Rapport TP3 – INF8225

Étudiants

Guillaume Thibault 1948612

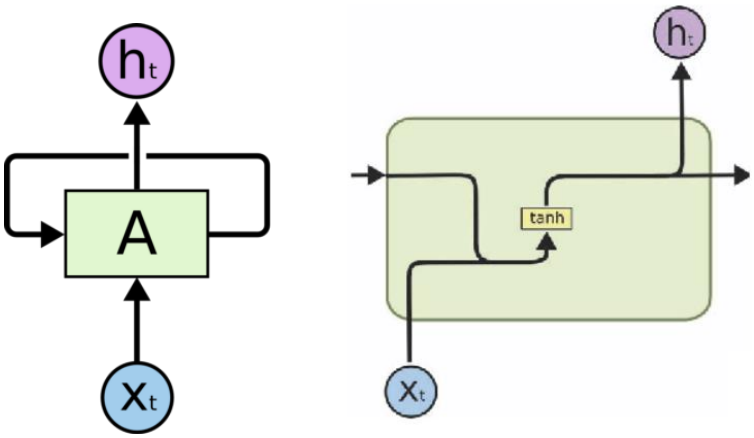
Julien Witty 1949837

11 avril 2022

Question 1

RNN

Les réseaux neuronaux récurrents (RNN) sont conçus pour fonctionner avec des données séquentielles. Le RNN utilise les informations précédentes dans la séquence pour produire la sortie. Dans notre cas, les données utilisées sont des phrases, alors les mots sont passés au modèle un à l'a suite de l'autre. Le model peu alors faire une prédiction du mot courant en se basant sur l'information résiduelle des mots précédant. Bien que ce modèle soit souvent illustré avec plusieurs couches, le model est récurrent et c'est toujours la même cellule qui est appelée (soit toujours les mêmes paramètres et biais). Le modèle comporte une seule fonction d'activation soit : $h_t = thanh(W_{hh}h_{h-1} + W_{xh}x_t)$. Les problèmes majeurs de ce modèle sont que les paramètres sont toujours réécrits et qu'il devient alors difficile de préserver l'information dans le temps de plus de souffrir de « vanishing gradient ».



[1] RNN vs GRU vs LSTM

GRU

Une architecture de RNN afin de diminuer l'impact de ces problèmes est l'architecture Gated Recurrent Units (GRU). Ce modèle fonctionne de la même façon que le modèle précédent, mais comporte une différente opération dans la cellule A. Le GRU comporte deux portes, soit «Reset» et «Update». Chaque porte comporte leurs propres paramètres et biais, ce qui améliore la mémoire de la cellule. Les différentes opérations du modèle sont :

Candidate cell :

$$n_t = \tanh\left(W_{in}x_t + b_{in} + r_t * \left(W_{hn}h_{(t-1)} + b_{in}\right)\right)$$

Reset cell :

$$r_t = sigmoid(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr})$$

Update cell :

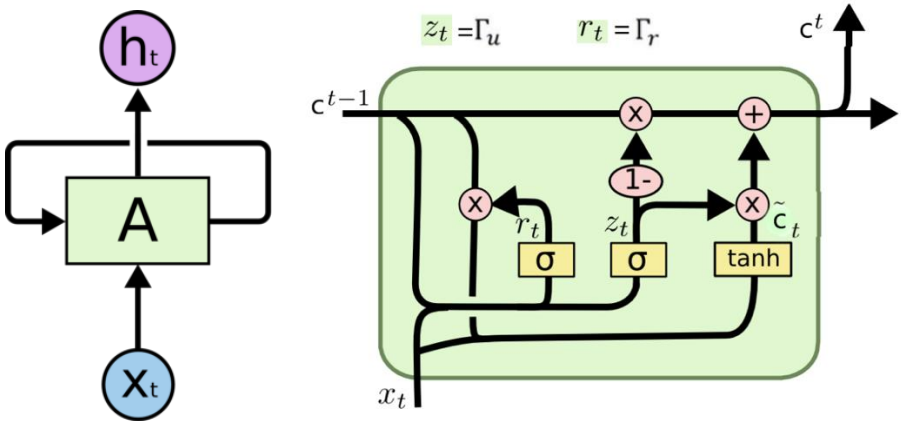
$$z_t = sigmoid(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}$$

Cell state :

$$h_t = (1 - z_t) * n_t + z_t * h_{(t-1)}$$

Ainsi qu'une fonction d'activation pour la sortie.
La cellule Update détermine si l'état de la cellule devrait être mis à jour avec la cellule candidate. Par la suite, la cellule Reset détermine si l'état est important ou non avant la mise à jour. Finalement, la cellule finale met à jour la mémoire en considérant la cellule Update.

Cette architecture permet de mieux préserver la mémoire et les informations des séquences précédentes.



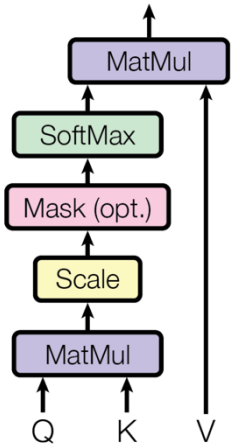
[1] RNN vs GRU vs LSTM

Transformer
L'architecture du troisième modèle réalisé dans ce travail est très différente. Au lieu de passer au modèle les données de façon séquentielles, toutes les données sont passées au modèle d'un coup et le modèle utilise un mécanisme d'attention qui permet de déterminer quelle partie de la séquence d'entrée est importante pour chaque étape.

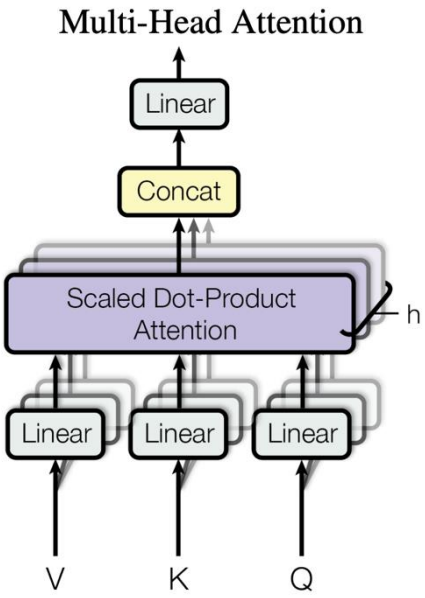
Le calcul permettant ce principe est calculé avec :

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

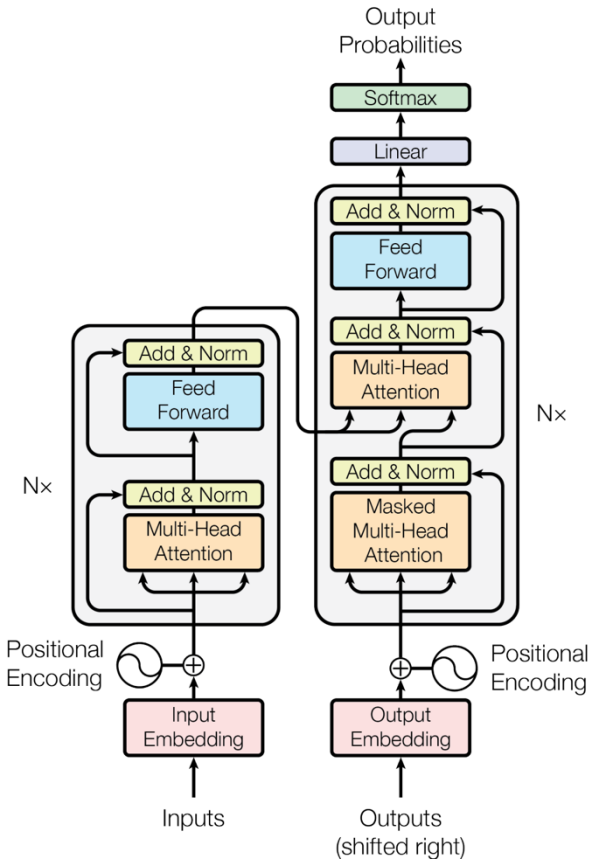
Ou Q est un vecteur contenant représentant un mot, K est un vecteur représentant tous les mots de la séquence et V est aussi un vecteur représentant tous les mots de la séquence.



Le modèle comporte plusieurs de ces attentions qui sont calculées dans en parallèle, soit les têtes d'attention. Ces attentions sont alors concaténées pour pouvoir extraire plusieurs relations entre les mots. Ce mécanisme d'attention peut aussi être utilisé avec des masques afin de cachées les des mots.



[2] Attention is all you need



Ce principe est utile, car le modèle comporte un encodé et un décodeur. Ceux-ci sont seulement des réseaux de neurone simple permettant d'encoder l'information de la séquence source et de décoder l'information dans le format de la séquence finale. Ils utilisent les résultats des têtes d'attentions combinés à l'entrée pour permettre de propager l'information quand l'attention n'est pas encore entrainée. Cette architecture utilise comme entrée la source et le résultat de façon d'écaler lors de l'entrainement. Cette façon de faire est très différente des architectures précédant. Donner la source et le résultat au modèle permet d'entrainer le décodeur pour qu'il puisse convertir la sortie de l'encoder dans le format des résultats. De plus, le fait que les données sont décalées et que la fonctionnalité de masque est utilisée dans le décodeur fait en sorte que le modèle peut seulement se baser sur les éléments qui viennent avant la réponse pour une certaine donnée.

Source

[1] RNN vs GRU vs LSTM, https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573
[2] Attention is all you need, https://arxiv.org/abs/1706.03762

Question 2

Dû au fait que toutes les données sont passées au modèle transformer simultanément, un principe permettant d'incorporer l'ordre des mots doit être introduit pour ce modèle. Le RNN et GRU n'ont pas besoin de ce principe, car les données leur sont passées de façon séquentielle ainsi, l'ordre des mots est conservé.

Le principe utilisé est le « Positionnal Encoding ». Afin que ce principe soit fonctionnel, celui-ci doit respecter plusieurs critères, soit :

- Il doit produire un encodage unique pour chaque pas de temps (position du mot dans une phrase).
- La distance entre deux pas de temps doit être cohérente entre des phrases de longueurs différentes.
- Le modèle doit pouvoir être généralisé à des phrases plus longues sans aucun effort. Ses valeurs doivent être bornées.
- Il doit être déterministe.

Une solution à ce problème est d'utiliser un vecteur à plus grande dimension contenant l'information à propos des différentes positions de la séquence. Pour cela, l'utilisation de fonction sinusoïdale est utile pour donner un aspect d'avancement dans les mots. Les calculs utilisés afin de réaliser cela sont :

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

where

$$\omega_k = \frac{1}{10000^{2k/d}}$$

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

L'intuition est que la fréquence des fonctions sinusoïdales diminue lorsque l'on avance dans la séquence. Cela permet alors de donner au modèle un aspect de position pour les éléments de la séquence.

Référence :

[3] The Positionnal Encoding https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Question 3

Avant de pouvoir entrainer les modèles, les données ont dû être normalisé en suivant le processus suivant :

1. Enlever les phrases trop longueur, car celle si prennent trop d'espace mémoire.
2. Enlever les caractères de fin de ligne « \n »
3. Créer un vocabulaire avec notre ensemble de données. Chaque mot doit avoir un index qui le référence.
4. Convertir toutes les phrases en tokens.

   Ex :
   ```
   >>> en_sentence = 'We are on the moon!'
   >>> eng_token = ['<bos>'] + en_tokenizer(en_sentence) + ['<eos>']
   >>> print(eng_token)
   ['<bos>', 'We', 'are', 'on', 'the', 'moon', '!', '<eos>']
   ```

   Si un mot inconnu par notre tokenizer `['<unk>']`

5. Ajouter du padding pour que tous nos phrases soit le la même longueur

   Ex avec une longueur de 9:
   ```
   ['<bos>', 'We', 'are', 'on', 'the', 'moon', '!', '<eos>', '<pad>']
   ```

6. Convertir tous les tokens par leur index dans le vocabulaire.

   Ex :
   ```
   [2, 563, 432, 24, 3244, 34, 987, 3, 0]
   ```

7. Passer le vecteur dans le embedding.

   Cette étape a pour but de convertir les valeurs discrètes en valeur continue. Cela se fait par l'utilisation de vecteur à faible dimension qui permet de convertir la valeur catégorique en une position. Ce vecteur à préalablement été appris. Nous l'utilisons un modèle pré-entrainer pour faire la conversion.

8. Pour les transformeurs, l'embedding des mots est passé dans le positional encoding
9. La donnée peut alors être utilisée modèle.

Résultat des différentes architectures

RNN

Résultats de la dernière époque (5):

```
Train -   loss: 2.39     top-1: 0.52     top-5: 0.71     top-10: 0.78
Eval -    loss: 2.19     top-1: 0.55     top-5: 0.74     top-10: 0.80
```

Résultats au cours de l'entrainement :

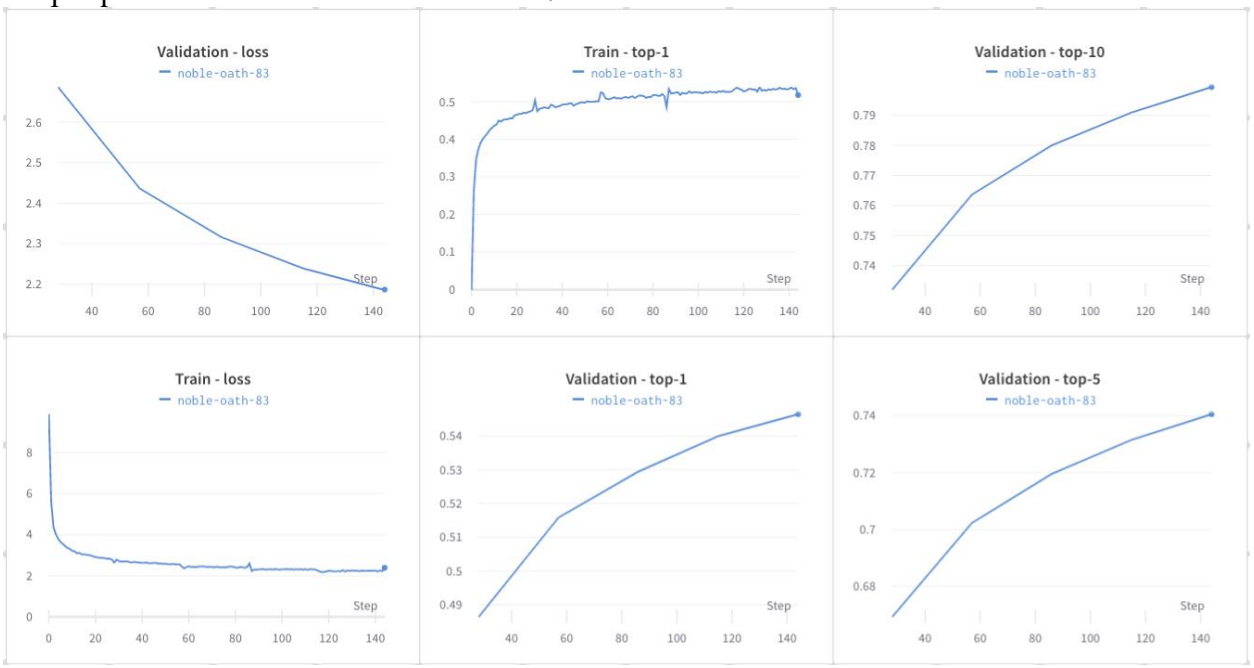
epoch	train - sourc	train - targe	train - predict	train - likelihc	validation - so	validation - tar	validation - pre	validation - lik
1	A dog suddenly jumped at me.	Un chien me sauta soudain dessus.	Une voiture s'est produit.	0.00009668	We know it's a hassle.	Nous savons que c'est pénible.	Nous savons très bien.	0.002807
2	We don't want them to decrease our paycheck.	Nous ne voulons pas qu'ils réduisent notre paie.	Nous ne voulons pas parler français.	0.0005623	Are your ankles swollen when you wake up in the morning?	Vos chevilles sont-elles enflées lorsque vous vous réveillez le matin ?	Sont-vous dans ta chambre ?	0.00003803
3	Today is Thursday.	Nous sommes jeudi.	Aujourd'hui.	0.04226	I know how to settle this.	Je sais comment régler ça.	Je sais Tom français.	0.008569
4	The capital of France is Paris.	La capitale de la France, c'est Paris.	La population de Tom est mort.	0.00003492	I knew there would be trouble when Tom walked into the room.	Je savais que Tom allait nous apporter des problèmes, dès qu'il est entré dans la pièce.	Je savais que Tom avait l'habitude de jouer au tennis.	5.894e-7
5	Keep an eye on my suitcases.	Gardez un œil sur mes valises.	Garde une tasse de café.	0.002291	These lemon and almond tarts are scrumptious.	Ces tartelettes au citron et amandes sont délicieuses.	Ces deux garçons sont dans la même classe.	0.000007636

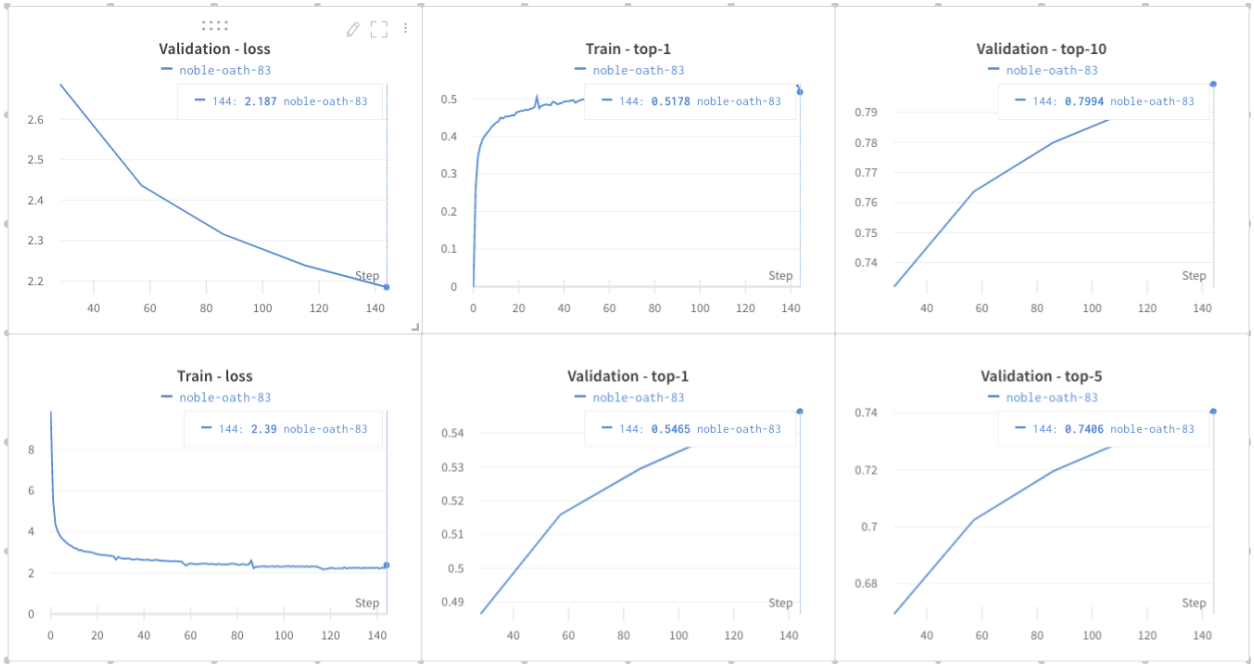
Test manuel :

```
sentence = "My cat eats his supper with a smile."
0. (0.00259%)     Ma mère ressemble à ses amis.
1. (0.00131%)     Mon chat a tué sa voiture.
2. (0.00129%)     Mon chat à un jeune homme.
3. (0.00081%)     Mon chat a tourné à ses amis.
4. (0.00077%)     Mon chat s'est cassé dans sa chambre.

sentence = "I called my friend to say hi."
0. (0.12164%)     J'ai demandé à Tom de faire ça.
1. (0.09858%)     J'ai demandé à Tom quoi faire.
2. (0.09048%)     J'ai demandé à Tom quelque chose.
3. (0.08970%)     J'ai donné quelque chose à manger.
4. (0.08403%)     J'ai demandé à Tom de le faire.
```

Graphiques de l’entrainements du modèle :





GRU

Résultats de la dernière époque (5) :

```
Train -   loss: 1.93     top-1: 0.60    top-5: 0.79    top-10: 0.85
Eval -    loss: 1.83     top-1: 0.60    top-5: 0.80    top-10: 0.85
```

Résultats au cours de l'entrainement :

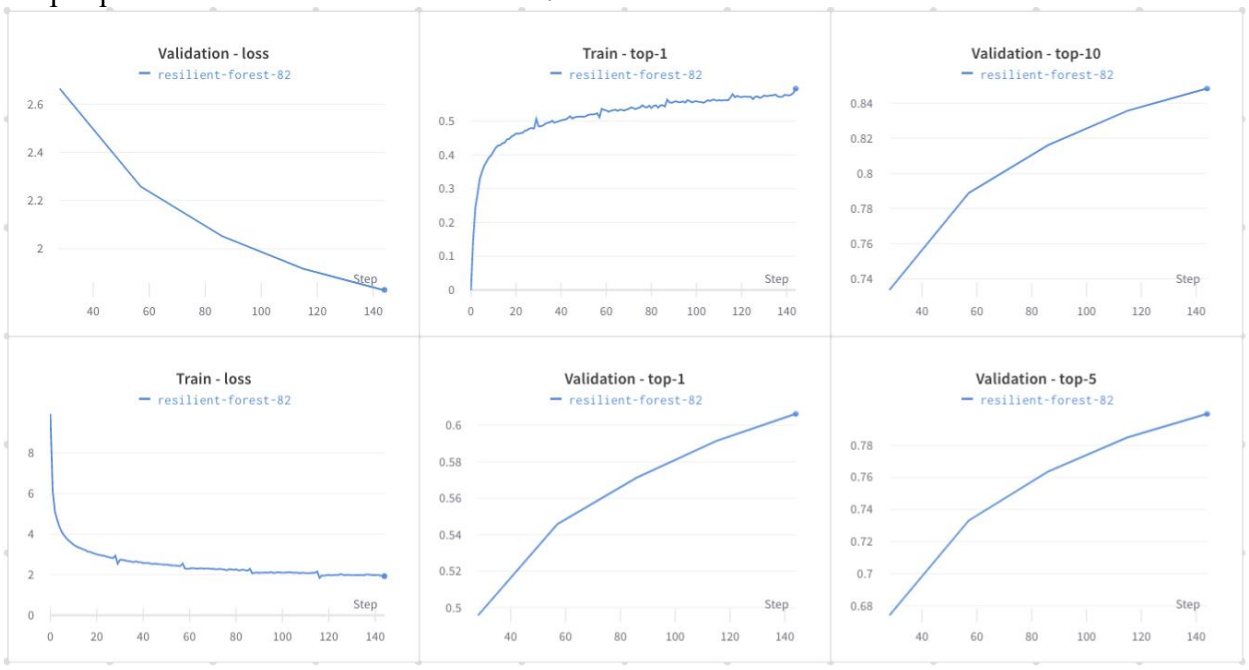
epoch	train - sourc	train - targe	train - predict	train - likelihc	validation - so	validation - ta	validation - pr	validation - lik
1	The landing was perfect.	L'atterrissage était parfait.	Le chien était différents.	0.00023	Tom is a sleepwalker.	Tom est somnambule.	Tom est vide.	0.002922
2	Can you remember his name?	Peux-tu te rappeler son nom ?	Peux-tu pu parler français ?	0.0004559	They lost their dog.	Elles ont perdu leur chien.	Ils ont perdu le livre.	0.004578
3	There is no hope of his recovery.	Il n'y a pas d'espoir pour son rétablissement.	Il n'y a pas d'argent.	0.0131	I saw the message.	J'ai vu le message.	J'ai vu le livre.	0.02676
4	Strange to say, he didn't know the news.	C'est étrange à dire, mais il ne connaissait pas la nouvelle.	Promets, il ne pouvait pas parler à Mary.	0.0005567	He let me go.	Il m'a laissée partir.	Il m'a vus.	0.05629
5	How do you want me to do that?	Comment veux-tu que je le fasse ?	Comment veux-tu que je le fasse ?	0.05752	Can you remember anything?	Pouvez-vous vous rappeler quoi que ce soit ?	Pouvez-vous me dire quoi que ce soit ?	0.0643

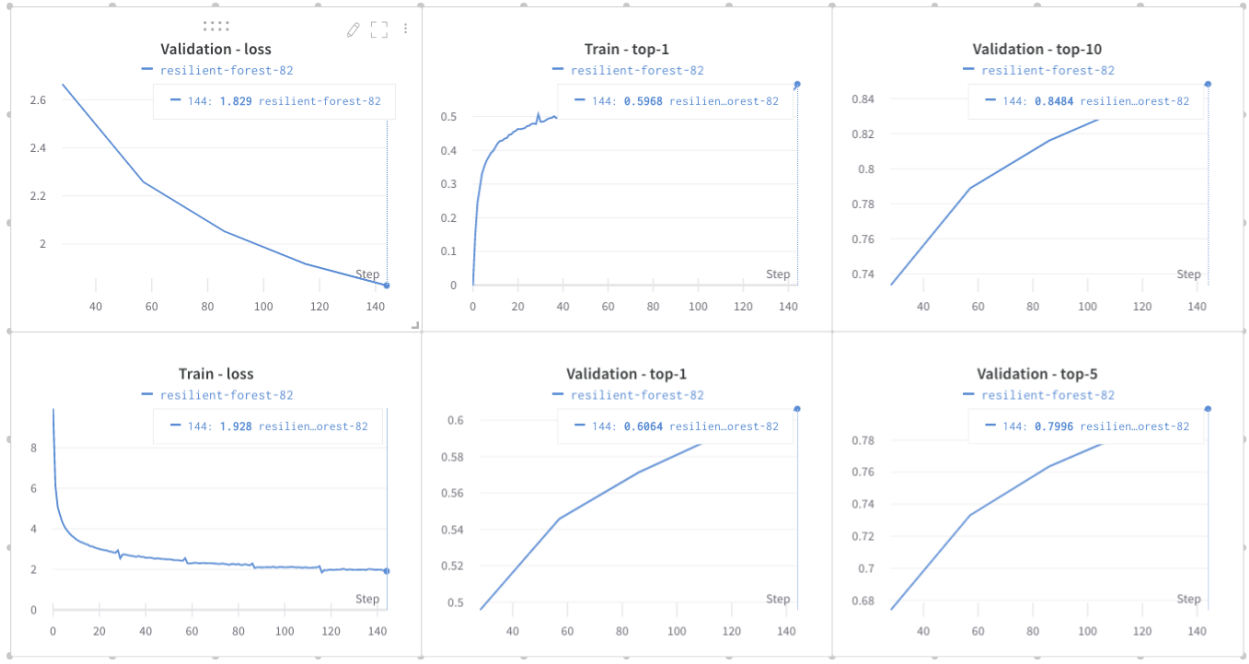
Test manuel :

```
sentence = "My cat eats his supper with a smile."
0. (0.11012%)     Ma mère aime son numéro de téléphone.
1. (0.09078%)     Ma mère étudie son numéro de téléphone.
2. (0.04530%)     Mon père aime son numéro de téléphone.
3. (0.04341%)     Mon chat aime son numéro de téléphone.
4. (0.04069%)     Mon père étudie son numéro de téléphone.

sentence = "I called my friend to say hi."
0. (0.20836%)     J'ai appelé mon nom.
1. (0.17567%)     J'ai appelé quelque chose à faire.
2. (0.13191%)     J'ai appelé mon père avec lui.
3. (0.11102%)     J'ai appelé mon nom avec lui.
4. (0.11069%)     J'ai appelé mon nom avec moi.
```

Graphiques de l’entrainements du modèle :





Transformer

Résultats de la dernière époque (5) :

```
Train -   loss: 1.58     top-1: 0.65    top-5: 0.84    top-10: 0.89
Eval -    loss: 1.36     top-1: 0.70    top-5: 0.88    top-10: 0.91
```

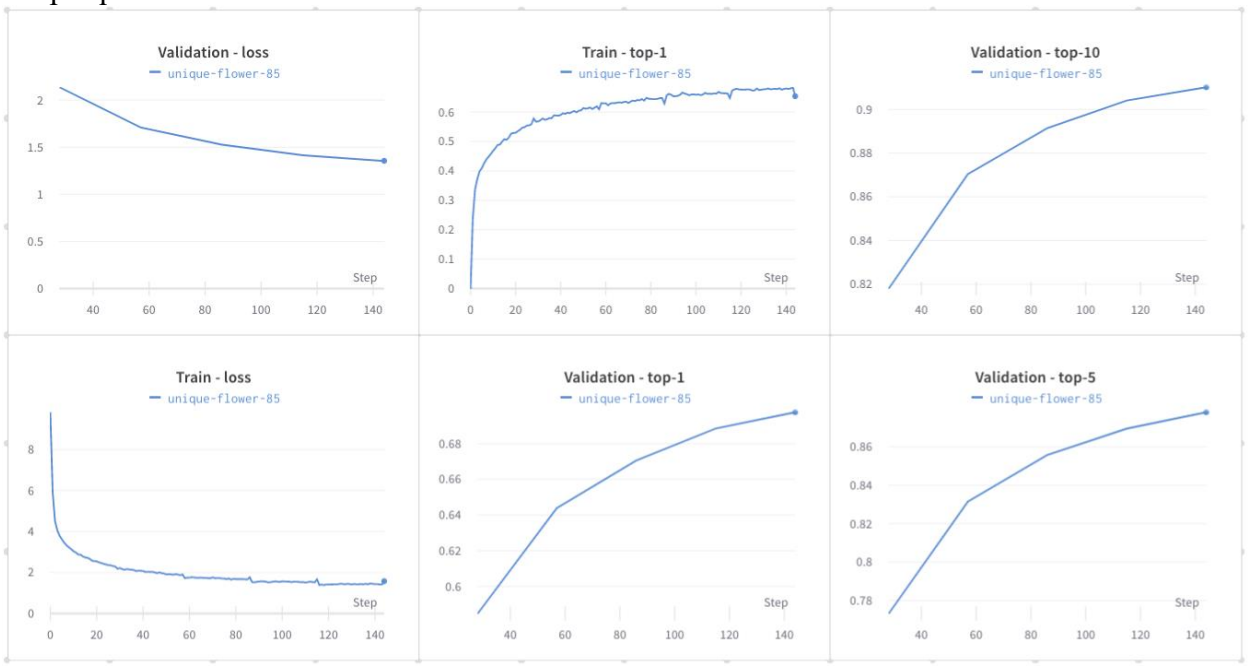
Résultats au cours de l'entrainement :

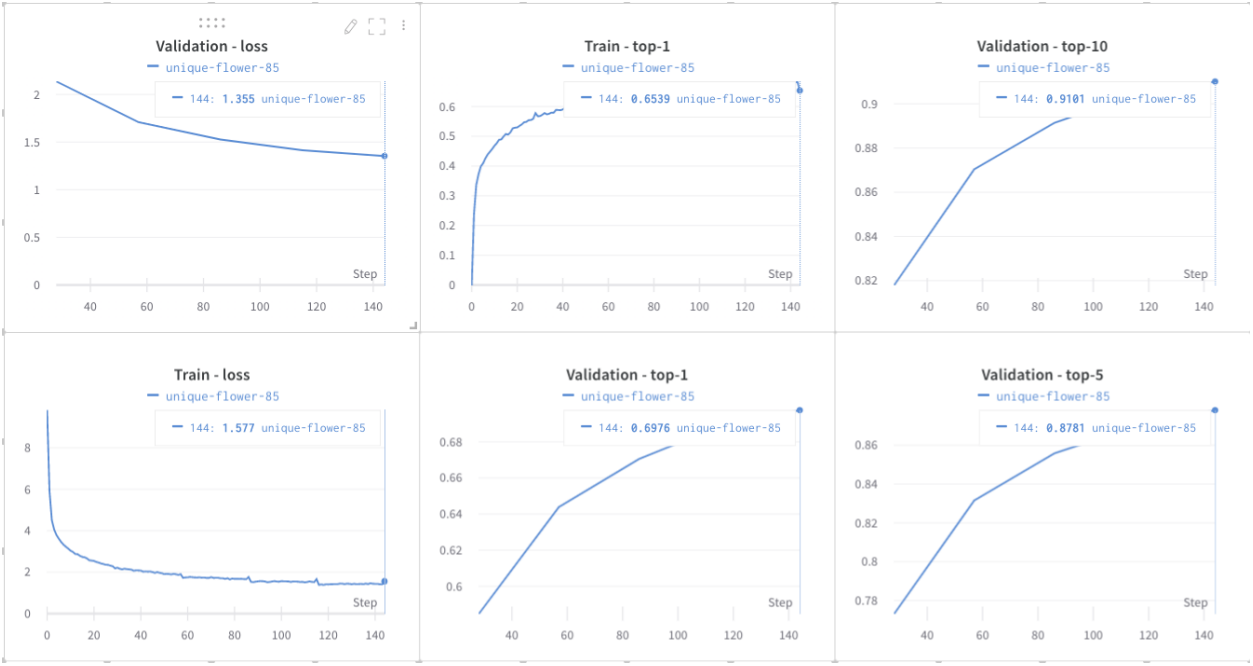
epoch	train - sourc	train - targe	train - predict	train - likelihc	validation - so	validation - ta	validation - pr	validation - lik
1	Tom sounds impressed.	Tom a l'air impressionné.	Tom semble impressionné.	0.03418	New stamps will be issued next month.	De nouveaux timbres seront émis le mois prochain.	La fois.	0.0005161
2	Do you have a lot of free time?	Est-ce que tu as beaucoup de temps libre ?	Avez-vous beaucoup de temps ?	0.08819	I want another cup of coffee.	Je veux une autre tasse de café.	Je veux une autre tasse de café.	0.1868
3	I forgot to give you back your umbrella.	J'ai omis de vous rendre votre parapluie.	J'ai oublié de te donner ton parapluie.	0.212	We're right here.	Nous sommes juste là.	Nous sommes juste ici.	0.1522
4	We have different ways of thinking.	Nous avons des façons de penser différentes.	Nous avons des expériences différents différente.	0.002979	I showed Tom the postcard I got from Mary.	J'ai montré à Tom la carte postale que j'ai reçue de Mary.	J'ai montré la carte de Mary.	0.03131
5	Which would you rather do, stay at home and watch TV or go out and see a movie?	Que préféreriez-vous faire, rester à regarder la télévision à la maison, ou sortir au cinéma ?	Quel film ou sortir-vous à la maison ?	0.000004847	I didn't tell you everything.	Je ne t'ai pas tout dit.	Je ne vous ai pas tout dit.	0.1978

```
sentence = "I called my friend to say hi."
0. (3.53737%)     J'ai appelé à mon ami.
1. (1.71857%)     J'ai appelé mon ami.
2. (0.85219%)     J'ai appelé avec mon ami.
3. (0.62319%)     J'ai appelé à dire à mon ami.
4. (0.40681%)     J'ai appelé mon ami à dire.

sentence = "My cat eats his supper with a smile."
0. (4.33348%)     Mon chat mange avec un sourire.
1. (2.56410%)     Mon chat mange un sourire avec son sourire.
2. (2.43644%)     Mon chat mange de sourire.
3. (2.02664%)     Mon chat mange avec son sourire.
4. (1.79241%)     Mon chat mange de sourire avec un sourire.
```

Graphiques de l’entrainements du modèle :





Expérimentation :

Dans ce laboratoire, nous avons implémenté le RNN, le GRU ainsi qu'un transformer pour traduire les mots dans une phrase.

Les résultats préliminaires de la configuration de base place le transformer en première place en termes de précision. Le GRU se place en deuxième position suivie du RNN.

Le tableau I compile les différents résultats du tableau :

Tableau I : Précision des modèles d'apprentissage après 5 epochs

Type	**Training set**		
	Top-1	**Top-5**	**Top-10**
RNN	0.55	0.74	0.80
GRU	0.61	0.81	0.85
Transformer	0.70	0.88	0.91

L'*accuracy* n'est pas nécessairement la meilleure mesure pour estimer la qualité d'une prédiction. C'est pourquoi que l'un des premiers réflexes que nous avons eu lors de nos expériences était d'ajouter une métrique permettant d'estimer l'efficacité de traduction. Celle choisie a été le score BLEU (BiLingual Evaluation Understudy) [1]. Cet indice de performance mesure la performance entre les données traduites et la référence. Le but est d'avoir le résultat le plus proche de 1 qui signifie un match parfait. Il aurait été possible d'utiliser d'autres métriques comme NIST, METEOR, TER et caractTER. Nous avons utilisé l'algorithme beam search pour trouver la séquence la plus probable.

Dans le cadre de ce laboratoire, nous nous sommes particulièrement intéressés à l'effet des hyperparamètres sur les performances des différents modèles. Le score bleu permet de mesurer la précision sur un corpus. Toutefois, il est important de mentionner qu'avant prit un échantillon de 50 phrases de l'ensemble de validation pour mesurer le score bleu dans nos expériences.

Mesure de l'impact des paramètres sur la performance du modèle

Hyperparamètre		**Score BLEU**	
		RNN	**Transformer**
Taille de l'encodage	100	0.10	0.16
	160	0.09	0.27
	180	0.07	0.17
	196	0.12	0.21
Dropout	0.1	0.12	0.24
	0.2	0.08	0.25
	0.3	-	0.21
	04	-	

*Les autres paramètres sont fixés à la configuration de base, l'architecture GRU n'a pas été représenté dans le tableau par manque de temps.

Il est intéressant de remarquer que les performances du Transformers diminuent lorsque la taille de l'encodage devient trop élevée. Cela peut s'expliquer le fait qu'une trop grande dimensionnalité peut avoir un impact sur la qualité des résultats.

Nos résultats corroborent les résultats de l'article. Nous obtenons un BLEU score très similaire à ceux obtenus dans l'article. De plus, le transformer dépasse en termes de performance les performances du RNN et du GRU. [2]

Une autre métrique qui a décidé d'être explorée est de faire varier le nombre de têtes d'attention dans le transformer. L'ajout du nombre de tête n'a pas eu un impact significatif sur la performance du modèle. Ce résultat est surprenant, car des modèles très puissants comme le modèle BERT ont utilisé 16 têtes d'attention.

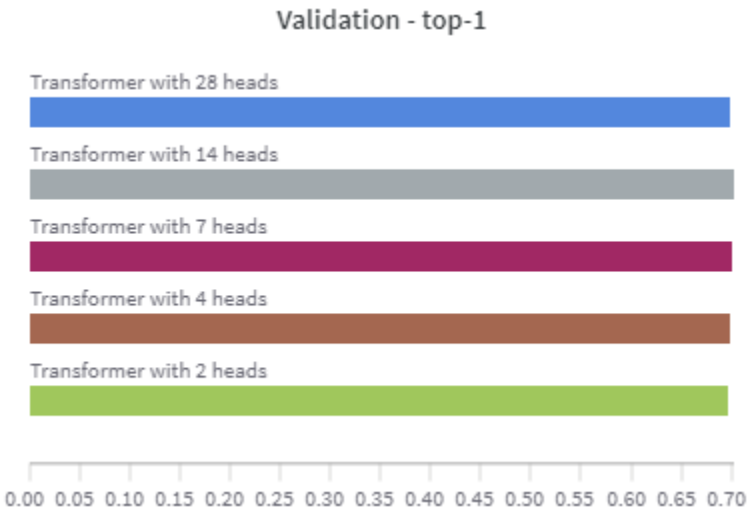


Figure 1. Accuracy sur le nombre de tête d'attention

L’hypothèse de ce résultat est que les modèles n’ont pas eu le temps de terminer l’entrainement après seulement 5 epochs ou bien que la mesure *d’accuracy* n’est pas assez précise pour faire la distinction. Il est important que l’ensemble d’entrainement ne présente pas de signe de surentrainement. Nous avons entrainé le modèle avec 10 epochs, il avait des signes de surentrainement, car la fonction de perte de l’ensemble d’entrainement donnait un coût plus petit que celle de l’ensemble de validation. Il aurait été intéressant de mesurer l’impact entre le nombre de têtes d’entrainement ainsi que le nombre d’epoch.

Finalement, le jeu de dropout testé n’a pas eu d’impact majeur sur les résultats du modèle.

Référence :

[1] Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311-318).
[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.