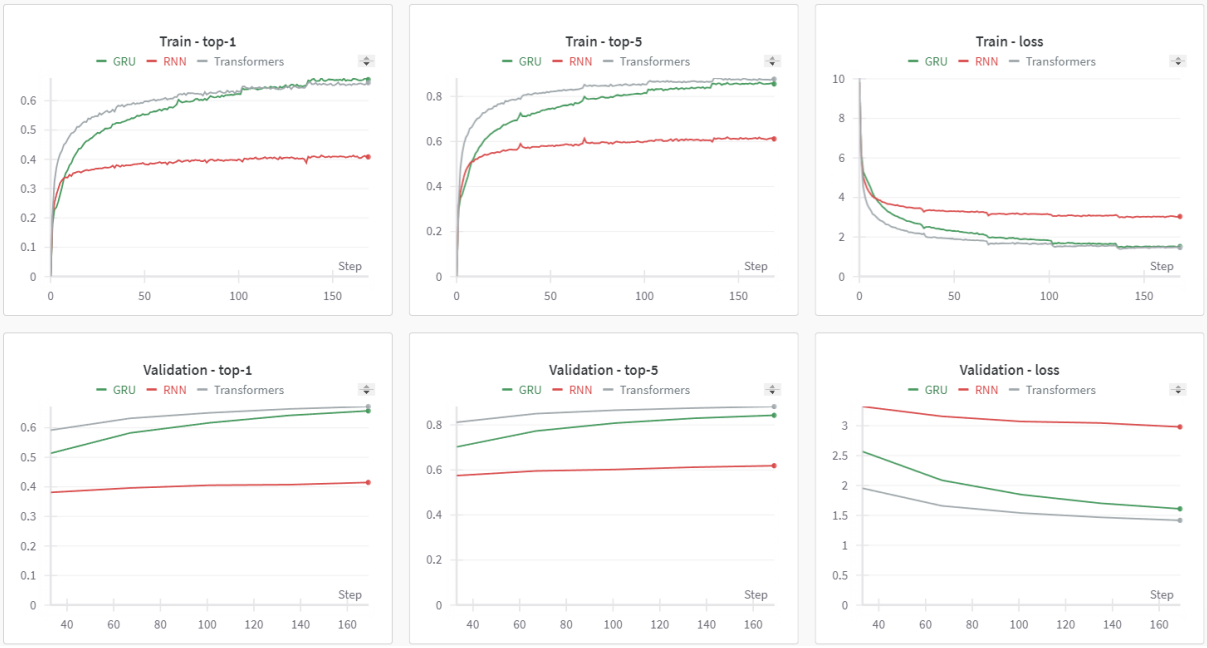

FIGURE 1 – Comparaison des méthodes

On observe bien le fait que le modèle GRU a de bien meilleurs résultats que le modèle RNN, montrant que le contrôle des informations transmises au cours du temps permet bien d'éviter les problèmes des RNN.

On observe que le modèle Transformer possède de très bonnes performances tout en étant bien plus rapide à entraîner comparé à RNN ou GRU (12 min en moyenne pour le Transformer, et 30 à 40 minutes en moyenne pour le RNN et le GRU). Cependant, on observe que la précision pour la réponse top 1 arrive à être légèrement meilleure pour le GRU que pour le Transformer, mais le GRU ne dépasse pas le Transformer lors de la validation, le GRU a donc plus tendance que le transformer à subir du surapprentissage. Tout cela montre les qualités du Transformer qui arrive à avoir les meilleures précisions tout en étant beaucoup plus rapide à calculer.

2 Etude des hyperparamètres

Pour observer l'influence des hyperparamètres sur les performances du Transformer, nous allons entraîner des modèles Transformers en modifiant un à un les hyperparamètres, tout en gardant tous les autres hyperparamètres du modèle et pour l'entraînement identiques à ceux utilisés dans la partie 1.

2.1 Nombre de têtes

Nous effectuons les entraînements avec n_heads = 2, 4 ou 8. Voici les résultats :

FIGURE 2 – Performance du transformer avec différents valeurs de n_heads

Avoir plusieurs têtes d'attention permet de réaliser plusieurs calculs de l'attention de manière indépendante, ce qui laisse supposer qu'un nombre élevé de têtes pourrait conduire à une meilleure précision. D'après nos résultats, il est clair qu'il y a une amélioration des performances du modèle pour n_heads = 4 par rapport à n_heads = 2, même si cette amélioration est légère. Toutefois, il semble qu'il existe une limite à l'amélioration de la précision apportée par l'augmentation du nombre de têtes, comme le démontre le modèle avec n_heads = 8, qui ne présente pas de meilleures performances comparativement au modèle avec n_heads = 4.

2.2 dimension des embeddings

Nous effectuons des entraînements avec dim_embeddings = 100, 196, ou 400. Voici les résultats :

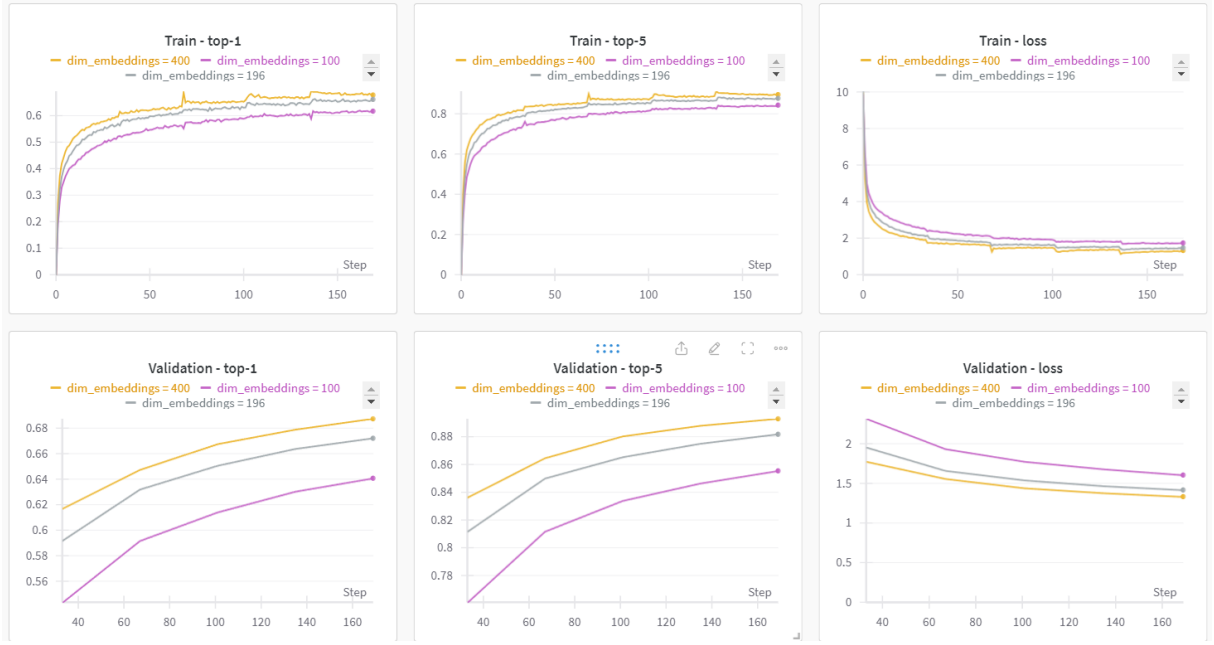

FIGURE 3 – Performance du transformer avec différents valeurs de dim_embeddings

La dimension des embeddings influence directement la quantité d'informations qu'ils peuvent contenir, ce qui implique qu'une dimension plus élevée pourrait améliorer la précision du modèle. Nos résultats confirment cette hypothèse : la précision augmente à mesure que la dimension des embeddings s'accroît. Cependant, l'augmentation de précision est moins prononcée entre 196 et 400 qu'entre 100 et 196, indiquant un rendement décroissant à mesure que la dimension augmente.

2.3 dimension des couches

Nous effectuons des entraînements avec dim_hidden = 128, 256, ou 512. Voici les résultats :

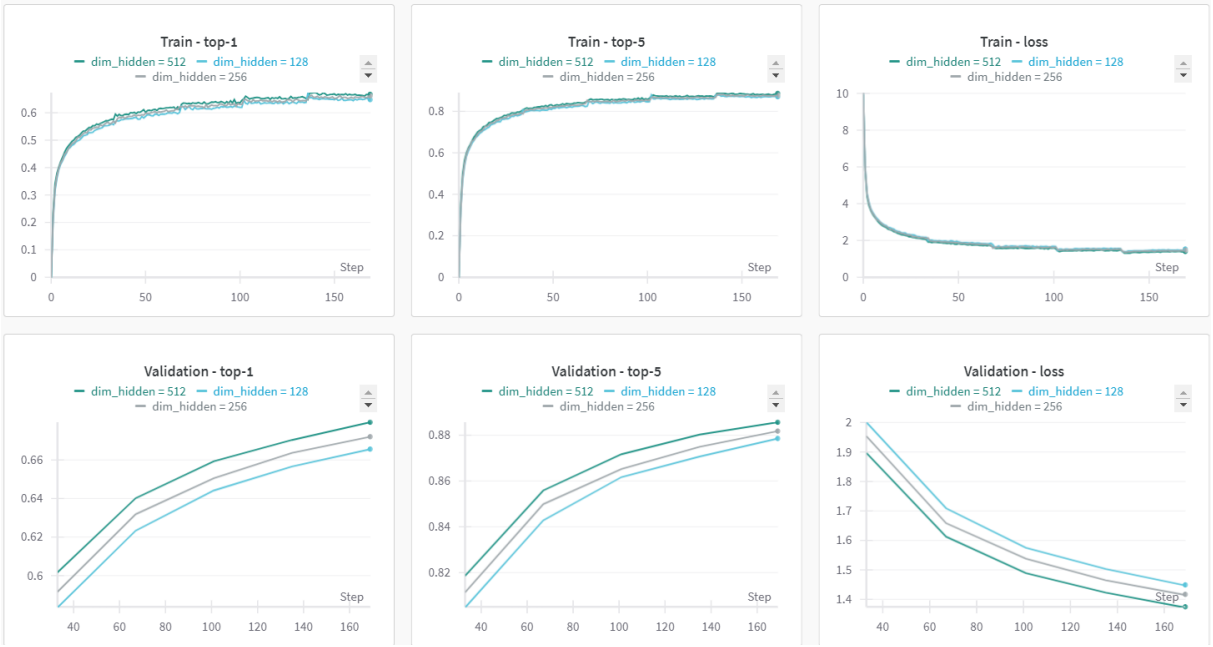

FIGURE 4 – Performance du transformer avec différents valeurs de dim_hidden

On constate que, lors de l'entraînement, la dimension des couches pour le réseau feed-forward a peu d'impact sur la précision. Cependant, sur le dataset de validation, une augmentation claire des performances est observée à mesure que la dimension des couches augmente.

2.4 nombre de couches

Nous effectuons des entraînements avec n_layers = 1, 2, ou 4. Voici les résultats :

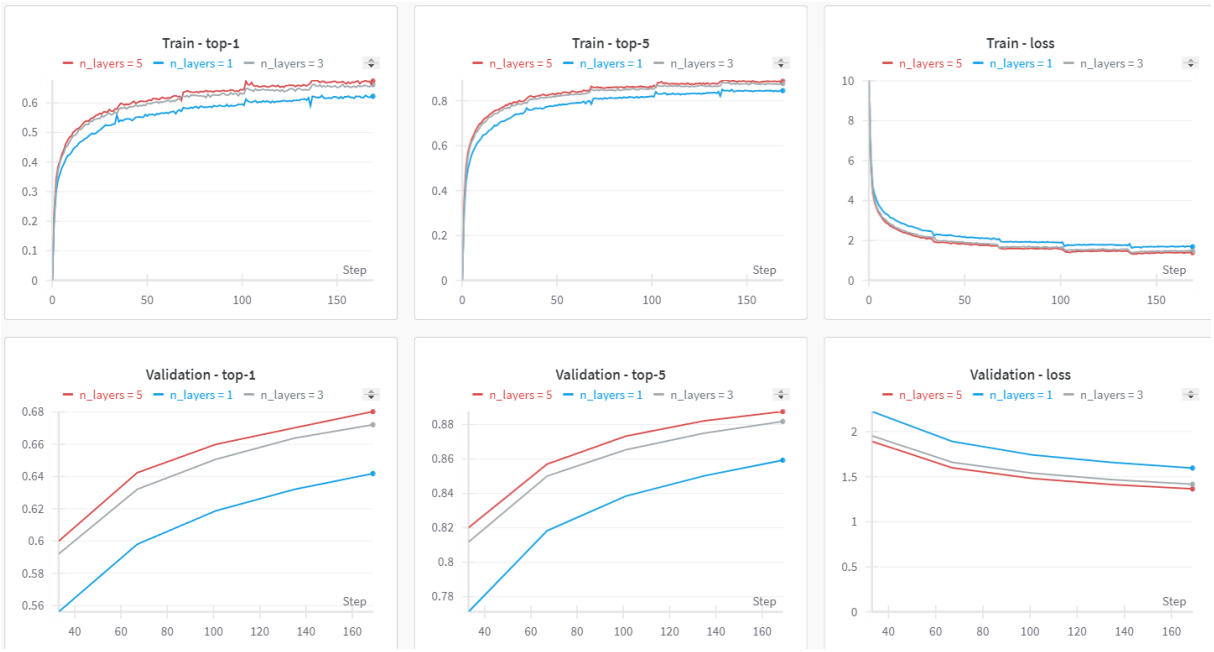

FIGURE 5 – Performance du transformer avec différents valeurs de n_layers

Nous constatons une nette amélioration de la précision lors du passage d'un modèle de 1 à 3 couches. Cependant, cette amélioration devient beaucoup plus modeste lorsqu'on passe de 3 à 5 couches.

2.5 dropout

Nous effectuons des entraînements avec dropout = 0.01, 0.1, 0.2, ou 0.4. Voici les résultats :

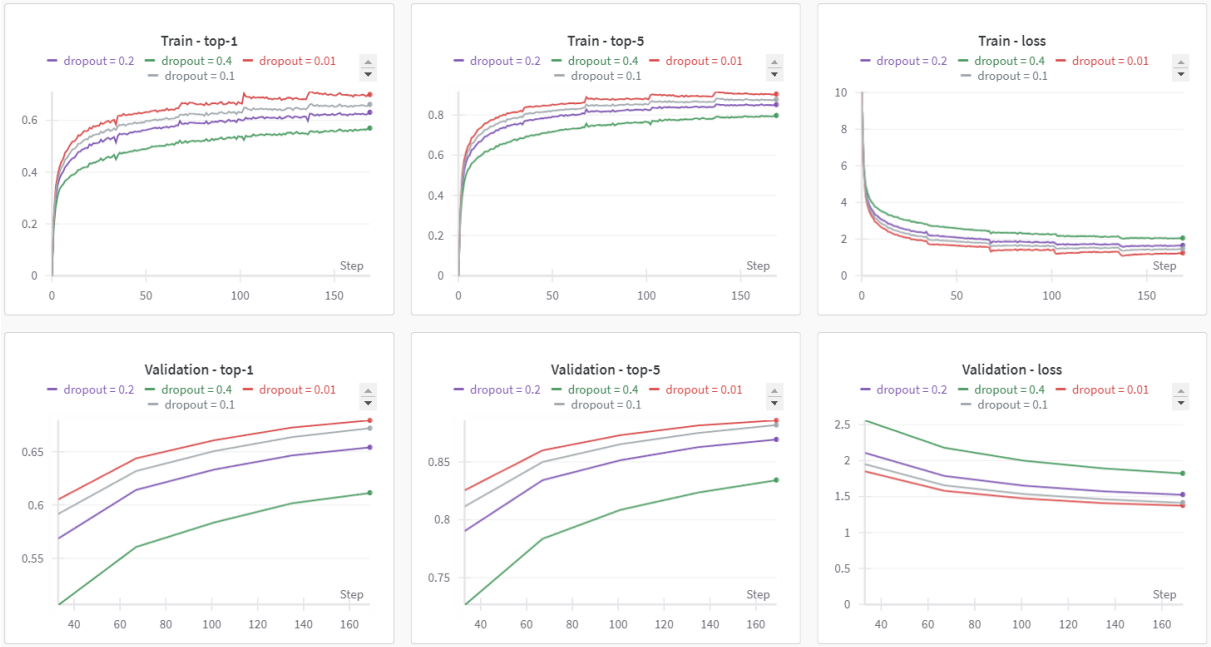

FIGURE 6 – Performance du transformer avec différents valeurs de dropout

Le dropout est une technique qui consiste à désactiver aléatoirement des neurones, ce qui aide le réseau à mieux généraliser en réduisant le surapprentissage. Nos observations révèlent que plus la valeur du dropout est faible, plus la précision est élevée. Il semble donc que le dropout, dans notre cas, ne fasse qu'entraver l'apprentissage du modèle.